Des cartes combinatoires pour la construction automatique de modèles d'environnement par un robot mobile

Delphine Dufourd

n° d'ordre :

THÈSE

présentée

pour obtenir

Le TITRE de DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE

par

Mlle Delphine DUFOURD

Titre de la thèse :

Des cartes combinatoires pour la construction automatique de modèles d'environnement par un robot mobile

soutenue le 9 novembre 2005 devant le jury composé de :

MM	**Didier**	**ARQUÈS**	**Président**
	Christian	**LAUGIER**	**Rapporteur**
	Roland	**SIEGWART**	**Rapporteur**
	Patrick	**RIVES**	**Examinateur**
	Raja	**CHATILA**	**co-Directeur de thèse**
	Dominique	**LUZEAUX**	**co-Directeur de thèse**

RESUME :

Des cartes combinatoires pour la construction automatique de modèles d'environnement par un robot mobile

Ce travail s'inscrit dans la problématique classique de localisation et de cartographie simultanées pour un robot mobile évoluant en milieu intérieur supposé inconnu. Son originalité réside dans la définition d'un modèle de carte très structuré fondé sur un outil algébrique appelé « carte combinatoire », qui combine plusieurs types de représentations géométriques (modèles surfaciques et cartes basées sur des primitives géométriques) et fournit des informations topologiques telles que les liens d'adjacence. Nous détaillons la chaîne algorithmique permettant de construire des cartes en ligne suivant ce modèle, avec un robot équipé d'un télémètre laser à balayage : il s'agit d'adapter les techniques habituelles basées sur le filtrage de Kalman afin de gérer les relations d'adjacence (appariement de chaînes polygonales, définition de points de cassure virtuels, mises à jour géométrique et topologique spécifiques). Des résultats expérimentaux illustrent et valident les divers mécanismes mis en œuvre.

Mots clefs : Robotique, Cartographie, Carte combinatoire, SLAM, Localisation, Couche topologique.

ABSTRACT :

Combinatorial maps for simultaneous localization and map building (SLAM) with a mobile robot

This thesis focuses on the well-known Simultaneous Localization And Map-building (SLAM) problem for indoor mobile robots. The novelty of this work lies in the definition of a well-structured map model based on an algebraic tool called "combinatorial map" which combines different kinds of geometric representations (space-based, grid-based as well as feature-based formats) and provides topological information such as adjacency links between map elements. We describe the whole algorithm designed to build maps on line according to this model, using a robot equipped with a laser scanner. Classical techniques relying on Kalman filtering are adapted in order to deal with adjacency relationships (via polyline matching, the use of virtual break-points and specific geometric and topological update operations). Exeprimental results are presented to illustrate and validate the various mecanisms involved in this process.

Key words : Robotics, Map building, Combinatorial map, SLAM, Localization, Topological layer.

REMERCIEMENTS

Tout d'abord, j'adresse mes plus chaleureux remerciements à mes deux co-directeurs de thèse, MM. Raja Chatila et Dominique Luzeaux, pour m'avoir proposé ce sujet passionnant, pour m'avoir fait confiance sur certaines pistes techniques et pour avoir orienté mon travail à travers leurs remarques très pertinentes, généralement sources de longs développements ultérieurs (la prise en compte de toutes les corrélations par exemple !)... En particulier, je remercie Dominique de m'avoir initialement accueillie au département GIP du Centre Technique d'Arcueil (aujourd'hui CEP Arcueil) de la DGA, d'avoir suivi avec enthousiasme mon DEA puis mon doctorat et pour les multiples éclairages qu'il m'a apportés dans le cadre de ma thèse (et plus généralement, dans le cadre de mes activités à la DGA). Et je remercie Raja d'avoir guidé avec un grand sérieux mon cheminement de doctorante, pour l'attention qu'il m'a toujours accordée, et pour m'avoir offert de nombreuses occasions d'approfondir mes connaissances en robotique sur Toulouse.

Je voudrais également remercier l'ensemble des membres du jury : le président, M. Didier Arquès, les rapporteurs, MM. Roland Siegwart et Christian Laugier, ainsi que M. Patrick Rives, pour avoir accepté d'examiner ce long manuscrit, pour m'avoir fait l'honneur d'évaluer mon travail et pour avoir soulevé des questions particulièrement intéressantes lors de la soutenance.

Je tiens ensuite à exprimer toute ma gratitude à André, pour sa disponibilité, son sérieux et sa gentillesse, pour m'avoir fait profiter de sa longue expérience (en traitement d'images, en électronique, en informatique, en mécanique et en robotique !) et notamment pour m'avoir initié aux mystères des architectures de contrôle. Un grand merci également à David pour son humeur toujours égale, pour son soutien et pour m'avoir apporté son point de vue éclairé sur de nombreux aspects de la robotique et en particulier sur le SLAM, notamment sur les approches complémentaires à celles que j'ai utilisées. Leur aide régulière m'a été particulièrement précieuse pour les expérimentations (et pour les autres affaires DGA bien sûr) et leurs encouragements m'ont été droit au cœur : c'est toujours un plaisir de travailler avec des collègues aussi motivés. Notre équipe a été renforcée ces derniers mois par de nouveaux arrivants, Eric et Marc, qui se plongent avec enthousiasme dans les problématiques roboticiennes. Quelle serait l'ambiance quotidienne sans les discussions animées et fructueuses de notre petit groupe sur les multiples aspects de la robotique (et sur d'autres sujets, légèrement moins techniques parfois...) ?

Je remercie également les autres thésards en robotique de GIP (Philippe notamment) et nos stagiaires très motivés, en particulier Olivier Masle (pour les cartes combinatoires discrètes) et Nicolas Sinègre (pour l'appariement de scans), qui ont largement contribué au développement et à la mise en oeuvre de l'interface homme / machine sur notre robot, facilitant ainsi les acquisitions de données, les expérimentations et les évaluations.

Je tiens aussi à remercier David Cazier et le LSIIT de m'avoir fourni le code source sur le raffinement des cartes combinatoires 2D (monochromes et en nombres flottants !), Sylvain Soliman de m'avoir donné son avis sur les algorithmes d'optimisation pour la mise en correspondance, ainsi que toutes les personnes qui m'ont apporté des informations sur les modèles géographiques : M. J.-Y. Bréard de l'IGN, Mme Karine Zeitouni du laboratoire PRiSM de l'Université de Versailles-St Quentin en Yvelines, ainsi que Pascal, Odile, Gildas et Annabelle de GIP.

Plus généralement, j'adresse mes plus sincères remerciements à toutes les personnes de GIP que j'ai eu le plaisir de côtoyer durant ces premières années à la DGA, pour leur motivation, leur curiosité, leur bonne humeur et leur soutien. En particulier, je remercie chaleureusement notre encadrement qui a tou-

jours soutenu activement notre activité (Dominique encore, Frédéric, Jean-Paul et Véronique), Didi pour son aide sur les plates-formes, les traiteurs d'images (Bertrand, Emmanuel, Jérôme...), notre fusionneur de données, les optroniciens, certains traiteurs de parole, Françoise (et Philippe) pour la documentation, l'équipe de secrétaires et depuis quelques mois... l'ensemble du groupe SRO.

Je voudrais également remercier les autres personnes du CEP Arcueil et de la DGA, celles du club robotique et toutes les personnes que j'ai pu côtoyer dans le monde professionnel et avec lesquelles nous avons eu des échanges enrichissants sur la robotique.

Je remercie en outre tous les membres de l'équipe RIA au LAAS pour leur accueil sympathique et plus particulièrement Simon, Michel, Thomas et Il-Kyun pour nos multiples discussions passionnées sur le SLAM.

Enfin, c'est aussi l'occasion d'adresser mes plus affectueux remerciements à mes parents et à toute ma famille pour leur précieux soutien et leurs encouragements et ce depuis... une petite trentaine d'années maintenant. Merci également à tous mes amis qui ont gentiment patienté lorsque j'étais un peu accaparée par mon travail ces derniers temps... Merci à ma « belle-famille » pour ses messages d'encouragement et pour être venue en nombre, depuis l'autre bout de la France, afin d'assister à la soutenance. Et « last but not least », je suis infiniment reconnaissante à Bruno qui m'a supportée (dans tous les sens du terme...) avec patience, compréhension et tendresse, au quotidien, durant ces années intenses et bien remplies !

Table des matières

Table des figures

Chapitre 1

Introduction

« L'observateur est pris dans une sphère qui ne se brise jamais, où il y a des différences qui seront les mouvements et les objets, et dont la surface se conserve close, bien que toutes les portions s'en renouvellent et s'y déplacent. (...) Il perfectionne l'espace donné en se souvenant d'un précédent. Puis, à son gré, il arrange et défait ses impressions successives. Il peut apprécier d'étranges combinaisons : il regarde comme un être total et solide un groupe de fleurs ou d'hommes, une main, une joue qu'il isole, une tache de clarté sur un mur, une rencontre d'animaux mêlés par hasard. Il se met à vouloir se figurer des ensembles invisibles dont les parties lui sont données. (...) Parfois, les traces de ce qu'il a imaginé se laissent voir sur les sables, sur les eaux ; parfois sa rétine elle-même peut comparer, dans le temps, à quelque objet la forme de son déplacement. »
P. Valéry (« Introduction à la méthode de Léonard de Vinci »).

1.1 La localisation : une capacité essentielle pour l'autonomie décisionnelle des robots

« Où suis-je ? » : ce n'est sans doute pas un hasard si cette question est devenue emblématique de la personne qui revient brusquement à la réalité. Pour reprendre pied dans le monde réel, pour décider des prochaines actions à accomplir, nous avons besoin d'informations sur ce qui nous entoure et en particulier, nous ressentons le besoin de savoir où nous nous trouvons. De même, comme nous allons le voir, pour accomplir seul des tâches de complexité variable, pour planifier ses mouvements et ses actes, un robot a besoin de se situer (même de manière locale ou qualitative) dans son environnement.

Les robots constituent aujourd'hui un formidable potentiel technologique : certains chefs d'entreprise visionnaires parlent même d'une prochaine révolution électronique, qui verrait les robots mobiles déferler dans notre quotidien. Ils sont déjà arrivés dans les foyers via les applications de divertissement (chiens électroniques, jouets humanoïdes, kits de construction de robots...) ou pour soulager les hommes de diverses tâches domestiques (robots aspirateurs, tondeuses à gazon...). Ils commencent à investir les espaces publics et les lieux de travail (robots de nettoyage, robots de surveillance, robots guides de musée...). Ils permettent d'atteindre des lieux inaccessibles pour l'homme, dans tous les milieux : les zones confinées

(inspection de conduits par exemple), les constructions élevées (ouvrages d'art comme les ponts), les fonds sous-marins, les autres planètes (avec les récents succès des robots martiens notamment) ou les environnements dangereux (centrales nucléaires...). En matière de sécurité et de défense [128], ils sont déjà employés pour les activités de déminage et d'inspection de colis piégés, commencent à être utilisés pour des applications de reconnaissance en milieu hostile ou d'inspection sous les véhicules et sont envisagés pour des missions toujours plus variées : assistance aux fantassins en zone urbaine, surveillance, ravitaillement, évacuation de blessés, relais de communication, etc.

A l'heure actuelle, la plupart de ces machines présentent une autonomie décisionnelle limitée, mais celle-ci est amenée à se développer dans les années à venir, afin de soulager le travail du téléopérateur et lui permettre de se focaliser sur des tâches plus difficiles à automatiser. Or pour accéder à un minimum d'autonomie, l'information de localisation est primordiale pour le robot mobile puisqu'elle lui permet de naviguer en sélectionnant les déplacements les plus efficaces et les plus sûrs afin de réaliser les actions qui lui sont confiées. Ainsi, un robot domestique doit pouvoir se positionner par rapport à sa station de recharge pour être en mesure de la rejoindre pour recharger ses batteries, un robot guide de musée doit être capable de se situer sur le circuit de visite afin de délivrer les bonnes indications aux visiteurs et de se rendre au prochain point d'intérêt, un robot de surveillance doit être capable de se localiser par rapport à la zone d'observation afin de trouver les bons angles de vue, etc.

1.2 Intérêt de la cartographie

Pour se localiser, un système robotisé dispose de capteurs dits « proprioceptifs » (qui mesurent des données internes au robot), capables de fournir des informations de position relatives par rapport au point de départ : odomètres, gyromètres, centrales inertielles, etc. Cependant, ces capteurs ont tendance à dériver en accumulant les erreurs au fil du temps : cette dérive peut vite devenir prohibitive pour les centrales peu onéreuses et peu encombrantes susceptibles d'être embarquées sur de petites plates-formes. On peut également équiper les robots de boussoles (compas), qui permettent de se recaler de manière absolue. De tels dispositifs sont toutefois très sensibles aux perturbations magnétiques, en particulier à l'intérieur des bâtiments qui contiennent de nombreuses structures métalliques et des câbles électriques.

Le GPS (« Global Positioning System »), qui repose sur le déploiement de satellites jouant le rôle de balises artificielles émettrices, est également destiné à fournir des informations de position absolues, mais il pose parfois des problèmes en zone urbaine (masquages ou existence de chemins multiples) et le signal passe mal à l'intérieur des bâtiments.

Ainsi, les concepteurs de robots ont été amenés à proposer d'autres approches pour la localisation, basées sur l'emploi de capteurs dits « extéroceptifs » (destinés à percevoir le monde extérieur au robot) : caméras monoculaires, systèmes stéréoscopiques, proximètres IR, télémètres laser ou à ultrasons,... En général, ces techniques alternatives s'appuient sur des modèles d'environnement, que le robot compare à ses perceptions courantes : il s'agit d'exploiter des plans de bâtiments, des cartes d'obstacles ou des reconstructions 3D par exemple.

Certains courants de robotique, comme les approches réactives introduites par Brooks [17], ont toutefois remis en cause la nécessité de recourir à de telles représentations : une connaissance locale et donc limitée de l'environnement peut suffire pour accomplir certaines tâches, telles que l'évitement d'obstacle ou certains comportements de groupe. Il existe également des méthodes de localisation comportementales, qui exploitent des règles de navigation simples : tourner à droite à chaque intersection par exemple [32], ce qui permet de revenir au point de départ en inversant la procédure. Des systèmes plus sophis-

tiqués, développés par Arkin notamment [3], apprennent quant à eux des structures internes qui peuvent être rejouées. Ces systèmes présentent toutefois des capacités de localisation limitées, directement conditionnées par leurs déplacements passés : les positions reconnues correspondent uniquement aux positions antérieures du système.

En fait, comme le souligne Lee [114], tout robot utilise au moins des suppositions (explicites ou implicites) sur l'environnement : scènes statiques, sol plan, obstacles de forme polygonale, etc. Les approches réactives conviennent bien aux environnements dynamiques et changeants pour lesquels les données de planification sont rapidement invalidées. Quant aux approches comportementales, elles peuvent être envisagées dans des cas où le robot est libre de ses mouvements (à la différence des robots téléopérés) et où les mouvements n'ont pas besoin d'être optimisés (elles paraissent peu adaptés à une recherche de plus court chemin par exemple). Il existe cependant des tâches qui exigent impérativement un modèle d'environnement : la surveillance d'une zone définie sur un plan ou la distribution de courrier dans une pièce spécifique par exemple. Plus généralement, une carte d'environnement confère en principe au robot une plus grande flexibilité (l'homme peut donner des ordres au robot grâce à la carte et de même, la carte fournit un retour d'information vers le téléopérateur) et une meilleure robustesse (l'efficacité du système est assurée dans un plus grand nombre de situations grâce à l'interprétation et à la validation des perceptions du robot par exemple).

On peut alors songer à donner au robot une carte a priori de son environnement. Une telle option serait néanmoins une source de problèmes ou de limitations : il est parfois coûteux et laborieux de construire des cartes manuellement, celles-ci ne sont pas toujours disponibles sur la zone d'évolution envisagée pour le robot (dans le cas de missions de reconnaissance notamment) et elles risquent d'être rapidement périmées (il suffit de déplacer un meuble en milieu intérieur par exemple). Alternativement, si un robot est capable de cartographier lui-même, en ligne, son environnement, cela offre des avantages considérables : possibilité de mises à jour fréquentes, adaptation de la représentation employée aux algorithmes du robot, modélisation de zones difficiles à atteindre pour l'homme, voire dangereuses ou hostiles, etc. C'est pourquoi cette fonctionnalité a fait l'objet de nombreuses recherches au cours des vingts dernières années.

1.3 Problématique générale du SLAM

Si la fonction de localisation bénéficie grandement des capacités de cartographie, la modélisation de l'environnement nécessite à l'inverse une estimation de la position du robot. En effet, pour savoir où positionner dans la carte les éléments observés par le robot (murs, frontières d'obstacles,...), il faut être capable de situer le système robotisé dans la carte en cours de construction. Or en général, le champ de perception du robot n'embrasse pas l'ensemble de l'environnement, du fait des occultations et des limites de portée des capteurs : le robot doit donc se déplacer pour réaliser une cartographie plus exhaustive et sa position doit être régulièrement recalculée. Il faut donc gérer le bouclage entre localisation et modélisation de l'environnement. Ce problème de l'œuf et de la poule est bien connu des roboticiens, qui le désignent sous le nom de « localisation et cartographie simultanée » ou SLAM en anglais (« Simultaneous Localization And Mapping »), voire plus rarement CML (« Concurrent Mapping and Localization »). Dans la suite de ce mémoire, c'est le terme de SLAM que nous utiliserons principalement.

Les modalités sensorielles employées pour la cartographie sont variées : la télémétrie (lasers, sonars, radar, proximètres IR), la vision (monoculaire ou stéréoscopique, et parfois panoramique), voire le toucher (utilisation de pare-chocs tactiles). Les mesures issues de ces capteurs extéroceptifs sont souvent combinées à des données proprioceptives (odométriques ou inertielles) voire goniométriques (boussoles) ou à des données GPS. Plus récemment, d'autres capteurs ont été utilisés dans le cadre d'applications

plus spécifiques : par exemple des dispositifs susceptibles de détecter la présence de gaz en vue d'en cartographier la concentration. Quant aux types d'environnements concernés, ils sont également très divers : milieu urbain (intérieur et abords des bâtiments), milieu naturel moins structuré, milieu sous-marin, cartographie de la terre vue du ciel, etc.

La problématique du SLAM implique en outre la gestion de multiples sous-problèmes [180]. Il faut d'abord gérer les erreurs de perception et de localisation (dérive des capteurs proprioceptifs, imprécision des mesures extéroceptives, présence de bruit, erreurs d'interprétation, etc.). A partir de ces mesures sensorielles imparfaites, il s'agit ensuite de sélectionner la meilleure carte dans un espace de recherche de très grande dimension. Il faut aussi pouvoir mettre en correspondance les observations locales entachées d'erreur avec les éléments de la carte globale en cours de construction. Dans certains cas, il faut également ajouter le traitement des environnements dynamiques et pour les robots totalement autonomes, des capacités d'exploration. Ainsi, selon certains chercheurs comme Thrun [180], la cartographie automatique peut être considérée comme le problème perceptuel le plus difficile en robotique mobile : les progrès dans ce domaine devraient avoir des retombées sur de nombreux autres aspects de la robotique tels que les interactions homme(s) / robot(s) ou la coordination multirobot par exemple.

1.4 La question des représentations

Pour traiter le problème de la localisation et de la cartographie simultanée, les roboticiens ont fait appel à des représentations variées de l'environnement : superpositions de scans laser bruts, grilles d'occupation surfaciques, cartes basées sur des primitives géométriques, etc. Comme nous le verrons plus en détail au chapitre 2, ces représentations s'avèrent souvent complémentaires : c'est le cas notamment des approches métriques et topologiques. En particulier, leur utilisation conjointe est susceptible de favoriser l'exploitation de la carte résultante pour les besoins de navigation du robot. C'est pourquoi les approches hybrides, qui combinent différents types de modèles élémentaires, se généralisent (cf. chapitre 3).

La plupart de ces représentations hybrides demeurent toutefois peu structurées, ce qui peut poser des problèmes de cohérence durant la construction de carte. En outre, comme nous le préciserons au chapitre 4, ce manque de structure risque de limiter les possibilités de raisonnements spatiaux de plus haut niveau. C'est pourquoi nous nous intéresserons en priorité à des représentations bien structurées, semblables à celles qui sont aujourd'hui employées dans les systèmes d'information géographiques (SIG), et plus précisément aux modèles incluant une « couche topologique » en plus des informations purement géométriques.

1.5 Objectifs de la thèse

Ainsi, dans le cadre de cette thèse, notre objectif est de développer un algorithme de cartographie automatique qui permette d'obtenir un modèle aussi précis et aussi structuré que possible. Cette carte est destinée au robot, pour ses propres besoins de navigation, mais elle est aussi destinée à l'homme, dans un contexte de reconnaissance en zone urbaine notamment [128] : elle doit donc être facile à interpréter par un opérateur et nous veillerons à réduire les ambiguïtés (distinction entre espace vide, plein ou inconnu, existence ou non d'une ouverture vers un espace ouvert,...) ou les éventuelles incohérences (intersection erronée de frontières d'obstacles, observation d'un objet à l'intérieur d'une zone occupée par un obstacle, etc.).

Comme nous le détaillerons au chapitre 4, les modèles très structurés tels que ceux employés dans les systèmes d'information géographiques impliquent notamment la gestion de liens d'adjacence entre tous les éléments constitutifs de la carte : sommets, arêtes et faces. En conséquence, la construction en ligne d'une représentation de ce type suppose une adaptation de toutes les étapes des algorithmes de cartographie habituels afin de prendre en compte ces liens d'adjacence. Il s'agit dans un premier temps de considérer des chaînes polygonales plutôt que des primitives géométriques individuelles (points, segments...) lors de la mise en correspondance entre carte globale et données d'observation locales. Il importe ensuite de mettre en place un processus d'estimation de position des éléments de la carte (à partir des observations successives) qui préserve ces liens d'adjacence tout en tenant compte des autres contraintes qui régissent ce processus d'estimation (corrélations entre erreurs d'estimation notamment). Il faut également gérer le lien avec l'étape précédente, à travers l'extraction des équations de mise à jour géométrique à partir de l'association de données entre carte globale et observations locales. Enfin, il s'agit de mettre à jour la carte globale en corrigeant concrètement la position des primitives selon les résultats du processus d'estimation précédent et en y ajoutant les nouvelles observations locales, tout en assurant la cohérence de la couche topologique.

Notre but est donc de développer une chaîne algorithmique complète qui permette de construire une carte aussi précise et exacte que possible, tout en maintenant la cohérence de notre modèle très structuré.

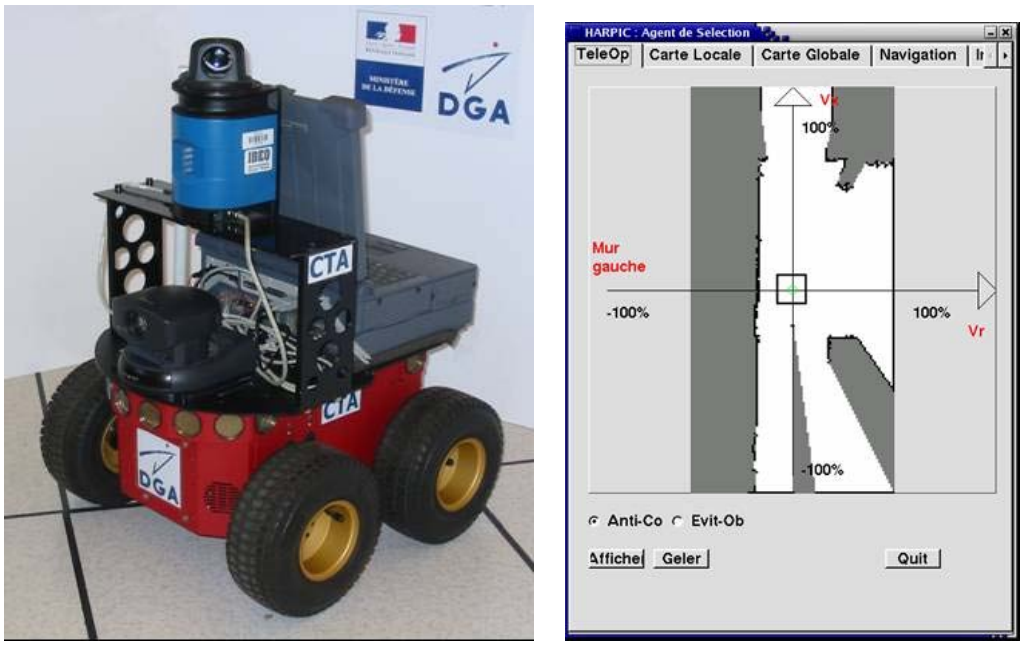
1.6 Hypothèses et contraintes

1.6.1 Hypothèses sur la plate-forme et sur ses équipements

Dans le cadre de cette thse, nous considérons un robot terrestre à roues, équipé d'un télémètre laser à balayage et d'un odomètre (constitué d'encodeurs destinés à compter le nombre de tours de roues depuis le point de départ de la plate-forme, afin de reconstituer approximativement sa trajectoire). En faisant tourner un faisceau laser dans un plan horizontal, le télémètre mesure à intervalle angulaire régulier (de l'ordre de 0.5° ou 1° en général) la distance à l'obstacle le plus proche. A chaque acquisition, il fournit un « scan », c'est-à-dire une coupe horizontale locale de l'environnement constituée des points de mesure bruts sur un certain champ angulaire (généralement compris entre 180° et 360°). La figure 1.1 (b) montre un tel exemple de coupe horizontale locale.

En pratique, la plate-forme que nous utilisons au Centre d'Expertise Parisien (CEP Arcueil - anciennement Centre Technique d'Arcueil ou CTA) de la Délégation Générale pour l'Armement (DGA) est un Pioneer 2AT de la société ActivMedia, équipé d'une ceinture de proximètres à ultrasons, d'une caméra couleur orientable et d'un télémètre laser à balayage (cf. Fig. 1.1 (a)). Cette plate-forme est destinée à expérimenter, à évaluer et à comparer des algorithmes de perception (tels que le SLAM) ainsi que des comportements sensorimoteurs (évitement d'obstacles, suivi d'amers, suivi de personnes, etc.) utilisables en zone urbaine, en particulier pour des missions de reconnaissance à l'intérieur des bâtiments [37].

Pour cela, notre système dispose d'une architecture de contrôle modulaire, basée sur le paradigme multiagents [36]. Cette architecture a été initialement conçue dans l'optique d'une autonomie décisionnelle complète pour le robot. Plus récemment, nous avons montré que moyennant quelques adaptations, elle pouvait également permettre l'intervention d'un téléopérateur, fournissant ainsi une base intéressante pour le test d'un concept d'autonomie ajustable [37] : différents modes de contrôle du robot, allant de la simple téléopération jusqu'au lancement de comportements autonomes sont disponibles pour le téléopérateur, lui permettant ainsi d'ajuster sa charge de travail tout au long de la mission, suivant sa propre situation et suivant celle du robot (les images de cartes du chapitre 2 et de la figure 5.20 du



(a) (b)

FIG. *1.1: (a) La plate-forme robotisée du CEP Arcueil, de type Pioneer 2AT. (b) Une coupe horizontale locale de l'environnement, vue à travers l'interface de téléopération du robot. L'espace libre apparaît en blanc, l'espace inconnu en gris, et les obstacles en noir. La position du robot est matérialisée par un cercle vert.*

chapitre 5 sont présentées à travers l'interface du robot, tout comme la carte locale de la figure 1.1 (b)).

L'algorithme de localisation et de cartographie simultanées développé dans le cadre de cette thèse est destiné à remplacer l'algorithme actuel, qui construit une simple représentation basée sur l'apparence (superposition de scans laser bruts), cette représentation étant transformée en grille d'occupation (représentation surfacique) pour les besoins de planification de trajectoires du système. Cet algorithme actuel est fondé sur un appariement de scans similaire à celui que nous utilisons (cf. chapitre 5), la position du robot étant simplement estimée par filtrage de Kalman. Il a fourni des résultats intéressants (cf. Fig. 5.20 au chapitre 5) mais il présente encore quelques problèmes de robustesse (en particulier lors des virages en dérapage du robot et lors du bouclage des cycles). En outre, comme nous l'expliquerons au chapitre 4, nous pensons qu'une plus grande structuration de la représentation permettrait une meilleure lisibilité de la carte et donc une meilleure interaction avec l'homme, ainsi que la mise en œuvre de raisonnements spatiaux de plus haut niveau pour les modes d'autonomie élevés du robot.

1.6.2 Hypothèses sur l'environnement et sur le contexte

Nous supposons en outre que le robot évolue dans un environnement intérieur relativement bien structuré (essentiellement composé de pièces, de couloirs et de quelques meubles). Les obstacles sont constitués de parois verticales, ce qui permet de représenter l'information en deux dimensions, vue d'en haut. Nous nous attachons toutefois à rester aussi génériques que possible : nous évitons l'hypothèse d'environnement isothétique (obstacles parallèles aux deux directions du repère orthogonal) et nous cherchons à assouplir l'hypothèse de polygonalité en autorisant la modélisation de surfaces courbes par des lignes polygonales (en gérant la non unicité d'une telle polygonalisation). Le sol est considéré comme plan et l'environ-

nement statique. Dans la mesure du possible, nous veillons toutefois à garder une certaine ouverture vers les environnements dynamiques et vers une modélisation en trois dimensions. En outre, nous nous interdisons de préparer ou de modifier l'environnement pour les besoins du robot, via l'ajout de balises par exemple, que cette transformation soit réalisée par un humain ou par le robot lui-même : dans un contexte de reconnaissance en zone urbaine, une telle préparation serait difficile à envisager et nuirait à la discrétion du système.

Enfin, il importe que la plate-forme robotisée puisse se déplacer où bon lui semble dans l'environnement : nous souhaitons éviter que le robot ait besoin d'exécuter des manœuvres spécifiques pour se localiser (retour à une position antérieure, parcours systématique des contours d'obstacles...) ou qu'il se cantonne à des chemins particuliers (diagramme de Voronoï par exemple). La cartographie doit notamment pouvoir se dérouler en parallèle ou en toile de fond par rapport à d'autres tâches du système, en particulier dans le cas où le robot est téléopéré par un homme : Engelson désigne ce mode de fonctionnement par l'expression « cartographie passive » [65].

1.7 Plan du mémoire

Dans ce mémoire, nous commençons par réaliser un état de l'art des représentations existantes et des méthodes de construction de cartes associées. Le chapitre 2 se focalise ainsi sur les modèles élémentaires en proposant un certain nombre de critères de comparaison, tandis que le chapitre 3 détaille les représentations hybrides qui combinent différents types de représentations élémentaires. Concernant ce dernier point, nous considérons à la fois le domaine de la robotique, mais aussi celui de la géographie avec les formats de cartes employés dans les systèmes d'information géographiques. Nous nous efforçons de tirer de cette analyse un certain nombre d'enseignements qui nous guident dans la suite de nos travaux. En particulier, nous étudions les complémentarités qui existent entre divers types de représentations et examinons l'intérêt d'une plus grande structuration des modèles.

Ce tour d'horizon nous permet alors de choisir un modèle particulier pour la construction de cartes. Ainsi, au chapitre 4, nous précisons les motivations qui nous poussent à proposer une représentation originale, riche et très structurée de l'environnement, basée sur un outil algébrique appelé « carte combinatoire ». Cet outil combine intrinsèquement topologie et géométrie et gère notamment les liens d'adjacence entre les divers éléments de la carte. Pour les besoins de cartographie automatique, nous y ajoutons une gestion des incertitudes et pour en garantir un usage robuste, nous en proposons une version discrétisée. Nous discutons également de la position de cette représentation par rapport à l'existant et fournissons quelques pistes pour la construction en ligne de ces cartes par le robot mobile. Nous introduisons alors les grandes étapes de notre algorithme de cartographie :

- mise en correspondance entre observations locales et carte globale ;
- correction géométrique de la carte globale via un filtre de Kalman étendu (EKF) ;
- mise à jour géométrique et topologique du modèle.

Les trois chapitres suivants sont consacrés à la description de ces différentes étapes. Ils sont en outre accompagnés de résultats expérimentaux qui illustrent les mécanismes mis en œuvre. Ainsi, au chapitre 5, nous expliquons le processus de mise en forme des données locales nouvellement observées par le robot (polygonalisation, filtrage) puis nous nous focalisons sur la mise en correspondance entre ces données (qui se présentent sous la forme d'une carte locale) et la carte en cours de construction (ou carte globale). Nous aborderons ainsi la problématique d'appariement de chaînes polygonales, qui nécessite la gestion des appariements multiples (tantôt autorisés, tantôt interdits), et qui est rarement considérée dans le cadre du SLAM.

Au chapitre 6, nous indiquons les adaptations apportées à la technique classique de cartographie par filtrage de Kalman, afin que cette méthode soit applicable à notre représentation. En particulier, nous expliquons la nécessité de traiter les « cassures » qui se produisent dans les arêtes et proposons d'introduire pour cela des points virtuels associés à chacun des segments.

Nous décrivons ensuite au chapitre 7 l'algorithme de mise à jour géométrique, déduit de ce filtrage, qui est appliqué à la carte globale existante : cet algorithme doit maintenir une structure de carte combinatoire cohérente. Nous détaillons également l'étape de mise à jour topologique qui permet véritablement de fusionner la carte locale avec la carte globale tout en mettant à jour des informations d'ordre topologique. Pour cela, nous définissons le concept de cartes combinatoires colorées et développons un certain nombre d'opérations permettant de les manipuler (raffinement coloré et complétion de labels associée, déplacement de sommets, ajout de sommets...). En particulier, ce cadre de travail permet la gestion des incohérences transitoires telles que des retournements de cellules ou des croisements.

Enfin, le chapitre 8 dresse un bilan du travail réalisé et propose plusieurs perspectives à horizons variés, qui sont détaillées dans l'annexe 1. L'annexe 2 fournit quant à elle quelques pistes en vue d'une évaluation plus systématique de la qualité des cartes.

Chapitre 2

Modèles élémentaires et techniques de cartographie associées

> *« Les formes naturelles (...) s'offrent à nous avec des caractères de forme réductibles à un petit nombre d'éléments généraux (...) [qui] constituent ce qu'on peut appeler l'alphabet des formes. Ces éléments généraux, qui écrivent les formes, comme les lettres écrivent les mots, rendent un compte précis de l'infinie variété (...) de la nature et de l'art. »*
> J. Bourgoin (« Grammaire élémentaire de l'ornement pour servir à l'histoire, à la théorie et à la pratique des arts et à l'enseignement »).

Le problème de la localisation et de la cartographie simultanées a commencé à être étudié en tant que tel dans les années 1980, avec quelques publications préliminaires à la fin des années 1970. L'augmentation des ressources de calcul et des capacités de perception des robots ont contribué à faire progresser les travaux du domaine tout au long des années 1990, pour laisser place à un véritable engouement depuis environ cinq ans, avec des publications toujours plus nombreuses, des sessions spéciales dans les conférences internationales, des écoles d'été, des applications variées et multimilieux (terrestres, aériennes et sous-marines) et même quelques produits commerciaux (plates-formes robotisées et simulateurs intégrant des algorithmes de cartographie, aspirateurs autonomes, logiciel de SLAM visuel, etc.).

Dans le cadre de cet état de l'art, nous nous sommes essentiellement focalisés sur les formats de cartes employés, en indiquant pour chacun de ces formats les techniques de construction disponibles. Comme nous allons le voir, les modèles d'environnement proposés dépendent à la fois des capteurs disponibles et du type d'environnement considéré. Par exemple, les grilles d'occupation se prêtent souvent mieux à des capteurs peu précis que les représentations fondées sur des primitives géométriques. De même, les représentations polygonales sont a priori mieux adaptées aux environnements intérieurs bien structurés qu'au milieu naturel. Comme nous l'avons indiqué dans le chapitre d'introduction, nous nous intéressons essentiellement aux représentations en deux dimensions d'environnements intérieurs structurés et statiques, construites par un robot unique : si ce thème a été longuement étudié, il n'est toujours pas considéré comme entièrement résolu à l'heure actuelle (en particulier dans le cas où l'environnement à cartographier est de grande dimension [180] [22], et justement à cause de la question des représentations, qui paraît déterminante dans ce cas). C'est pourquoi cet état de l'art est essentiellement axé sur les travaux qui se rapportent à ce type de milieu : nous ne détaillerons pas les approches destinées à modéliser

des environnements non structurés (caractéristiques statistiques locales [167], modélisations par sommes de gaussiennes [58], modèles numériques de terrain [96]...) ni les modèles 3D d'environnements urbains exploitant des représentations volumiques [138] ou des primitives 3D (plans, segments et points 3D [139] [179] [40]...), sauf si ces modèles ont un intérêt dans le type d'environnement qui nous intéresse. De même, nous ne détaillerons pas non plus les stratégies de cartographie en environnement dynamique (cf. [84] ou [158] par exemple), ni les stratégies de cartographie multirobot, sauf si celles-ci apportent de nouveaux éléments dans un cadre monorobot.

Afin de pouvoir comparer les différents modèles de cartes, nous nous sommes fixé une grille de lecture proposant un certain nombre de critères de comparaison (en partie inspirés de [114], [182] ou [194]). Certains de ces critères sont intrinsèques au modèle considéré (compacité par exemple), mais beaucoup sont en fait liés aux méthodes de construction disponibles sur le modèle :

- **compacité** : taille mémoire nécessaire au stockage de la représentation ;

- **généricité par rapport au type d'environnement**. Ce critère regroupe différentes notions : adaptation automatique à l'échelle des objets de l'environnement (par exemple, les grilles d'occupation se révèlent peu efficaces lorsque l'environnement comprend à la fois des objets de grande et de petite taille puisqu'elles nécessitent un découpage en petites cellules peu adapté aux grands objets), adaptation automatique à la densité et à la nature des obstacles (par exemple, les représentations polygonales sont mal adaptées aux courbes et peu efficaces par rapport aux représentations de type balises lorsque l'environnement présente de nombreux objets ponctuels tels que des pieds de meubles) ;

- **généricité par rapport aux capteurs** : il s'agit de vérifier que le modèle est adapté à l'emploi de tous les types de capteurs classiques en robotique (capteurs précis ou non, denses tels que la vision ou moins denses tels que les ceintures de télémètres à ultrasons, etc.), notamment en vue de réaliser une fusion de données multisensorielles ;

- **adaptation à la représentation de grands environnements** : au-delà du critère de compacité, il s'agit de vérifier si le modèle et les techniques de constructions associées restent efficaces lorsque le volume de données augmente et si elles permettent d'assurer la cohérence de la carte. Ce dernier point concerne aussi la représentation de l'incertitude (à la fois pour les éléments de la carte et pour la position du robot) : gestion des ambiguïtés de positionnement du robot, gestion des bouclages de cycles dans l'environnement avec rétropropagation des corrections d'erreurs de position le long du cycle (par exemple, les méthodes de construction de grilles d'occupations ne permettent généralement pas de bien prendre en compte des corrélations entre cellules, ce qui peut engendrer des problèmes de cohérence de la carte lors d'une fermeture de boucle) et convergence du processus (par exemple, des problèmes de linéarisation peuvent apparaître lors de l'utilisation du filtre de Kalman étendu, ce qui peut mener à une divergence du filtre) ;

- **souplesse des modes de contruction existant**. Ce critère couvre deux aspects : possibilité de construction de la carte en temps réel (ou bien nécessité d'un apprentissage de la carte préalablement à la navigation du robot, ou encore construction hors ligne en raison de temps de calcul prohibitifs) et possibilité de construction « passive », sans induire de trajectoire particulière pour le robot (en particulier, compatibilité du processus de construction de cartes avec la téléopération du robot selon une trajectoire quelconque) ;

– **exploitation par le robot** : il s'agit de vérifier si le robot peut faire une exploitation efficace de la carte pour naviguer, exécuter les tâches qui lui sont allouées et planifier ses actions. Ce critère concerne donc notamment l'adaptation de la représentation aux algorithmes de navigation et de planification classiquement employés par les robots, aux algorithmes de recherche opérationnelle et aux raisonnements spatiaux de plus ou moins haut niveau ;

– **exploitation par un humain** en ligne (dans le cadre d'interactions homme / machine) ou a posteriori : il s'agit de vérifier si la représentation est facile à appréhender par un humain (par exemple, certaines représentations topologiques se révèlent assez hermétiques pour l'homme, même si elles sont bien adaptées aux algorithmes et aux moyens de perception du robot) et de s'assurer que l'on peut attacher des informations sémantiques sur l'espace (sur des objets ou sur des régions) permettant une meilleure interaction entre l'homme et le robot.

Dans un premier temps, nous décrivons un certain nombre de modèles « élémentaires » (catégories couramment citées dans la littérature). Dans le chapitre suivant, nous nous attachons à classer les modèles hybrides issus de combinaisons de ces représentations élémentaires.

2.1 Lieux reconnaissables

Comme le souligne Lee [114], la reconnaissance de lieux ou de points de référence constitue une aptitude fondamentale pour la navigation. Mataric [132] rappelle en effet que chez l'humain ou chez l'animal, les représentations internes de l'environnement (aussi appelées « cartes cognitives ») modélisent l'espace comme un ensemble de balises servant de points de référence. Les travaux du domaine des sciences cognitives montrent à quel point ces balises ou lieux de référence jouent un rôle central dans les représentations spatiales, en tant que briques élémentaires définies à partir de stimuli essentiellement visuels mais aussi sonores, olfactifs ou auditifs. Piaget [155] précise en outre que les enfants représentent d'abord l'espace comme des lieux séparés, avant d'ajouter des informations de distances entre lieux. Ainsi, certains travaux de robotique ont d'abord cherché à définir des lieux ou des balises caractéristiques, dont les spécificités pourraient suffire à elles seules à localiser le robot sans ambiguïté.

Chez Nehmzow et Smithers [144] par exemple, les lieux sont définis et reconnus exclusivement selon les mouvements réalisés par le robot. Le robot est doté d'un comportement de suivi de mur et cherche à détecter des virages significatifs correspondant à des coins de la pièce. Dans ce contexte, les capteurs sont utilisés pour maintenir le robot à distance constante du mur mais ils n'interviennent pas directement dans le processus de localisation. Le système semble fonctionner correctement dans certaines configurations puisqu'il apprend les coins lors des circuits successifs autour de la pièce, jusqu'à être capable de les reconnaître individuellement à la fin de l'expérimentation. Toutefois, il éprouve des difficultés dès que l'environnement présente des régions de forme similaire.

Dans les travaux de Donnet [47], lors d'une phase préparatoire, le robot est placé de façon systématique à un certain nombre de positions dans l'environnement. A chaque position, le robot enregistre les informations (intensité, direction, distance) relatives à diverses balises sonores, ultrasonores et infrarouges. Ensuite, lors de la phase de navigation proprement dite, le robot parvient à se localiser par mise en correspondance entre ses observations et les informations stockées sur les balises, suivant une technique de classification bayésienne.

Dans le domaine de la vision, Yeh et Kriegman [204] proposent eux aussi une approche bayésienne pour sélectionner automatiquement parmi un ensemble de primitives géométriques 3D (points ou segments indistingables - en pratique, les lignes verticales de l'image) le sous-ensemble qui a le plus de chances d'être reconnu dans une image unique (et le moins de chances d'être confondu avec un autre sous-

ensemble de primitives). Toutefois, il était prévu dès le début une extension vers l'utilisation de graphes (graphe d'aspects ou représentation du type de celle proposée par Taylor et Kriegman [177], appelée « boundary place graph », qui garde en mémoire l'ensemble des balises visibles depuis les frontières de chaque obstacle) pour permettre au robot de naviguer de façon plus robuste dans un cadre réaliste.

Plus récemment, Gechter et Charpillet [77] ont expérimenté une méthode de localisation fondée uniquement sur l'exploitation d'une base de données d'images de l'environnement construite a priori. Cette base de données peut être vue comme une carte basique, élaborée en amont de la phase de navigation du robot. La reconnaissance de lieu est réalisée par comparaison des observations du robot avec les images de la base de données initiale, en calculant une probabilité de « proximité » (similitude entre images). L'algorithme choisit l'image de la base qui ressemble le plus aux perceptions courantes : ce calcul est basé sur une classification préalable des données par analyse en composantes principales (ACP) comme chez Sim et Dudek [171], de manière à limiter les calculs lors de la phase de navigation. Cette méthode présente cependant des limitations et conduit à des ambiguïtés de positionnement. Elle a donc été améliorée par combinaison avec une technique à base de processus décisionnels de Markov partiellement observables (PDMPO) qui tient compte des déplacements du robot et le modèle de cartes devient plus sophistiqué : l'espace est alors décomposé en états correspondant chacun à une position et à une orientation, de manière similaire aux travaux de Simmons et Koenig [173] que nous verrons plus loin.

Ainsi, ces différentes représentations très simples, constituées exclusivement d'une liste de description de lieux présentent d'importantes limitations. Concernant la localisation, la mise en correspondance des observations du robot avec ces descriptions de lieux pré-enregistrées peut générer des ambiguïtés de positionnement difficiles à lever en l'absence de tout lien entre lieux (adjacence, proximité, ordre de succession...). L'adjonction des relations d'adjacence permettrait notamment de limiter le nombre d'images candidates à tester [187]. La connaissance de l'ordre de succession de ces lieux pour une trajectoire donnée peut également faciliter leur identification le long de cette trajectoire (cf. les travaux de Pradalier et al. par exemple sur le suivi de trajectoires sensorimotrices [157]). On est toutefois contraint, dans ce cas, à évoluer sur la trajectoire prédéfinie : il n'est pas prévu d'utiliser des raccourcis par exemple. Enfin, comme l'indique Lee [114], les simples représentations à base de lieux sont adaptées à la navigation des robots lorsque tous les lieux recherchés (bornes de recharge par exemple) sont visibles en permanence. En revanche, si le point de ralliement recherché n'est pas visible, une certaine planification s'avère nécessaire, mais celle-ci impose de connaître l'existence de chemins entre lieux. On s'oriente dès lors vers des représentations dites « topologiques ».

2.2 Cartes topologiques

2.2.1 Description

Selon Thrun [178], les cartes topologiques auraient vu le jour à la fin des années 70, avec les travaux de Kuipers qui modélise le monde comme un graphe de lieux et utilise les arcs pour représenter les mouvements possibles entre lieux [105]. Plus généralement, les cartes topologiques peuvent être vues comme des représentations abstraites qui décrivent les relations entre éléments de l'environnement, sans utiliser de repère de référence absolu [68]. Elles se présentent sous la forme de graphes, dont les sommets correspondent à des lieux, souvent associés à des informations perceptuelles (histogrammes de couleurs, images, données télémétriques...), et dont les arêtes indiquent qu'il existe un chemin traversable par le robot, reliant les deux extrémités de l'arête (cf. Fig. 2.1). Il n'existe pas de sémantique unique sur ces représentations et la signification des sommets et arêtes diffère beaucoup selon les approches [68]. Plus précisément, les principales différences entre cartes se situent ainsi dans :

- la nature des sommets, leur densité et la manière dont ces sommets sont ajoutés dans la carte ;
- la nature des arêtes ;

– le degré d'information métrique ajoutée à la représentation purement topologique, comme nous le verrons dans la section consacrée aux représentations hybrides.

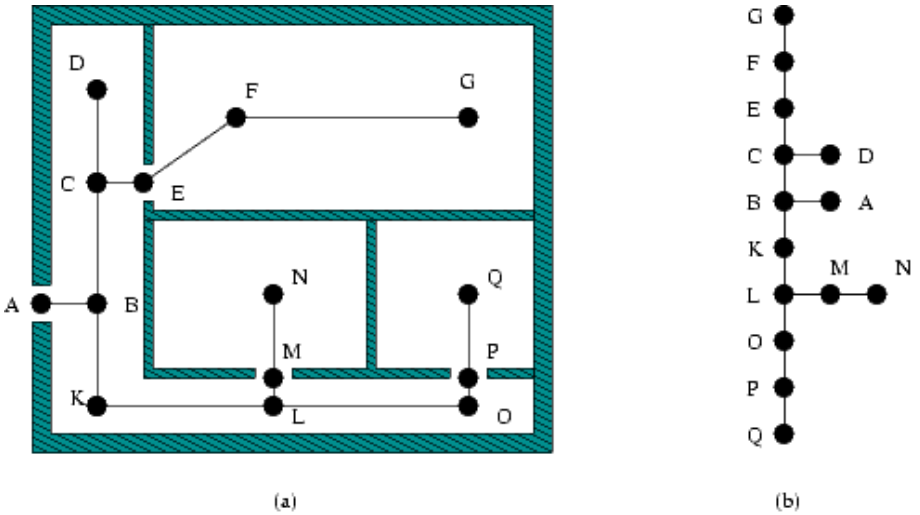


FIG. *2.1: Un exemple de carte topologique. a) Représentation métrique de l'environnement associée à des lieux topologiques reliés entre eux par des relations d'adjacence. b) Représentation purement topologique correspondante.*

Les sommets des cartes topologiques sont souvent déduits des informations perceptuelles, mais certains travaux, inspirés par la neurobiologie notamment, les définissent en lien avec les mouvements du robots. Par exemple, chez Mataric [132], ils correspondent à des combinaisons de mouvements du robot et de perception par une ceinture de télémètres à ultrasons. Ainsi, un couloir est défini comme une combinaison de déplacements en ligne droite et de mesures télémétriques peu profondes dans les directions latérales au robot. En outre, ces sommets sont augmentés de données métriques brutes fournies par l'odométrie et de directions indiquées par une boussole (par exemple : « mur à gauche, avec un déplacement de 10 m vers le nord »).

Globalement, à l'instar de Filliat [70], on peut proposer la classification suivante pour les sommets :

– **Sommets définis par l'homme** : le robot détecte des types de lieux ou des balises (portes, couloirs, jonctions,... [107] [41]) entièrement prédéfinis par le concepteur du robot, et exploite des connaissances a priori sur l'apparence de ces lieux ou balises (descriptions « exhaustives »permettant l'extraction de ces divers éléments). Il dispose ainsi d'algorithmes explicites lui permettant de reconnaître ces objets : par exemple, chez Dedeoglu et Mataric [41], le robot détecte les portes en recherchant un espace libre entre deux murs parallèles dans des mesures issues de sonars. Dans ce type d'approche, il existe toutefois un fort risque d'ambiguïté perceptuelle (« perceptual aliasing ») puisque deux balises de même nature peuvent être ensuite difficiles à discriminer par le robot, en l'absence d'autres éléments de description de l'objet. En outre, la définition des sommets risque de ne pas être adaptée aux capacités perceptuelles du robot : le système ne prendra pas en compte une balise très discriminante (au regard de ses capacités sensorielles) si celle-ci ne fait pas partie des objets à rechercher et, à l'inverse, il peut arriver que l'homme définisse un type de balise difficile à reconnaître par le robot.

– **Sommets définis à des lieux canoniques** : le concepteur du robot ne spécifie pas complètement la nature des sommets mais définit les lieux où un sommet peut être créé et laisse le soin au robot de le trouver et de le caractériser précisément. Ainsi, chez Kuipers [106], les sommets, appelés « lieux distinctifs » (« distinctive places »), sont définis comme des points uniques correspondant aux extrema locaux de mesures dites de « distinctiveness » (par exemple, la mesure « distances égales entre le robot et les objets voisins » qui fournit les nœuds d'un diagramme de Voronoï). Des lois de commande basées elles aussi sur ce type de mesures guident localement le robot vers ces lieux canoniques, tels que le « centre » d'une intersection entre deux couloirs. La stratégie de cartographie d'Engelson est fondée sur le même principe [65]. Chez Kortenkamp et Weymouth [103], le robot détecte des transitions entre régions d'espace libre distinctes (passages, portes, jonctions), désignées par le terme générique de « gateways ». Une stratégie de navigation locale lui permet en outre de se placer précisément au milieu de ces « gateways ». Ensuite, comme les sommets correspondent à des lieux précis (des points et non des zones de surface non nulle), il devient plus facile d'y associer des informations facilitant la reconnaissance ultérieure du sommet au cours de la navigation du robot (on peut aussi parler d'indexation perceptuelle [65]) : scan sonar, grille d'occupation locale [106], signature visuelle [103]. En effet, on règle ainsi le problème classique du « point de vue » [70] : lorsque les lieux sont définis par l'opérateur, correspondent à une zone de surface non nulle et peuvent éventuellement être détectés à distance (par exemple, les portes chez Dedeoglu et Mataric [41]), les informations supplémentaires que l'on pourrait y attacher risquent d'être dépendantes de la position du robot au moment de l'observation. Ici, ces informations sont attachées à un lieu précis, qui peut être retrouvé par le robot en appliquant localement des lois de commande spécifiques : il n'y a plus de risque de variation locale de ces informations. En revanche, cela implique une trajectoire particulière pour le robot (pas de cartographie « passive »). De plus, dans le cas où le robot utilise une fonction de « distinctiveness », le bon fonctionnement du système repose sur l'habileté et l'intuition du concepteur dans la définition de cette fonction.

– **sommets définis automatiquement par classification non supervisée**, souvent par réseau de neurones (« self-organizing map de Kohonen » [100] chez Kurz [108], perceptron chez Nehmzow et al. [143]). Cette fois, la définition des sommets ne dépend plus de l'homme (si ce n'est dans la définition du réseau de neurones). En général, on suppose que la situation sensorielle du robot est relativement constante au sein d'un lieu donné : on détecte donc un nouveau lieu lorsque la variation de situation sensorielle dépasse un certain seuil. Cette situation sensorielle peut être constituée de la configuration locale des directions et des distances entre balises ([120] [76]), de la valeur des capteurs de proximité ([143]), ou des caractéristiques des images panoramiques ([187]). Chez Kurz par exemple [108], les sommets (« situations areas ») correspondent à des régions où les données télémétriques sont similaires et la carte est entraînée pour reconnaître ces groupements de données sensorielles. Chez Nehmzow et al. [143], le réseau de neurones apprend le lien entre la perception à l'instant t et à l'instant $t + 1$: les endroits où la prédiction du modèle est mauvaise sont automatiquement sélectionnés comme sommets. Ainsi, les sommets sont bien adaptés aux modalités de perception du robot. En contrepartie, ils risquent de ne plus être très signifiants pour un humain.

– **sommets définis à intervalles réguliers dans l'espace** : dans certains systèmes robotisés, comme celui de Yamauchi et Beer [202], un nouveau sommet est créé dans la carte dès que la distance parcourue dépasse un certain seuil. Ce sommet ne correspond donc pas à un lieu spécifique de l'environnement : il s'agit simplement de discrétiser l'espace de manière relativement régulière (d'un point de vue métrique) en une carte topologique comportant un nombre fini de sommets. Cette stratégie peut toutefois conduire à un encombrement inutile de la carte, au risque d'omettre des sommets intéressants et facilement reconnaissables par le robot.

On comprend donc que ces diverses stratégies induisent des différences de densité dans les sommets, selon que l'espace est échantillonné régulièrement pour créer de nouveaux sommets [202] ou que le système effectue un choix en fonction de ses perceptions sensorielles ([106] [41] [143]...). Nous verrons également dans le chapitre consacré aux représentations hybrides que les nœuds ne sont pas forcément insérés dans le même ordre : dans les représentations purement topologiques, les nœuds sont plutôt ajoutés tout au long du trajet du robot tandis que chez Thrun par exemple [178], ceux-ci sont définis a posteriori.

Les arêtes topologiques, quant à elles, procèdent toutes en général d'une notion d'adjacence. Toutefois, comme le souligne Filliat [70], la nature de ces arêtes est variable :

- Celles-ci peuvent être orientées (on parle alors d' « arcs », comme chez Taylor et Kriegman par exemple [177]) ou non orientées (dans le cas des graphes d'adjacence par exemple).

- Certaines arêtes peuvent demeurer implicites : par exemple, le fait que des sommets partagent les mêmes balises peut fournir des informations sur l'adjacence entre lieux (cf. Levitt et Lawton [120]).

- L'existence d'une arête peut recouvrir d'autres notions que l'adjacence : une notion de visibilité chez Taylor et Kriegman [177], une information d' « atteignabilité » suivant un comportement sensorimoteur donné chez Kuipers [106], etc.

- Ainsi, une arête peut parfois être véritablement ancrée dans l'espace (métrique) même s'il s'agit d'un ancrage implicite (lien déduit d'une mesure de discernabilité chez Kuipers [106], en particulier si le robot suit le diagramme de Voronoï [192]) et parfois cet ancrage n'existe pas (par exemple, chez Kortenkamp, on a juste une notion d'adjacence entre « gateways » [101]).

- Plus généralement, comme on le verra plus loin lors de la description des modèles de cartes hybrides, une arête peut être enrichie d'informations métriques (notamment la longueur du chemin reliant les deux extrémités). Cependant, dans un cadre purement topologique, cette information n'est pas utilisée dans un repère de référence commun pour inférer des informations métriques globales, mais plutôt pour faciliter la levée d'ambiguïtés ou pour faciliter la recherche opérationnelle dans le graphe (recherche de plus courts chemins notamment).

- Enfin, quelques auteurs ont ajouté un degré de confiance sur l'existence de ces arêtes : dans le réseau adaptatif de lieux de Yamauchi et Beer [202], cet indice de confiance est plutôt utilisé pour gérer les modifications éventuelles de l'environnement (obstacles qui bougent et portes qui se ferment).

Ces définitions variables des arêtes et des sommets ont donné lieu à des représentations diverses. On peut toutefois en dégager globalement un certain nombre de caractéristiques communes, comme l'indique la discussion qui suit.

2.2.2 Discussion

Avantages :

Selon [102], l'avantage principal des cartes topologiques revient à s'abstraire des problèmes d'incertitude dans le mouvement des robots : les incertitudes ne s'accumulent pas globalement car le robot se contente de naviguer localement, entre endroits. Il n'y a donc pas besoin d'estimation précise de la

position de la plate-forme robotisée [178], ce qui limite les ressources de calcul nécessaires. En outre, ces représentations sont bien adaptées aux algorithmes de planification puisque la taille de l'espace de recherche est petite comparée à l'ensemble des trajectoires possibles dans l'espace 2D continu [70].

Concernant l'interaction avec les humains, certaines de ces cartes peuvent présenter un découpage de l'espace qui facilite l'utilisation par l'homme [114] (notamment si les lieux correspondent à des pièces ou à des couloirs [103]) : par exemple, on peut imaginer de donner l'ordre au robot de se rendre dans la salle de séjour plutôt qu'aux coordonnées cartésiennes (x,y) [70]. Thrun souligne ainsi que ces modèles sont bien adaptés aux planificateurs et aux systèmes de résolution de problèmes symboliques, ainsi qu'à l'interaction en langage naturel car on peut facilement y ajouter des informations symboliques.

Par ailleurs, ces cartes sont souvent considérées comme compactes dans leur représentation de l'espace car elles ne matérialisent que les lieux intéressants, dans un graphe d'adjacence. Cependant, cette propriété dépend de la définition des sommets : les cartes où de nouveaux sommets sont créés dès que la distance au sommet le plus proche dépasse un certain seuil [202] sont souvent moins économiques en terme de place mémoire, mais la discrétisation de l'espace métrique reste tout de même plus lâche que dans une grille d'occupation par exemple [178].

Concernant l'adaptation aux différents types de capteurs, la définition des sommets est souvent relative au capteur utilisé (mesures de « distinctiveness » associées aux capteurs télémétriques par exemple [106]). En revanche, si la position de sommet est bien définie localement (lieu ponctuel comme chez Kuipers [106]), on peut facilement attacher de multiples données sensorielles, issues de capteurs variés (signatures visuelles ou télémétriques notamment).

Enfin, comme l'indique Filliat [69], divers systèmes robotisés inspirés du fonctionnement cérébral du rat (et notamment des cellules de lieu existant dans son hippocampe) reposent sur des cartes topologiques, ce qui suggère que ce type de cartes est biologiquement plausible.

Inconvénients :

Comme le souligne Kortenkamp [101], le principal inconvénient des modèles purement topologiques est justement la qualité qui les rend attractifs : l'absence d'informations géométriques. Cette lacune peut empêcher le robot de réaliser des raisonnements spatiaux sur l'ensemble de son environnement. En particulier, le système robotisé peut avoir des difficultés à sélectionner le chemin optimal entre deux lieux [178] : d'une part, le manque d'information sur la longueur des chemins (sur les arcs) peut l'empêcher de choisir entre deux branches du graphe qui mènent au même endroit, et d'autre part, il n'est pas possible de trouver un chemin plus direct dans l'espace métrique 2D que ceux qui sont implicitement codés dans les arêtes du graphe (par exemple, les chemins qui suivent implicitement le diagramme de Voronoï). Plus généralement, selon Prescott [161], la capacité de déterminer des raccourcis ou des routes directes dans un espace non exploré nécessite la connaissance de relations spatiales de plus haut niveau.

De plus, nous avons vu que les cartes topologiques posent parfois des problèmes pour distinguer des lieux différents (« perceptual aliasing »), et souffrent en particulier d'une sensibilité au point de vue, ce qui génère parfois des ambiguïtés de positionnement (qui pourraient être résolues avec informations topométriques comme les coordonnées cartésiennes des sommets).

En outre, selon Prescott [161], si les sommets sont impossibles à distinguer (ou du moins séparables localement sur la base des informations sensorielles qui leur ont été attachées, mais pas globalement, ce qui est souvent le cas), la construction d'une carte purement topologique nécessite de mettre en œuvre un processus très coûteux et peu efficace lors de la création d'un nouveau sommet. En effet, il faut alors comparer les observations courantes du robot aux informations sensorielles attachées aux sommets existants pour vérifier que l'on n'est pas revenu à un lieu connu. Comme ces informations ne sont pas forcément discriminantes, cette procédure (appelée « rehearsal procedure » chez Kuipers [106]) exige un test spatial sur chaque sommet candidat dans lequel les relations d'adjacence de ces candidats sont

comparées à celles du sommet à créer : les sommets voisins sont comparés de manière récursive. Ce test ne s'arrête que si les données sensorielles permettent de retrouver l'identité des sommets testés. Ainsi, il importe de savoir combien de places globalement distinctives sont nécessaires pour réaliser ce test : Kuipers en utilise une seule [106] mais Prescott montre que cela est insuffisant [161]. Une alternative consiste à limiter le répertoire de comportements du robot : simple suivi de mur ou suivi du diagramme de Voronoï, de manière à faciliter la segmentation et l'identification de lieux en limitant le nombre de points où le robot est face à un véritable choix de navigation (sommets de degré supérieur à 2). Si cette stratégie peut fonctionner dans un espace de taille limitée, en contrepartie, les zones ouvertes sont moins bien représentées et cela limite considérablement les chemins possibles pour le robot.

Ainsi, dans un certain nombre d'approches topologiques ([132] [106]), la cartographie n'est pas une tâche « passive » puisqu'elle ne peut pas être réalisée en parallèle d'une tâche quelconque (ou d'une téléopération) : elle nécessite souvent un comportement de navigation particulier pour le robot, tel que le suivi de mur, et implique une trajectoire spécifique. De plus, l'arête entre deux sommets n'est généralement ajoutée que si le robot a réellement parcouru un chemin entre les deux extrémités de l'arête (excepté dans quelques systèmes où cette information peut être inférée « à distance » [189]) : cela nécessite souvent une longue période de navigation dans l'environnement afin de parcourir tous les chemins possibles et apprendre la carte de manière exhaustive.

Selon Lee [114], les représentations topologiques conviennent lorsque les balises ou les lieux distinctifs dominent l'environnement et que les sommets sont reliés par des chemins bien définis (notamment par des lois de commandes locales comme chez Kuipers [106] ou Kortenkamp et Weymouth [103]), mais elles peuvent poser problème dans des environnements ouverts tels que des entrepôts, qui ne posséderaient pas de marqueurs distinctifs (si ce n'est des informations métriques) et où les robots n'approcheraient pas forcément des balises ou des références nécessaires à la navigation (murs, portes...) situées en périphérie.

Enfin, concernant l'interaction homme / robot, de nombreuses cartes topologiques ne correspondent pas à la représentation « idéale » décrite ci-dessus, comportant un sommet par pièce et par jonction : les représentations topologiques sont parfois perçues comme relativement « hermétiques », propres aux robots et à leurs capacités sensorielles particulières (et même dans certains cas, propres à un seul robot donné, sans possibilité de partage avec d'autres robots), et donc difficiles à appréhender pour un humain (à moins de mettre en œuvre un lourd traitement de mise en forme).

2.3 Cartes métriques

Les systèmes basés sur les cartes métriques visent à produire une représentation géométrique plus ou moins détaillée de l'environnement à partir des données perceptuelles. Dans ce type de cartes, des informations de longueur, de distance ou de position apparaissent explicitement et sont en principe définies dans un référentiel unique. Ainsi, elles sont souvent faciles à appréhender par l'homme car elles offrent une relation bien définie avec le monde réel [114].

En revanche, la construction de cartes métriques cohérentes nécessite de gérer des contraintes géométriques particulières, notamment lorsque le robot retourne sur un lieu connu après avoir réalisé une boucle dans l'environnement (couloir qui se referme sur lui-même par exemple). En effet, on constate souvent, à la fermeture d'un tel cycle, que les observations successives ne se recouvrent pas correctement. Plusieurs raisons permettent d'expliquer ce phénomène [179] :

- dans certains cas (par exemple dans le cas des grilles d'occupation classiques), le système ne peut corriger les positions antérieures du robot (et donc indirectement des éléments observés par le robot à ces positions antérieures), ce qui paraît pourtant indispensable pour assurer une bonne superposition des données présentes et passées ;
- parfois, le robot ne maintient qu'une seule hypothèse sur sa position (par exemple dans le cas des filtres de Kalman où la position du robot est modélisée par une distribution gaussienne unimodale) :

il ne peut pas gérer correctement certaines ambiguïtés de positionnement ;
– dans certains cas, le robot ne tient pas vraiment compte de l'incertitude sur sa position, ce qui gêne l'appariement entre l'observation et la carte courante. En effet, cet appariement est souvent réalisé par des méthodes de descente de gradient qui nécessitent une bonne estimation initiale de la position du robot (celle-ci doit se trouver dans le bassin d'attraction de l'extremum), alors que l'incertitude sur cette position peut croître sans borne lors du parcours du cycle.

Ainsi, plus encore que pour les cartes topologiques, la gestion des cycles est l'un des enjeux majeurs de la construction de cartes métriques puisque outre la détection du cycle, elle nécessite la gestion de la cohérence géométrique de l'ensemble de la représentation.

2.3.1 Représentations basées sur l'apparence

Description

Les représentations basées sur l'apparence sont constituées des données brutes issues des capteurs et ne font l'objet d'aucun traitement de mise en forme tel que l'extraction de primitives géométriques : c'est pourquoi on parle aussi d'approches directes [194]. En général, il s'agit d'un ensemble de scans lasers recalés les uns par rapport aux autres de manière à maximiser leur correspondance (et donc visuellement leur superposition) [126] [81] [179] [166]. On peut aussi placer dans cette catégorie les ensembles d'images recalées en vue de former une mosaïque [188].

FIG. *2.2: Une carte de notre couloir (dans les locaux du CEP Arcueil) constituée de scans laser bruts, construite selon une méthode similaire à celle de Röfer [166].*

Méthodes de construction de ces cartes

Avant de détailler les méthodes de construction de ces cartes, qui sont pour la plupart basées sur le filtrage bayésien, nous rappelons brièvement le principe de ce filtrage.

*** Filtrage bayésien :**

Dans plusieurs articles, Thrun montre que la grande majorité des méthodes de cartographie actuelles recourent à la théorie des probabilités et à l'inférence probabiliste, de manière à transformer les mesures sensorielles imprécises en cartes métriques [179] [180]. Certaines approches sont plus explicites que d'autres sur leur origine probabiliste mais on peut souvent se ramener à un tel cadre en s'appuyant sur les hypothèses adéquates.

Ainsi, la plupart de ces approches reposent sur la règle de Bayes, qui dans sa forme la plus générale, s'énonce comme suit :

$$p(x \mid d) = \mu p(d \mid x) p(x)$$

où x représente la quantité à estimer et d l'observation, tandis que μ est un facteur de normalisation (ici, $1/p(d)$ en fait). Elle exprime que la probabilité *a posteriori* $p(x \mid d)$ de la variable aléatoire x connaissant l'observation d est proportionnelle au produit de deux termes : une probabilité *générative* $p(d \mid x)$ (qui décrit comment est générée une observation d selon l'hypothèse du « monde » x) et une probabilité *a priori* $p(x)$ (qui indique une supposition sur le « monde », préalablement à toute observation).

Dans le problème de la cartographie et de la localisation simultanées, les données arrivent au fur et à mesure. Suivant des notations classiques, on peut désigner les données sensorielles extéroceptives (issues notamment de télémètres laser, de sonars ou de caméras) obtenues à l'instant t par la variable z_t et les commandes générées entre l'instant t et l'instant $t+1$ (qui sont parfois remplacées par les données proprioceptives issues de l'odométrie ou de centrales inertielles) par la variable u_t. L'exposant ^{t} fait quant à lui référence à l'ensemble des données obtenues entre l'instant 1 et l'instant t. Les filtres bayésiens étendent la règle de Bayes pour l'appliquer de manière récursive à des problèmes d'estimation temporelle. La quantité x_t est appelée *l'état* et le filtre bayésien permet de calculer récursivement sa probabilité a posteriori via l'équation suivante [180] :

$$p(x_t \mid z^t, u^t) = \mu p(z_t \mid x_t) \int p(x_t \mid u_t, x_{t-1}) p(x_{t-1} \mid z^{t-1}, u^{t-1}) dx_{t-1}$$

où μ représente un facteur de normalisation. Dans le cas du SLAM, l'état x_t contient généralement la carte m_t et la position du robot s_t, qui influent toutes deux sur les observations. Selon l'hypothèse d'un environnement statique (donc d'une carte m statique), cette équation devient [180] :

$$p(s_t, m \mid z^t, u^t) = \mu p(z_t \mid s_t, m) \int p(s_t \mid u_t, s_{t-1}) p(s_{t-1}, m \mid z^{t-1}, u^{t-1}) ds_{t-1} \tag{2.1}$$

où la probabilité $p(z_t \mid s_t, m)$ est considérée comme le *modèle d'observation* et la probabilité $p(s_t \mid u_t, s_{t-1})$ comme le *modèle de déplacement*.

Comme l'indique Thrun [180], il est clair que cette équation ne peut être résolue telle quelle par un ordinateur. En effet, elle fait intervenir une distribution de probabilité sur un espace continu, qui présente donc une infinité de dimensions. Comme nous le verrons, les algorithmes de SLAM sont donc contraints de recourir à des hypothèses supplémentaires, ce qui conduit à diverses techniques de résolution telles que le filtrage de Kalman, les algorithmes EM (« Expectation - Maximisation »), ou la « Rao-Blackwellisation ».

*** Différentes manières de gérer les environnements cycliques**

Ainsi, il existe diverses méthodes de construction de cartes, qui gèrent de manière plus ou moins élégante la rétropropagation des corrections d'erreur de positionnement des scans lors de la fermeture de boucles dans l'environnement :

**** Méthode de maximum de vraisemblance incrémentale couplée à une méthode heuristique pour la gestion des cycles**

Chez Röfer [166], lors de l'acquisition d'un nouveau scan, le robot recherche parmi les scans de la carte ceux qui sont susceptibles d'être appariés : il utilise pour cela un simple critère de distance. De plus, l'auteur privilégie les scans les plus anciens, c'est-à-dire ceux dont la position est la moins susceptible d'avoir subi une dérive importante de l'odométrie. Ensuite, le système met en œuvre un algorithme d'appariement par histogrammes (cf. chapitre 4) entre le scan courant et les scans candidats de manière à estimer la position du scan courant avant de l'ajouter à la carte et de recaler par la même occasion l'estimation de position du robot.

Cette approche « gloutonne » peut aussi s'énoncer de la manière suivante [179] : « Etant données une observation (un balayage du télémètre laser) et une lecture d'odométrie, calculer la position la plus probable pour le robot, puis ajouter à la carte cette observation avec cette position et les figer de manière définitive ». Par rapport au cadre théorique du filtrage bayésien que nous venons de décrire, cette technique peut être vue comme un algorithme incrémental très populaire, basé sur une estimation au sens du maximum de vraisemblance [179] [180]. A l'instant $t-1$, le robot dispose d'une estimation de sa pose $\hat{s}^{t-1}$ et de la carte $\hat{m}(\hat{s}^{t-1}, z^{t-1})$. Après déplacement et acquisition d'une nouvelle observation z_t, le robot détermine sa position la plus vraisemblable selon l'équation suivante (dérivée de l'équation 2.1 en considérant les estimations $\hat{m}(\hat{s}^{t-1}, z^{t-1})$ et $\hat{s}^{t-1}$ comme certaines) :

$$\hat{s}_t = \arg\max_{s_t} \{p(z_t \mid s_t, \hat{m}(\hat{s}^{t-1}, z^{t-1})).p(s_t \mid u_{t-1}, \hat{s}_{t-1})\}$$

Quant à la probabilité a posteriori sur la carte, elle n'a pas besoin d'être estimée explicitement puisque les représentations fondées sur l'apparence permettent de construire directement la carte à partir des estimations successives s^t de la pose du robot : la pose de chaque scan est confondue avec la pose du robot au moment de l'acquisition de ce scan. De manière incrémentale, pour mettre à jour la carte, il s'agit donc simplement d'ajouter à la carte $\hat{m}(\hat{s}^{t-1}, z^{t-1}))$ le scan courant positionné selon $\hat{s}_t$.

Ce type d'approche est relativement courant en cartographie [13] [62] [202]. Elle peut toutefois conduire à des incohérences lors de la fermeture de cycle, pour les raisons que nous avons évoquées plus haut : cette technique ne maintient pas l'estimation d'incertitude sur la position du robot, ce qui peut l'empêcher de détecter la fermeture de boucle, et elle ne propose pas de mécanisme de rétropropagation des erreurs d'estimation le long du cycle. C'est pourquoi Röfer a ajouté dans son algorithme un mécanisme de fermeture de cycle *ad hoc* : lorsqu'un cycle est détecté par appariement de scans, la correction des erreurs en translation et en rotation est propagée sur l'ensemble des scans de la boucle, selon une pondération calculée d'après la valeur de confiance de l'appariement au moment de l'insertion de ces scans dans la carte.

**** Méthode de maximum de vraisemblance incrémentale couplée à une localisation de type Monte Carlo pour la gestion des cycles**

Dans l'un de ses articles, Thrun [179] propose un algorithme basé lui aussi sur une estimation

incrémentale par maximum de vraisemblance. Toutefois, pour gérer la fermeture de cycles, cet algorithme est défini comme une méthode hybride qui implante en outre un second estimateur estimant la probabilité a posteriori complète sur les positions du robot (mais pas sur les cartes). L'auteur indique que cela revient à employer un algorithme de localisation markovienne [173] [19], à la différence près que l'on ne suppose pas qu'une carte complète de l'environnement est disponible. Cet estimateur est implanté via un filtre particulaire (cette technique est connue sous le nom d'algorithme de condensation en vision et de « Monte-Carlo Localization » ou « MCL » en robotique mobile). Ensuite, le sytème calcule la correction d'erreur qui doit être rétropropagée le long du cycle, par différence entre la meilleure estimation obtenue par maximum de vraisemblance incrémental (qui utilise ici une montée de gradient à partir d'une estimation initiale de position du robot souvent éloignée de la position réelle, au risque de tomber sur un maximum local) et la meilleure estimation utilisant la probabilité a posteriori complète sur la pose du robot (qui permet de faire des hypothèses plus proches de la position réelle du robot et augmente les chances de trouver le maximum global). Cette correction est alors distribuée de manière proportionnelle le long de la boucle, avant d'appliquer itérativement une montée de gradient sur toutes les poses pour maximiser la vraisemblance dans le cycle. Si ces approches hybrides ont donné de bons résultats dans de larges environnements cycliques, il semble que la qualité de la carte dépende directement de la précision et de la richesse de l'information issue du capteur.

**** Optimisation des positions des scans ou des images selon un réseau de contraintes locales**

La plupart des méthodes de cartographie fondées sur un réseau de contraintes entre scans reviennent à modéliser la carte comme un système de ressorts dont les longueurs au repos correspondent aux distances observées entre les scans (cf. Fig 2.3). Lors de la fermeture d'un cycle, si la carte est incohérente, le réseau de ressorts n'est pas en équilibre : pour corriger la carte, il suffit de calculer la position de repos du système de ressorts [126] [78]. En effet, cette position de repos correspond à la position dans laquelle leur écart à la longueur observée est la plus faible.

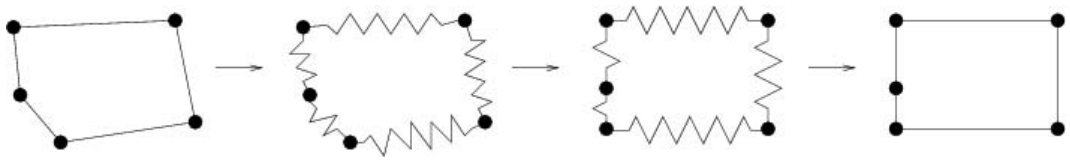


FIG. *2.3: Modélisation du réseau de scans selon un système de ressorts. Chaque sommet représente un scan. Le calcul de la position d'équilibre du système de ressorts permet d'assurer la cohérence de la carte.*

Ainsi, Lu et Millios proposent de construire des cartes constituées de scans laser par résolution d'un système sur-contraint de mesures (qui correspondent à des contraintes de distance entre scans) [126] : ils calculent ainsi la perturbation au sens des moindres carrés sur les positions absolues des scans (optimisation par maximum de vraisemblance). Les contraintes locales entre poses sont des mesures de distance associées à leurs covariances : elles sont issues soit de l'odométrie, soit de l'appariement de scans par un algorithme d'ICP (« Iterative Closest Point » - cf. chapitre 4). Cet algorithme est cependant très coûteux, d'une complexité de l'ordre de $O(n^3)$, n étant le nombre de scans. Les auteurs en proposent une version incrémentale, mais celle-ci n'apporte pas réellement de gain par rapport à la méthode initiale qui nécessite à chaque fois de retraiter l'ensemble des données. Ils proposent également des raffinements pour limiter la complexité : transformer les contraintes fortes (à faible variance) en liens rigides, travailler momentanément sur des cartes locales composées de sous-ensembles de scans et reliés avec les autres

cartes locales par une seule contrainte (toutefois, s'il s'avère impossible de trouver un lien unique entre ces sous-cartes, la méthode devient sous-optimale).

Gutmann et Konolidge étendent cette approche pour définir un algorithme applicable en ligne pour cartographier un environnement cyclique au moyen d'un télémètre laser [81]. Ils améliorent la méthode de Lu et Milios au sens où ils mettent en œuvre les accélérations proposées et en ajoutent quelques autres. D'abord, ils proposent de faire tourner l'algorithme de Lu et Milios seulement sur un nombre réduit et fixe de scans voisins, afin de réaliser un suivi de position du robot (le nombre de scans considérés a été réglé empiriquement). En parallèle, un test régulier de fermeture de cycle est réalisé par corrélation du scan local avec la carte globale, au moyen de grilles d'occupation. On peut ainsi considérer cette approche comme une méthode hybride, au sens où l'entend Thrun [179] puisque le test de corrélation permet d'estimer une probabilité a posteriori plus complète sur la pose du robot. Lorsqu'une fermeture de cycle s'impose, le système applique l'algorithme de Lu et Milios [126] sur l'ensemble de la boucle. De plus, il recherche les liens forts entre scans (liens à faible variance) et les considère comme rigides, ce qui revient à supprimer des sommets (des scans) du réseau de contraintes. Enfin, les auteurs réduisent le temps de calcul en exploitant les propriétés des matrices creuses (la matrice d'observation des contraintes de distance est creuse car de nombreuses paires de scans ne font pas l'objet d'une contrainte de distance), mais la complexité de l'estimation lors du bouclage de cycle reste de l'ordre de $O(n^3)$ (n étant le nombrede scans).

Unnikrishnan et Kelly [188] effectuent eux aussi une optimisation sur un réseau de contraintes pour redresser des mosaïques d'images aériennes qui rebouclent sur elles-mêmes, avec une technique un peu différente de celle de Lu et Milios, mais plus proche du filtre relatif de Newman [148] que nous verrons plus loin. Quant à Golfarelli et al., ils exploitent directement l'analogie avec le système de ressorts, chaque arête du graphe étant modélisée comme une paire de ressorts : un ressort axial linéaire et un ressort rotationnel. Les paramètres d'élasticité des ressorts sont déduits des incertitudes odométriques. La complexité de l'algorithme global correspond à l'inversion d'une matrice de taille $4n \times 4n$. Enfin, Bosse et al. ont développé le système ATLAS, [16] basé sur un réseau de cartes locales reliées par des contraintes géométriques, qui permet de cartographier de très larges environnements. La représentation employée peut être vue comme une combinaison de cartes topologiques et métriques : c'est pourquoi elle est détaillée dans le chapitre suivant. Cependant, on peut noter qu'ATLAS propose un cadre de travail modulaire qui peut fonctionner directement sur des scans laser.

**** « Rao-Blackwellisation »**

Enfin, plus récemment, une technique dite de « FastSLAM » [137] a été appliquée à des ensembles de scans [83] : il s'agit de combiner l'appariement de scans à une méthode de filtrage particulaire, selon le principe de la Rao-Blackwellisation (expliqué plus en détail à la section 2.3.3). Cette approche représente la trajectoire du robot s^t par un ensemble de particules, chacune de ces particules maintenant sa propre estimation de la carte. En fait, pour une particule i donnée, chaque scan k du modèle (acquis à l'instant k) est positionné selon la position correspondante *exacte* $s_{i,k}$ du robot dans la trajectoire s_i^t représentée par cette particule : l'estimation de la carte m_i pour une particule donnée est donc immédiate. L'avantage de cette méthode est que les particules approximent à chaque instant la probabilité a posteriori complète sur les poses du robot et sur la carte, ce qui permet de gérer correctement la fermeture de cycles. A chaque pas de temps, pour chaque particule, l'algorithme tire au sort une nouvelle estimation de position du robot selon la probabilité indiquée par le modèle de déplacement du robot $p(s^t \mid u^t, s_{t-1})$. Ensuite, chaque particule i est pondérée selon la vraisemblance de l'observation z_t pour la position courante $s_{i,t}$. Un ré-échantillonnage des particules est alors réalisé selon cette pondération.

Dans l'algorithme proposé, l'appariement de certains scans est utilisé pour pallier l'odométrie dans

le modèle de déplacement du robot : comme la mesure de distance déduite de l'appariement est plus précise que l'odométrie, la faible variance qui en résulte réduit le besoin de ré-échantillonnage et limite ainsi le problème de déplétion des particules (qui peut conduire au manque de particules près de la position réelle du robot), classique en FastSLAM [137]. Les autres scans, qui ne sont pas employés pour corriger l'odométrie, constituent les observations et servent à estimer la carte dans le processus de filtrage particulaire. L'algorithme résultant permet de cartographier en temps réel de larges environnements cycliques (de l'ordre de 10m × 30m) en utilisant seulement une centaine de particules. Le principal problème de ces algorithmes de « FastSLAM » est cependant que l'influence du nombre de particules sur le fonctionnement du processus d'estimation est mal connue.

Discussion

*** Avantages** :

Les représentations basées sur l'apparence offrent l'avantage d'être très génériques. Elles sont a priori adaptées à tout type d'environnement puisqu'elles peuvent modéliser des objets de taille ou de forme variables, qu'il s'agisse d'obstacles polygonaux de grande dimension, de petits objets quasiment ponctuels ou de surfaces courbes. De plus, comme les données ne subissent aucun traitement de mise en forme, cela limite les erreurs d'interprétation qui peuvent arriver lors de l'extraction de primitives géométriques ou la reconnaissance d'objets par exemple.

Ce type de cartes est en outre très flexible : comme la forme et la position des obstacles se précisent peu à peu par accumulation de points de mesure et non par fusion successive des données (comme c'est le cas dans la plupart des autres représentations), il est assez facile de revenir a posteriori sur les mises en correspondance puisque les informations ne sont pas intégrées de manière irréversible. De même, il est aisé de corriger les positions passées du robot, qui sont confondues avec les positions des éléments de la carte. En particulier, les méthodes de construction existantes permettent de gérer la construction en ligne de cartes de grande dimension en utilisant les méthodes de rétropropagation des erreurs le long des cyles proposées par Thrun et al. par exemple [179].

Enfin, elles sont adaptées à une cartographie passive puiqu'elles ne nécessitent pas de suivre une trajectoire particulière.

*** Inconvénients** :

En contrepartie de leur flexibilité, ces cartes manquent de structuration, ce qui les rend mal adaptées à l'application directe d'algorithmes de planification. En général, pour exploiter les représentations à base de scans laser, il faut d'abord mettre en œuvre des traitements de mise en forme qui les transforment en grille d'occupation ou qui extraient des primitives géométriques. A priori, ces traitements ne sont pas prévus pour être utilisés de manière incrémentale, notamment si la position des scans évolue, ce qui signifie qu'il faut les appliquer sur l'ensemble des données à chaque fois que l'on veut réaliser un raisonnement spatial sur la carte.

De plus, comme ces représentations sont composées des données brutes non structurées, elles manquent de compacité : l'encombrement mémoire pourrait être réduit par une compression en primitives géométriques par exemple.

Concernant la compréhension par l'homme, elle est satisfaisante si les données sont suffisamment denses et nombreuses pour faire apparaître clairement les objets, par accumulation de points. Toutefois, la position des obstacles est parfois floue du fait du manque de structuration et des ambiguïtés peuvent résulter du manque d'information sur « le plein et le vide » (zones occupées ou non par un obstacle).

Enfin, ces méthodes sont plutôt adaptées aux capteurs riches tels que les caméras ou les télémètres

laser (Gutmann et Konolige les classent parmi les méthodes à base de capteurs denses [81]), qui facilitent la mise en correspondance : notamment, il paraît difficile d'appliquer ces techniques sur des données sonar par exemple, du fait des ambiguïtés d'appariement. En revanche, on peut envisager de superposer aux données initiales (les scans laser) des données multicapteurs complémentaires acquises simultanément (images panoramiques par exemple).

2.3.2 Représentations surfaciques

Description et méthodes de construction

Dans les modèles surfaciques, l'espace est partitionné en un ensemble de régions (ou cellules) distinctes. A chacune de ces cellules est attaché un nombre (ou plusieurs) représentant une propriété de la région : le plus souvent, dans les représentations 2D, il s'agit du degré d'occupation par un obstacle (indice de confiance sur le fait que la cellule correspondante est occupée ou non par un obstacle). Les différences entre modèles surfaciques proviennent essentiellement de la forme et de la taille des cellules, ainsi que de la signification exacte des nombres attachés et de la manière de les mettre à jour.

*** Grilles d'occupation**

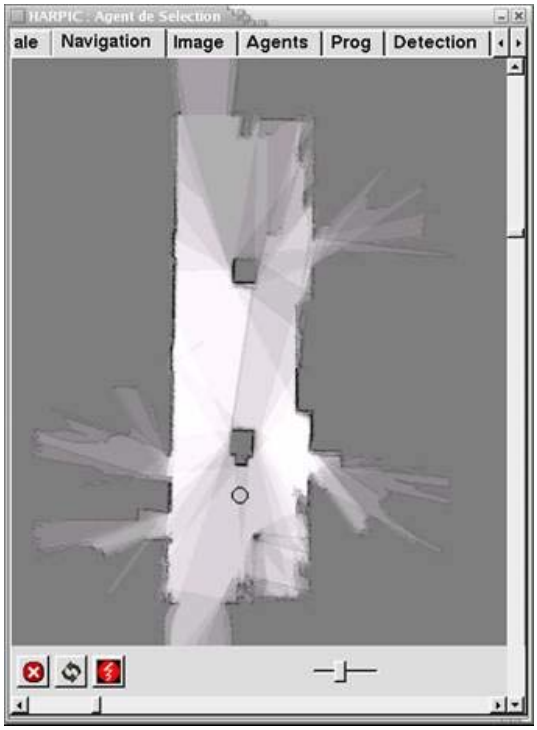


FIG. *2.4: Une grille d'occupation de notre couloir dérivée de la représentation basée sur les scans laser de la figure précédente. Le niveau de gris correspond à un simple degré d'occupation.*

Les grilles d'occupation constituent la représentation surfacique la plus simple et la plus populaire (cf. Fig. 2.4). Dans ce type de modèle, l'espace est discrétisé selon une grille régulière en cellules carrées ou rectangulaires de même taille. Chaque cellule contient un indice (probabilité, histogramme...) indiquant si l'espace correspondant est plutôt libre ou occupé. Chez Borenstein et Koren [13], l'indice d'occupation correspond à un entier compris entre 0 et 15 et il est simplement incrémenté ou décrémenté d'une valeur fixe à chaque observation. Il s'agit donc d'une approche simple et plutôt heuristique, qui ne permet

pas de construire des cartes de grande précision, mais qui est efficace pour la détection d'obstacles à grande vitesse à partir de sonars. Chez Elfes [61], cet indice représente d'abord un statut discret d'occupation (inconnu, occupé et vide) associé à un facteur de certitude compris entre 0 et 1. Quelques années plus tard, cet indice est traité plus formellement comme une probabilité d'occupation [62]. Lim et Cho [122] ajoutent pour leur part une probabilité d'orientation pour chaque cellule. Dans ces approches probabilistes, la cartographie mise en œuvre correspond en général à une approche bayésienne. Il s'agit cette fois d'estimer la probabilité de la carte indépendamment pour chaque cellule de la grille : on note $p(m_{x,y} \mid z^t, s^t)$ la probabilité que la cellule de coordonnées (x, y) soit occupée par un obstacle. L'équation récursive du filtre bayésien s'écrit cette fois de la manière suivante [62] [180] :

$$\log \frac{p(m_{x,y} \mid z^t, s^t)}{1 - p(m_{x,y} \mid z^t, s^t)} = \log \frac{p(m_{x,y} \mid z_t, s_t)}{1 - p(m_{x,y} \mid z_t, s_t)} + \log \frac{1 - p(m_{x,y})}{p(m_{x,y})} + \log \frac{p(m_{x,y} \mid z^{t-1}, s^{t-1})}{1 - p(m_{x,y} \mid z^{t-1}, s^{t-1})}$$

où $p(m_{x,y} \mid z^t, s^t)$ représente un *modèle capteur inverse* et $p(m_{x,y})$ la probabilité d'occupation a priori, qui est généralement fixée à $0,5$, ce qui le fait disparaître de l'équation.

Cette approche bayésienne, très populaire, fournit des cartes de bien meilleure qualité que la simple technique histogrammique de Borenstein et Koren par exemple. Elle souffre pourtant de certaines limitations, qui proviennent notamment de l'hypothèse d'indépendance entre cellules voisines (alors que les obstacles sont souvent regroupés : une cellule donnée a plus de chances d'être occupée si sa voisine l'est) et de l'hypothèse initiale des probabilités d'occupation à 0,5 (choix du maximum d'entropie en l'absence de données a priori) pour toutes les cellules, ce qui nécessite de nombreuses mesures pour assurer la convergence de l'algorithme [152].

Ainsi, certains chercheurs ont proposé d'autres cadres de travail, basés par exemple sur la théorie des possibilités et sur la logique floue : chaque cellule contient alors un nombre réel qui quantifie la possibilité que cette cellule appartienne à un obstacle. Par exemple, chez Oriolo et al. [152], l'ensemble E des cellules vides d'obstacle et l'ensemble O des cellules occupées sont modélisés par des ensembles flous, qui ne sont donc plus complémentaires, à la différence des ensembles classiques. Ainsi, une cellule peut présenter à la fois une appartenance partielle à E et à O. La mise à jour des degrés d'appartenance se fait selon une mise à jour classique en théorie des possibilités, ici via l'opérateur d'union de Dombi. Ce cadre de travail permet de déterminer dans quelles zones les informations sonar sont conflictuelles ou insuffisantes, de manière à construire des cartes « conservatives », qui correspondent à une estimation prudente, où les cellules libres d'obstacles n'incluent ni les cellules ambiguës de l'ensemble $E \cap O$, ni les cellules indéterminées de l'ensemble $\overline{E} \cap \overline{O}$. On peut alors envisager d'aller raffiner les cartes dans les zones où les informations sont insuffisantes. Les expérimentations menées par les auteurs permettent également de comparer l'algorithme flou à l'approche bayésienne classique. Elles montrent que l'algorithme flou est en principe plus robuste aux fausses alarmes issues des rélexions multiples, semble converger plus rapidement et s'avère plus prudent que l'algorithme bayésien (par exemple, les cellules situées au-delà d'un obstacle sont considérées comme non sûres, au contraire de la carte bayésienne qui les maintient à une valeur indéterminée de 0,5).

D'autres chercheurs recourent plus généralement à la théorie de l'évidence : le but est alors de déterminer le support des propositions E (vide) et O (occupé). Ainsi, chez Pagac et al. par exemple [154], l'ensemble de discernement utilisé est $\Omega = \{E, O\}$, l'ensemble de propositions correspondant à $\{\emptyset, E, O, \{E, O\}\}$. L'état de chaque cellule est alors caractérisé par des masses de croyance $m(E)$, $m(O)$, et $m(E \cup O)$ telles que leur somme vaille 1, avec $m(\emptyset) = 0$. Deux modèles de capteurs sont mis en œuvre, l'un pour E et l'autre pour O. Ensuite, la règle de combinaison de Dempster-Shafer est appliquée pour mettre à jour les masses de la carte à partir des masses d'observation. L'intérêt par rapport à la méthode bayésienne classique est de permettre un support simultané de plusieurs propositions à la fois.

En particulier, on évite les ambiguïtés liées à la signification d'une probabilité à $0,5$ pour une cellule, qui peut correspondre à la valeur initiale (inconnu) ou au fait que les informations reçues sur cette cellule sont contradictoires. De plus, comme la théorie de l'évidence fournit une borne inférieure (plausibilité) et une borne supérieure (croyance) sur les probabilités, elle permet de modéliser correctement le manque de données (ignorance), en fonction de largeur d'intervalle. Elle permet également de quantifier la probabilité de masses non distribuées, par évaluation de la qualité des probabilités a posteriori. Enfin, elle ne nécessite pas de définition précise des probabilités conditionnelles a priori sur les capteurs, ce qui permet d'incorporer un modèle de capteur plus réaliste.

Ribo et Pinz ont comparé expérimentalement ces trois méthodes de construction de grilles d'occupation [164] : une méthode classique bayésienne et probabiliste, une méthode basée sur la logique floue et une méthode fondée sur la théorie de l'évidence. Ces trois approches ont été testées avec des robots réels sur deux types d'environnements relativement bien structurés. En conclusion, il semble que les trois approches soient globalement satisfaisantes en environnement structuré, en l'absence de réflexions multiples du sonar (qui correspondent à des angles d'incidence élevés). L'approche basée sur la théorie des possibilités (logique floue) semble la plus robuste aux fausses alarmes et aux réflexions multiples. En contrepartie, elle fournit les cartes les moins précises (ce qui est dû semble-t-il à l'approche « conservative » mentionnée ci-dessus) et se révèle la plus gourmande en place mémoire. L'approche probabiliste s'avère quant à elle la plus rapide.

Enfin, on peut mentionner les travaux de Wijk et Christensen qui proposent un algorithme basé sur la triangulation des mesures sonar [198]. Cette technique permet de repérer des balises ponctuelles (les arêtes verticales de la scène) avec une grande précision compte tenu des limitations du sonar. L'algorithme garde en mémoire les dernières mesures puis, par une méthode de vote simple, regroupe les mesures sonar susceptibles d'avoir touché le même objet de l'environnement. Ensuite, une triangulation entre ces mesures permet d'atteindre une précision de localisation de l'obstacle de l'ordre de quelques centimètres. L'algorithme permet aussi de calculer la variance de cette localisation en utilisant une grille d'occupation locale. Ensuite, il est possible de construire une carte d'occupation globale : la mise à jour de la grille consiste par exemple à augmenter la probabilité d'occupation sur l'ellipse d'incertitude de l'objet détecté et à réduire la probabilité le long de la ligne de visée. Les résultats obtenus sont manifestement de bonne qualité grâce à la précision accrue de localisation des objets de l'environnement.

*** Représentations hiérarchiques**

Pour remédier au problème d'encombrement mémoire des grilles d'occupations, différents chercheurs ont proposé des représentations hiérarchiques. Nillson aurait déjà employé dans les années 70 une représentation pyramidale par « quadtrees » (tétra-arbres). Zelinsky reprend ce modèle dans lequel l'espace est subdivisé récursivement en quatre régions carrées jusqu'à ce que chaque carré présente un statut d'occupation homogène ou atteigne la taille minimale autorisée [205]. Outre le statut d'occupation, Zelinsky ajoute un indice de confiance pour chacune des cellules, correspondant au pourcentage de la région qui a été visitée durant l'exploration. La méthode de construction de cette représentation est liée au capteur particulier du robot : un capteur tactile. Le robot explore son environnement en se fixant un point à rallier et planifie sa trajectoire à partir d'une « transformation en distance » dans la carte en cours de construction. L'auteur recourt de plus à l'hypothèse forte que la position de la plate-forme est connue. Si le robot rencontre un obstacle sur son trajet, il remet à jour sa carte localement et replanifie sa trajectoire. Ainsi, au fur et à mesure des explorations, la carte devient représentative de l'environnement. Il semble toutefois que cette technique ne soit pas adaptée à une mise à jour extensive (de type bayésien par exemple), qui incorporerait itérativement de nombreuses mesures (issues d'une ceinture de télémètres

à ultrasons plutôt que d'un capteur simple tactile), car cela nécessiterait de trop grands coûts de calcul. En outre, cette méthode de construction n'est absolument pas passive puisqu'elle nécessite de nombreux passages dans l'environnement.

Chez Poncella et al. [156], la grille d'occupation est également traitée pour obtenir un partitionnement de résolution variable, où l'environnement est discrétisé en cellules représentant des régions homogènes. Les auteurs soulignent que dans les quadtrees, la complexité de la décomposition dépend beaucoup de la dispersion des obstacles. Pour y remédier, ils introduisent des étapes supplémentaires de fusion de cellules adjacentes homogènes et une classification des cellules homogènes (afin de réduire le nombre de classes). Cette représentation hiérarchique peut être construite en ligne au sens où il est rapide de la regénérer entièrement à partir d'une grille d'occupation, mais la méthode de construction n'est pas à proprement parler incrémentale.

Enfin, Arleo et al. [4] proposent eux aussi un modèle d'environnement multirésolution, dans lequel seules deux directions orthogonales sont autorisées pour les arêtes. Les obstacles apparaissent ainsi comme des ensembles connexes de rectangles isothétiques (c'est-à-dire de bords parallèles aux axes du repère). La résolution de la carte pour le découpage en rectangles dépend de la résolution des moyens de perception et de la forme des obstacles. L'apprentissage de la carte utilise un parcours particulier destiné à cartographier chaque obstacle rencontré (suivi du périmètre de l'obstacle) : à chaque fois qu'un nouvel obstacle est introduit dans la carte, celle-ci est redécoupée, puis les cellules voisines sont fusionnées si besoin. La méthode de partitionnement implique donc des modifications ponctuelles du modèle et risque de ne pas être adaptée aux mises à jour très fréquentes que l'on réalise dans une approche bayésienne par exemple. De plus, une telle représentation est peu efficace dans le cas où l'environnement contient des régions de forme irrégulière et des objets dont les arêtes ne sont pas parallèles aux axes du repère orthonormal global.

Discussion

*** Avantages** :

Les grilles d'occupation permettent de représenter de grandes densités d'information et sont adaptées à des environnements de forme quelconque. Elles fournissent en outre une estimation statistique de la confiance dans les données, et certaines approches permettent même de détecter les zones conflictuelles ou les régions qui nécessitent des compléments d'observation. De plus, contrairement aux représentations composées de scans lasers bruts, elles fournissent des informations « de plein et de vide » puisqu'elles indiquent directement où sont placés les obstacles. Ainsi, elles sont souvent utilisées lorsque l'application visée repose sur la connaissance de l'espace libre, en particulier la planification de trajectoires (à partir de transformations en distance [205] ou de champs de potentiels [112] par exemple) ou, dans une optique plus réactive, l'évitement d'obstacles à grande vitesse [13]. Elles sont par ailleurs relativement aisées à interpréter par l'homme même si elles présentent parfois des zones floues et ambiguës : dans le cas bayésien notamment, on a vu qu'il est difficile de déterminer si ces zones imprécises sont dues à un manque d'information ou à la présence d'informations contradictoires.

Concernant leurs méthodes de constructions, les grilles d'occupation sont plutôt économiques en ressources de calcul : leur mise à jour s'avère rapide et facile. Il est de plus possible de les construire en ligne, sans contrainte particulière sur la trajectoire à suivre pour le robot. En revanche, les représentations hiérarchiques se prêtent manifestement moins bien aux approches incrémentales.

Enfin, les grilles d'occupation permettent une intégration aisée de divers types de capteurs : l'algorithme de fusion est immédiat. Elles sont de plus bien adaptées à des capteurs bruités (vision stéréoscopi-

que, sonars, radars...) où les primitives géométriques sont assez difficiles à extraire du fait de l'incertitude sur les données [194]. Toutefois, seules certaines approches permettent de bien prendre en compte les modèles de capteurs, comme chez Thrun [181].

*** Inconvénients** :

Le principal inconvénient des grilles d'occupation réside dans leur manque de compacité : elles sont plutôt adaptées à la représentation d'environnements encombrés mais s'avèrent particulièrement inefficaces dans les grands espaces vides. De plus, la finesse de la discrétisation étant prédéfinie, elles ne sont pas capables de s'adapter automatiquement à la densité ou à la taille des obstacles. En conséquence, si les grilles d'occupations se prêtent bien à certains algorithmes de planification, ceux-ci peuvent cependant se révéler inefficaces du fait du manque d'adaptation à l'échelle de l'environnement (en raison de la multiplication des cellules libres dans les espaces ouverts par exemple). Les approches hiérarchiques telles que les quadtrees peuvent toutefois limiter ces problèmes de compacité et d'adaptation à l'échelle.

On peut également reprocher aux représentations surfaciques le processus de positionnement complexe et coûteux qu'elles entraînent pour le véhicule (notamment par rapport à une carte de balises où les objets à comparer sont moins nombreux). De plus, elles paraissent peu appropriées, contrairement aux cartes topologiques par exemple, aux algorithmes de résolution de problèmes exprimés de manière symbolique [102].

Les algorithmes classiques de construction de grilles d'occupation présentent par ailleurs d'importantes limitations pour la cartographie d'environnements de grande taille, en particulier si ces environnements présentent des cycles. On a vu que les algorithmes de construction de grilles d'occupation bayésiennes décomposent traditionnellement le problème en un ensemble de sous-problèmes à une dimension où les cellules de la grille sont estimés indépendamment les unes des autres. Ces techniques font donc une hypothèse forte d'indépendance entre cellules voisines. Un article récent de Thrun résout partiellement le problème en faisant appel à un algorithme EM (« Expectation - Maximization ») pour trouver la carte qui maximise la vraisemblance des mesures [181]. L'estimation est réalisée dans tout l'espace des cartes possibles en maintenant les dépendances entre cellules voisines. En particulier, elle utilise un modèle direct de capteurs ($p(z \mid m)$) plutôt qu'un modèle inverse ($p(m_{x,y} \mid z)$), ce qui permet aussi de mieux prendre en compte les phénomènes physiques liés à l'acquisition de données. En contrepartie, cette approche augmente considérablement le coût de calcul puisque l'algorithme nécessite de parcourir plusieurs fois l'ensemble des données. De plus, même avec cette méthode, on ne tient pas compte du fait que l'environnement peut présenter des structures : le modèle de carte a priori ($p(m)$) suppose l'indépendance entre toutes les cellules, ce qui n'est plus exact lorsque l'environnement présente beaucoup de murs continus par exemple (si une cellule est occupée, certaines cellules voisines ont une forte probabilité a priori d'être également occupées puisque les obstacles sont souvent regroupés ou étendus sur plusieurs cellules voisines).

Enfin, l'une des principales limitations actuelles provient du fait que les incertitudes sont exprimées dans un repère très local (une cellule) : ainsi, la position du robot (et l'incertitude associée) n'intervient pas dans les équations d'estimation de la carte. Cette position est donc considérée comme exacte : à la différence des approches basées sur le filtre de Kalman (que nous détaillons dans la section suivante), on ne dispose pas d'une représentation intégrée de l'incertitude de position du véhicule et de la carte via leurs corrélations, ce qui pose problème lors des fermetures de cycles. En effet, les méthodes actuelles ne permettent pas de réestimer le degré d'occupation des cellules le long du cycle une fois qu'il est bouclé : il manque un mécanisme de rétropropagation des corrections d'erreurs [79] [194]. De plus, comme on ne dispose pas d'une estimation de l'incertitude globale sur la carte, l'espace de recherche de l'autre extrémité de la boucle (sur la partie ancienne de la carte) peut être très important lors du bouclage du

cycle, d'où une augmentation du temps de calcul. On observe que ce type de problème introduit souvent un certain flou dans la grille [8].

Des approches récentes offrent toutefois des pistes pour gérer les fermetures de cycle.

Un algorithme baptisé DP-SLAM s'inspire de la technique de FastSLAM [137] pour la transposer à une représentation basée sur une grille régulière de l'environnement [63] [64]. Le FastSLAM exploite notamment des propriétés d'indépendance conditionnelle soulignées par Murphy [142] : lorsque la trajectoire s^t du robot est connue, les observations individuelles des éléments de l'environnement deviennent indépendantes. Ainsi, l'estimation de chaque élément (ici chaque cellule de la grille) peut réellement être traitée comme un problème d'estimation indépendant. En outre, le problème de fermeture de cycle est résolu naturellement comme chez Hähnel [83] dans le cas des cartes de scans bruts. Chez Eliazar et Parr [64], chaque particule représente une hypothèse de trajectoire s^t associée à une estimation de grille d'occupation. La difficulté consiste à stocker de manière efficace ce grand nombre de grilles d'occupation (une centaine voire un millier, selon le nombre de particules utilisées). Pour cela, les auteurs définissent une structure de données efficace à base d'arbres : ces arbres représentent les instants où les trajectoires des particules ont divergé suivant le processus de rééchantillonnage. Finalement, les cartes obtenues modélisent avec une très grande précision des environnement cycliques de grande taille (environ 24m $\times$ 60m) mais nécessitent encore des temps de calcul très importants (plusieurs heures selon [64]).

Une autre piste pour la gestion des cycles revient à utiliser un réseau topologique de grilles d'occupation locales, comme chez Duckett et al. [49] : comme nous le verrons dans le chapitre consacré aux représentations hybrides, la cohérence globale de la carte peut alors être assurée par optimisation globale de réseau de contraintes métriques induit par les arêtes du graphe topologique. La représentation surfacique globale est ensuite obtenue par fusion des grilles d'occupation locales.

2.3.3 Modèles basés sur des primitives géométriques ou sur des balises

Description

Guivant et al. [79] définissent les primitives géométriques (« features » en anglais) pouvant servir de balises (on parle aussi d'*amers* suivant un terme de marine couramment employé en robotique) comme des parties distinctives de l'environnement que l'on peut facilement extraire via un type de capteur donné et qui admettent une description paramétrique. Selon Rives et Devy [113], la sélection des amers géométriques pour la localisation devrait répondre à divers critères, en particulier : pouvoir discriminant, domaine de visibilité important, stabilité, invariance et bonne adaptation à la métrologie. Ces propriétés sont cependant locales et doivent donc être associées à une région de validité. De plus, ces critères ne sont pas toujours évidents à formaliser : en pratique, toutes les primitives géométriques employées ne remplissent pas correctement ce cahier des charges.

Ainsi, on trouve dans la littérature différentes sortes de balises géométriques, auxquelles peuvent être attachées plusieurs propriétés métriques (localisation, orientation, taille, etc.) :

- **points** ou objets considérés comme ponctuels [115] [147] [120] [46] [15] : dans les cartes 2D, ils représentent notamment les objets verticaux tels que des poteaux ou des troncs d'arbres en environnement extérieur et des coins de murs en environnement intérieur ;
- **coins** : par rapport aux points, ils incluent également une information d'orientation, et éventuellement un angle relatif entre les segments adjacents à ce coin ;
- **segments**, qui correspondent généralement aux frontières d'obstacles [139] [115] [18] [75] [2] [21] [191] [15] (cf. Fig 2.5) : en général, ces segments sont représentés par leurs droites supports (de préférence en coordonnées polaires [170]) et portent éventuellement des informations sur la position géométrique de leurs extrémités ;

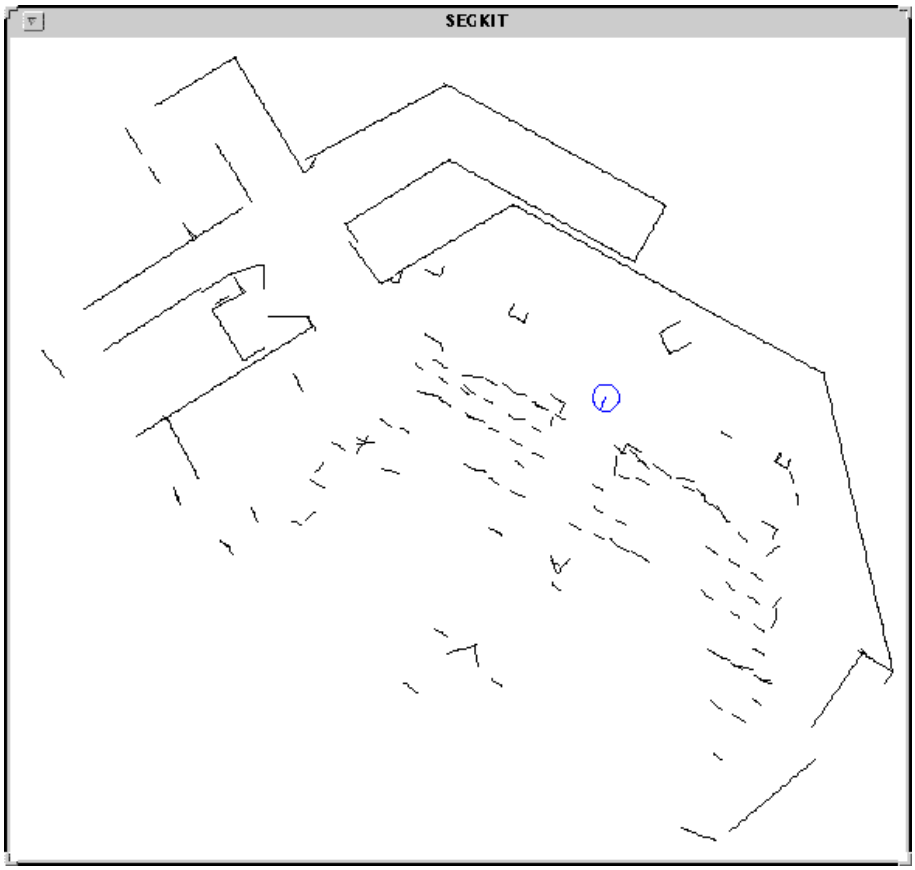


FIG. *2.5: Représentation de la salle de conférence Altaïr de la Cité de l'Espace à Toulouse selon une carte d'amers constituée de segments, construite avec les algorithmes développés par Moutarlier au LAAS [139].*

- **polygones** de forme et taille quelconques correspondant aux frontières d'obstacles dans le plan horizontal [139] et plus généralement, « polysoïdes » (polygones comportant des trous) chez [87] ;
- **portions de courbes** et notamment arcs de cercle [115] [206].

Pour maintenir une représentation cohérente de l'environnement, les cartes d'amers doivent de plus modéliser les incertitudes sur ces primitives géométriques. En général, comme nous le verrons, ces incertitudes sont représentées par des distributions gaussiennes sur les paramètres géométriques de la primitive (positions cartésiennes ou polaires, longueurs...). Toutefois, quelques travaux isolés font plutôt appel aux sous-ensembles flous [75].

Au-delà de ces formes élémentaires simples, certains travaux modélisent même de véritables **objets structurés**, qui peuvent combiner différentes primitives [42] [21]. Par exemple, Devy et Bulata [42] [18] distinguent plusieurs niveaux de représentations : un niveau géométrique où chaque primitive est décrite dans un repère local et un niveau symbolique où les balises sont constituées d'un ensemble de primitives (points et segments). Dans ces balises, les orientations des segments ne sont pas imposées et ceux-ci peuvent être reliés les uns aux autres pour former des chaînes polygonales, donnant lieu à une grande variété de représentations (cf. Fig 2.6). Les auteurs proposent en outre des mécanismes permettant de modifier la nature d'un objet particulier durant la construction de la carte, par ajout de segments ou de points. Des règles simples permettent également d'attacher à chaque balise un repère local dont la pose est estimée dans une carte globale. Cette représentation par objets offre ainsi l'avantage de réduire le nombre d'éléments à stocker dans la carte globale.

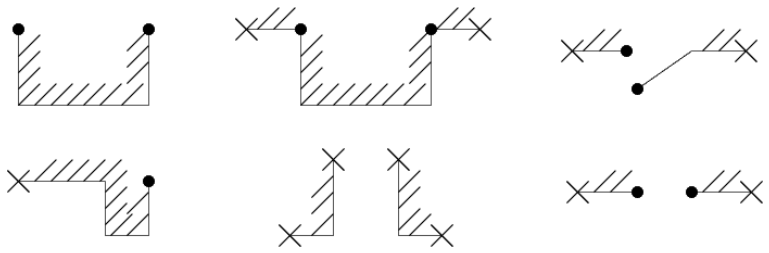


FIG. *2.6: Exemples de modèles d'objets structurés. Les sommets marqués par un point représentent des sommets référencés explicitement dans la carte tandis que les sommets marqués d'une croix représentent des extrémités libres de segments (qui n'ont pas été observées). Les hachures indiquent de quel côté se trouve la matière de l'obstacle.*

Certains auteurs proposent de mélanger différents types d'amers au sein d'une même carte : il est courant d'utiliser à la fois des points et des segments [15] [21], en ajoutant éventuellement des arcs de cercles [206] ou différents types d'objets [18] [21]. Borges définit notamment une représentation unifiée pour les amers de type point et segment [15]. Plus généralement, Castellanos et Tardos proposent le concept de SP-Map (« Symmetries and Perturbation model ») qui permet de représenter selon le même formalisme l'ensemble des balises ou objets de la carte, quelle que soit leur forme. Comme chez Bulata et Devy [18], chaque balise de la SP-Map est attachée à un repère local, ce qui permet de définir également l'incertitude de manière locale sur la position des objets. Ainsi, cette incertitude ne dépend plus de la position du repère de référence, comme c'est le cas dans les représentations polaires (r, θ) des droites par exemple, où une légère imprécision sur l'orientation d'un mur a plus d'influence sur l'incertitude du paramètre r si le mur est loin du centre du repère. En outre, les règles d'attachement du repère local tirent parti des symétries des amers.

Méthodes de construction de ces modèles

Les représentations basées sur les primitives géométriques ont été largement étudiées dans la littérature : si les approches probabilistes telles que le filtrage de Kalman sont prédominantes, il existe aussi quelques méthodes alternatives fondées sur la logique floue, voire sur des techniques ensemblistes.

*** Approches ensemblistes**

Ces méthodes ensemblistes reposent sur une hypothèse de bruits bornés : les incertitudes sont donc représentées sous la forme d'un sous-ensemble fermé, centré sur l'estimée de la mesure (par exemple en 2D : rectangle, disque, ellipse...). Elles ont donné des résultats intéressants pour la localisation des robots puisqu'elles se révèlent particulièrement robustes à la présence de fausses alarmes [97]. Toutefois, le problème de la cartographie s'avère plus complexe à traiter par ces techniques puisque les quantités à estimer se situent dans un espace de très grande dimension, ce qui risque de générer des coûts de calcul rédhibitoires selon le type de méthode ensembliste considéré. Dans sa thèse, Csorba a développé pour le SLAM un filtre appelé BOR (« BOunded Region » filter) qui modélise ainsi les incertitudes sur la position d'amers ponctuels et sur la position du robot par des rectangles de côtés parallèles aux axes du repère de référence [35]. Comme la mise à jour des estimations ne dépend plus des corrélations entre erreurs, l'algorithme est de complexité linéaire (en le nombre d'amers) et ne requiert qu'un espace mémoire réduit. Toutefois, les expérimentations sont peu probantes puique le filtre diverge sauf si l'on réduit artificiellement les bornes d'erreur. Il est possible que cette divergence soit liée au modèle d'erreur utilisé, très imprécis : les rectangles ne modélisent pas finement la forme des incertitudes, contrairement aux techniques basées sur l'analyse par intervalles [97] par exemple.

*** Logique floue**

Quelques rares publications exploitent la logique floue pour estimer la position des amers de la carte. Gasós et Martín utilisent par exemple une représentation basée sur des segments de droite flous, qui représentent les frontières polygonales des obstacles [75]. Ils ont choisi ce cadre théorique parce qu'une partie du système de navigation de leur robot est basée sur des instructions linguistiques, or la logique floue facilite la fusion de données dans le cas de descriptions linguistiques des objets. Les ensembles flous utilisés représentent l'incertitude sur la position réelle des objets. Toutefois, le degré d'appartenance d'une zone de la carte à une frontière floue n'exprime pas seulement l'incertitude sur la position de cette frontière mais aussi le degré de similarité avec les frontières situées à proximité. Ainsi, un degré d'appartenance faible (resp. élevé) à une frontière floue pour une zone indique que lorsqu'une autre frontière est trouvée dans la même zone, elle doit être considérée comme peu (resp. hautement) similaire à l'originale.

Pour construire la carte de manière incrémentale, les auteurs proposent d'abord de regrouper les mesures sonar alignées afin de construire des segments flous, en prenant en compte les diverses sources d'erreur (erreur d'estimation de la position du robot, imprécision du capteur, incertitude sur la manière dont les ultrasons sont réfléchis par l'obstacle...). Ensuite, le système repère les segments colinéaires qui s'intersectent : ces segments peuvent être issus de capteurs différents ou d'observations multiples de la même frontière d'obstacle selon des points de vue différents. Selon leur degré d'appariement, ils sont alors fusionnés en *frontières* plus longues (et non redondantes) et les incertitudes originales sont propagées dans la nouvelle représentation. Les petits segments conflictuels sont également supprimés lors d'une opération de nettoyage de carte. Les résultats expérimentaux montrent des résultats encourageants, avec des cartes nettes et de bonne qualité compte tenu de l'imprécision des proximètres à ultrasons.

*** Filtre de Kalman étendu**

La grande majorité des travaux basés sur une carte d'amers géométriques reposent sur le filtrage de Kalman et sur ses variantes. Si l'on se ramène au cadre général de l'estimation bayésienne [180] vu précédemment, l'équation 2.1 est résolue en modélisant les distributions de probabilités par des gaussiennes. Le vecteur d'état x_t contient alors la pose du robot s_t et les coordonnées cartésiennes de tous les éléments de la carte. Quant aux modèles de déplacement et d'observation, le filtre de Kalman impose qu'ils soient linéaires et perturbés par un bruit gaussien. Le modèle de déplacement de la carte répond bien à ce critère puisque la carte est immobile. En revanche, le déplacement du robot ne vérifie pas une équation linéaire du fait des rotations : il faut donc linéariser ce modèle, via un développement en série de Taylor [180] :

$$p(x_t \mid u, x_{t-1}) = Ax_{t-1} + Bu + \epsilon_{commande}$$

où A et B représentent des matrices et $\epsilon_{commande}$ un bruit gaussien, centré et de covariance connue. De même, l'équation d'observation linéarisée s'écrit :

$$p(z \mid x) = Cx + \epsilon_{mesure}$$

où C est une matrice et ϵ_{mesure} un bruit gaussien, centré et de covariance connue. Le filtre résultant est connu sous le nom de *filtre de Kalman étendu* (EKF pour « Extended Kalman Filter » en anglais).

Les bases théoriques de l'application du filtrage de Kalman au problème du SLAM ont été jetées dans les articles de Smith et Cheeseman [174] et de Moutarlier et Chatila [140], qui développent formellement le concept de cartes d'environnement stochastiques. Dès lors, on sait que l'espace mémoire requis pour le stockage de ces cartes est en $O(N^2)$ (taille de la matrice de covariance de l'état), N étant le nombre d'amers représentés dans la carte. On montre également que le coût de calcul de l'EKF est de l'ordre de $O(N^3)$ pour une implémentation naïve (coût essentiellement lié à la multiplication et à l'inversion des matrices) ou de $O(N^2)$ en tenant compte de la structure creuse des matrices de déplacement (puisque la carte est statique) et d'observation (puisque seules quelques balises sont observées à chaque pas de temps) [139] [80]. En outre, il est indispensable de maintenir l'ensemble des corrélations entre amers et entre robot et amers (via la matrice de covariance de l'état), au risque de voir le filtre diverger.

Plus récemment, les thèses de Csorba [35] et Newman [148] ont étudié la structure particulière du filtre de Kalman appliqué au problème du SLAM. Newman et Dissanayake [148] [46] démontrent en particulier trois propriétés fondamentales de ce filtre :

- la covariance de chaque objet de la carte décroît de façon monotone ;
- les estimations de balises tendent à devenir complètement corrélées ;
- leur covariance tend vers une borne inférieure liée à la covariance initiale du véhicule.

Toutefois, comme l'expliquent Castellanos et al. [22], ces propriétés ne sont plus garanties lorsque l'on passe à une version linéarisée du filtre : en effet, les linéarisations des équations d'évolution et d'observation ont tendance à rendre l'EKF trop optimiste sur les estimations d'erreur, ce qui peut mener à une divergence du filtre.

*** Améliorations du filtre de Kalman étendu**

Les techniques de cartographie basées sur le filtre de Kalman étendu souffrent d'un certain nombre de limitations :

- Comme la position du robot est modélisée par une distribution gaussienne unimodale, le système ne peut pas maintenir différentes hypothèses sur sa localisation, ce qui peut le gêner en présence

d'ambiguïtés perceptuelles. De plus, le système ne peut pas résoudre le fameux problème du « robot kidnappé », dans lequel le robot est déplacé à son insu et doit tenter de se relocaliser : les systèmes basés sur le filtrage de Kalman se contentent de réaliser un suivi en position, où l'incertitude de positionnement demeure limitée.
- Il arrive fréquemment que les différentes hypothèses d'indépendance entre les bruits du modèle ne soient pas vérifiées, ce qui peut conduire à une divergence du filtre.
- Il peut exister des biais sur les estimations de position ou d'observation : ici encore, on ne satisfait plus aux hypothèses du filtre de Kalman.
- Certaines balises peuvent correspondre à des fausses alarmes : il importe de pouvoir filtrer les amers incertains.
- Ce filtre est très sensible aux erreurs d'associations de données, qui peuvent conduire à une divergence.
- La linéarisation qui intervient dans l'EKF conduit à des estimations trop optimistes et peut également conduire à une divergence.
- Enfin, le coût de calcul quadratique devient prohibitif si le nombre de balises est trop important. En pratique, la formulation classique ne fonctionne plus au-delà d'un millier de balises environ.

Les approches qui suivent s'attachent ainsi à réduire les effets de ces limitations.

**** Gestion du problème de distribution unimodale**

Durrant-Whyte et al. introduisent des sommes de gaussiennes pour modéliser les distributions de probabilité sur le modèle de déplacement et sur les observations [58]. La carte elle-même est représentée par des sommes de gaussiennes, ce qui permet de représenter des objets complexes en environnement extérieur notamment. L'approche est intéressante mais il n'est pas certain qu'elle possède les mêmes propriétés de convergence que le filtre de Kalman classique [46]. De plus, elle paraît plutôt adaptée aux environnements non structurés.

Une autre piste pour résoudre ce problème réside dans l'utilisation d'un filtre de Kalman multi-hypothèses pour suivre la position du robot. On verra que la technique de FastSLAM, qui combine le filtrage de Kalman (pour l'estimation de la carte) au filtrage particulaire (pour l'estimation de position du robot), offre également une solution intéressante.

**** Prise en compte des corrélations entre bruits**

Moutarlier a développé par exemple un filtre plus général que le filtre de Kalman classique puisqu'il gère directement les corrélations entre bruit d'état et bruit de mesure [140] [139]. Il fonctionne également avec des bruits colorés sur l'état ou sur l'observation. On se ramène toutefois à la formulation exacte du filtre de Kalman étendu classique si l'on fait l'hypothèse d'indépendance des bruits d'état et de mesure, ainsi que l'hypothèse de bruit blanc.

**** Gestion des biais**

Par ailleurs, Moutarlier montre que ce type de filtre est très sensible à la présence de biais. Or selon lui, ces biais sont inévitables du fait des linéarisations et de l'imperfection des modèles de capteurs. Ainsi, il propose une méthode sous-optimale de recalage-fusion (inspirée d'Ayache [7]) pour limiter leur influence. Dans cette approche, l'information d'observation est d'abord utilisée pour relocaliser le robot avant de corriger les positions des amers à partir des observations. Ainsi, on ne prend pas en compte l'information fournie par l'erreur d'estimation sur la position du robot pour corriger la position des amers.

**** Gestion des ambiguïtés d'extraction de balises et d'associations de données**

L'opération d'extraction de balises nécessite une interprétation précoce des données observées, alors que celle-ci peut être reportée à un traitement a posteriori dans les représentations fondées sur l'apparence. Il existe donc des risques d'erreurs et de fausses alarmes lors de la structuration des observations locales en primitives géométriques et en balises. Pour y remédier, Leonard et Durrant-Whyte ajoutent à chaque balise un paramètre de confiance qui est incrémenté lorsque celle-ci est bien détectée depuis différentes positions. Seules les balises très fiables sont utilisées pour la mise à jour de la position du robot [115].

Léonard et al. ont aussi proposé plus récemment un nouveau cadre de travail stochastique pour effectuer des décisions retardées sur la mise en correspondance [118] [117]. Lorsque le système observe un amer, trois décisions distinctes sont en effet possibles : appariement avec un amer existant, création d'un nouvel amer ou élimination de la mesure considérée comme une fausse alarme. L'objectif des auteurs était de gérer de manière efficace (d'un point de vue calculatoire) différentes hypothèses sur l'origine d'une mesure. En particulier, l'initialisation d'un amer doit pouvoir utiliser des observations provenant de plusieurs points de vue. Pour cela, ils étendent le vecteur d'état afin d'autoriser la prise en compte des corrélations temporelles : ils ajoutent donc dans l'état les positions successives du robot (qui sont appelées « états de trajectoire »). Le coût de calcul s'en trouve augmenté mais il est possible de supprimer d'anciens états de trajectoires (et les covariances associées) lorsque toutes les mesures effectuées à un pas de temps donné ont été traitées. Les auteurs dressent un parallèle avec le « fixed-lag Kalman smoother » où les états dont l'âge dépasse un certain seuil sont automatiquement supprimés : dans leur approche cependant, la suppression ne se fait pas forcément dans l'ordre chronologique et dépend du fait que les informations d'observation ont été utilisées ou non. Le système permet ensuite de définir une fonction d'initialisation d'amer impliquant des données issues de différents pas de temps, au lieu d'un seul dans l'approche classique. Cette stratégie permet également d'attendre que les matrices jacobiennes de la fonction d'initialisation indiquent que l'estimée du nouvel amer est bien conditionnée. On peut même définir un nouveau point de vue à rallier par le robot, de manière à réduire les valeurs de ces jacobiennes pour réaliser une initialisation plus stable. Enfin, une fois le nouvel amer initialisé, il est possible de raffiner la carte en utilisant toutes les mesures passées qui peuvent lui être associées, à travers un processus de mise à jour a posteriori. Cela permet non seulement d'extraire l'information maximale des mesures passées mais aussi de créer des objets composites à partir de plusieurs primitives.

Par ailleurs, pour éviter les erreurs d'association de données, certains auteurs tels que Neira et Tardos ont proposé des méthodes d'appariement globales et robustes telles que le « joint compatibility test » [145]. Nous y reviendrons plus en détail au chapitre 4.

**** Gestion des problèmes de linéarisation**

Comme le souligne Knight [98], si l'on peut prouver théoriquement que le filtre de Kalman fournit la meilleure estimée a posteriori de la carte en présence de bruits gaussiens, il n'en va pas de même pour le filtre de Kalman étendu : l'EKF n'est qu'une approximation. Même si le temps est discrétisé très finement (avec des pas de temps de durée négligeable), la linéarisation produit des résultats incorrects simplement parce que les estimations sont calculées autour d'estimations d'états erronées (sauf si le bruit est nul, bien entendu : dans ce cas, les estimations d'état sont parfaites et correspondent exactement à la réalité). Tardos et al. expliquent que l'EKF n'est valide que dans le cas du robot à une dimension (auquel cas le problème devient linéaire) et pour des cartes de petite taille (typiquement de dimension inférieure à la centaine de mètres) [176]. En réalité, le problème de linéarisation apparaît avant le problème du coût de calcul quadratique. Julier et Uhlman ont eux aussi montré expérimentalement les problèmes de divergence de l'EKF classique sur des représentations constituées de balises ponctuelles.

Ainsi, Lionis et Kyriapopoulos tentent de remédier au problème en utilisant deux filtres de Kalman

distincts sur une représentation à base de segments de droite [123]. Le premier filtre estime l'orientation du robot ainsi que les orientations absolues des segments : il n'est pas nécessaire de recourir à l'EKF pour ce premier estimateur puisque le problème est linéaire. Quant au second filtre, il exploite les résultats du premier pour estimer les positions du robot et des segments de la carte. Les résultats expérimentaux sont encourageants mais l'approche est théoriquement sous-optimale car elle suppose que les incertitudes sur la translation du robot sont indépendantes des erreurs de rotation. Des hypothèses d'indépendance similaires sont appliquées aux balises. Ainsi, une validation plus poussée s'avère nécessaire.

Castellanos et al. proposent pour leur part une cartographie dite « robocentrique » [22]. Cette méthode consiste à construire la carte relativement au robot c'est-à-dire directement dans le repère du robot, qui évolue, plutôt que dans le repère de référence fixe. Ainsi, le vecteur d'état contient la position $\hat{x}^R_{F_i}$ des balises F_i ainsi que la position $\hat{x}^R_W$ du repère de référence W par rapport à la position du robot R. A l'instant k, suite au déplacement du robot, la carte entière devrait être mise à jour par composition avec ce déplacement $\hat{x}^{R_{k-1}}_{R_k}$, de manière à ce que les estimations soient définies par rapport à la nouvelle position $\hat{x}^{R_k}_W$ de la plate-forme. Les nouvelles estimées de covariance d'une balise F seraient elles aussi mises à jour par linéarisation autour des valeurs $\hat{x}^{R_{k-1}}_{R_k}$ et $\hat{x}^{R_{k-1}}_{F_{k-1}}$. Toutefois, avant d'effectuer cette opération, les auteurs préfèrent réduire les erreurs de linéarisation en améliorant l'estimation de déplacement calculée par l'odométrie. C'est pourquoi le vecteur d'état est augmenté du déplacement $\hat{x}^{R_{k-1}}_{R_k}$ du robot, traité comme une balise indépendante. Ensuite, l'ensemble du vecteur d'état est mis à jour via un EKF classique, en utilisant l'observation courante, puis chaque élément du vecteur d'état est composé avec la nouvelle estimation de déplacement du robot. De cette manière, on limite les erreurs de linéarisation car l'incertitude des amers observés reste faible (sauf lors d'une fermeture de cycle), contrairement au cas classique où elle a tendance à augmenter au fur et à mesure du déplacement du robot, jusqu'à ce que celui-ci revienne à un endroit déjà visité. Les tests en simulation montrent que cette technique améliore effectivement l'estimation de covariance (qui ne pêche plus par excès d'optimisme) mais il semble que la méthode souffre de biais.

Enfin, Julier et Uhlman ont eux aussi proposé une solution partielle au problème de linéarisation, via l'utilisation d'une variante du filtre de Kalman appelée « unscented Kalman filter » ou UKF [93]. L'UKF utilise une forme de filtrage particulaire : la distribution gaussienne est représentée par un ensemble de *points sigma* dans l'espace des paramètres. Ces points sigma sont réestimés via une fonction adéquate à partir des observations et du déplacement du robot et une nouvelle distribution gaussienne est estimée à partir de ces nouvelles estimations. En théorie, cette technique transfère l'erreur au troisième ordre (et même au quatrième dans certains cas), sans augmentation du coût de calcul (en général, ce coût est même réduit).

**** Réduction des coûts de calcul**

Comme nous allons le voir, les efforts destinés à limiter les coûts de calcul peuvent recourir à diverses stratégies, optimales (au sens où elles conduisent à des estimations identiques à l'EKF classique) ou non :

- ignorer ou limiter les corrélations ;
- simplifier la carte (en limitant le nombre de balises) ;
- travailler momentanément sur une région limitée de la carte.

Ainsi, une première approche pour réduire les coûts de calcul consiste à **ignorer les corrélations entre objets ou** à **limiter leur prise en compte**.

Dans ce cadre, Julier et Uhlmann [94] proposent une technique fondée sur l'intersection de covariance (« Covariance intersection » ou CI en anglais), qui généralise le filtrage de Kalman : la règle de mise à jour

utilisée est en théorie optimale lorsqu'aucune information sur les corrélations n'est maintenue (lorsque le système n'a pas de mémoire en quelque sorte). Le principe est le suivant : lorsque l'on fusionne l'estimée d'incertitude sur une balise de la carte avec l'incertitude d'une nouvelle mesure de cette balise, le résultat tombe nécessairement dans l'intersection des deux covariances. On est donc assuré que toute règle de mise à jour de l'incertitude contenant l'intersection des covariances est valide.

Ainsi, étant données deux estimations de moyenne et de covariance $\{a, A\}$ et $\{b, B\}$, la mise à jour $\{c, C\}$ par CI est la suivante :

$$C = (\omega A^{-1} + (1-\omega)B^{-1})^{-1}$$
$$c = C(\omega A^{-1}a + (1-\omega)B^{-1}b)$$

où ω est compris entre 0 et 1 et peut être choisi de manière à minimiser une mesure donnée de l'incertitude et liée à la matrice de covariance résultante C : déterminant ou trace par exemple. Intuitivement, ω permet de pénaliser l'estimation, suite au manque d'information sur les corrélations [95]. Julier et Uhlmann vérifient la validité de cette règle de mise à jour en démontrant que la matrice $C - E\left[\tilde{c}\tilde{c}^T\right]$ est semi-définie positive pour toute valeur de covariance entre les estimées a et b, $\tilde{c}$ correspondant à l'erreur d'estimation : le résultat tombe bien dans l'intersection des deux covariances (l'estimation est bien « conservative »).

On se ramène donc à un ensemble de $n+1$ filtres indépendants : un pour le robot et un pour chacune des n balises de la carte. Cette technique s'avère toutefois bien trop prudente et pessimiste : la convergence est considérablement plus lente qu'avec un filtre de Kalman. C'est pourquoi certaines études ont cherché à la combiner avec d'autres techniques de SLAM : avec des cartes relatives par exemple [91]. Julier et Uhlmann ont également proposé une extension, appelée « Split Covariance Intersection » (SCI), qui permet d'introduire une information partielle sur les corrélations [95]. Cette méthode décompose l'erreur de prédiction en une partie corrélée (dans le cas du SLAM, elle provient des corrélations entre les positions de balises et la position du robot, dont la structure n'est pas maintenue) et une partie indépendante (qui provient des hypothèses de bruit blanc pour l'évolution du système et les observations). Intuitivement, cet algorithme revient à appliquer la pénalité ω uniquement à la partie corrélée. Toutefois, dans ces méthodes, le gain en terme de coût de calcul est obtenu au détriment de la précision : ces approches fournissent encore des estimations sous-optimales ou pessimistes.

Csorba et Newman introduisent pour leur part les filtres relatifs [35] [148]. Dans ce cadre, les cartes sont formulées en termes de positions relatives entre balises (distances ou déplacements) plutôt qu'en positions absolues dans un repère de référence. L'intérêt de ces représentations réside dans leur structure, qui permet de s'abstraire des corrélations : en effet, leur matrice de covariance reste diagonale par blocs. Ainsi, la mise à jour est réalisée en temps constant. En outre, une telle approche aide à limiter les problèmes de biais sur les mesures odométriques puisque l'on découple le mouvement du robot et l'estimation des positions de balises [131]. Par ailleurs, la forme de la carte relative se prête bien à des algorithmes d'association de données exploitant les graphes de correspondance [159].

Ces gains sont toutefois obtenus au prix de plusieurs limitations. D'abord, cette carte ne peut pas être directement employée pour mettre à jour la position du véhicule : en effet, cette position n'est généralement pas incorporée dans le vecteur d'état car elle complique le modèle de transition et ne permettrait plus de conserver une matrice creuse pour la covariance de l'état (du fait de la structure des covariances du SLAM, qui proviennent des corrélations entre balises et véhicule). De plus, comme les cartes relatives n'expriment pas les balises dans un référentiel absolu, il est nécessaire d'utiliser des opérateurs de projection, qui ne garantissent pas l'unicité de la carte selon le chemin pris à travers les états pour réaliser cette transformation. Newman montre cependant qu'il est possible d'appliquer des contraintes de cohérence à la carte relative quand une carte absolue est requise, afin d'assurer cette unicité [148]. Toutefois, ces contraintes ont une complexité de l'ordre de N_c^3, où N_c est le nombre de contraintes indépendantes. L'application de ces contraintes risque cependant d'être fréquente si le véhicule a besoin de se localiser. Newman suggère donc de limiter cette quantité N_c en recherchant des boucles aussi com-

plexes que possible dans la carte, mais cette recherche risque elle-même d'être coûteuse. En outre, la mise en correspondance des observations avec la carte n'est pas aisée, puisque l'on ne dispose ni de la position du véhicule, ni de la carte absolue de la zone locale (à moins d'appliquer les opérateurs de projection avec les contraintes de cohérence). Enfin, si l'on considère une carte absolue comme une carte relative géante où toutes les positions relatives entre balises sont maintenues, une véritable carte relative risque de converger un peu moins vite, puisqu'elle ne stocke qu'un nombre réduit de relations [99]. Elle peut même présenter des problème de stabilité du fait de l'absence des corrélations induites par le véhicule : certaines parties éloignées de la carte risquent de s'en trouver mal connectées [99].

Enfin, des extensions récentes de ces travaux donné lieu à des reformulations complètes du problème de SLAM sous forme de « SLAM découplé » (ou « D-SLAM »), pour lesquelles la cartographie et la localisation du robot sont traités comme des processus indépendants : un problème d'estimation statique non linéaire pour la cartographie et un problème d'estimation dynamique de faible dimension pour la localisation [195]. Pour maintenir une estimation en temps constant, différentes stratégies peuvent être considérées. Chez Martinelli et al. par exemple [131], la matrice de covariance de la carte est maintenue creuse via l'utilisation d'un vecteur d'état contenant des éléments redondants et pour lesquels toutes les relations ne sont pas explicitées. Toutefois, l'application de contraintes de cohérence (qui explicitent les relations entre éléments, pour obtenir des cartes absolues à grande échelle notamment) se fait au détriment de la complexité de la mise à jour de la carte, puisque la structure creuse de la matrice de covariance est détruite. Chez Wang et al. [195], des hypothèses particulières sur l'observation (par exemple, on suppose qu'à chaque aquisition des données perceptuelles, le robot peut reconnaître au moins deux balises) permettent de maintenir la matrice d'information exactement creuse. Le modèle de carte correspond alors soit à une carte relative compacte, soit à une carte absolue, et conserve les corrélations entre estimations de position de balises. En contrepartie, il semble que la carte converge un peu moins vite puisque comme dans les approches précédentes concernant les filtres relatifs, le processus de construction n'exploite pas les informations de positionnement du robot.

Plutôt que de changer la structure de la carte ou la nature du filtre, une autre idée consiste à **négliger la mise à jour des éléments presque sûrs ou faiblement corrélés aux balises observées** (en particulier les objets lointains) dans un filtre de Kalman classique.

Guivant et Nebot remarquent que les calculs les plus importants de l'EKF portent sur la mise à jour de la covariance de l'erreur d'estimation sur l'état [80]. Pour réduire ce coût de calcul, ils proposent une approche sous-optimale qui consiste à ne prendre en compte qu'un sous-ensemble des balises de la carte. Le vecteur d'état X est donc divisé en deux groupes : un groupe X_P d'éléments préservés et un groupe X_D d'éléments non pris en compte. Pour sélectionner les éléments qui ne seront pas considérés dans la mise à jour, le système utilise deux critères : soit la valeur de leur covariance dans l'état se trouve en-dessous d'un certain seuil (c'est-à-dire que leur position est pratiquement sûre), soit leur contribution à la mise à jour de la covariance de l'état est négligeable par rapport à la covariance courante (on ne regarde que la diagonale de la matrice de mise à jour). La matrice de covariance s'écrit ensuite :

$$P = \begin{bmatrix} P_{PP} & P_{PD} \\ P_{DP} & P_{DD} \end{bmatrix}$$

Il est alors possible de générer des estimées cohérentes (« conservatives ») en mettant à jour les états X_D mais pas la covariance P_{DD}. Tant que la taille du bloc P_{DD} reste bornée, la complexité résultante est linéaire en le nombre global de balises dans l'état X. Les auteurs proposent également une méthode de structuration de la carte en *constellations* qui facilite le découpage du vecteur d'état. Chaque constellation est basée sur deux balises qui spécifient un repère local dans lequel sont définies les autres balises de la

constellation. Cette représentation permet de limiter les corrélations entre balises de régions différentes et d'assurer que l'algorithme reste proche de la méthode optimale. Comme nous le verrons plus loin, en réalité, ce type de carte introduit implicitement des éléments topologiques dans la représentation métrique, puisque chaque constellation peut être vue comme un sommet d'un graphe dont les arêtes traduisent des liens de corrélation entre repères locaux (les éléments de base des constellations).

Julier a proposé une méthode équivalente nommée « Sparse Weight Filter », de complexité linéaire également, qui force la matrice de gain de Kalman à une valeur nulle pour toutes les balises sauf le véhicule et un petit groupe d'éléments de la carte [92]. La matrice de gain de remplacement est obtenue en temps borné en minimisant la trace de la nouvelle matrice de covariance de l'état. En réalité, Julier montre que cette technique est équivalente à la méthode de Guivant et Nebot [80] si la partition de la carte (en éléments conservés et en éléments négligés) est identique.

Enfin, Knight [98] suggère de réaliser un « découplage partiel » en ne mettant à jour que le bloc P_{PP} dans la matrice de covariance de Guivant et Nebot. L'algorithme sous-optimal résultant est en temps constant si le nombre d'éléments du groupe X_P reste borné. Toutefois, pour éviter la divergence du filtre, quelques modifications *ad hoc* ont été apportées pour gérer la sortie d'un élément hors du groupe X_P et sa rentrée ultérieure dans ce même groupe. D'après les évaluations empiriques, cette méthode fournit une précision métrique et permet de fermer des cycles de l'environnement sur une échelle qui dépend du nombre d'éléments maintenus dans X_P.

Au lieu de travailler directement sur la matrice de covariance de l'état, Thrun et al. exploitent le fait que son inverse, la matrice d'information, est naturellement creuse (du moins dans sa forme normalisée) : on constate empiriquement que la plupart de ses éléments ont des valeurs très basses, ce qui illustre des liens faibles entre les balises correspondantes [183] [124]. En effet, comme l'expliquent les auteurs, si toutes les balises deviennent complètement corrélées à la limite de convergence [46], ces corrélations proviennent essentiellement d'un réseau limité de contraintes locales connectant des balises proches. Thrun et al. proposent donc de passer à la forme équivalente de l'EKF, basée sur la matrice d'information : il s'agit du filtre d'information étendu (« Extended Information Filter » ou EIF en anglais). Ils en développent ensuite une forme approchée (SEIF pour « Sparse Extended Information Filter ») où la matrice d'information est maintenue artificiellement creuse par application d'une opération de nettoyage adéquate. Dans le cas linéaire, la mise à jour du SEIF peut être réalisée en temps constant (si le nombre de coefficients non nuls reste borné). Toutefois, dans le cas non linéaire qui nous intéresse, des opérations supplémentaires et des approximations (descente de gradients notamment) s'avèrent nécessaires pour conserver cette mise à jour en temps constant : si les premiers résultats sont prometteurs, des expérimentations plus poussées sont donc indispensables pour valider l'approche. Ici encore, comme nous le préciserons plus loin, on peut déceler une composante topologique dans la représentation employée : en effet, l'opération de nettoyage utilisée permet de maintenir un réseau dynamique de balises voisines.

Plus généralement, d'autres approches que nous développerons dans le chapitre suivant s'appuient explicitement sur un découpage de la carte, selon une partition [116] [28] ou un réseau de cartes locales [67] : nous verrons que ces approches permettent à la fois de limiter les temps de calcul et, dans certains cas, de réduire les problèmes de linéarisation.

Une seconde piste d'accélération réside dans la **limitation de la taille du vecteur d'état, via la suppression de balises** notamment [57] : si la carte peut s'en trouver considérablement réduite, la localisation n'en est guère affectée. Pour justifier cette approche dans le cadre du SLAM, Knight remarque qu'en général, les animaux n'ont pas besoin d'une carte détaillée de l'ensemble de l'environnement [98] : il leur suffit de naviguer localement et peu leur importe ce qui se trouve au-delà d'un voisinage proche.

Ainsi, Dissanayake et al. proposent de supprimer les balises qui ne sont pas visibles depuis le lieu courant, à moins qu'elles ne présentent un contenu informationnel important [45]. Plus précisément, Durrant-Whyte et al. soulignent que la vitesse de convergence du processus de cartographie et de localisation du robot est gouverné par les balises de meilleure qualité, c'est-à-dire celles qui fournissent le plus d'information au sens de Fisher [57]. De plus, il s'agit souvent d'un petit sous-ensemble des balises disponibles. Il est donc possible, en général, de trouver quelles balises ont le plus d'influence sur la convergence et de supprimer les autres éléments de la carte. Knight [98] remarque que Davison utilisait un principe similaire dans son système de SLAM basé sur la stéréovision : pour limiter les temps de calculs liés à l'observation stéréoscopique, il choisissait les balises à observer en fonction des covariances sur l'innovation pour celles qui étaient visibles [40].

Enfin, une dernière piste d'accélération consiste à **travailler momentanément sur une partie limitée de la carte**, en retardant la mise à jour coûteuse de la carte globale complète.

Davison propose ainsi dans sa thèse une méthode dite de « postponement » (report temporel), qui revient à réaliser un compromis entre le temps de calcul et la place mémoire requise pour le stockage des données [40]. Pour cela, il réarrange les équations de l'EKF, ce qui lui permet de ne traiter qu'une partie bornée de l'état et de reporter la mise à jour du reste, en enregistrant à la place un ensemble de taille bornée de données de report temporel. Ces données sont utilisées pour mettre à jour la carte complète dès qu'une puissance de calcul suffisante est disponible. Chez Davison, la méthode est limitée à l'observation d'une balise unique à chaque itération. Chez Knight et al. [99], l'approche est étendue à un groupe de balises qui peut augmenter (une sous-carte) sans nécessiter une mise à jour complète.

Une idée similaire est proposée par Williams et al. à travers le « Constrained Local Sub-map filter » [200]. Dans ce système, le robot initialise régulièrement une carte locale de l'environnement et durant un certain laps de temps, il travaille exclusivement sur cette sous-carte, sans mettre à jour les amers de la carte globale. La position x_L du repère local est enregistrée dans le vecteur d'état global : il s'agit de la position du robot au moment de la création de la carte locale. Ainsi, pour fusionner la carte locale dans la représentation globale, le système procède en deux temps. D'abord, les amers locaux sont ajoutés dans le vecteur d'état global et dans la matrice de covariance associée par composition de x_L et de leur position locale. Ensuite, des contraintes sont appliquées pour fusionner les amers communs entre les cartes locale et globale. Le gain de temps dépend de la périodicité des créations de cartes locales : il correspond approximativement au ratio du nombre d'observations par le nombre de contraintes à appliquer. Les résultats obtenus sont en outre très proches du filtre de Kalman global. Les auteurs indiquent également que cet algorithme améliore l'association de données. En effet, lors de la construction de la carte locale, les éléments candidats à l'appariement sont peu nombreux. Ensuite, lors de la fusion avec la carte globale, le système peut exploiter une sous-carte structurée plutôt qu'une simple observation, ce qui limite les ambiguïtés.

Guivant et Nebot implémentent cette idée de report temporel à travers le « Compressed Extended Kalman Filter » [80]. Là encore, le système travaille momentanément sur une petite partie de la carte en ne mettant à jour que les N_a balises locales. Ainsi, tant que le robot opère sur cette zone locale, l'information peut être maintenue avec une complexité constante de l'ordre de $O(N_a^2)$. Les auteurs montrent que cette information locale peut ensuite être transférée ponctuellement, en une seule itération et sans perte d'information, à l'ensemble de la carte, au coût classique du SLAM complet $O(N^2)$ (N étant le nombre total d'amers). La démonstration implique un découpage de l'état X en une partie X_A qui correspond à la carte locale et une partie X_B qui correspond au reste de la carte globale : l'application de l'EKF sur ces différents blocs (en considérant uniquement des observations d'amers de X_A) conduit à une formulation

du filtre permettant la mise à jour ultérieure de X_B. Ainsi, la méthode proposée est optimale, au sens où elle est équivalente à l'EKF global classique. En pratique, pour définir les zones de travail locales, les auteurs proposent un partitionnement régulier de la carte en parties rectangulaires et exploitent les informations de voisinage entre zones : on se rapproche donc encore une fois des combinaisons de cartes métriques et topologiques.

Pour finir, Tardos et al. proposent pour leur part une méthode séquentielle de jointure de cartes (« sequential map-joining »), qui s'appuie sur une séquence de cartes locales [176]. Périodiquement, au fur et à mesure de sa progression, le robot commence une nouvelle carte locale, relative à sa position courante. Tant que cette carte est active (tant que le robot n'a pas défini de nouvelle carte), un EKF classique est appliqué sur cette carte locale pour la mettre à jour. Aucune information ne doit être partagée par deux cartes locales distinctes A et B : elles sont totalement décorrélées (même si elles contiennent sans le savoir des balises communes). Ensuite, si un élément B_j de la carte B est reconnu comme identique à un élément A_i de la carte A, une opération de jointure de cartes (« map joining ») peut être appliquée pour fusionner ces deux cartes locales indépendantes. Cette opération se déroule en deux étapes : d'abord, on déplace la base (le référentiel) de la carte B vers l'élément commun B_j, puis on fusionne les informations des deux cartes en une carte unique. Ensuite, les balises communes peuvent être aisément fusionnées par l'application de contraintes d'égalité. La covariance de la carte globale résultante s'avère correcte et le biais reste très faible : à travers cette approche (optimale par rapport à l'EKF classique), on a tendance à limiter les problèmes de linéarisation. De plus, le gain en terme de temps de calcul est de l'ordre d'un facteur 50 sur l'expérimentation réalisée (ce gain correspond au ratio $(k-1)/(q-1)$ où k est le nombre moyen d'observations d'une balise au cours de l'expérimentation et q le nombre moyen de cartes locales incluant cette balise).

*** Filtrage particulaire « rao-blackwellisé »**

Le filtrage particulaire n'est guère adapté aux espaces d'état de grande dimension car il requiert un trop grand nombre d'échantillons pour représenter leur probabilité a posteriori. C'est pourquoi certains chercheurs ont eu l'idée de partitionner l'espace d'état X en deux sous-espaces X_1 et X_2 de telle manière que $p(X_1 \mid X_2, z^t, u^t)$ puisse être mise à jour de manière analytique et efficace tandis que $p(X_2 \mid z^t, u^t)$ est mis à jour par filtrage particulaire (ne nécessitant qu'un nombre de particules réduit) [48]. Ainsi, dans cette méthode dite de filtrage particulaire *rao-blackwellisé*, la probabilité a posteriori sur l'état se calcule suivant la règle de Bayes :

$$p(X \mid z^t, u^t) = p(X_1 \mid X_2, z^t, u^t) p(X_2 \mid z^t, u^t)$$

en utilisant les notations introduites plus haut (z^t et u^t représentent respectivement l'ensemble des observations et des commandes jusqu'à l'instant t).

La technique de FastSLAM peut être considérée comme une instance de filtrage particulaire rao-blackwellisé [137]. Elle décompose le problème du SLAM en un problème de localisation du robot (X_2 correspond à la trajectoire s^t du véhicule, traitée par filtrage particulaire) et en un problème d'estimation de position de balises, conditionnée par l'estimation de position du robot (X_1 correspond à la carte m, traitée de manière analytique par filtrage de Kalman étendu). Il se trouve que dans le cas du SLAM, on peut exploiter des propriétés d'indépendance conditionnelle : si les positions de robot sont connues, les observations individuelles des amers deviennent indépendantes, ce qui permet de découpler l'estimation de position des N balises en N problèmes d'estimation indépendants. Ainsi, à chaque pas de temps, la mise à jour des estimations se déroule en trois étapes :

- pour chaque particule, tirage au sort (échantillonnage) de sa position d'après le modèle de déplacement du robot ;
- réestimation de la carte à partir des observations ;
- calcul de pondérations sur les particules (« facteurs d'importance ») en utilisant l'observation et rééchantillonnage des particules suivant ces pondérations, de manière à ce que les nouvelles particules représentent bien la nouvelle probabilité a posteriori sur la position du robot.

Une implémentation naïve de cette stratégie conduit à un algorithme en $O(MN)$ où M représente le nombre de particules employées. En fait, Montemerlo et al. proposent une structure de données efficace basée sur des arbres qui réduit le temps de calcul à $O(M \log N)$. Outre ce coût de calcul réduit, le FastSLAM offre un grand intérêt dans le cadre de la problématique d'association de données : en effet, il permet de maintenir plusieurs hypothèses sur la mise en correspondance des observations avec la carte, en attachant à chaque particule sa propre hypothèse de carte. Les expérimentations montrent que cette approche permet de traiter des cartes contenant un très grand nombre d'amers (de l'ordre de 50000) en utilisant une centaine de particules. Cependant, à l'heure actuelle, on ne sait pas encore déterminer le nombre de particules nécessaire pour assurer la convergence du filtre et il est à craindre que ce nombre soit exponentiel en la taille de la carte [136].

Ainsi, Montemerlo et al. ont proposé une seconde version de cet algorithme, intitulée FastSLAM 2.0 [136]. Cette nouvelle stratégie vise notamment à remédier au problème de déplétion bien connu en filtrage particulaire : si le bruit de déplacement est élevé par rapport au bruit de mesure, les particules ont tendance à tomber dans des zones où la probabilité de l'observation est basse. Pour cela, l'échantillonnage de la pose dans la première étape de l'algorithme ne dépend plus uniquement du modèle de déplacement, mais aussi de la mesure. Comme dans le cas du SLAM traité par filtre de Klaman classique, un théorême de convergence peut être appliqué au FastSLAM dans le cas linéaire (mais comme pour l'EKF, il n'existe pas de théorème dans le cas non linéaire qui nous intéresse) : le FastSLAM avec $M = 1$ particule(s) converge vers la carte correcte si tous les amers sont observés infiniment souvent, et si la position de l'un des amers est connue à l'avance. Les résultats expérimentaux concernant le FastSLAM 2.0 sont très prometteurs, même lorsque le nombre de particules utilisées est particulièrement bas (y compris dans le cas où $M = 1$). De plus, la carte est construite en temps réel. Toutefois, comme l'indiquent les auteurs, l'utilisation de plusieurs particules se révèle nécessaire si l'association de données est ambiguë, mais on ne sait pas encore définir la quantité de particules adéquate.

Discussion

*** Avantages** :

Cette représentation par amers offre un cadre de travail élégant pour résoudre le problème du SLAM, à travers les approches basées sur le filtre de Kalman étendu (EKF) où la fermeture de boucles est gérée de manière transparente via la matrice de covariance. Bailey signale des faiblesses liées à l'association de données, qui est souvent réalisée de manière locale et « gloutonne » par des méthodes de plus proche voisin notamment [8]. Des progrès ont toutefois été faits dans ce domaine, en particulier avec le test de « joint compatibility » de Neira et Tardos [145]. En outre, le nombre de balises étant limité, les mises en correspondance sont souvent moins coûteuses que dans une grille d'occupation par exemple. Bailey indique aussi que l'EKF offre un cadre efficace pour la fusion de données multicapteurs et permet de prendre en compte explicitement les incertitudes des modèles de capteurs [8]. De plus, comme nous l'avons vu, des approches récentes permettent de limiter les inconvénients de la linéarisation et de la complexité quadratique du filtre classique : il devient possible de cartographier des environnements comportant des boucles de plusieurs centaines de mètres.

Par ailleurs, les cartes d'amers offrent une description plus compacte des larges espaces ouverts que les grilles d'occupation notamment, même si elles sont moins adaptées aux environnements encombrés de petits objets, où les balises risquent de se multiplier. Par rapport aux approches basées sur l'apparence, la décomposition des observations brutes selon des primitives géométriques permet également de comprimer l'information [194]. De plus, les cartes d'amers peuvent en principe atteindre une meilleure précision que les grilles d'occupation car les objets sont localisés selon des coordonnées en nombres flottants alors que dans les grilles, l'espace est discrétisé.

Enfin, le filtrage de Kalman autorise une cartographie passive et en ligne dans le cas des améliorations décrites précédemment.

*** Inconvénients** :

Les représentations fondées sur des balises sont souvent peu denses et contrairement aux grilles d'occupation, elles ne contiennent pas les informations de « plein et de vide » nécessaires à la planification de trajectoires. En effet, les amers ponctuels renseignent peu sur la position des obstacles continus de grande dimension et les balises de type « segment » ne sont généralement pas jointives : le contour des obstacles est rarement continu et fermé, ce qui génère des ambiguïtés (par exemple, on peut se demander si l'on a observé une ouverture dans l'interstice entre deux murs non jointifs). De plus, on constate fréquemment des incohérences dans la carte, comme l'illustre la figure 2.5 pour une représentation à base de segments : segments redondants suite à un appariement manqué, intersections aberrantes entre frontières d'obstacles, etc. Selon Thrun [180], les représentations à base d'objets sont plus faciles à appréhender par les hommes que les grilles d'occupations. On peut en effet songer au format répandu des plans d'architectes, qui reposent sur l'utilisation de formes géométriques simples. En revanche, les cartes constituées de primitives géométriques trop basiques (amers ponctuels notamment) informent peu sur la nature de l'environnement.

La généricité par rapport à l'environnement n'est pas acquise pour les cartes constituées de primitives géométriques. Ce type de représentation est relativement bien adapté aux environnements structurés tels que l'intérieur des bâtiments, où des structures artificielles permettent de définir et d'extraire plus aisément les balises. Toutefois, ces représentations sont souvent limitées à des régions pouvant être décrites sous forme d'éléments géométriques simples [180]. La liste de formes et d'objets possibles risque alors de s'avérer incomplète : en particulier, les approximations polygonales sont mal adaptées aux environnements présentant des courbes. Pour y remédier, Thrun suggère par exemple de permettre au système d'apprendre des modèles d'objets ou de mettre en œuvre des cartes hybrides combinant grilles d'occupation et objets. En revanche, on constate que les cartes de balises sont généralement capables de s'adapter automatiquement à la densité des objets et aux changements d'échelle, ce qui constitue un avantage par rapport aux grilles d'occupation.

2.4 Conclusion sur l'état de l'art des modèles élémentaires

2.4.1 Complémentarité des modèles

Lorsque l'on compare les différents modèles « élémentaires » de cartes (cartes topologiques, représentations basées sur l'apparence, cartes d'amers géométriques et représentations surfaciques), on peut être frappé par leur complémentarité : aucun modèle ne permet de gérer toutes les facettes de la cartographie mais chacune de ces représentations offre des avantages spécifiques. Nous avons tenté de dresser un tableau récapitulatif des qualités et des défauts de ces différents modèles selon les critères définis en introduction. Au vu des nuances introduites dans les discussions précédentes, nous nous en tenons

dans cette synthèse à un jugement très qualitatif : le lecteur est donc invité à se référer aux sections précédentes pour obtenir les détails de cette notation. En outre, nous nous restreignons pour les cartes basées sur l'apparence à celles qui sont constituées de scans lasers (par opposition aux approches fondées sur l'indexation d'images ou sur les mosaïques.

critères vs **types de cartes**	apparence	surfacique	primitives	topologique
compacité	*	*/**	**	**
généricité par rapport à l'environnement	**	*/**	*/**	**
généricité par rapport aux capteurs	*	**	*	*/**
gestion de grands environnements	**	*	**	**
souplesse des méthodes de cartographie	**	**	**	*/**
exploitation par les robots	*	*/**	*/**	**
exploitation par l'homme	*	**	*/**	*

TAB. *2.1: Jugement des différents types de cartes. Notations : * = faible ; */** = moyen ; ** = bon. Une note moyenne indique souvent que la valeur du critère dépend des méthodes de construction de cartes utilisées.*

On constate donc une complémentarité évidente entre représentations topologiques et géométriques, mais aussi entre les différentes représentations métriques. Cette remarque encourage donc à proposer des modèles plus sophistiqués, combinant divers types de représentations « élémentaires », comme nous le verrons au chapitre suivant.

2.4.2 L'indispensable modélisation des incertitudes

De manière générale, pour obtenir une carte correcte, nous avons vu qu'il est indispensable de modéliser les incertitudes. Ces incertitudes concernent non seulement le modèle d'environnement mais aussi la position du robot. En effet, les différents capteurs (proprioceptifs et extéroceptifs) sont entachés d'erreurs, qui se répercutent sur la représentation d'environnement construite. Pour maintenir la cohérence de la carte, lors de la fermeture de cycles notamment, il importe de pouvoir estimer ces imprécisions : cela permet de corriger les estimations de position du robot et les estimations de position des éléments de la carte, voire même de revenir sur d'anciennes décisions (nature des primitives extraites, mises en correspondance, etc.). Il existe en fait différents types d'incertitudes :

- les incertitudes d'ordre topologique : dans les cartes topologiques, il s'agit des incertitudes sur l'existence de sommets ou de liens d'adjacence du graphe par exemple, avec la possibilité de créer ou de supprimer certains sommets [66] [189] et l'adjonction d'indices de confiance sur l'existence des arêtes entre sommets [202] ; dans les représentations surfaciques (à mi-chemin entre cartes métriques et topologiques via les informations implicites de connexité entre cellules voisines), la modification des degrés d'occupation peut aussi avoir un impact sur la topologie de l'espace libre (ajout de trous notamment) ; dans les cartes de balises enfin, il peut s'agir de la nature des amers ou des objets composés (nombre de sommets d'une ligne polygonale, liens d'adjacence entre primitives géométriques...) ;
- les incertitudes d'ordre métrique : distributions probabilistes sur les erreurs de position des éléments de la carte (sur les amers des cartes métriques ou sur les sommets des graphes topologiques ancrés dans le plan), incertitudes sur les distances ou sur les transformations entre amers, objets, sous-cartes voire sommets topologiques, position des zones de degré d'occupation élevé dans les représentations surfaciques, etc.

Comme le rappellent Neira et al. [147], il existe deux grands types de modèles d'imprécision métrique :
- les modèles ensemblistes à erreurs bornées (avec des régions d'incertitude bornées de forme prédéfinie telle qu'une ellipse ou un rectangle), dans lesquels la fusion d'information géométrique est accomplie par intersection de régions ;
- les modèles probabilistes (avec des distributions discrétisées, échantillonnées ou paramétriques telles que les gaussiennes), dans lesquels la fusion est réalisée par des méthodes d'estimation (comme le filtrage de Kalman par exemple).

On peut également rajouter la logique floue et la théorie de l'évidence. En pratique, nous avons vu que l'essentiel des algorithmes de cartographie métrique sont basés sur des modèles probabilistes et sur l'estimation bayésienne : filtre de Kalman, algorithme EM, grilles d'occupation bayésiennes, FastSLAM, etc. Outre certaines approches topologiques, seules quelques approches alternatives ont été développées : méthodes ensemblistes, théorie de l'évidence et logique floue.

Enfin, parmi toutes ces techniques, certaines sont relativement génériques : le FastSLAM par exemple a déjà été appliqué aux grilles d'occupation, aux représentations basées sur l'évidence et aux cartes d'amers. D'autres sont presque exclusivement reservées à un type de modèle donné : en particulier, l'approche statistique de Thrun qui réalise une correction a posteriori des mises en correspondance nécessite une représentation flexible (à base de scans par exemple) [182].

Chapitre 3

Combinaisons de représentations élémentaires

> *« Il (...) faut d'ordinaire plusieurs [éléments graphiques] pour donner en se groupant des « formes » ou des « objets » ou autres choses subsidiaires : par exemple, surfaces engendrées par des lignes en rapport les unes avec les autres (...), ou formations spatiales dessinées par des énergies soutenant des rapports à trois dimensions (...). Un tel enrichissement de l'orchestre des formes multiplie de manière illimitée les possibilités de variation et, conjointement, les possibilités réelles d'expression. »*
> P. Klee (« Théorie de l'art moderne »)

Nous avons vu au chapitre précédent combien les différents modèles de cartes sont complémentaires. Ainsi, de nombreux travaux ont proposé de les combiner dans des représentations plus sophistiquées. La plupart des combinaisons concernent des mélanges de cartes topologiques et de cartes métriques mais certains travaux proposent également d'hybrider plusieurs types de modèles métriques.

3.1 Combinaisons de représentations topologiques et métriques

L'intérêt de combiner des cartes métriques et topologiques est reconnu depuis de nombreuses années, dès 1985 pour Chatila et Laumond [26] ou 1991 pour Kuipers et Byun [106] : en effet, les deux types de représentations offrent clairement des avantages complémentaires. Par exemple, comme l'indique Bailey [8], la topologie permet de subdiviser l'environnement de manière naturelle, elle induit un espace de stockage et des temps de calcul réduits et assure une connexité ainsi qu'une cohérence à grande échelle. En contrepartie, les cartes métriques induisent une précision locale élevée, une quantification de l'incertitude et une certaine généricité. Quant à la mise en correspondance des données observées avec la carte, elle se fait plutôt par reconnaissance de caractéristiques locales dans le cadre topologique, par opposition à une association de données contrainte par estimation de la pose du robot dans le cas métrique. Ainsi, on trouve dans la littérature un grand nombre de modèles hybrides qui proposent des combinaisons plus ou moins étroites entre informations métriques et topologiques. De plus, comme le souligne Thrun [180], la distinction entre cartes métriques et cartes topologiques n'est pas nette car la grande majorité des systèmes topologiques robustes recourent également aux informations métriques.

3.1.1 Coexistence de deux types de représentations

Au plus bas niveau de combinaison, on peut imaginer de construire séparément mais simultanément, selon deux techniques différentes, une carte topologique et une carte métrique de l'environnement. Ces constructions indépendantes peuvent cependant mener à des incohérences : la structure topologique sous-jacente de la carte métrique risque de ne pas correspondre à la carte topologique. Par exemple, on pourrait trouver un trajet praticable entre deux lieux de la carte métrique alors que le graphe topologique n'indique pas de chemin entre ces deux lieux. Finalement, il est assez rare de trouver dans la littérature un découplage total entre les représentations métrique et topologique si toutes deux sont construites.

Chez Dedeoglu et Matarič [41], le but est de générer en temps réel et de façon autonome des cartes topologiques de l'environnement : les sommets correspondent à des jonctions, des coins, des impasses ou des portes fermées. Ils portent des compteurs indiquant le nombre de fois où ils ont été visités. Ces lieux sont reliés par des arcs augmentés d'informations sur leur nature (espace libre, barrière, porte), mais aussi d'informations de distance et d'orientation : en cela, ces cartes entrent plutôt dans les catégories suivantes de cartes topologiques augmentées d'informations métriques. Cependant, les auteurs ont choisi de construire en parallèle une représentation purement géométrique basée sur des segments de droite. Le système ne cherche pas à les fusionner ou à les aligner correctement : si le robot utilise exclusivement l'information topologique, cette information géométrique apporte en revanche des informations à un éventuel observateur humain.

3.1.2 Augmentation de cartes topologiques par des informations métriques sur les arêtes

Comme le souligne Lee [114], lorsque l'on ajoute des informations de distance aux représentations topologiques, et notamment la longueur des chemins entre sommets fournie par l'odométrie (comme chez Mataric [133] ou Kuipers par exemple [106]), cela permet d'améliorer la navigation entre deux sommets (le robot sait approximativement à quel moment s'arrêter) ainsi que la planification de trajectoires (on peut notamment choisir le chemin topologique le plus court entre deux lieux). Cela favorise en outre la levée d'ambiguïtés lors de la reconnaissance de balises ou de lieux caractéristiques. On ne tombe pas pour autant dans un véritable cadre métrique puisque ces données métriques peu précises ne sont généralement pas utilisées pour inférer la position globale des sommets dans un repère de référence.

Ainsi, Simmons et Koenig [173] ont élaboré un système de navigation basé sur les processus de décision markoviens partiellement observables (PDMPO) qui exploite des informations métriques imprécises contenues dans les arcs de la carte topologique. Leur représentation se fonde sur la discrétisation de l'environnement 2D (ensemble S de positions s), de l'ensemble A des actions a possibles (« avancer d'un pas », « tourner à droite de 90 »...) et de l'ensemble O des observations o envisageables pour le robot (« percevoir une porte à droite », « percevoir un mur devant »...). Elle repose également sur des hypothèses simplificatrices de jonctions à angles droits et de couloirs rectilignes. Cette représentation est avant tout basée sur un graphe topologique dont les sommets correspondent aux jonctions entre couloirs et/ou aux portes, associées aux éléments pouvant être perçus dans chacune des quatre directions cardinales. Les arcs orientés sont enrichis d'informations métriques, sous la forme de distributions de probabilité sur toutes les longueurs possibles (cf. Fig. 3.1). Un modèle de Markov est ensuite construit à partir de cette représentation topologique. Les états de ce modèle contiennent à la fois la position et l'orientation du robot (seules les quatre directions cardinales sont possibles) : quatre états suffisent donc à représenter entièrement un sommet du graphe (même si le modèle a été ensuite raffiné pour contenir huit états à chaque jonction) tandis que les arcs sont modélisés par des séries de chaînes de Markov dont la longueur variable traduit l'incertitude sur la longueur du couloir (cf. Fig. 3.1).

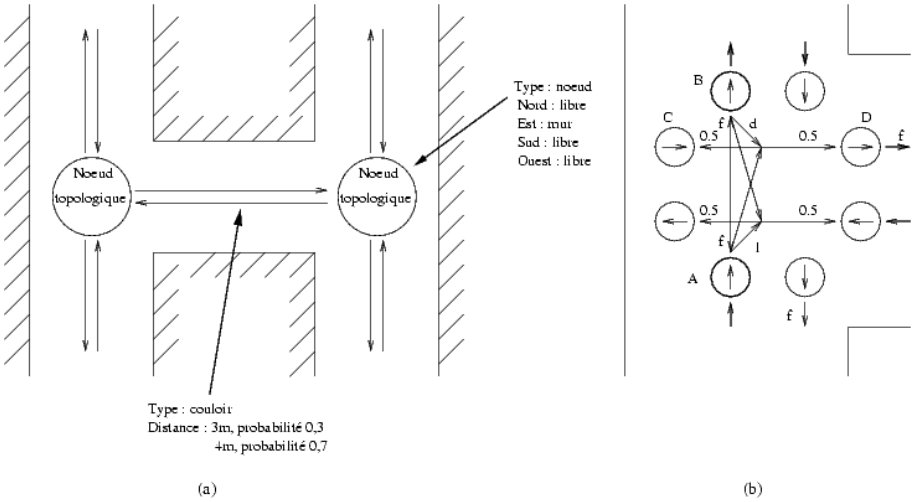


FIG. *3.1: Modélisation markovienne de Simmons et Koenig [173] a) Représentation topologique. b) Modèle de Markov correspondant.*

Ensuite, la localisation du robot s'effectue en suivi de position. Le robot maintient une croyance sur son état courant (et donc sur sa pose) sous la forme d'une distribution de probabilité $p(s)$ sur les états $s \in S$. La mise à jour de cette croyance s'effectue directement par application de la règle de Bayes. Si le robot exécute l'action a, les probabilités deviennent :

$$p_{posterior}(s) = K \times \Sigma_{s' \in S | a \in A(s')} p(s \mid s', a) \times p(s')$$

où K est un facteur de normalisation. Si le robot fait l'observation o à partir du capteur i, la mise à jour s'effectue de la manière suivante :

$$p_{posterior}(s) = K \times p_i(o \mid s) \times p(s)$$

Ce type de représentation s'avère particulièrement efficace et robuste pour la localisation et la navigation du robot en présence de fortes ambiguïtés perceptuelles. La localisation markovienne [19], utilisée dans de nombreuses applications telles que les robots guides de musées (qui doivent en outre faire face à des éléments mobiles dans l'environnement), est d'ailleurs basée sur un principe similaire mais utilise pour sa part des grilles d'occupation plutôt qu'un graphe topologique. Cependant, Simmons et Koenig ne précisent pas comment construire ces modèles topologiques de manière automatique, ni comment définir les distributions de probabilités nécessaires à la mise en œuvre des PDMPO (probabilités de transition notamment) : les cartes topologiques sont construites manuellement à partir des connaissances préalables sur l'environnement.

Chez Filliat, les informations de distance sur les arcs sont cette fois utilisées pour générer une véritable représentation métrique dans un repère de référence unique [69]. La carte topologique proposée est relativement dense puisqu'elle présente un espacement moyen de 50 cm entre les nœuds : elle permet donc de se faire une idée générale de la topographie des lieux au fur et à mesure des passages successifs du robot dans l'environnement (cf. Fig. 3.2). Dans la représentation topologique proposée, chaque nœud est associé à des informations extéroceptives omnidirectionnelles et directionnelles (informations visuelles et télémétriques) tandis que les arcs sont enrichis de données proprioceptives (distances imprécises). Un processus de relaxation permet de calculer une carte métrique plus précise et cohérente à partir de toutes les informations de distance observées sur les arcs : il s'agit d'un algorithme similaire à ceux qui sont appliqués aux réseaux de scans dans les représentations basées sur l'apparence [49]. L'information métrique

attachée aux nœuds contribue donc à la construction d'un modèle métrique et facilite la localisation du robot. Pour cela, une activité est associée à chaque nœud de la carte, afin d'estimer la probabilité que le robot se trouve à la position correspondante de l'environnement. Le processus incrémental de cartographie et de localisation simultanées enchaîne différentes étapes : calcul de l'activité pour tous les nœuds de la carte, reconnaissance du lieu courant ou création du nouveau nœud, mise à jour des données extéroceptives pour le nœud reconnu, mise à jour des paramètres attachés à l'arête reliant le nœud précédent au lieu courant à partir des données odométriques, application du processus de relaxation sur la carte globale pour assurer sa cohérence. Les expérimentations ont montré une bonne robustesse du système pour la localisation du robot, malgré le peu d'information perceptuelles utilisées.

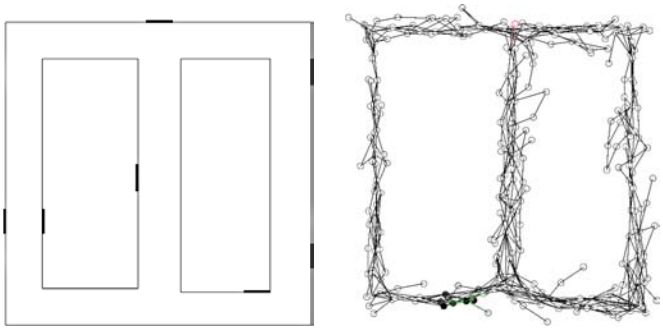


FIG. *3.2: Carte topologique ancrée dans le plan, construite selon l'algorithme de Filliat (à droite). La vue de gauche correspond à l'environnement réel vu selon une représentation métrique.*

3.1.3 Augmentation des cartes topologiques par des informations métriques sur les sommets

L'augmentation des cartes topologiques par des informations géométriques sur les sommets répond à différents objectifs :

- faciliter la levée d'ambiguïtés lors de la reconnaissance de lieux ou de balises : dans ce cas, on reste dans un cadre plutôt topologique ;
- calculer la localisation exacte du robot : nous avons déjà rencontré des cas similaires dans la section consacrée aux lieux reconnaissables ;
- obtenir simultanément une carte métrique, qui peut être exploitée lors de la navigation du robot.

Dans le premier cas, l'information de positionnement peut rester approximative, contrairement aux deux cas suivants où le but est plutôt d'obtenir une information aussi précise que possible.

Dans l'optique de faciliter la reconnaissance de lieux au sein d'une approche topologique, Tapus et al. exploitent ainsi le concept d'empreintes (« fingerprints ») [175]. Une empreinte correspond à une liste circulaire d'éléments caractéristiques de l'environnement (taches de couleur, arêtes verticales, coins et balises issus de données laser) pour laquelle l'ordre des éléments dans la liste correspond à leur ordre relatif autour du robot. Pour favoriser la distinction entre deux empreintes, les informations géométriques de distance angulaire entre deux éléments consécutifs de la liste ont été ajoutées. En outre, ce concept peut incorporer une gestion des incertitudes afin de garantir un bon appariement entre les empreintes observées et les empreintes de la carte : pour cela, chacun des éléments de l'empreinte est associé à un facteur d'incertitude dont la nature dépend du type de l'élément caractéristique. Chacune des empreintes étant codée selon une séquence de lettres, la mise en correspondance revient à un appariement de chaînes de caractères incertaines, dérivé d'un algorithme de minimisation d'énergie utilisé en stéréovision. Dans

cette approche, la navigation du robot et la construction de la carte topologique correspondante est assurée au moyen d'un modèle de processus de décision markovien partiellement observable (PDMPO). Chaque nœud de la carte contient la topologie du lieu (par exemple couloir, intersection en forme de T, porte fermée devant le robot, etc.) ainsi que l'empreinte associée. La fonction d'observation du PDMPO est assurée par la mise en correspondance des empreintes. Lorsque le robot est confiant à propos de son état, il peut également mettre à jour les éléments de l'empreinte ainsi que leur incertitude, ce qui permet de prendre en compte une certaine dynamique au sein de l'environnement.

Van Zwynsvoorde expose dans sa thèse une méthode de construction automatique d'un graphe topologique basé sur le diagramme de Voronoï [189]. Les sommets du graphe correspondent aux nœuds du diagramme de Voronoï. Chaque sommet indique le nombre d'arêtes incidentes et les angles relatifs entre ces arêtes, ainsi que la distance aux obstacles de l'environnement et enfin ses coordonnées cartésiennes approximatives (issues de l'odométrie). Quant aux arcs, ils sont découpés en une liste de « portions » : à chaque portion est associée l'information de distance aux obstacles qui l'ont générée ainsi qu'un type (segment ou parabole suivant la nature de l'équidistance aux obstacles de l'environnement : mur/mur, mur/sommet ou sommet/sommet). Ainsi, ces portions maintiennent une information riche sur l'espace libre environnant (largeur, longueur approximative et courbure). Plus globalement, chaque arc précise aussi les manœuvres possibles vers les arcs suivants, et contient un pointeur vers le sommet suivant ainsi qu'une direction moyenne (cf. Fig. 3.3).

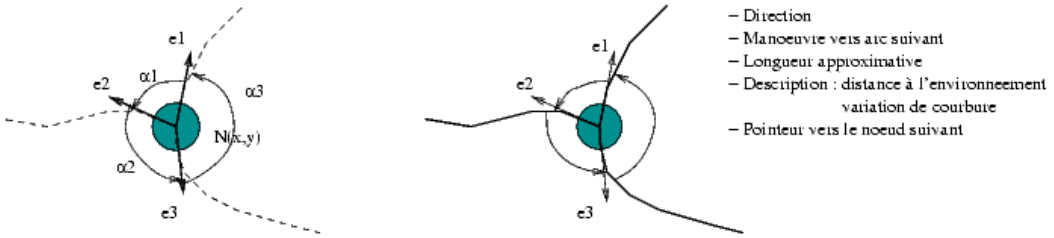


FIG. *3.3: Modélisation topologique proposée par Van Zwynsvoorde.*

La méthode de construction proposée est incrémentale. Chaque scan laser local est transformé en polygone d'espace libre dont sont extraites les portions locales du diagramme de Voronoï. Le système en distingue les parties certaines des parties temporaires, induites par les segments « non obstacles » du polygone (les lignes de visée du capteur liées aux occultations). Un filtrage du diagramme est également appliqué pour éviter la multiplication des arcs suite aux légères irrégularités de l'environnement réel. Ensuite, pour ajouter cette carte locale à la représentation globale, l'algorithme détermine le lieu de la connexion entre ces deux cartes et gère la fusion des parties communes en fonction de leur caractère certain ou temporaire, via différentes opérations (ajouts et suppressions de sommets, mise à jour des attributs des sommets et des arêtes...). Pour définir le lieu de connexion, le système utilise divers critères de comparaison entre sommets : angles de départ relatifs entre arcs successifs, longueur des arcs incidents, variation de la distance aux obstacles le long des arcs, direction globale des arcs et position incertaine du sommet. Cette position métrique des sommets sert donc avant tout à limiter les ambiguïtés perceptuelles, plutôt qu'à estimer la position du robot ou à construire une carte métrique en bonne et due forme.

On peut également noter que cette méthode de construction du modèle est passive : en particulier, contrairement à d'autres approches exploitant le diagramme de Voronoï [106] [192], le robot n'est pas contraint de s'asservir sur les arêtes du diagramme. En contrepartie, l'approche est sans doute plus sensible aux problèmes de localisation du robot : la dérive odométrique constatée risque de nuire à la mise en correspondance des sommets et à la cohérence topologique du modèle obtenu. Enfin, on a vu que cette carte topologique est très augmentée d'informations métriques, même si le cadre de travail pour

sa construction et son exploitation reste plutôt topologique. Comme l'indique l'auteur, ces informations métriques peuvent servir à introduire une certaine sémantique dans les modes de navigation du robot. Par exemple, elles peuvent permettre d'interpréter qualitativement la nature du chemin (couloir, coin, passage, hall, chicane...) afin de lancer des modes de navigation appropriés.

Selon Choi et al., le diagramme de Voronoi n'est cependant pas toujours adapté aux formes arbitraires des objets et se révèle assez coûteux en temps de calcul [27]. Les auteurs proposent donc un modèle topologique alternatif basé sur l'opération morphologique de « squelettisation » (« thinning » en anglais) appliquée à une grille d'occupation locale déduite des données laser. Par rapport au diagramme de Voronoi, la squelettisation fournit également un graphe dont les sommets sont ancrés dans le plan mais d'après les auteurs, le calcul du graphe serait plus rapide. En outre, ce modèle fournit des nœuds supplémentaires (extrémités, points de branchement, coins) susceptibles de lever les ambiguïtés perceptuelles, puiqu'ils apportent localement un surcroît d'information géométrique. Les auteurs proposent une méthode incrémentale de construction de cette représentation. Les cartes topologiques locales sont ainsi fusionnées dans la carte globale qui maintient la position des nœuds dans un repère unique de référence ainsi que les informations de connexité entre nœuds. Toutefois, le système ne gère pas la problématique de localisation et de cartographie simultanée puisque la carte est construite a priori depuis un ensemble de positions prédéterminées pour le robot : durant la construction de la carte, la position du robot est donc connue avec précision et le modèle construit présente une information métrique précise et cohérente. Une fois cette phase d'apprentissage terminée, le robot exploite la carte obtenue pour se localiser : il échantillonne aléatoirement les positions possibles dans l'environnement, extrait par fenêtrage la carte topologique locale de chaque position candidate, la compare à la carte locale extraite par squelettisation de sa perception courante (comparaison des distances relatives entre nœuds) et en déduit sa position selon un critère bayésien.

Enfin, les représentations basées sur l'apparence, que nous avions placées dans la catégorie des modèles géométriques, peuvent être considérées comme des cartes topologiques permettant de produire une carte métrique dans un repère global unique. En effet, on a vu que ces représentations sont généralement fondées sur un réseau de scans (ou d'images) qui constitue un graphe topologique. Le but est ensuite d'estimer aussi précisément que possible la position des sommets du graphe à partir des estimations de distance entre ces sommets, ce qui permet de recaler les scans (ou les images) les uns par rapport aux autres. Avec ce type de modèle, on dispose donc à la fois des informations métriques sur les arêtes et sur les nœuds.

Einsele [60] propose lui aussi la construction en ligne d'une carte métrique basée sur un réseau topologique de données laser. Toutefois, à la différence des représentations basées sur l'apparence, les scans laser sont transformés en polygones constitués des segments élémentaires reliant deux points de mesures successifs : ce sont ces segments qui apparaissent dans le rendu final de la carte. Ainsi, la carte construite peut être assimilée à un graphe dont chaque sommet correspond à un scan polygonalisé et dont les arêtes indiquent une relation de voisinage entre ces scans. Les transformations entre deux scans sont estimées par mise en correspondance de ces segments selon une technique de programmation dynamique : ainsi, les positions géométriques des sommets du graphe de polygones peuvent être estimées. Toutefois, on n'obtient pas forcément une carte métrique « pure » au sens où les positions ne sont pas forcément indiquées dans un référentiel global unique. En effet, à la création d'un nouveau sommet (lors de l'acquisition d'un scan impossible à apparier avec un scan existant), le système cherche à vérifier la plausibilité de l'estimation de position fournie par le système. Par exemple, lors de la progression le long d'un couloir, la distance parcourue par le robot est parfois difficile à estimer puisque les observations successives sont très similaires et l'appariement de scans ambigu : il arrive que le système prétende que le robot reste immobile, ce qui est manifestement inexact. Si une telle situation d'incohérence est mise en évidence par

le robot, celui-ci abandonne le système de coordonnées courant pour instancier un nouveau repère attaché au nouveau sommet. La nouvelle arête reliant ce nouveau sommet au sommet précédent peut également indiquer comment passer d'un repère à l'autre. Toutefois, l'article n'indique pas précisément comment gérer les problèmes de fermeture de cycles.

3.1.4 Réseaux de cartes métriques locales

Au-delà d'une simple augmentation des sommets de la carte topologique par leurs coordonnées cartésiennes, de nombreux travaux proposent de créer des réseaux topologiques de cartes métriques locales : dès lors, comme le souligne Bailey [8], les sommets de la représentation topologique ne sont plus réduits à des lieux discrets mais peuvent décrire des régions de taille et de forme arbitraire. Pour Prescott [161], il existe d'ailleurs deux grandes approches pour segmenter l'environnement géométrique selon les composantes élémentaires d'un graphe topologique :

- soit on associe les sommets à des lieux de dimension nulle et les arêtes à un réseau de chemins 1D qui connectent ces lieux,
- soit on associe les sommets à des régions plus étendues de l'espace et on assimile les arêtes à l'existence de frontières entre ces régions (on retrouve la notion de « tesselation » ou « pavage »).

Ces deux types de représentations sont duales. En effet, si l'on réduit les régions du pavage à leur centre ponctuel, on obtient un graphe d'adjacence dont les sommets sont de dimension nulle. Inversement, si l'on construit par exemple le diagramme de Voronoï ponctuel des sommets d'un graphe topologique ancré dans le plan, on obtient naturellement une partition de l'environnement (cependant, on n'est pas assuré que les arêtes du graphe initial correspondent bien à l'existence de frontière entre régions du diagramme de Voronoï). On obtient ainsi deux classes de modèles hybrides fondées sur un réseau de cartes métriques locales, comme l'illustre la figure 3.4 : les représentations topologiques issues d'un partitionnement (régulier ou non) du plan et les réseaux plus « lâches » de cartes métriques locales dont l'ensemble ne forme pas une partition du plan (il peut exister des zones de superposition entre ces cartes et elles ne recouvrent pas nécessairement tout l'espace).

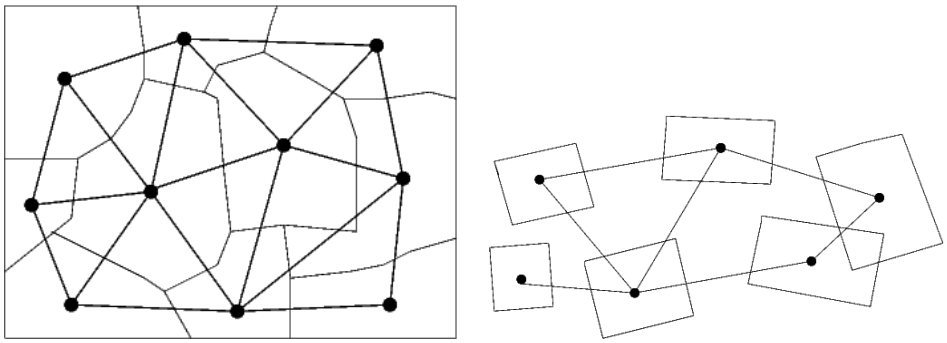


FIG. *3.4: Deux types de réseaux de cartes métriques locales a) Graphe d'adjacence issu d'un partitionnement du plan. b) Graphe de cartes métriques locales qui ne forment pas une partition du plan.*

En outre, on peut distinguer plusieurs objectifs lors de la construction de tels réseaux :

- soit il s'agit de mettre en œuvre une navigation topologique entre cartes locales : dans ce cas, le système exploite uniquement des liens qualitatifs entre sommets de la carte, sans chercher à se localiser précisément dans un repère de référence global ;
- soit le but est de construire une carte métrique précise et cohérente permettant de déterminer *in fine* la position géométrique de chaque carte locale par rapport à un référentiel absolu (ou du moins

de retrouver ponctuellement des informations métriques précises le long de chemins topologiques de la carte, en vue de gérer les cycles par exemple) : le système cherche alors à retrouver les distances exactes entre cartes locales, voire des relations spatiales de plus haut niveau (transformations de systèmes de coordonnées).

Dans le cas du partitionnement d'une carte métrique, on obtient en principe directement les informations géométriques dans un repère absolu : libre au système de les exploiter ou non. Dans le cas de réseaux plus « lâches », tout dépend du but poursuivi : certains systèmes s'attachent à maintenir la cohérence globale des informations métriques, tandis que d'autres se contentent de naviguer localement, entre lieux.

Extraction de la topologie selon un partitionnement de la carte métrique

On peut là encore distinguer deux types de représentations hybrides fondées sur un partitionnement de carte métrique : soit cette carte métrique est constituée de balises ou de primitives géométriques, soit elle correspond à une représentation surfacique de type tessellation.

*** Découpage d'une représentation géométrique fondée sur des primitives géométriques**

**** Représentations multiniveaux**

Dans les années 1980 et 1990 sont apparus divers systèmes robotisés fondés sur plusieurs couches de représentations.

Chatila et Laumond suggèrent ainsi de représenter l'environnement selon trois niveaux : un niveau géométrique, un niveau topologique et un niveau symbolique [26]. Le modèle géométrique est directement déduit des données perceptuelles : dans l'illustration proposée, l'environnement est modélisé par des objets polygonaux, dont la position des sommets est définie dans un référentiel absolu. Le modèle topologique contient en réalité toute une hiérarchie de représentations. Il se présente à chaque niveau sous la forme d'un graphe dont les sommets correspondent à des lieux (unités fonctionnelles telles qu'une station de travail ou unités topologiques telles qu'une pièce, un étage voire un bâtiment entier selon le degré hiérarchique) et dont les arcs représentent des connecteurs entre lieux adjacents (portes, couloirs, escaliers ou ascenseurs par exemple). Au plus bas niveau de cette hiérarchie, la topologie de l'espace est directement déduite du modèle géométrique : ce dernier est partitionné en cellules convexes sur la base des sommets et des arêtes d'obstacles. Ce graphe de cellules peut alors être utilisé pour planifier des chemins de manière plus efficace qu'avec le modèle géométrique. Des modèles topologiques plus abstraits sont ensuite construits en structurant cette première représentation topologique : les composantes résultantes peuvent être étiquetées de manière à introduire des informations sémantiques (portes, pièces, couloirs, etc.). Le modèle sémantique ainsi produit doit également contenir des informations sur les propriétés des objets et de l'espace et sur leurs relations, afin d'être exploité par les niveaux décisionnels les plus élevés du système. Cet article insiste également sur la modélisation des incertitudes afin d'assurer la cohérence de la carte. Ces idées ont été développées ultérieurement par la même équipe du LAAS, à travers les thèses de Moutarlier [139] et de Bulata notamment [18].

Le système Qualnav de Levitt et Lawton [120] propose également une représentation spatiale multiniveaux fondée sur la reconnaissance de balises. Le niveau métrique contient des informations d'angles relatifs et des informations de distance entre balises. Au niveau topologique, l'estimation de distances entre balises est oubliée et l'on relie toute paire de balises par des frontières virtuelles : cette opération génère une subdivision topologique (le système ne tient pas compte de la partition métrique sous-jacente). En effet, lorsque le robot franchit l'une de ces frontières virtuelles, l'ordre (droite / gauche) des deux ba-

lises générant cette frontière s'inverse (cf. Fig. 3.5)). Ainsi, chaque région possède une description unique, de nature topologique, en fonction de l'ordre de ces paires de balises. Les deux niveaux peuvent travailler de concert dans la planification. Il semble que ce système n'ait toutefois été testé qu'en simulation.

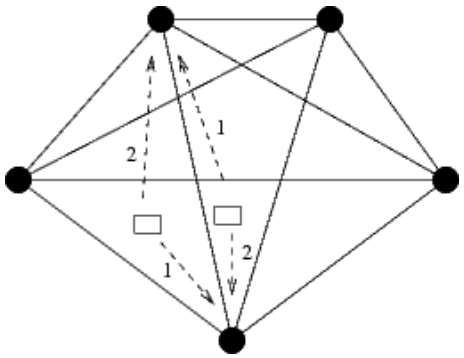


FIG. *3.5: Modélisation topologique du système Qualnav : toute paire de balises (points noirs sur la figure) est reliée par une frontière virtuelle, ce qui génère une partition métrique du plan. Cette partition est en fait exploitée de manière topologique en considérant l'ordre des balises (par exemple avec la convention que la première est celle de droite et la seconde celle de gauche, lorsque les deux balises se trouvent devant le robot, ce dernier étant représenté sur la figure par un rectangle) : cet ordre s'inverse lorsque l'une de ces frontière est franchie.*

**** Régions de visibilité de balises**

Suivant une idée similaire, d'autres travaux proposent un partitionnement de l'espace libre en régions de visibilité associées à des amers. Par exemple, Kosecka découpe l'environnement en un ensemble de lieux qui couvrent l'ensemble de l'espace libre [104] : des régions de visibilité associées à une balise (balise distinctive constituée de plusieurs primitives géométriques) et des régions de visibilité associées aux frontières (murs notamment) qui conduisent aux balises. Pour faciliter la navigation, l'environnement est alors représenté comme un graphe dont les sommets correspondent aux zones de superposition de régions de visibilité de balises (appelées « gateways ») ou les frontières menant à des balises. Un sommet est ainsi caractérisé par l'ensemble des balises visibles ou approchables par rapport auxquelles le robot peut se positionner. Quant aux arêtes du graphe, elles correspondent aux stratégies d'asservissement pouvant être appliquées depuis un lieu donné. Ce modèle n'est cependant pas construit en ligne puisque l'opérateur dispose d'une connaissance a priori de l'environnement et sélectionne lui-même les amers.

**** Découpage en cartes locales pour accélérer les calculs de l'EKF**

Le partitionnement d'une carte de balises a également été proposé dans le but d'accélérer les calculs dans les approches à base de filtre de Kalman étendu. Leonard et Feder ont ainsi développé une approche nommée « Decoupled Stochastic Mapping » (DSM), qui s'appuie sur un ensemble de sous-cartes dont la position est référencée dans un repère absolu [116]. Lorsque le robot se trouve dans une sous-carte donnée, il ne travaille que localement, ce qui réduit considérablement les coûts de calcul. Les auteurs définissent également des règles de transition heuristiques entre sous-cartes, permettant de transférer l'information de position du véhicule d'une carte à l'autre. En pratique, la taille et la forme de chacune de ces cartes locales sont spécifiées a priori : il s'agit d'une partition régulière du plan, dans laquelle chaque cellule est légèrement élargie pour créer une petite zone de superposition avec la cellule voisine. Ainsi, comme dans les grilles d'occupation, la topologie n'est pas explicite mais grâce à la connaissance implicite des liens d'adjacence entre cellules, il est possible de limiter le nombre de sous-cartes candidates à la transition.

On évite ainsi de réaliser une véritable recherche spatiale. Cette méthode présente globalement une complexité en temps constant mais elle demeure sous-optimale, du fait des règles de transition *ad hoc* : si les résultats expérimentaux sont prometteurs, elle nécessiterait donc des tests expérimentaux plus poussés afin de vérifier sa validité.

Comme nous l'avons vu précédemment, le « Compressed Kalman filter » de Guivant et Nebot [80] propose également de travailler momentanément sur une région limitée de la carte globale, contenant seulement un nombre restreint de balises. En pratique, les auteurs proposent un découpage régulier de l'espace en zones rectangulaires, dont la taille est au moins égale à la portée du capteur. Tant que le robot opère sur une zone rectangulaire locale r, l'information recueillie peut être maintenue avec une complexité de l'ordre de $O(N_a^2)$, N_a étant le nombre de balises contenues dans la région r et dans chacune des huit régions voisines (en 8-connexité). Lorsque le robot quitte cette région r, l'information recueillie localement peut ensuite être transférée sans perte d'information à l'ensemble de la carte en une seule itération, au coût classique du SLAM complet $O(N^2)$. Ainsi, la méthode proposée est optimale : elle ne nécessite pas de faire appel à des heuristiques de transitions entre sous-cartes comme chez Leonard et Feder [116].

*** Extraction de la topologie de représentations surfaciques**

Thrun et al. [102] [178] ont proposé de générer a posteriori une carte topologique à partir d'une grille d'occupation bayésienne. L'objectif est d'abord de réduire la place mémoire occupée par la grille d'occupation, puis d'utiliser la représentation topologique compacte pour réaliser une planification de chemins efficace. L'algorithme proposé découpe la carte métrique en régions cohérentes, séparées par des « lignes critiques » qui correspondent à des passages étroits tels que des portes et qui sont issues de points critiques sur diagramme de Voronoï. On gagne ainsi plusieurs ordres de grandeur dans la taille de la représentation. De plus, une évaluation pratique des cartes topologiques obtenues a été menée pour les comparer aux représentations initiales. Cette évaluation est basée sur des critères de cohérence, de perte et d'efficacité dans la planification de chemins. Elle est menée sur trois configurations différentes de l'environnement. Le critère de cohérence est vérifié puisque chaque solution de planification trouvée dans l'une des cartes peut également être extraite de l'autre carte. La perte est caractérisée par la longueur du chemin trouvé : les auteurs constatent que la longueur du chemin issu du modèle topologique n'est que légèrement supérieure à celle du chemin extrait de la grille d'occupation. Enfin, l'efficacité en terme de temps de calcul est bien meilleure dans le cas topologique. On remarque toutefois que le sytème ne permet pas une construction simultanée des deux types de représentations : le modèle topologique n'est issu que d'un post-traitement de la grille d'occupation initiale qu'il faudrait a priori recommencer à chaque fois que l'on souhaite effectuer une planification.

Les représentations hiérarchiques que nous avions évoquées dans la section consacrée aux modèles surfaciques peuvent aussi être considérées comme des représentations topologiques. A l'extrême, une simple grille d'occupation contient d'ailleurs implicitement un graphe d'adjacence qui relie chaque cellule à ses 4 voisins si l'on travaille en 4-connexité, voire à ses 8 voisins si l'on préfère la 8-connexité. Cette propriété est souvent exploitée dans les algorithmes de planification, par exemple lorsqu'ils utilisent la transformation en distance de Jarvis [205]. On a vu qu'Arleo et al. [4] construisent une carte géométrique multirésolution qui correspond à une partition de l'espace avec des cellules rectangulaires de taille variable. Ils en déduisent en outre un graphe d'adjacence pour la planification : en réalité, cette représentation topologique est déjà pour ainsi dire présente dans leur représentation en régions. Quand ils structurent la grille d'occupation selon un modèle hiérarchique, Poncella et al. [156] génèrent également un graphe topologique dont les sommets appartiennent à différents niveaux hiérarchiques. Ils maintiennent de plus un lien explicite entre la couche métrique et la couche topologique, ce qui permet d'appliquer des algorithmes de planification et

d'exploration efficaces s'appuyant successivement sur les deux couches. Au niveau topologique, les nœuds à explorer sont d'abord ordonnés de manière efficace puis l'algorithme recherche des chemins permettant de les relier. Au niveau métrique, le chemin topologique est précisé au moyen d'un algorithme de planification classique basé sur les champs de potentiel. L'algorithme, même s'il ne conduit pas toujours à une solution optimale, se révèle efficace à la fois en environnement connu et inconnu. On peut cependant noter que la représentation topologique obtenue avec ces modèles hiérarchiques n'est pas très explicite pour l'homme : en particulier, les nœuds ne correspondent pas à des éléments caractéristiques tels que des pièces ou des jonctions.

*** Extraction de la géométrie des lieux topologiques issus de la partition**

Fabrizi et Saffiotti [68] exploitent deux niveaux d'informations géométriques. Le premier niveau consiste à construire une grille d'occupation dans un cadre de logique floue (cf. Oriolo et al. [152]), tandis que le second niveau s'appuie sur le réseau topologique extrait par partitionnement de la grille d'occupation : il s'agit de caractériser en termes géométriques les régions qui constituent les sommets du graphe topologique. Dans ce travail, la grille d'occupation est vue comme une image : les degrés d'occupation flous correspondent à des niveaux de gris. Ainsi, il est possible d'appliquer des techniques classiques de traitement d'images telle que la morphologie mathématique : ici, les auteurs recourent à une ouverture puis à une segmentation selon les lignes de partage des eaux pour extraire des zones d'espace libre de grande dimension délimitées par des obstacles ou des passages étroits. Ces zones sont considérées comme les sommets du graphe topologique tandis que les arêtes correspondent aux liens de connexité entre régions adjacentes dans la représentation surfacique. Ensuite, l'algorithme extrait différents attributs géométriques sur chaque zone, en exploitant notamment la notion de moment : centre, orientation, largeur, longueur et excentricité de la région. Ces attributs permettent ensuite d'introduire une certaine sémantique dans la représentation : par exemple, l'attribut d'excentricité permet de déterminer si la région correspond plutôt à une pièce ou à un couloir. Le robot peut alors exploiter la carte pour mettre en œuvre des comportements sensorimoteurs adaptés : navigation à l'estime pour traverser une pièce ou suivi de mur pour traverser un couloir par exemple.

Réseaux non issus d'un partitionnement

*** Navigation topologique entre cartes métriques locales**

Selon Yeap, qui s'inspire du domaine des sciences cognitives, il n'est pas souhaitable de garder des relations métriques à grande échelle [203]. C'est pourquoi il propose un modèle hybride dit « relatif-absolu » (R-A). Ce modèle consiste en une représentation globale appelée RSR (Relative Space Representation) qui est lui-même une composition qualitative de représentations locales appelées ASR (Absolute Space Representations). Il s'agit en fait d'un ensemble de cartes métriques précises connectées topologiquement par des zones floues et partiellement inconnues.

D'un point de vue plus biologique, Nehmzow précise en outre que si les animaux utilisent leur sens magnétique, des balises terrestres et des balises de références (étoiles, soleil) pour se localiser, ils préfèrent recourir à des chemins bien définis même si ceux-ci impliquent des distances plus importantes (notamment s'ils sont proches de leur habitat).

Plus récemment, Simhon et Dudek ont également proposé un modèle basé sur des zones précises (« islands of reliability ») entre lesquelles le robot peut naviguer sans exploiter d'informations géométriques [172]. Dans sa thèse, Victorino a développé un système hybride combinant une navigation topologique

avec des représentations métriques locales ou « semi-globales » (représentation globale segmentée en « lieux ») [192]. Ces cartes métriques locales sont composées de segments de droite dont la position est progressivement raffinée par filtrage de Kalman étendu. Dans ce cadre, le robot navigue sur un diagramme de Voronoï et ne met à jour la carte métrique qu'aux points de bifurcation de ce diagramme : cette restriction de la représentation métrique à certains points d'intérêt vise notamment à réduire les coûts de calcul. Une nouvelle carte locale est créée lorsque la branche entre deux points de bifurcation est trop longue pour permettre d'identifier des éléments caractéristiques communs avec la carte globale mémorisée. Par ailleurs, la convergence du robot vers ces points de bifurcation est assurée par des techniques de commande référencée capteurs, qui permettent notamment de borner les erreurs de déplacement du robot et de limiter les problèmes liés à une dérive de l'odométrie.

Dans le même esprit, Tomatis et al. [184] proposent un paradigme hybride dans lequel le modèle métrique correspond à un ensemble de lieux tels que des pièces individuelles, ces lieux étant reliés entre eux via un graphe topologique. En pratique, le modèle métrique est composé de droites infinies situées le long des frontières d'obstacles locaux. Il est mis à jour par filtrage de Kalman étendu. Les nœuds du modèle topologique représentent quant à eux des lieux topologiques tels que des ouvertures ou des coins, ces nœuds étant connectés par le biais d'arêtes topologiques. Les connexions entre un nœud topologique et un lieu métrique sont matérialisées par des arêtes spéciales permettant d'effectuer la transition entre les paradigmes métrique et topologique. Pour déterminer précisément le point de transition vers le modèle métrique et permettre ainsi une re-localisation précise du robot dans la carte métrique locale, ce système impose l'existence d'une balise métrique détectable à cet endroit (par exemple une porte). Ce basculement du modèle topologique au modèle métrique pet également être réalisé au moyen des empreintes que nous avons évoquées précédemment [130]. La navigation topologique du robot entre lieux est ensuite basée sur une estimation de position du robot selon un processus de décision markovien partiellement observable (PDMPO). On peut noter que cette approche permet une fermeture de cycles explicite pour la construction de carte : pour cela, le système détecte que la distribution de probabilité de position du robot converge en deux pics qui se déplacent de la même manière.

*** Définition d'un niveau géométrique à partir du niveau topologique de la carte**

Au début des années 1990, Kuipers a développé un concept appelé « spatial semantic hierarchy » [106]. Ce système se décompose selon :

1) un niveau sensorimoteur qui représente les relations d'entrée / sortie avec l'environnement ;

2) un niveau procédural qui dispose de procédures apprises et enregistrées définies en termes de primitives sensorimotrices pour accomplir des tâches de recherche de lieux et suivi de chemin ;

3) un niveau topologique qui décrit l'environnement en termes d'entités prédéfinies telles que des lieux, des chemins, des balises et des régions, reliées par des relations topologiques telles que la connectivité, l'inclusion et l'ordre ;

4) un niveau métrique qui décrit l'environnement selon les mêmes entités prédéfinies, reliées par des relations métriques telles que la distance et les angles relatifs, ou des angles et distances absolus par rapport à un repère de référence.

Selon Kuipers, la construction du modèle d'environnement doit se faire du premier niveau vers le quatrième, avec un raffinement progressif du modèle en termes de structure et de géométrie. On passe ainsi du paradigme classique « perception → géométrie → topologie » (comme chez Chatila et Laumond par exemple [26]) au paradigme « perception + contrôle (comportements sensorimoteurs) → topologie → géométrie ».

Le système multiniveaux NX de Kuipers et Byun [106], développé en simulation, est inspiré de ce modèle hiérarchique. Au premier niveau du modèle se trouvent les lieux distinctifs et les arêtes de déplacement, définis à partir de critères de « distinctiveness » (« d-mesures ») que nous avons évoquée dans le chapitre consacré aux cartes topologiques (par exemple le critère « equidistance aux objets proches »). Les lieux correspondent aux maxima locaux ponctuels trouvés par des techniques de montée de gradient selon les d-mesures tandis que les arêtes sont définies à la fois par une d-mesure et une stratégie de contrôle (par exemple « suivre le centre d'un couloir » ou « suivre le mur gauche ») qui correspond au second niveau (procédural). Au troisième niveau ou niveau topologique, les lieux sont reliés entre eux par les arêtes de déplacement. Enfin, au quatrième niveau apparaît une carte géométrique, locale d'abord (autour des lieux et des chemins possibles) puis éventuellement globale dans un repère de référence unique. Kuipers suggère de générer cette carte globale grâce à une méthode de relaxation. Ces différents niveaux ne sont pas indépendants : ils peuvent travailler de concert pour résoudre des problèmes de génération de trajectoires par exemple.

*** Utilisation d'une carte topologique pour accroître la robustesse de construction de la carte métrique**

Thrun et al. [182] ont proposé une approche originale de l'intégration des cartes topologiques et géométriques. Ils soulignent que les deux types de cartes sont complémentaires pour construire des cartes à grande échelle : si la topologie permet de résoudre des problèmes d'alignement global, la géométrie permet en revanche une résolution très fine et un alignement local plus précis. Selon eux, les problèmes géométriques et topologiques peuvent être résolus comme différentes instances d'une même classe de problèmes d'estimation statistique, dans lesquels le robot cherche la carte la plus probable à partir d'observations et de commandes de mouvement. Pour les résoudre, ils préconisent l'algorithme EM (« Expectation - Maximisation ») de Dempster qui effectue une montée de gradient dans l'espace des vraisemblances, en alternant de façon itérative une phase dite d'estimation (« expectation ») et une phase de maximisation. Ces deux phases correspondent respectivement à une étape de localisation (calcul des densités de probabilité a posteriori sur les positions présente et passées du robot, la carte étant fixée) et à une étape de cartographie (calcul de la carte la plus probable selon ces distributions de probabilité sur les positions du robot). L'intérêt de l'algorithme EM est notamment de réviser en permanence et dynamiquement l'ensemble des croyances en fonction des données au cours du temps.

Dans la carte topologique, l'algorithme cherche à définir la position des lieux significatifs (sommets du graphe) et l'ordre dans lequel ils sont visités par le robot. Les auteurs supposent que les lieux sont indiscernables et que le robot n'en connaît pas le nombre. A chaque instant, le système sait seulement s'il se trouve ou non en un lieu significatif (en pratique, l'opérateur appuie sur un bouton pour indiquer que la plate-forme a atteint un tel lieu mais des méthodes automatiques sont envisageables). Les expérimentations montrent que l'algorithme EM est capable de trancher quand il existe des ambiguïtés de mise en correspondance entre le lieu courant et les sommets topologiques de la carte. En outre, l'implémentation proposée exploite une représentation discrétisée des densités de probabilités (positions successives du robot, positions des sommets de la carte) par le biais de grilles régulières, d'où une possibilité de distributions multimodales. En fait, la représentation topologique ressemble un peu à une carte métrique des lieux significatifs (indicernables) avec des liens d'adjacence qui indiquent s'il existe un chemin traversable par robot : on sort du cadre topologique strict car la reconnaissance de lieux se fait sur la base d'informations de distance (odométrie, position des nœuds dans l'espace...). Quant à la carte métrique, elle est basée sur l'apparence puisqu'elle est constituée d'un réseau de scans recalés (comme chez Lu et Milios par exemple [126]). Les densités de probabilités sont définies comme des gaussiennes d'où une précision en nombres flottants et une meilleure résolution que dans la représentation topologique. En revanche, comme les probabilités ne sont pas multimodales, l'erreur de positionnement du robot

doit rester limitée : la carte topologique grossière est justement à même de fournir une bonne estimation initiale de la localisation du robot.

En résumé, cet article propose un cadre théorique unifié, basé sur l'algorithme EM, utilisant une carte topologique relativement grossière pour accroître la robustesse du processus de construction de carte métrique précise. En contrepartie, l'algorithme EM n'est pas incrémental et exige des temps de calcul prohibitifs pour une utilisation en temps réel. C'est pourquoi des méthodes simplifiées, dites « hybrides », ont été proposées par la suite (cf. par exemple [179] décrit plus haut dans le cadre des représentations basées sur l'apparence) [180]. A la base, ces méthodes implémentent un algorithme populaire de « maximum de vraisemblance incrémental », qui peut être vu comme un algorithme EM dégénéré, auquel on a retiré l'étape E (expectation) : la carte à l'instant t est la plus probable compte tenu de la carte à l'instant $t-1$, de la pose du robot à l'instant t et de l'observation courante (comme dans le cas des grilles d'occupation classiques par exemple). A priori, cet algorithme basique fige définitivement la carte, sans possibilité de correction ultérieure. Pour gérer les boucles, les méthodes hybrides maintiennent en outre une notion d'incertitude en calculant une distribution de probabilité a posteriori sur la position courante du robot qui peut être employée pour calculer la position la plus probable du robot dans la partie ancienne de la boucle. Ainsi, une action corrective peut être mise en œuvre, par rétropropagation de l'erreur d'estimation sur l'ensemble de la boucle : cette action peut être vue comme une simplification de l'étape M (maximisation). Toutefois, comme cette action corrective est seulement utilisée de manière ponctuelle, ces méthodes hybrides ne permettent pas de raffiner l'association de données comme dans l'algorithme EM initial et peuvent même être sujettes à échec si la décision de correction est erronnée. On note par ailleurs que toutes ces méthodes fournissent une carte unique, sans estimation d'incertitude associée.

*** Construction d'un réseau de cartes locales pour générer des représentations métriques à grande échelle**

Chong et Kleeman ont proposé une stratégie de cartographie fondée sur un réseau de cartes locales indépendantes générées au cours du déplacement du robot [28]. Seules sont maintenues les informations de covariance entre les repères des cartes locales et entre amers d'une même carte locale. Ainsi, les coûts de calcul et la place mémoire requise sont considérablement réduits (l'algorithme est en temps constant si les cartes locales sont bornées), puisque les covariances entre amers appartenant à des cartes locales distinctes ne sont pas estimées. Les transitions entre cartes locales sont assurées par un algorithme qui recherche la position du robot lorsque celui-ci a une forte probabilité de se situer dans la zone d'une autre carte locale. Toutefois, les auteurs ne proposent pas de méthode susceptible de générer une carte globale cohérente de l'environnement : ils suggèrent simplement une optimisation au sens des moindres carrés. Williams montre en fait que la construction de cartes globales cohérentes (et équivalentes à celles fournies par un EKF global) peut être réalisée en appliquant une série de contraintes entre sous-cartes [199] : son algorithme de« Constrained Relative Submaps Filter » est en fait une extension de l'algorithme de « Constrained Local Submap Filter » décrit au chapitre 2, permettant la gestion de plusieurs cartes locales au lieu d'une seule.

Bailey a également développé dans sa thèse un algorithme appelé « Network Coupled Feature Map » [8] dans lequel la transformation entre deux cartes locales adjacentes est estimée via des contraintes imposées entre des balises communes. Cependant, si elle fournit des résulats intéressants, la méthode ignore les corrélations qui apparaissent entre différentes estimations. Les auteurs conjecturent que cela produit quand même des résultats cohérents.

Pour construire des cartes métriques de taille importante, certains travaux appliquent directement

les techniques de construction de représentations basées sur l'apparence, qui sont aussi utilisées sur les graphes topologiques dont les sommets sont ancrés dans le plan. Il suffit de disposer d'une fonction permettant de fusionner en ligne ou a posteriori les cartes métriques locales, déplacées suite à l'optimisation du réseau de contraintes. Ainsi, dans le cadre de la construction de modèles surfaciques, Duckett et al. ont proposé un algorithme exploitant un graphe topologique pour générer des grilles d'occupation cohérentes et de grande taille [49]. Une stratégie d'exploration relativement similaire à celle de Yamauchi et Beer [202] permet de construire en ligne une carte topologique de l'environnement, en supposant que le robot peut reconnaître les différents lieux (sommets) de la carte. A chaque sommet du graphe est attachée une grille d'occupation locale, ce qui permet d'estimer la transformation (et les incertitudes associées) entre deux sommets adjacents, par appariement des grilles d'occupation correspondantes selon une méthode par histogrammes. On obtient donc un réseau de contraintes locales. Ainsi, lorsqu'un cycle est détecté, une carte globale cohérente peut être générée : les auteurs emploient pour cela une technique de relaxation efficace, qui fonctionne en temps linéaire et peut donc être appliquée en ligne. Une fois l'exploration terminée, une grille d'occupation globale peut être calculée selon une technique d'occupation bayésienne classique [62].

Le système ATLAS de Bosse et al. propose un cadre de travail générique pour intégrer des algorithmes de cartographie robustes à petite échelle en vue d'obtenir des performances temps réel sur des environnements de très grande dimension [16]. La représentation de l'environnement est une fois de plus basée sur un graphe d'adjacence de cartes métriques locales. Le système est conçu pour être modulaire : on peut y intégrer un algorithme de cartographie locale quelconque, pourvu qu'il fonctionne en temps constant : appariement de scans appliqué aux données laser brutes, représentation basée sur des amers ponctuels, etc.

Contrairement aux algorithmes classiques, ce système n'utilise pas de repère de référence unique. Les arêtes du graphe d'adjacence global fournissent une estimation de la transformation permettant de passer du référentiel d'une carte locale au référenciel de la carte adjacente. Lorsque des transformations de coordonnées sont nécessaires entre différents repères locaux, le système recherche le chemin le plus court, c'est-à-dire le moins incertain (la métrique employée est le déterminant des covariances des transformations), dans le graphe global. Pour cela, il utilise soit l'algorithme de Dijsktra (en $O(\log n)$), soit un algorithme de recherche en largeur d'abord (en $O(1)$). Cette technique permet notamment d'estimer la position des différentes cartes locales par rapport à la position du robot pour rechercher une éventuelle fermeture de cycle. Le système offre de plus un cadre multi-hypothèses pour la localisation du véhicule puisque plusieurs hypothèses sont maintenues sur la sous-carte dans laquelle se trouve le robot (via une gestion de transitions efficace entre cartes voisines, appelée « traversal »).

Outre la projection d'incertitude sur les plus courts chemin et la gestion des hypothèses de transition entre cartes, trois autres opérations constituent la base du système de cartographie : la genèse (création d'une nouvelle carte locale), la mise en correspondance de cartes locales (pour estimer la transformation entre les deux cartes) et le raffinement des arêtes du graphe (lorsque le robot se localise de façon fiable dans deux cartes adjacentes, l'estimation de leur transformation peut être précisée par une méthode d'intersection de covariance par exemple). Enfin, un algorithme d'optimisation non-linéaire au sens des moindres carrés permet de calculer rapidement (en quelques secondes) une représentation globale cohérente par rapport à un repère unique, en appliquant des contraintes issues de la fermeture des cycles.

Finalement, on obtient des cartes de bonne qualité sur des trajets du robot de l'ordre de 2 km. L'algorithme de recherche de plus courts chemins conditionne la complexité globale de l'algorithme puisque toutes les autres opérations s'effectuent en temps constant (hormis la mise en forme finale de la carte) : la représentation est donc maintenue à un coût très faible, en complexité constante ou logarithmique. L'utilisation de cartes locales permet en outre de limiter les erreurs de linéarisation. Toutefois, comme dans l'algorithme de Constant Time SLAM de Newman [149], la fermeture de cycles n'est pas réalisée

en ligne et la cohérence de la carte n'est assurée qu'a posteriori. De plus, les auteurs ne précisent pas comment fusionner des cartes locales contenant des amers communs, si ce n'est en superposant ces amers sans les fusionner réellement.

Très récemment, Estrada et al. ont introduit une technique dite de « SLAM hiérarchique » qui vise à gérer à la fois les erreurs de linéarisation et les coûts de calculs prohibitifs de l'EKF classique [67]. Contrairement à la méthode précédente, le SLAM hiérarchique permet de maintenir en ligne des cartes globales cohérentes. Au fur et à mesure de sa progression, comme dans le cas du « map-joining » séquentiel vu précédemment [176], le robot construit des cartes stochastiques locales indépendantes les unes des autres. La taille de ces sous-cartes est bornée (au sens du nombre d'amers), ce qui permet de les construire en temps constant.

Au niveau global, le modèle maintient un graphe d'adjacence entre cartes locales dont les arcs ij représentent une relation topologique connue entre deux cartes locales M_i et M_j (la transformation permettant de passer d'un repère local à l'autre). Le système met aussi à jour une carte stochastique relative globale qui contient uniquement ces transformations relatives connues entre cartes locales et les incertitudes associées. La position du robot par rapport à toutes les cartes locales est également maintenue. Ainsi, le système est capable de détecter qu'il revient sur une carte locale connue M_i. Un algorithme de mise en correspondance de cartes [146] est alors appliqué pour définir une référence commune (une balise de type « coin » par exemple) entre M_i et la carte courante M_j et pour fusionner ces deux cartes via une opération de recollage local de cartes (« map-joining ») [176].

Ensuite, il faut imposer une contrainte de cohérence sur l'ensemble du cycle : pour limiter les erreurs de linéarisation sur les grandes boucles, les auteurs proposent une optimisation non linéaire sous contraintes. L'algorithme proposé pour cette optimisation s'appelle « sequential quadratic programming » (SQP) : les auteurs montrent qu'il est équivalent à l'EKF itératif avec une hypothèse de bruits de mesure nuls. Cet algorithme doit être itéré jusqu'à convergence, chaque itération étant linéaire en le nombre de cartes dans la boucle. En pratique, la convergence est obtenue en 2 ou 3 itérations, même pour des boucles de très grandes dimensions (jusqu'à 300 m), ce qui offre des capacités de cartographie en temps réel. Une généralisation permet également de gérer la fermeture simultanée de plusieurs cycles. Dans ce processus, pour obtenir une complexité linéaire, des méthodes de calcul efficace tirent parti du fait que certaines des matrices manipulées sont creuses, du moins avant la fermeture de cycle (par construction, la matrice de covariance de la carte relative globale est diagonale par blocs et les jacobiennes des contraintes de cohérence sont creuses).

Les auteurs précisent que la paramétrisation locale permet de limiter les problèmes de linéarisation et d'assurer la cohérence sur des cycles de grande taille, comme le montrent les expérimentations. En revanche, comme dans le cas d'ATLAS, si le même objet apparaît dans plusieurs cartes, les auteurs ne proposent pas de mécanisme permettant de le mettre à jour à partir de plusieurs observations.

Enfin, comme nous l'avons indiqué plus haut, l'algorithme « Sparse Extended Information Filter » de Thrun et al. met en œuvre des opérations spécifiques permettant de maintenir artificiellement la matrice d'information sous une forme creuse [183]. Comme l'indiquent les auteurs, on peut voir ce mécanisme comme une manière d'introduire une certaine topologie dans la carte. En effet, les coefficients non nuls représentent des liens forts entre amers voisins et l'application des opérations spécifiques permet donc de définir un réseau (un graphe) de relations fortes entre un petit nombre d'amers corrélés. En outre, contrairement aux représentations précédentes, ce réseau peut évoluer avec le temps : d'après les auteurs, cette représentation dynamique illustre donc mieux les liens entre balises au fur et à mesure de la construction de la carte, contrairement aux réseaux statiques plus classiques.

*** Plongement d'une carte métrique dans un graphe non planaire**

Howard propose une représentation originale des cartes en deux dimensions, appelée représentation « variété » [87] : au lieu de les projeter directement dans un plan comme habituellement, la carte est une surface immergée dans un espace de plus grande dimension (cf. Fig. 3.6). Dans le cadre de la cartographie monorobot, l'intérêt principal de cette représentation réside dans la gestion des boucles, dont la fermeture peut être indéfiniment retardée : les extrémités de la boucle n'étant pas situées dans le même plan, elles se chevauchent sans être nécessairement fusionnées. Le modèle facilite également d'autres capacités de navigation autonome telles que le retour sur traces robuste. Dans un cadre multirobot, des comportements actifs de fermeture de cycles peuvent être mis en œuvre, en fixant des rendez-vous entre robots par exemple : si les véhicules se retrouvent au point de rendez-vous spécifié, il s'agit bien du même lieu.

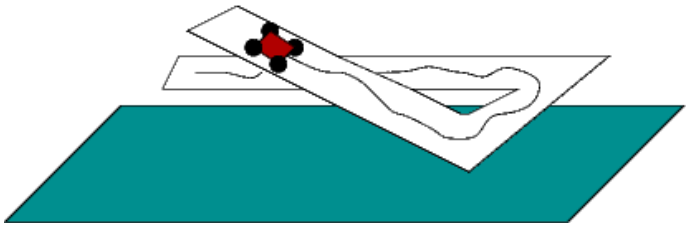


FIG. *3.6: Illustration de la représentation « variété » de Howard* [80].

L'algorithme de cartographie proposé exploite un graphe topologique de cartes métriques locales dans lequel chaque carte locale correspond à l'union de quelques polygones d'espace libre issus de scans laser. Les arêtes de la carte topologique indiquent les projections dans le plan des transformations entre cartes locales, estimées par mise en correspondance de scans. Lorsque le robot décide de fermer un cycle, il connecte topologiquement les deux extrémités du cycle, les plonge sur la même surface et réestime la position des projections des cartes locales sur le plan via l'optimisation du réseau de contraintes sur la boucle, par une méthode similaire à celle de Lu et Milios [126]. Cette technique permet de générer des cartes cohérentes d'environnements étendus : les expérimentations décrites dans l'article génèrent ainsi des grilles d'occupation de grande dimension par fusion des grilles d'occupation locales extraites des données laser.

3.2 Combinaison de différentes représentations métriques

Le mélange de différentes représentations métriques est moins répandu que l'hybridation entre cartes métriques et topologiques. Pourtant, on a vu que les différents types de représentations métriques sont complémentaires. En particulier, les représentations surfaciques fournissent des informations bien plus denses sur la forme de l'espace libre et sur la présence d'obstacles que dans le cas des balises ou des représentations basées sur l'apparence : ces informations sont essentielles dans les tâches de planification de trajectoires notamment. En revanche, les cartes constituées de primitives géométriques sont souvent beaucoup plus compactes que les autres représentations, ce qui favorise a priori les processus d'association de données. Enfin, les grilles d'occupation et les cartes de scans bruts s'adaptent plus facilement à la représentation d'objets de formes variées (courbes, objets isolés, etc.).

3.2.1 Fusion de polygones d'espace libre bornés par les frontières d'obstacles

Un modèle surfacique global de l'environnement peut être généré par fusion successive des polygones d'espace libre locaux acquis par le robot (cf. Fig. 3.7). Dans l'absolu, ces polygones locaux sont bornés

d'une part par des arêtes correspondant aux frontières d'obstacles (au-delà desquelles le capteur est occulté), et d'autre part par des arêtes « virtuelles » correspondant aux limites du champ de perception du capteur (portée maximale du télémètre, cône d'ouverture, bords d'une image, ligne de visée entre deux obstacles, etc.). Pour définir en général la forme de l'espace libre global à construire, Howard étend la notion de polygone en introduisant le concept de « polysoïde » (polygone pouvant contenir des « trous »), qui correspond à la fusion successive de polygones classiques [87].

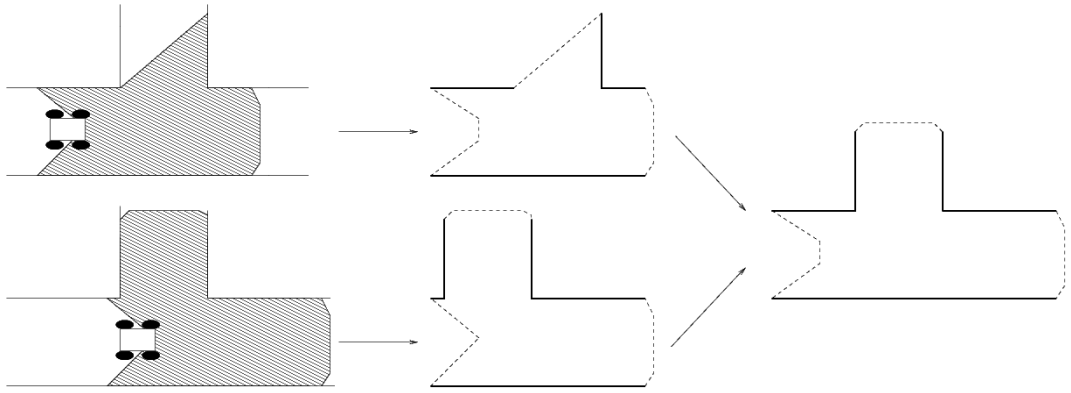


FIG. *3.7: Construction de carte globale par fusion de polygones d'espace libres locaux. Le champ de perception du robot est noté en hachuré dans les figures de gauche, qui montrent l'environnement réel. Dans les cartes locales au centre et dans la carte globale de droite résultant de la fusion des deux cartes locales, les arêtes virtuelles (limites de champ de perception du capteur) sont représentées en pointillés et les frontières d'obstacles en traits gras.*

Ainsi, dans une optique de planification (exploration par sélection incrémentale des meilleurs points de vue ou « Next Best View »), Gonzales et al. construisent une représentation de type « polysoïde » fondée sur ce principe [153]. Leur algorithme extrait de chaque scan laser local un polygone m d'espace libre f partiellement borné par des lignes polygonales p correspondant aux frontières d'obstacles. Le modèle de carte globale s'écrit $M = (P, F)$ où P est l'ensemble des lignes polygonales et F la région d'espace libre associée. Pour fusionner un nouveau polygone d'espace libre local $m = (p, f)$ à la carte globale $M = (P, F)$, l'algorithme commence par effectuer un recalage entre la représentation globale et la représentation locale, par un algorithme d'appariement de type « plus proches voisins » sur des paires de segments. Ensuite, l'espace libre de la nouvelle carte globale est calculé par union de F et f, tandis que le nouvel ensemble de contours polygonaux correspond aux arêtes solides (frontières d'obstacles) de ce nouvel espace libre. Le processus de fusion des arêtes p et P est peu détaillé dans l'article. Nous verrons toutefois que cette opération de fusion n'est pas nécessairement triviale en présence d'erreurs de mesure, comme le suggèrent d'ailleurs les résultats expérimentaux fournis. En effet, ceux-ci présentent des incohérences entre frontières d'obstacles et espace libre (les frontières d'obstacles n'apparaissent pas toujours sur les contours des polygones d'espace libre). Les auteurs précisent simplement qu'ils évitent la fragmentation de contours en fusionnant des segments adjacents presque alignés et que les petits obstacles (correspondant à des indentations étroites du polygone d'espace libre) ont tendance à disparaître lors de la fusion. Une représentation séparée gère donc ces objets de petite taille. L'article explique également que le système ne dispose pas de mécanisme permettant de prendre en compte les imprécisions et de corriger les erreurs : en particulier, il ne peut pas gérer les fermetures de cycles.

Zhang et Ghosh proposent un système de cartographie « 2,5 D » fondé sur trois types de primitives géométriques : segments, arcs de cercles et groupes de points (ces groupes de points correspondent à des

zones dans lesquelles aucun segment ni arc de cercles n'a pu être extrait des points de mesure bruts) [206]. Ces différentes primitives géométriques sont extraites de données issues d'un télémètre laser à balayage, de deux ceintures de télémètres à ultrasons et de capteurs tactiles : comme ces capteurs ne se trouvent pas dans le même plan horizontal, on obtient différentes couches de primitives : la représentation est donc qualifiée de « 2,5 D ». Pour chaque arc et chaque segment de la carte, le système retient de quel côté il a été observé. L'algorithme modélise également les arêtes « virtuelles » (qui ne correspondent pas aux frontières d'obstacle) des polygones d'espace libre locaux. Cela lui permet de construire une représentation globale de l'espace libre par union des cartes locales. Cependant, comme précédemment, l'article apporte peu de précisions sur le déroulement de l'opération de fusion. Il suppose semble-t-il que chaque primitive est fusionnée avec une primitive de même type, ce qui n'est pas acquis en présence de bruits de mesure (qui peuvent induire des erreurs d'interprétation lors de l'extraction des primitives). Ainsi, la gestion des erreurs de mesure ne paraît pas évidente dans ce système.

Dans sa thèse, Moutarlier s'est attaché à développer une méthode de construction de cartes cohérentes, mêlant primitives géométriques (lignes polygonales) et espace libre [139] [141]. L'algorithme de construction incrémentale procède également par fusion de polygones d'espace libre. La modélisation des frontières d'obstacles est réalisée par filtrage de Kalman étendu, avec une prise en compte rigoureuse de toutes les corrélations : les erreurs de mesure sont donc gérées explicitement. Toutefois, l'opération de mise à jour de l'espace libre a donné lieu à quelques difficultés : des incohérences sur ses frontières (polygones croisés...) ont dû être traitées par des méthodes *ad hoc*. De plus, le système ne semble pas capable de revenir sur certaines décisions de modélisation : par exemple, si le robot constate en s'approchant une cassure dans un mur déjà cartographié comme continu, il aura du mal à répercuter cette observation sur la carte. On constate donc qu'une fusion rigoureuse de polygones d'espace libre n'est pas si aisée, en particulier lorsque l'on souhaite prendre en compte les incertitudes et corriger les erreurs de la carte.

3.2.2 Combinaisons de grilles d'occupation et de balises

Devy et Bulata [42], de même que Hermosillo et al. [86], proposent de construire d'une part une carte d'amers géométriques pour les besoins de localisation du robot et d'autre part une grille d'occupation pour les besoins de planification de trajectoires (en vue d'explorer l'environnement ou de rejoindre un point de ralliement indiqué par l'opérateur). Toutefois, ces articles ne précisent pas comment assurer la cohérence entre les deux types de représentations, qui semblent construites indépendamment l'une de l'autre.

Pour assurer cette cohérence, une première approche consiste à construire la grille d'occupation à partir des primitives géométriques. Dans la thèse de Lee, les primitives utilisées sont des points et des segments qui sont projetés sur le plan discrétisé de la grille d'occupation [114]. Les cellules de la grille peuvent présenter quatre niveaux d'occupation différents : inconnu, occupé (cellules intersectant une primitive géométrique), dangereux (cellules correspondant à une zone de sécurité autour des obstacles d'une taille égale au diamètre du robot) ou libre. Lee explique comment définir les zones occupées de la grille à partir des primitives géométriques et comment définir les cases libres de façon cohérente, sans intersecter les obstacles. Toutefois, la thèse ne précise pas comment corriger les cellules de la grille lorsque la position des balises est remise à jour. En effet, les cellules occupées et dangereuses doivent alors être déplacées : faut-il recalculer l'ensemble de la grille en utilisant des contraintes de cohérence ou bien existe-t-il un algorithme plus incrémental ? De plus, l'estimation de position des balises fait appel à un simple moyennage : les modèles d'incertitude paraissent très simplifiés et le système ne gère pas les corrélations, ce qui ne permet pas, en principe, de traiter correctement les fermetures de cycles de l'environnement.

L'approche inverse consiste à construire les frontières d'obstacles à partir des cartes d'occupation. Ainsi, Schultz et al. utilisent des grilles d'occupation bayésiennes locales pour effectuer des tâches d'évitement d'obstacles et de localisation, et des grilles globales pour réaliser une planification de trajectoire [169]. Pour cela, ils déduisent de la grille les frontières d'obstacles et les limites entre espace libre connu et espace inconnu (arêtes « virtuelles ») permettant de définir des stratégies d'exploration. Il s'agit en fait d'une approche analogue à celle de Yamauchi et al. pour l'exploration basée sur les frontières : chez Yamauchi et al., ces limites entre espace libre connu et espace inconnu sont extraites par détection de contours et par extraction de régions via des algorithmes de traitement d'images [201]. Cette extraction a posteriori permet d'assurer la cohérence entre l'espace libre et ses frontières mais il n'est manifestement pas prévu qu'elle se déroule de façon réellement incrémentale, avec mise à jour en fonction des corrections apportées par la cartographie.

Très récemment, Nebot et al. ont proposé une nouvelle représentation qui permet de construire de manière incrémentale et cohérente une représentation surfacique de l'environnement s'appuyant sur des primitives géométriques [79]. L'espace est segmenté en triangles dont les sommets correspondent à des balises ponctuelles. Chaque triangle peut contenir une représentation dense de type grille d'occupation, modélisée dans le repère local défini par les trois balises et dont la cohérence locale peut facilement être assurée. Ensuite, la cohérence d'ensemble de la représentation est gérée via la mise à jour des balises, à travers un filtre de Kalman par exemple. De cette manière, les corrélations entre positions de balises et position du véhicule étant maintenues, il est possible de générer des représentations globalement cohérentes. Cette représentation semble particulièrement bien adaptée aux environnements extérieurs peu structurés. Dans le cas des environnements intérieurs polygonaux, on peut se demander comment gérer un segment (frontière d'obstacle) à cheval sur deux triangles adjacents (cf. Fig 3.8). En effet, une mise à jour de la position des sommets de ces triangles pourrait entraîner une cassure du segment (configuration (a) de la figure 3.8). Si l'on empêche cette cassure en contraignant le segment à rester rectiligne et en ne mettant à jour que ses extrémités, on risque alors de provoquer un décalage entre ce segment et les régions qu'il délimite (représentées via des grilles d'occupation par exemple : cf. configuration (b) de la figure 3.8).

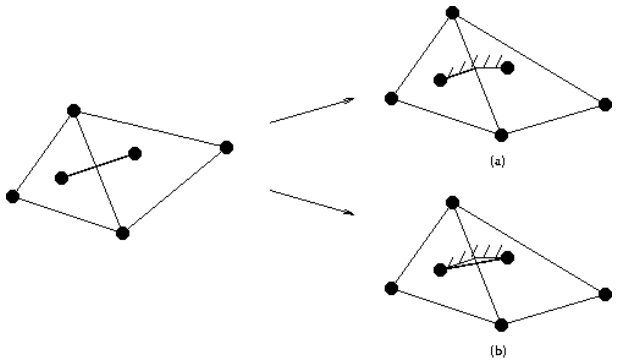


FIG. *3.8: Illustration du problème de cohérence qui pourrait survenir si un segment se trouvait à cheval sur deux régions triangulaires voisines. Les hachures correspondent à la matière d'un obstacle.*

3.2.3 Structuration progressive en ligne ou a posteriori

Wang et Thorpe proposent une approche hiérarchique basée sur différents paradigmes de représentations métriques de l'environnement [194]. La méthode est qualifiée de hiérarchique car elle opère successivement sur trois niveaux de structuration différents, correspondant respectivement à des représentations basées

sur l'apparence, à des grilles d'occupation et à des cartes d'amers géométriques. Au plus bas niveau, les scans lasers sont mis en correspondance afin de créer des cartes locales. Pour apparier ces scans et calculer les transformations géométriques permettant de passer de l'un à l'autre, l'algorithme utilisé fonctionne directement sur les points de mesures et recourt à une technique d'ICP (cf. chapitre 5). Pour estimer les incertitudes sur les transformations calculées, les auteurs proposent un échantillonnage sur les hypothèses de position initiales du scan local par raport au scan final (exploitant les informations odométriques), ce qui produit un ensemble d'échantillons finaux résultant de la mise en correspondance effective entre scans. En outre, pour prendre en compte les bruits de mesure du capteur, chaque scan est transformé en une grille d'occupation élémentaire et les échantillons finaux sont pondérés par le degré de corrélation entre grilles d'occupation élémentaires recalées. Cet ensemble d'échantillons pondérés représente la distribution d'erreur finale sur le recalage. Ensuite, les grilles d'occupation élémentaires provenant de scans voisins sont fusionnées pour obtenir une carte surfacique locale : ces grilles d'occupation locales correspondent à un niveau de représentation supérieur. Enfin, pour générer des cartes à grande échelle et pour gérer les cycles de l'environnement, les auteurs passent à un troisième niveau de représentation : une carte d'amers construite par filtrage de Kalman étendu. Au départ, il était prévu d'extraire des primitives géométriques à partir des grilles d'occupation. Pour simplifier le processus, chaque carte locale est considérée comme un amer unique et le filtrage de Kalman assure la cohérence de cette carte globale. Finalement, on obtient des cartes de milieu urbain extérieur de la taille d'un quartier entier. Toutefois, la simplification mise en œuvre au niveau de la carte d'amers ne permet pas de mettre à jour l'intérieur des grilles d'occupation locales, ce qui peut engendrer des problèmes de cohérence sur les zones de superposition de cartes locales. Selon les auteurs, cela reste suffisant pour les applications de planification de trajectoires envisagées.

D'autres approches proposent une structuration a posteriori des cartes obtenues. Par exemple, de nombreux systèmes transforment les représentations à base de scans laser en grilles d'occupation pour faciliter la lisibilité du « plein » et du « vide » et pour faciliter la planification de trajectoire par le robot. Un article récent explique également comment extraire des lignes polygonales à partir de cartes formées de scans laser bruts recalés [190].

3.2.4 Combinaisons de représentations métriques augmentées d'informations topologiques

Enfin, quelques rares publications proposent de combiner à la fois différentes représentations géométriques et des informations topologiques.

Nous avons vu que la représentation proposée initialement par Chatila et Laumond comporte plusieurs niveaux, en particulier un niveau géométrique et un niveau topologique (graphe d'adjacence de lieux) [26]. La représentation métrique est d'abord composée de primitives géométriques regroupées en objets de plus haut niveau. On remarque que l'introduction d'objets ou même de lignes polygonales implique déjà une certaine augmentation topologique au sens où l'on définit des relations d'adjacence (entre segments dans les lignes polygonales ou entre primitives d'un même objet), voire des sous-cartes (objets) reliées les unes aux autres par les transformations géométriques adéquates. Quant au processus de structuration topologique, il fait appel à une partition de l'espace libre selon les lignes de visibilité, cet espace libre (représentation métrique surfacique) étant construit de façon incrémentale par fusion des polygones locaux. D'autres partitions plus macroscopiques peuvent être définies dans des niveaux topologiques supérieurs : les sommets du graphe d'adjacence correspondent alors à des lieux tels que des pièces ou des couloirs par exemple.

Bulata et Devy ont implémenté un système de ce type [18]. Au niveau métrique, la représentation contient des primitives géométriques (points et segments) qui sont regroupées en objets de plus haut

niveau (objets composés d'ensembles de points et de segments que nous avons vu au chapitre précédent : cf. Fig. 2.6) : ces primitives sont seulement référencées dans le repère local de l'objet correspondant. Au niveau symbolique se trouvent des régions correspondant à des entités sémantiques (pièces, couloirs...) ou à des zones de taille plus limitée en raison des limites de portée des capteurs : ces régions constituent des repères locaux pour les objets qu'elles contiennent. Enfin, au niveau topologique, la représentation peut être assimilée à un graphe reliant les régions du niveau symbolique via des passages particuliers (portes...). Les positions de ces régions sont définies par rapport au repère de référence global via des repères locaux centrés sur les passages.

Enfin, dans l'optique de faciliter la navigation du robot, outre la construction d'un modèle surfacique et d'une carte d'amers par fusion de polygones d'espace libre locaux, Moutarlier a également ajouté un niveau topologique en structurant l'espace libre selon un graphe d'aspect (fondé sur l'espace navigable et sur le calcul du graphe de visibilité), qui s'appuie sur les primitives géométriques constituant les frontières d'obstacles [139].

3.3 Un cadre formel hiérarchique pour la combinaison de différents types de cartes

Dans sa thèse, Diard propose une représentation inspirée de la méthodologie de programmation bayésienne pour les robots (BRP ou « Bayesian Robot Programming ») [43]. Selon cette méthodologie, un programme est une structure composée de deux parties [44] :

- une composante déclarative, dans laquelle l'utilisateur définit une description. L'objectif d'une description est de spécifier une distribution conjointe $P(X_1\, X_2 ... X_n \mid \delta\, \Pi)$ sur un ensemble $\{X_1, X_2, ..., X_n\}$ de variables, étant donné un ensemble δ de données expérimentales et des connaissances préalables Π. Pour spécifier cette distribution, le programmeur commence par lister les variables pertinentes (ainsi que leur domaine), puis décompose la distribution conjointe en un produit de termes plus simples (en précisant éventuellement des hypothèses d'indépendance conditionnelle permettant de simplifier le modèle ou le calcul), pour finalement assigner à chacun des facteurs une forme particulière (formes paramétriques directes ou questions récursives vers d'autres programmes bayésiens). Tous les paramètres libres doivent être estimés : soit ils sont fournis a priori par le programmeur, soit ils sont calculés par le biais d'un mécanisme d'apprentissage (spécifié par le programmeur) à partir de données expérimentales ;

- une composante procédurale, qui consiste à exploiter la description de la composante déclarative pour formuler une question, c'est-à-dire pour calculer une distribution de probabilité de la forme $P(X_{Recherche} \mid X_{Connu})$. Répondre à une question consiste à décider d'une valeur pour la variable $X_{Recherche}$ selon $P(X_{Recherche} \mid X_{Connu})$. Ce processus fait appel aux techniques d'inférence bayésiennes qui peuvent parfois s'avérer difficile à mettre en œuvre.

Selon ce formalisme, une carte bayésienne c est une description qui définit une distribution conjointe $P(P\, L_t\, L_{t'}\, A \mid c)$ dans laquelle :
- P est une variable de perception (le robot lit ses valeurs à partir de capteurs physiques ou de variables de plus bas niveau) ;
- L_t est une variable de localisation à l'instant t (cette localisation correspond à un ensemble de lieux possibles pour le robot) ;
- $L_{t'}$ est une variable de même domaine que L_t, mais à un temps t' (on suppose $t' > t$) ;

– A est une variable d'action (qui correspond aux commandes du robot).

L'intérêt de ce formalisme très général est qu'il reste particulièrement ouvert : la forme de la décomposition n'est pas contrainte (toute structure de dépendance probabiliste peut être utilisée), de même que les formes des distributions de probabilité (gaussiennes, particules, etc.) et les mécanismes d'apprentissage. En particulier, le programmeur choisit un ensemble de lieux pertinents pour accomplir la tâche considérée, dans la classe d'environnements que le robot est susceptible de rencontrer : ainsi, le choix de la nature de ces lieux (métriques ou topologiques, denses ou espacés...) ne doit être qu'une conséquence de ces considérations.

Par ailleurs, pour être utile, une carte bayésienne doit permettre de générer des comportements. Un comportement élémentaire correspond à une question du type $P(A^i \mid X)$ où A^i est un sous-ensemble de A et X un sous-ensemble des autres variables de la carte (c'est-à-dire qui ne se trouvent pas dans A^i). Des comportements plus complexes peuvent par exemple être bâtis comme des séquences de comportements élémentaires. En particulier, il est utile de disposer de cartes qui répondent de manière pertinente (c'est-à-dire par le biais de distributions de probabilité informatives, d'entropie élevée et donc différentes de distributions uniformes) à trois questions principales : la localisation ($P(L_t \mid P)$), la prédiction ($P(L_{t'})$) et le contrôle ($P(A \mid L_t\, L_{t'})$).

Il existe en outre un opérateur dit de « superposition », qui assemble deux cartes en superposant les lieux des cartes sous-jacentes : dans ce cadre, le robot se localise simultanément dans les deux cartes, ce qui lui permet d'enrichir son vocabulaire de description de son interaction avec l'environnement. Il existe également un opérateur « d'abstraction », permettant de construire une carte c plus élaborée, dont les « lieux »(positions de localisation pour la variable L_t) sont d'autres cartes c^1, c^2,..., c^n. Pour relier entre elles différentes cartes (différents lieux de la carte abstraite), les comportements a^1, a^2,..., a^k des cartes élémentaires peuvent être utilisés. Ainsi, la carte abstraite exploite un espace interne de taille limitée, chaque carte sous-jacente étant réduite à un symbole et les « détails » n'ayant pas besoin d'être précisés. Enfin, la variable de perception P de la carte abstraite correspond à la liste des variables P^i, L^i_t, $L^i_{t'}$ et A^i des cartes élémentaires. Concernant les détails de la mise en œuvre des décompositions et des questions, le lecteur pourra se référer à [44] par exemple. En particulier, pour décider de ses actions, le robot n'a pas besoin de savoir exactement dans quelle carte il se trouve : le processus considère toutes les possibilités, qui sont pondérées selon leur probabilité.

Une validation expérimentale proposée par Diard et al. [44] propose une application simple dans laquelle le robot cherche à se cacher dans les coins, comme s'il fuyait l'espace libre. Pour cela, trois situations sont considérées : soit le robot se situe près d'un mur et doit le longer pour rejoindre le coin le plus proche, soit le robot est déjà dans un coin et doit rester immobile, soit il se trouve au milieu de l'espace libre et avance tout droit pour quitter aussi vite que possible la zone exposée. Ces situations sont représentées par des cartes bayésiennes élémentaires dont les perceptions sont liées aux mesures sonar acquises par le robot, tandis qu'une carte abstraite est construite sur la base de ces représentations élémentaires. L'analyse des probabilités de positionnement du robot dans la carte abstraite permet alors de tracer sur un modèle métrique les zones de l'environnement où chaque situation (chaque carte élémentaire) est dominante : on constate que les coins, les murs et les espaces libres sont bien reconnus sur la base des informations sensorimotrices.

Ainsi, dans cette application, la carte obtenue dépasse la simple distinction entre représentations métriques ou topologiques : il s'agit plutôt d'une représentation fondée sur des situations sensorimotrices. Toutefois, le formalisme très général des cartes bayésiennes hiérarchiques permet a priori de représenter de multiples combinaisons de modèles (métriques ou topologiques) : ensembles ou réseaux de lieux ou

de cartes locales (via la notion d'abstraction), superpositions de différentes couches de représentations (via l'opérateur de superposition), etc. Ce formalisme inclut également les actions possibles pour le robot (comportements sensori-moteurs notamment), permettant de relier ensemble différents lieux ou sous-cartes (constituant ainsi potentiellement des arêtes de graphes topologiques) : il se rapproche donc des systèmes de navigation basés sur les PDMPO [173] [69] [184], qui constituent d'ailleurs un cas particulier des cartes bayésiennes. Leur mise en œuvre (et, à notre connaissance, les expérimentations réalisées) correspond(ent) cependant a priori à une problématique de navigation de robot autonome, plutôt qu'à la construction d'une représentation qui soit également bien lisible pour un opérateur humain et n'induise pas de contraintes sur le déplacement du robot (ce qui correspond plus à l'objectif de notre thèse). La question de la construction automatique de ce type de cartes (notamment les hiérarchies) semble également devoir être précisée pour les nouvelles applications considérées (même si la problématique d'apprentissage est déjà abordée dans la thèse de Diard [43]).

3.4 Modèles géographiques

Les modèles de cartes employés dans les systèmes d'information géographiques (SIG) sont rarement mentionnés dans la communauté robotique, bien qu'ils fournissent des idées intéressantes pour la définition des formats de cartes. En particulier, ils introduisent une plus grande structuration des modèles, susceptible de permettre aux robots de réaliser des raisonnements spatiaux de plus haut niveau. On ne pourra cependant pas décrire réellement d'algorithmes de construction de ces cartes géographiques : en effet, à l'heure actuelle, ces cartes sont souvent construites manuellement, même s'il existe des aides semi-automatiques (détection de réseaux routiers dans des images aériennes, traitement de données de type « raster » pour extraire des vecteurs, etc.). Toutefois, nous verrons que certaines structures de données sont mieux à même que d'autres à gérer les mises à jour.

3.4.1 Utilisation des données géographiques : des similitudes avec la robotique ?

Selon David, l'utilisation classique des données géographiques peut être divisée en deux catégories [39] :

- prévision, planification, aide à la décision (en général, les modèles utilisés sont à petite échelle) ;
- gestion, inventaire, suivi (à plus grande échelle).

On peut retrouver des applications similaires dans le domaine de la robotique en ce qui concerne la planification ou le suivi de mission par exemple. En outre, dans un contexte de gestion de crise (civile ou militaire), l'aide à la décision, la prévision, la planification, la gestion et l'inventaire peuvent avoir un intérêt en vue d'établir la situation tactique et de prévoir les mouvements des forces sur un fond cartographique, qui pourrait être fourni en partie par des robots.

Plus précisément, concernant les traitements géographiques, on peut distinguer des classes de complexité variable [39] :

- **traitements locaux** : il s'agit de traitements qui s'effectuent pixel par pixel (ils sont souvent proches du traitement d'image). Parmi les exemples, on peut citer la sélection spatiale ou thématique, la correction géométrique, le calcul d'histogrammes, le filtrage, la corrélation entre images, etc.

- **traitements élémentaires** : ces opérations s'effectuent cette fois élément géographique par élément géographique. Il s'agit par exemple de calculs de longueur, de périmètres ou de surfaces, de calculs d'indices de connexité ou de convexité, d'intersection géométrique avec une figure donnée, d'opérations d'agrégat spatial (union géométrique, somme d'attributs pondérés par leur surface ou

leur longueur...), de superposition (« map overlay »), etc. Cependant, ces traitements n'incluent pas de calculs tenant compte du voisinage (adjacence...) car ceux-ci font partie de la classe suivante.

– **traitements de voisinage** : dilatation, agrégation spatiale tenant compte de l'adjacence pour trouver des agrégats connexes (par exemple redécoupage c'est-à-dire recherche de partition spatialement connexe avec agrégation de certaines caractéristiques alphanumériques vérifiant certains critères), recherche opérationnelle, principalement dans le domaine de la théorie des graphes (plus court chemin dans un réseau ou un découpage, optimisation de tournée, analyse de flux, calculs d'intervisibilité, choix d'emplacement de magasins, etc.).

Ces opérations peuvent encore être regroupées par type de traitement selon cinq catégories : projection, sélection, agrégat, jointure, opérations de niveau supérieur. Enfin, le document de description du format VPF cite pour sa part 8 opérations basiques supportées par le modèle relationnel des cartes vecteur [151] : sélection, projection, produit, jonction, union, intersection, difference, division. Ces opérations peuvent éventuellement être combinées pour former des opérations plus complexes (en particulier : requête, création, suppression, modification). Le modèle relationnel correspondant est fondé sur la théorie des ensembles : les opérations citées sont elles-mêmes basées sur des principes mathématiques fondamentaux. Elles permettent de manipuler et de créer les objets de la base de manière spécifique, menant à un résultat stable.

On devine que la plupart de ces opérations très élémentaires peuvent potentiellement être invoquées dans le cadre de la planification et de la navigation des robots (ou dans un contexte de planification d'opérations de défense à base de cartes fournies par systèmes robotisés par exemple).

Pour préciser cette impression, nous pouvons tenter de caractériser les traitements impliquant des données géographiques qui interviennent dans le cadre de la robotique mobile. Dans un contexte applicatif plutôt orienté vers la navigation en milieu ouvert, inspiré des véhicules développés dans le cadre de programmes lancés par la DGA (Délégation Générale pour l'Armement), Larue propose une typologie des traitements utilisés en robotique [111] :

– **préparation de mission** : recherche opérationnelle dans des graphes (recherche de plus courts chemins sur le réseau routier...), calculs d'intervisibilité ;

– **planification locale** : découpage du terrain en zones navigables ou non navigables (obstacles) sous la forme d'un graphe planaire (ce qui ressemble au SLAM). Sur ce graphe sont réalisés des traitements de calcul de visibilité et de recherche opérationnelle pour trouver le meilleur chemin possible, sous la forme d'un ensemble de zones navigables. Pour effectuer le zonage de l'espace, les réseaux routiers sont traités comme des réseaux surfaciques : si dans les données de base, le réseau est représenté par des objets linéaires, il faut effectuer un pré-traitement exploitant les caractéristiques sémantiques des routes qui le composent (importance, nombre de voies...) afin de construire le réseau surfacique correspondant ;

– **navigation / guidage** : traitements d'intervisibilité (GPS par exemple, voire sur les communications) ;

– **perception** : les algorithmes correspondants travaillent sur le sur-sol (objets tels que des bâtiments placés sur un modèle numérique de terrain) et peuvent faire intervenir des images d'objets de la base vus par les capteurs du robot.

En fait, toujours selon Larue [111], ces traitements peuvent être répartis selon cinq classes : sélection des objets géographiques utiles au traitement robotique, traitements de recherche opérationnelle, algorithmes d'intervisibilité, traitements de projection (transformation des données de la base dans le format de l'applicatif robotique demandeur), zonage (partition) selon une propriété (espace libre ou occupé par un obstacle par exemple). Ainsi, selon la typologie des traitements de bases de données, on peut distinguer : l'extraction (qui isole l'information brute), le calcul (recherche opérationnelle, intervisibilité, zonage) et le reformatage (transcription des données dans un format applicatif). Larue précise également que les besoins en terme de modèle interne pour les données robotiques se déclinent selon les actions suivantes : modéliser l'ensemble des données représentées dans des espaces différents, exprimer des constantes spatiales, effectuer des traitements sous forme de prédicats et de fonctions, mettre à jour efficacement la base de données au rythme des besoins (activité de renseignement). On constate donc que tout ceci fait potentiellement appel à des traitements élémentaires similaires à ceux cités ci-dessus pour l'utilisation plus classique des données géographiques (opérations booléennes, extraction, requêtes sur la base d'informations sémantiques...). Nous donnerons quelques exemples pratiques de ces opérations au chapitre suivant.

3.4.2 Différents formats de cartes géographiques

Comme le souligne David [39], pour représenter l'association entre des informations alphanumériques et leur localisation, il est nécessaire de découper l'espace ou le plan en éléments auxquels seront associés une localisation et une description alphanumérique. Il existe plusieurs manières de réaliser ce découpage, ce qui conduit à distinguer [39] [111] :

- **les modèles raster**, d'une part, qui sont proches des grilles d'occupation (tessellations) ;
- **les modèles vecteurs**, d'autre part, qui font intervenir divers types de primitives géométriques de dimension variable : objets ponctuels, linéiques (lignes polygonales) et surfaciques (régions).

Les modèles raster et les tessellations correspondent à des découpages a priori du plan : le découpage le plus utilisé est une décomposition régulière en carrés ou en rectangles (pixels), chaque cellule étant associée à une description alphanumérique (intensité lumineuse, code correspondant à un élément topographique...). De simples structures matricielles peuvent ainsi être employées pour représenter ces pavages réguliers. Dans le but de compacter ces tessellations, on peut également recourir à divers types de représentations plus économiques en terme de place mémoire [39] : codage par plages, quadtree, clés de Péano, etc. Larue [111] cite également un certain nombre de maillages irréguliers : polygones de Voronoï, maillages hybrides, etc. On retrouve donc à travers ce modèle des représentations très proches des grilles d'occupations utilisées classiquement en robotique, ainsi que les modèles hiérarchiques associés [205] [156].

Les modèles vecteurs, quant à eux, localisent l'information à travers des figures géométriques simples définies dans un plan ou un espace cartésien. Ils font appel à différents niveaux de représentations des ensembles de primitives spatiales, qui diffèrent essentiellement sur l'expression des relations topologiques entre composantes d'objets [39] [165] [151] :

- **les modèles appelés « spaghettis »** (niveau 0 de topologie pour le format VPF), qui ne gardent en mémoire aucune topologie et considèrent les primitives de la carte comme indépendantes les unes des autres ;

- **les modèles dits de « réseaux »** (niveau 1 de topologie), qui furent d'abord définis pour représenter les réseaux dans les applications basées sur les graphes, pour les services de transport par exemple ;

– **les modèles de graphes planaires** (niveau 2 de topologie), qui sont similaires aux modèles de réseaux mais qui imposent en outre que le réseau soit un graphe planaire. Des contraintes d'intégrité doivent être vérifiées lorsque les entités topologiques sont plongées dans le plan : les arêtes doivent s'intersecter sur un sommet et les faces ne doivent pas se superposer ;

– **les modèles « topologiques »** (niveau 3 de topologie) : le graphe planaire induit une subdivision planaire en faces contiguës et le modèle contient les informations d'adjacence qui relient les différents éléments (ponctuels, linéaires et surfaciques) entre eux.

Ainsi, outre les représentations fondées sur des tesselations, la plupart des modèles de cartes utilisées actuellement en robotique mobile correspondent aux spaghettis (modèles basés sur les primitives géométriques de type points, segments, etc.). On trouve également quelques représentations en réseau (basées sur un diagramme de Voronoï, bien que souvent, les sommets et les arcs du graphe correspondant ne soient pas explicitement localisées dans l'espace) et quelques très rares représentations métriques assorties de contraintes d'intégrité ([114]). De plus, peu de représentations mélangent ces deux types de données, or comme le souligne Larue [111], les deux sont indispensables à des traitements robotiques.

3.4.3 Intérêt de la couche topologique

La topologie étudie les relations entre des objets localisés dans l'espace qui sont invariantes par des transformations topologiques appelées homéomorphismes. Un homéomorphisme est une bijection continue, ainsi que son inverse, d'une région de l'espace dans elle-même : intuitivement, il peut être vu comme une déformation élastique de l'espace dans laquelle il n'y a ni déchirure, ni pliage du plan sur lui-même. Dans les SIG, selon David, la topologie permet ainsi d'exprimer des relations entre entités géographiques qui sont indépendantes des transformations topologiques, comme par exemple l'inclusion d'un objet dans un autre, l'intersection d'objets ou l'adjacence entre objets [39]. Plus concrètement, comme l'indique la spécification du format VPF [151], la couche topologique a pour but de maintenir la connaissance des relations thématiques et spatiales entre cellules voisines. Pour garantir la validité d'un modèle topologique, ces relations doivent être préservées au-delà des changements d'échelle, de forme ou de taille.

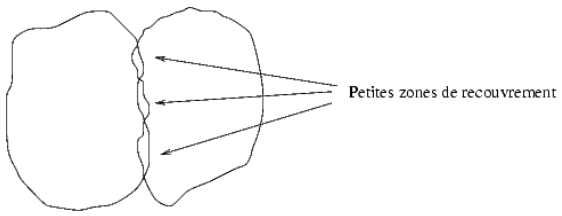


FIG. *3.9: Exemple de « slivers » : apparition de petites régions à la frontière des polygones adjacents, dues à une mauvaise supersposition des arêtes communes.*

Initialement, selon [39] et [165], la couche topologique des modèles géographiques visait essentiellement à limiter les problèmes d'imprécisions géométriques, tels que les « slivers », de petites régions qui apparaissent dans la carte suite à une mauvaise supersposition des arêtes de polygones adjacents (cf. Fig. 3.9). En effet, ces problèmes d'imprécision rendent difficile tout traitement fondé sur les relations de proximité spatiale tels que l'intersection, l'inclusion ou l'adjacence. Les structures de données topologiques permettent d'y remédier en introduisant des données redondantes par rapport à la géométrie, afin de définir des relations de proximité entre éléments [135]. Ainsi, cette couche topologique peut être critiquée du fait

qu'elle introduit souvent une certaine redondance par rapport aux informations purement géométriques. En réalité, elle s'avère très efficace pour la manipulation des cartes et leur mise à jour, car elle stocke en mémoire de l'information pré-calculée, qui n'a plus à être réévaluée systématiquement à partir de la géométrie [39]. Par exemple, les requêtes classiques dans les systèmes d'information géographique font intervenir des opérations booléennes (union, intersection, différence...) qui tirent avantageusement parti des relations topologiques. Le document de description du format VPF souligne d'ailleurs que de nombreuses opérations topologiques complexes peuvent être dérivées des seules relations d'adjacence [151]. La structure topologique permet également d'introduire plus facilement de la sémantique dans le modèle [39] (bonne définition des objets, étiquetage possible de toutes les entités topologiques, opérations booléennes pouvant faire intervenir ces informations sémantiques...).

Ainsi, cette structuration peut également se révéler utile pour les applications de robotique. Elle peut être utilisée lors du processus de cartographie et de localisation simultanées pour vérifier la cohérence des données. Elle peut aussi être exploitée pour les raisonnements spatiaux globaux, en vue de planifier les actions du robot et les tâches de navigation. Dans sa thèse, qui vise à développer un modèle de serveur de données géographiques pour robot mobile, Larue souligne justement que les avantages de la couche topologique sont nombreux [111] : outre l'optimisation de certains traitements topologiques, elle offre la possibilité de garantir la cohérence spatiale de la base de données géographique lors de l'évolution des données (elle garantit le partitionnement notamment). Cette dernière propriété peut s'avérer très utile lors d'une construction incrémentale de carte d'environnement, effectuée par un robot mobile au cours de sa progression. Larue n'a cependant pas opté pour une représentation des données selon un tel type de modèle topologique. En effet, selon lui, les principales limitations de ces modèles sont les suivantes :

- De nombreuses applications nécessitent de définir différentes couches thématiques qui correspondent chacune à un partitionnement différent de l'espace. Or les traitements topologiques optimisent les calculs à l'intérieur d'une même couche mais pas entre deux couches différentes. Le croisement des deux couches nécessite une opération coûteuse de superposition (« map overlay ») qui impose de recalculer toutes les relations topologiques pour l'intersection des deux couches.
- Des considérations similaires peuvent être appliquées au cas de la superposition d'une couche topologique avec une primitive donnée en paramètre au système (nouvel objet non contenu dans la base : zone d'intervisibilité, zone de contamination chimique...).

Cependant, ces difficultés sont en grande partie levées dès lors qu'on dispose d'une fonction de superposition performante qui permette de fusionner ces différentes couches en une seule couche cohérente. Or comme nous le verrons dans le chapitre suivant, les travaux de Cazier [23] ont permis de définir un algorithme très performant pour l'opération dite de « raffinement », qui effectue la fusion de plusieurs cartes topologiques : la complexité de cet algorithme est approximativement en $O(n \log n)$, n étant le nombre d'arcs dans la structure de carte. En fonction du nombre de couches thématiques à fusionner, il demeure certes un risque de « sur-segmentation », au sens où l'espace risque d'être découpé en cellules de très petite taille, mais ce point est à vérifier plus finement en fonction des tâches envisagées pour le système robotisé considéré.

Il existe plusieurs manières d'introduire ces informations topologiques [39]. Il faut d'abord fragmenter tous les éléments géométriques en éléments topologiques élémentaires de manière à ce qu'un élément topologique n'en intersecte pas un autre (par exemple, si deux lignes s'intersectent, alors elles sont chacune divisées en deux élements linéiques appelés arcs et se coupent en un élément ponctuel appelé nœud). On peut ainsi décrire différentes structures de données :

- **structure de graphe planaire** [39] (cf. Fig 3.10) : les arcs et les nœuds sont agencés en un graphe, qui, de par sa planarité, définit également des faces. Les éléments géométriques initiaux

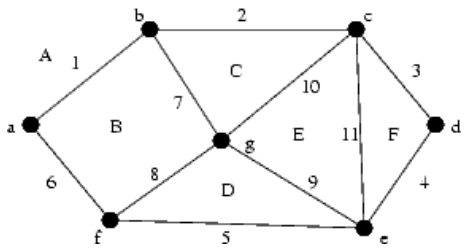


FIG. *3.10: Exemple de graphe planaire. Dans cette figure, les faces sont notées en lettres majuscules, les sommets en lettres minuscules et les arêtes sont numérotées.*

sont transformés en éléments topologiques. Un élément ponctuel est soit un nœud du graphe, soit un élément isolé. Un élément linéaire est représenté par un chemin du graphe composé d'arcs. Un élément surfacique est représenté par un ensemble de faces. Ces données sont étiquetées et structurées dans un tableau d'arcs (indiquant les étiquettes du nœud initial, du nœud final, de la face droite et de la face gauche), un tableau de nœuds (indiquant les coordonnées cartésiennes du ponctuel correspondant) et éventuellement un tableau de faces. David souligne que cette représentation est plus théorique que pratique, car elle présente divers inconvénients [39]. D'abord, le graphe n'est pas forcément connexe (en particulier, pas de gestion des « trous » dans les faces). En outre, cette structure ne permet pas de retrouver facilement la définition géométrique d'une face : par exemple, pour retrouver le contour d'une face, il faut parcourir l'ensemble du tableau des arcs pour trouver ceux dont la face droite ou la face gauche correspond à cette face, puis trier cet ensemble d'arcs pour obtenir une liste dans l'ordre du contour (comparaison des nœuds finaux et initiaux).

– **structure d'arêtes ailées** introduite par Baumgart en 1970 [9] dans le contexte de la vision par ordinateur : le format très répandu des cartes vecteurs VPF (Vector Product Format) [151] est basé sur cette représentation. Chaque arête de cette structure est représentée par des pointeurs orientés vers chacun de ses deux sommets, les deux faces partageant cette arête, et vers quatre arêtes émanant de ses sommets [72] (cf. Fig 3.11) . En outre, chaque sommet possède un pointeur vers l'une des arêtes dont il émane et chaque face pointe vers l'une de ses arêtes. Par rapport aux graphes, cette représentation a notamment l'avantage de permettre un calcul très rapide du contour d'une face, avec un coût de parcours proportionnel au nombre d'arêtes formant le parcours de la face.

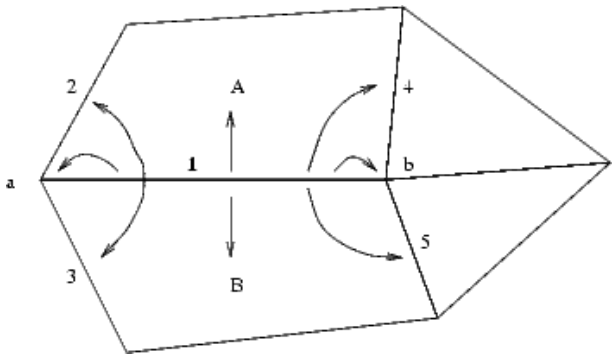


FIG. *3.11: Structure d'arêtes ailées pour l'arête 1. Dans cette figure, les faces sont notées en lettres majuscules, les sommets en lettres minuscules et les arêtes sont numérotées.*

– **structures DCEL** (« Doubly connected edge lists ») [160] [39] (cf. Fig 3.12) : dans cette structure, un arc est orienté et correspond à un segment de droite défini par deux points et les arcs sont

référencés entre eux par une double liste : chaque arc pointe vers un arc suivant et un arc précédent. L'arc suivant est le premier arc qui sort du nœud final en tournant autour de ce nœud dans le sens trigonométrique. De même, l'arc précédent est le premier arc qui sort du nœud initial dans le sens trigonométrique. De plus, un signe est associé à la référence pour donner le sens de l'arc suivant (resp. précédent) : positif si le nœud final (resp. initial) de l'arc courant est le nœud initial de l'arc suivant (resp. précédent), négatif dans le cas contraire. Cette structure est aisée à représenter par une table d'arcs étiquetés, l'orientation étant indiquée par un signe positif ou négatif. Le numéro signé référence l'une des deux orientations possibles pour un arc : chacune de ces orientations correspond à un brin (chaque arc est donc composé de deux brins, correspondant chacun à l'une des deux orientations possibles de l'arc).

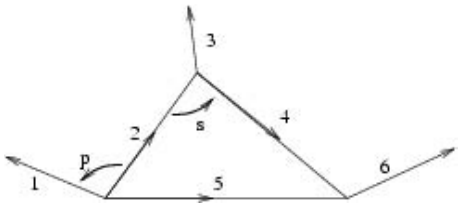


FIG. *3.12: Exemple de structure DCEL (« Doubly Connected Edge List »). L'arc* 2 *pointe vers l'arc suivant (*4*) via le pointeur s et vers l'arc précédent (*1*) via le pointeur p.*

On peut ainsi définir la fonction « brin suivant » notée ϕ de la manière suivante (cf. Fig 3.13) : pour $b > 0$, $\phi(b) = suivant(b)$ et $\phi(-b) = précédent(b)$. L'application itérative de la fonction ϕ définit des cycles de brins qui correspondent aux faces. De même, l'application itérative de la fonction $\sigma = \phi^{-1}$ définit des cycles de brins qui correspondent aux nœuds. Enfin, si l'on définit la fonction α qui associe à un nombre son opposé, les arêtes peuvent être vues comme des cycles de cette fonction. Par rapport aux graphes, cette représentation permet également un calcul très rapide du contour d'une face, avec un coût de parcours proportionnel au nombre d'arêtes. Par rapport aux arêtes ailées [9], la manipulation de cette représentation est mieux formalisée. En effet, si la notion de graphe a l'avantage de définir une base théorique bien définie et largement utilisée, il existe aussi une théorie, dite des « cartes combinatoires », qui correspond aux concepts qui viennent d'être décrits. De plus, toute carte définissant un graphe sous-jacent, tous les concepts de la théorie des graphes peuvent être définis dans la théorie des cartes. Ce modèle de **carte combinatoire**, qui peut être considéré comme une généralisation plus formelle de la structure de données DCEL, sera détaillé au chapitre suivant car il sert de base au modèle de cartes que nous avons choisi dans le cadre de cette thèse.

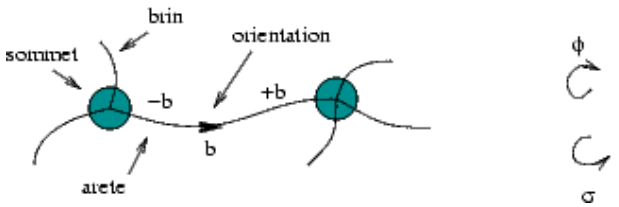


FIG. *3.13: Notations pour les cartes combinatoires.*

Enfin, David souligne que l'un des problèmes importants de la gestion de l'information géographique est la vérification de la qualité de ces données, et en particulier leur cohérence, ce qui permet notamment de simplifier les programmes d'application [39]. Ces contraintes sont de deux types : sémantiques et structurelles. Les contraintes sémantiques correspondent aux requêtes que l'on peut exprimer dans le langage

de requêtes du système géographique. Quant aux contraintes structurelles, elles interviennent directement sur les structures de données géographiques en imposant que certaines propriétés soient vérifiées sur certains éléments de la représentation. David en cite six qui lui paraissent particulièrement importantes dans la gestion des données géographiques [39] : connexité, recouvrement, disjonction, planéité, partition, partitions emboîtées (découpage hiérarchique). De même, le format VPF introduit des contraintes d'intégrité pour relier les objets et les opérations de manipulation (création, suppression, récupération, modification) : par exemple, quand on élargit une route sur la carte, il importe d'en rétrécir le bas-côté.

Cette notion de contraintes pourrait également avoir un intérêt dans le contexte de la construction automatique pour les robots : les contraintes permettraient de vérifier la cohérence du modèle construit, en évitant par exemple des intersections aberrantes entre primitives (cf. le cas des segments qui s'intersectent dans les cartes métriques à base de primitives géométriques, ou l'espace libre qui s'étend de part et d'autre des frontières d'obstacles dans les combinaisons de modèles métriques). Dans le cadre de la cartographie par des robots, Engelson propose d'ailleurs déjà d'introduire des contraintes structurelles pour assurer la cohérence de sa carte topologique [65] : un « restructureur » diagnostique explicitement les incohérences structurelles. Les contraintes considérés dans ce cas sont cependant plus d'ordre géométrique (position trop proche de deux nœuds de la carte) ou fonctionnelle (problème de « perceptual aliasing » ou « biais perceptuel » qui induit des comportements aberrants sur le robot). L'application de ces contraintes a cependant un effet sur la topologie de la carte construite : elle implique des ajouts ou des suppression de nœuds.

Comme nous le verrons au chapitre suivant, l'utilisation d'une couche topologique dans le contexte du SLAM peut également permettre la mise en œuvre de contraintes d'ordre topologique (disjonction, partition, connexité...) afin de détecter automatiquement des incohérences liées par exemple à des problèmes d'imprécision géométrique (problèmes similaires aux « slivers » mentionnés ci-dessus ou problèmes de polygones croisés mentionnés chez Larue [111]).

3.5 Conclusion sur l'état de l'art des représentations mixtes

Les différents types de cartes rencontrés depuis le début de cet état de l'art présentent clairement **des niveaux de structuration très variables**. En remontant aux modèles élémentaires, on peut ainsi distinguer :

- **les modèles simples et flexibles** :
  - ensemble de données brutes (images par exemple) non triées, non recalées ;
  - représentations directes basées sur l'apparence : données brutes recalées en position ;
  - grilles d'occupation : ce modèle permet une fusion des informations successives (presque brutes) dans la carte ;

- **un premier niveau de structuration dans lequel on extrait des primitives géométriques** à partir des données brutes et où l'on peut opérer une certaine sélection :
  - cartes basées sur des primitives « élémentaires » telles que les points (sélectionnés), les segments ou les arcs de cercle ;
  - représentations fondées sur des primitives de plus haut niveau (combinaisons simples de primitives élémentaires, telles que les lignes polygonales) ;
  - définition d'objets (combinaisons quelconques de primitives élémentaires [18] [21]) ;

- **l'unification des représentations de divers types de primitives géométriques** :
  - avec un nombre réduit de primitives prédéfinies (points et segments par exemple [15]) ;
  - avec un formalisme générique incluant tout type d'objets (SP-Maps [21]...) ;

- **un mélange de différents formalismes** :
  - combinaison de cartes métriques et topologiques : cartes topologiques augmentées d'informations métriques et graphes topologiques ancrés plus ou moins implicitement dans le plan, graphes d'adjacence construits sur le partitionnement d'une carte métrique, systèmes multiniveaux, réseaux de cartes métriques locales, etc.
  - combinaisons de représentations surfaciques et de cartes d'amers : création d'une représentation surfacique à partir des frontières d'obstacles, extraction des contours d'une grille d'occupation, construction simultanée des deux types de représentations par fusion des polygones d'espace libre locaux, etc.
  - cadres formels très généraux permettant la combinaison de différents formalismes : cartes bayésiennes hiérarchiques notamment [44] ;

- **la production et l'attachement d'informations sémantiques** :
  - définition a priori des objets possibles (jonctions, portes, impasses...[41]) ;
  - extraction de la sémantique à partir des informations métriques attachées à la représentation topologique [189], qui peut elle-même être issue d'une segmentation de carte métrique surfacique [68] ou d'une carte de primitives géométriques [26], exploitation d'informations sensorimotrices [44] ;

On remarque toutefois que l'on peut distinguer la carte présentée à l'opérateur de la représentation nécessaire au processus de cartographie (réseaux bayésiens plutôt que primitives géométriques individuelles, graphe de cartes locales plutôt que carte globale présentée dans un repère unique, etc.). Il existe souvent plus de structuration de la carte dans l'algorithme de construction que dans la représentation fournie en ligne à l'opérateur (où les liens entre balises n'apparaissent pas forcément par exemple). En revanche, la carte finale peut avoir subi un traitement de mise en forme a posteriori : transformation de scans bruts en grilles d'occupation ou en lignes polygonales, optimisation globale pour retrouver les transformations entre cartes globales, etc.

On constate cependant de façon générale que **les modèles d'environnement utilisés en robotique sont souvent de bas niveau par rapport aux formats géographiques**. On compte un certain nombre d'hybridations métriques et topologiques, avec souvent un glissement subtil de la représentation métrique vers la représentation topologique. Toutefois, il existe encore peu de travaux sur les mélanges de cartes métriques et encore moins sur l'adjonction d'informations topologiques à des combinaisons de cartes métriques. En particulier, la couche topologique des représentations utilisées pour le SLAM reste de très bas niveau. Pourtant, nous avons vu qu'**une couche topologique élaborée peut se révéler intéressante pour plusieurs raisons** :
- elle induit une accélération de certains traitements spatiaux à travers le stockage d'informations précalculées ;
- elle permet de vérifier la cohérence globale de la carte par adjonction de contraintes structurelles ;
- elle facilite l'introduction de la sémantique.

Ainsi, on pourrait dire qu'il existe globalement deux options extrêmes pour la cartographie (avec un continuum entre ces deux extrêmes) :
- soit on construit rapidement des cartes peu structurées (avec des modèles flexibles tels que les

représentations basées sur l'apparence) : ensuite, on peut effectuer a posteriori des opérations visant à structurer les données pour faciliter la navigation (hiérarchie de représentations [194], extraction de polylignes [190]...) ou définir des algorithmes adaptés à ce faible niveau de structuration (planification basée sur les grilles d'occupation), aux dépens peut-être de l'efficacité et de la robustesse ;
– soit on construit directement une carte très structurée (avec une gestion explicite des contraintes de cohérence du modèle dans le processus de construction), au risque d'introduire des opérations coûteuses en temps de calcul, mais en favorisant la compression et la structuration des données, ainsi que l'attachement d'informations sémantiques ; cette structuration constitue une sorte de précalcul accélérant l'exploitation de ces cartes (sinon, il faut recommencer le traitement a posteriori à chaque fois que l'on inclut une nouvelle observation).

Comme nous le préciserons au chapitre suivant, nos travaux se placent plutôt dans la deuxième optique.

Chapitre 4

Choix d'un modèle

« Le point, la ligne et la surface sont des éléments des formes de l'espace. C'est de leur lien génétique que découle l'ordre dans lequel ils sont inclus. L'élément le plus simple de l'espace est le point. Sa trace est la ligne. La trace de la ligne est la surface. Toutes les formes de l'espace sont pensées de ces trois éléments. »
D. Bourliouk.

4.1 Motivations

Nous nous plaçons dans un cadre où la carte doit servir d'une part au robot pour réaliser de manière semi-autonome les tâches qui lui sont assignées et d'autre part au téléopérateur pour comprendre l'environnement dans lequel évolue le robot et pour mieux le guider. Nous favoriserons donc une représentation métrique qui offre l'avantage d'être facile à appréhender pour un humain et d'être également exploitable par un robot. Plus précisément, nous nous attacherons à construire une représentation métrique dense qui fournisse les informations « de plein et de vide » nécessaires à la compréhension de l'environnement et à la planification de trajectoire. Cela ne nous interdit pas, toutefois, d'ajouter des informations topologiques en vue d'améliorer l'efficacité des algorithmes de planification par exemple. Enfin, cette carte doit pouvoir être générée en ligne et de manière passive, c'est-à-dire de manière transparente lors de la progression du robot dans l'environnement, sans nécessiter de stratégie de navigation particulière (telle que le suivi de mur, le suivi du graphe de Voronoï ou le contournement exhaustif et systématique des obstacles rencontrés).

Ainsi, l'objectif de notre travail est de tirer parti de toutes les observations réalisées afin de construire de manière incrémentale la carte métrique la plus précise et la plus complète possible. En particulier, on cherchera à éviter les incohérences liées aux erreurs d'appariement ou au bouclage de cycles : intersections ou dédoublements aberrants de primitives géométriques, empiètements de l'espace libre sur les frontières d'obstacles, zones « floues » résultant de données contradictoires, etc. Si ces incohérences se révèlent impossibles à éviter dans un processus de construction incrémental, nous chercherons au moins à les détecter efficacement de manière à pouvoir mettre en œuvre les processus de correction adéquats.

Nous avons vu au chapitre précédent que la notion de couche topologique pouvait présenter un intérêt pour les applications de « haut niveau » en robotique (raisonnements spatiaux avancés pour la planification notamment). Nous nous sommes donc orientés vers une représentation basée sur les cartes combinatoires, qui ont été brièvement introduites au chapitre précédent. Cette représentation permet de gérer

différents niveaux de représentation dans un modèle relativement compact : représentation métrique des frontières d'obstacles, modélisation surfacique de l'espace libre et informations topologiques d'adjacence. Elle permet également d'assurer la cohérence entre ces différents niveaux, via l'adjonction de contraintes structurelles. L'enjeu de la thèse consiste donc à construire ces cartes de façon incrémentale en gérant les incertitudes associées aux erreurs de mesure.

Les sections suivantes précisent de manière plus formelle la définition de ces cartes combinatoires, avant de préciser le modèle de carte utilisé pour représenter l'environnement.

4.2 Définition des cartes combinatoires

Une carte combinatoire est un outil algébrique permettant de décrire et de manipuler la topologie d'une subdivision de surface fermée orientable, en termes de sommets, d'arêtes et de faces. La première fois que l'on a songé à employer une telle structure algébrique pour décrire la topologie de polyèdres semble remonter à Edmonds en 1960 [59]. La notion de constellation, introduite par Jacques en 1970 [90], précise cette idée. Cette notion fut généralisée sous différentes formes, par exemple dans les travaux de Cori en 1975 [33] ou de Lienhardt en 1991 [121]. De nombreux travaux théoriques en précisent les propriétés et en énumèrent les caractéristiques topologiques [5] [6]. Tutte étudie en 1984 [186] les constellations sous le nom de *cartes combinatoires orientées*, que nous gardons.

Les applications des cartes combinatoires concernent notamment la synthèse d'images avec la construction et la manipulation d'objets 3D, mais aussi la géographie avec l'utilisation de cartes en deux dimensions. En fait, cette représentation a exactement la même puissance modélisatrice que la structure de données classique d'« arêtes ailées » proposée par Baumgart en 1970 (cf. [9] [72]), mais elle est définie de manière plus précise et concise, algébriquement.

Dans le cadre de cette thèse, nous utilisons uniquement des cartes dites *planaires*, dont la définition est liée à la notion de *genre* (intuitivement, le nombre de trous dans la surface ou dans la carte).

Plus précisément, soit S le nombre de sommets, A le nombre d'arêtes, F le nombre de faces et C le nombre de composantes connexes de la subdivision. La caractéristique d'Euler-Poincaré s'écrit alors : $\chi = S - A + F$. On peut ensuite définir le *genre* g de la manière suivante : $g = C - \chi/2$. On admet couramment le théorème suivant, appelé *théorème du genre* : g est un entier positif ou nul.

Une carte combinatoire modélise toujours une subdivision d'une surface de même genre. Si le genre est nul, la carte est dite *planaire*. Lorsque la subdivision d'une surface de genre non nul est plongée dans le plan, la carte combinatoire orientée correspondante contient nécessairement des auto-intersections (par exemple des arêtes croisées). Inversement, lorsque le genre est nul (c'est-à-dire dans le cas des cartes planaires que nous utilisons), il est toujours possible de plonger la carte dans le plan sans auto-intersection, par ce que l'on appelle un *plongement planaire*.

4.2.1 Définition topologique

Une *carte combinatoire orientée (à 2 dimensions)* est un triplet $M = (D, \alpha_0, \alpha_1)$, où D est un ensemble fini d'éléments, appelés *brins*, α_0 est une involution sans point fixe dans D, et α_1 est une permutation dans D. Soient k égal à 0 ou 1, et $z \in D$. Les brins $\alpha_k z$ et $\alpha_k^{-1} z$ sont respectivement les *k-successeur* et *k-prédécesseur* de z. Ils sont dits *k-liés* ou *k-cousus* par des *k-liaisons* ou *k-coutures*.

Pour abréger, nous écrirons *carte* pour *carte combinatoire orientée de dimension 2*.

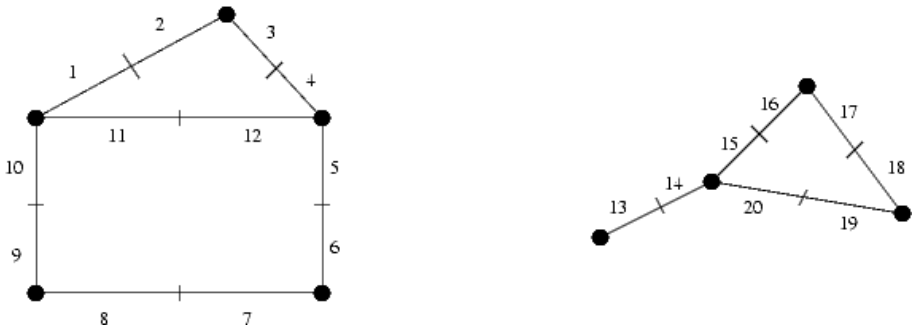


FIG. *4.1: Une carte combinatoire plongée dans le plan.*

Par exemple, la figure 4.1 montre une carte M avec $D = [1..18]$,
$\alpha_0 = (1\,2)(3\,4)(5\,6)(7\,8)(9\,10)(11\,12)(13\,14)(15\,16)(17\,18)(19\,20)$
et $\alpha_1 = (1\,10\,11)(2\,3)(4\,12\,5)(6\,7)(8\,9)(13)(14\,20\,15)(16\,17)(18\,19)$.
La carte est plongée dans le plan avec les conventions suivantes. Chaque brin étiquette un demi-segment de courbe délimité par un point noir et un petit trait. Un trait entre deux demi-segments représente une 0-couture entre ses étiquettes. Les demi-segments sont ordonnés dans le sens trigonométrique (sens inverse des aiguilles d'une montre) autour des points noirs, suivant leur ordre de succession par α_1. Le brin 13 est un point fixe pour α_1. Nous verrons à la section suivante une définition plus précise du plongement planaire.

La vraie nature des brins n'a pas d'importance. Nous les considérerons dans les schémas comme des entiers, mais dans une implémentation efficace, ils peuvent être des pointeurs sur des structures de données. Notons que pour $d \geq 3$, les cartes combinatoires de dimension d peuvent se définir de la même façon, à l'aide de d involutions ou permutations. Dans ce qui suit, nous nous limiterons cependant aux cartes à 2 dimensions, puisque nous considérons des subdivisions du plan.

Grâce à la notion d'*orbite*, les cartes permettent d'exprimer facilement les définitions des cellules usuelles d'une subdivision et facilitent leur accès.

Soit D un ensemble et Φ un ensemble d'applications dans D. Les composantes connexes du multigraphe fonctionnel (D, Φ) sont appelés *orbites* par rapport à Φ. L'orbite par rapport à Φ qui contient l'élément z de D est noté $< \Phi > (z)$. Soit $M = (D, \alpha_0, \alpha_1)$ une carte. Les orbites par rapport à $< \alpha_0 >$ sont les *0-orbites* ou *arêtes topologiques* de M, les orbites par rapport à $< \alpha_1 >$ sont les *1-orbites* ou *sommets topologiques* de M, les orbites par rapport à $< \alpha_1^{-1} \circ \alpha_0 >$ sont les *faces directes*, les orbites par rapport à $< \alpha_1 \circ \alpha_0 >$ sont les *faces indirectes*, et enfin les orbites par rapport à $< \alpha_1, \alpha_0 >$ sont les *composantes connexes*.

De par les propriétés des permutations, les sommets, arêtes, faces directes et indirectes forment des circuits élémentaires simples (c'est-à-dire des chemins qui ne repassent pas deux fois par le même arc ou nœud, excepté pour leurs extrémités, qui sont confondues) dans leurs graphes respectifs (D, α_0), (D, α_1), $(D, \alpha_1^{-1} \circ \alpha_0)$, $(D, \alpha_1 \circ \alpha_0)$. Par exemple, le sommet $< \alpha_1 > (8)$ de la figure 4.1 peut s'écrire (8 11 10). Notons que chaque cellule, ou orbite, d'une carte peut être repéré de façon uniforme par les permutations correspondantes, et surtout, par un brin unique. Les cartes fournissent donc un moyen aisé de parcourir les faces.

Par exemple, dans la carte de la figure 4.1, lorsque l'on réduit toute orbite à son ensemble support : $< \alpha_0 > (1) = \{1, 2\}$ et $< \alpha_0 > (8) = \{7, 8\}$ sont des arêtes ; $< \alpha_1 > (1) = \{1, 10, 11\}$, $< \alpha_1 > (17) = \{16, 17\}$ et $< \alpha_1 > (13) = \{13\}$ sont des sommets ; $< \alpha_1^{-1} \circ \alpha_0 > (2) = \{2, 11, 4\}$ et $< \alpha_1^{-1} \circ \alpha_0 > (1) = \{1, 3, 5, 7, 9\}$ sont des faces directes ; $< \alpha_1 \circ \alpha_0 > (9) = \{9, 11, 5, 7\}$ et $< \alpha_1 \circ \alpha_0 > (10) = \{10, 8, 6, 4, 2\}$ sont des faces indirectes ; $< \alpha_0, \alpha_1 > (17) = \{17, 18, 19, 20, 13, 14, 15, 16\}$ est une composante connexe.

Pour plus de précisions, le lecteur pourra se référer à la thèse de Cazier [23] par exemple.

4.2.2 Définition du plongement planaire

La géométrie d'une carte planaire est définie à travers son plongement planaire, c'est-à-dire par le plongement de chacun de ses brins. Ainsi, les sommets sont associés à des points et les arêtes à des arcs de Jordan. Ici, nous ne considérons que le cas linéaire, où ces arcs sont des segments de droites.

Une carte M plongée (linéairement) dans le plan est un 4-uplet $(D, \alpha_0, \alpha_1, \pi_0)$, où (D, α_0, α_1) est une carte et où π_0 est une fonction qui fait correspondre à chaque brin le point sur lequel son sommet est 0-plongé.
Les plongements en dimensions 1 et 2 sont définis implicitement à partir de π_0. Ainsi, le 1-plongement π_1 du brin x est l'intérieur du segment $[\pi_0(x), \pi_0(\alpha_0(x))]$. De même, le 2-plongement π_2 du brin x est l'intérieur du polygone $\{\pi_0(y), \pi_1(y)\}_{y \in < \alpha_1 \circ \alpha_0 > (x)}$, défini par la séquence de points et de segments de sa frontière.
Grâce à l'orientation de la carte, nous considérons par convention que pour tous les brins x, le polygone $\pi_2(x)$ se trouve toujours à droite du segment $\pi_1(x)$: il correspond à la face directe de x.

4.3 Quelques opérations sur les cartes combinatoires

Les cartes combinatoires s'avèrent particulièrement efficaces dans diverses manipulations géométriques et topologiques [55]. En outre, on montre que la conception d'opérations de base en programmation géométrique peut être grandement facilitée par l'utilisation de spécifications formelles, dans le but de manipuler ces modèles de cartes combinatoires [163]. Cette approche permet d'exhiber des contraintes d'intégrité pour les objets manipulés, et de se concentrer sur les aspects logiques et conceptuels de l'opération géométrique. En particulier, la définition du raffinement tel qu'il sera décrit dans la section suivante a largement bénéficié de cette méthode de conception, à travers l'emploi de systèmes de réécriture.

4.3.1 Présentation du raffinement

Le co-raffinement de deux subdivisions, c'est-à-dire de deux partitions de l'espace 2D ou 3D en cellules distinctes (sommets, arêtes, faces, et volumes pour le cas 3D), consiste à produire une nouvelle subdivision contenant les cellules des anciennes, mais coupées le long de leurs intersections (cf. Fig. 4.2). Il s'agit donc de partitionner les cellules jusqu'à ce qu'aucune intersection n'existe plus entre elles et de réaliser leur restructuration topologique. Les relations d'incidence et d'adjacence sont conservées et complétées durant tout le processus. Ajoutons qu'en dimension 2, le raffinement est une généralisation du problème des arrangements de droites (cf. par exemple les travaux de Bentley et Ottmann [10] ou Nievergelt et Preparata [150]).

Cette opération de raffinement est une opération de base en modélisation géométrique : en particulier, elle est fréquemment employée pour l'évaluation d'opérations booléennes (union, intersection...) de modeleurs utilisant des représentations par frontières. Nous verrons que cette opération se révèlera

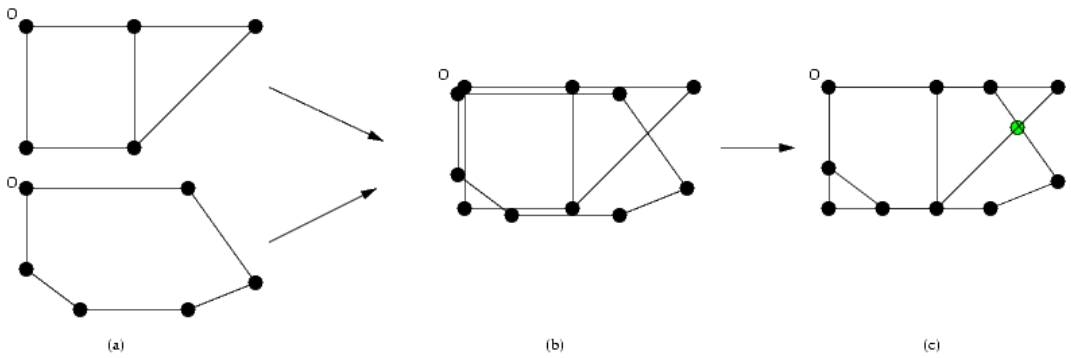


FIG. *4.2: Un exemple de co-raffinement de deux subdivisions. (a) Les deux subdivisions à raffiner (O étant l'origine d'un repère commun). (b) Superposition initiale de ces deux subdivisions (avec un léger décalage pour les besoins de l'illustration). (c) Résultat du raffinement (avec fusion des arêtes et sommets superposés et création d'un sommet plongé sur le point d'intersection en vert).*

également fort utile dans certaines étapes de la construction de cartes d'environnement. Le raffinement 2D servira notamment lors de la fabrication de la carte locale, lors des corrections respectives des cartes locale et globale avant fusion, et surtout, lors de la fusion de ces deux cartes.

Les définitions et algorithmes de raffinement qui suivent sont tous issus des travaux de thèse de Cazier [23] [24]. Ils concernent en réalité l'opération d'auto-raffinement d'une seule carte combinatoire. Dans le cas de la fusion de deux cartes, il suffit d'ajouter au préalable les éléments de la seconde carte à la première sans réaliser de restructuration.

4.3.2 Définition formelle du raffinement

Nous avons vu qu'une carte combinatoire à 2 dimensions plongée peut modéliser un ensemble de segments et de polygones dans le plan, avec des relations d'incidence et d'adjacence. Sans contraintes géométriques, une carte peut être plongée avec des intersections et des superpositions entre ses cellules et ses composantes connexes. Une telle carte est cependant très utile pour modéliser l'ensemble de départ contenant les subdivisions 2D que nous souhaitons raffiner. Le raffinement 2D consiste alors à le transformer en une carte modélisant une partition du plan, c'est-à-dire une carte planaire plongée sans auto-intersection. Par la suite, nous dirons qu'une telle carte est *bien plongée*. Notons au passage qu'une carte bien plongée dans le plan est nécessairement planaire [186]. Inversement, une carte non planaire ne peut jamais être bien plongée dans le plan.

Soit $\Pi(D) = \{\pi_0(x), \pi_1(x), \pi_2(x)\}_{x \in D}$ un ensemble de plongements de la carte M dans le plan $\mathbb{R}^2$. Alors M est *bien plongée* si :
A. $\bigcup_{p \in \Pi(D)} p = \mathbb{R}^2$
B. $\forall p, q \in \Pi(D), p \neq q \Rightarrow p \cap q = \emptyset$.

Comme les plongements des arêtes et des faces sont définis de manière implicite, nous pouvons exprimer ces deux conditions de façon plus pratique. Soit $angle(x)$ l'angle du segment $\pi_1(x)$ par rapport à un axe donné du plan (dans le sens trigonométrique). Alors une carte est bien plongée si les cinq conditions suivantes sont satisfaites pour tous les brins x et y de M :
(cbp 1) $< \alpha_1 > (x) \neq < \alpha_1 > (y) \Rightarrow \pi_0(x) \neq \pi_0(y)$
(cbp 2) $\pi_0(x) \notin \pi_1(y)$

(cbp 3) $< \alpha_0 > (x) \neq < \alpha_0 > (y) \Rightarrow \pi_1(x) \cap \pi_1(y) = \emptyset$
(cbp 4) la séquence $\{angle(\alpha_1^k(x_0))\}_{k\in\{0,\ldots,n\}}$ est croissante, n étant le plus petit entier tel que $\alpha_1^{n+1}(x_0) = x_0$ et $x_0 \in < \alpha_1 > (x)$ tel que $angle(x_0) = \min\{angle(y)\}_{y\in<\alpha_1>(x)}$
(cbp 5) $\pi_1(x) \neq \emptyset$.

La condition (cbp 1) signifie que tous les brins 0-plongés sur le même point appartiennent au même sommet. Les conditions (cbp 2) et (cbp 3) assurent qu'il n'existe aucune arête sécante, et aucun sommet incident à une arête. La condition (cbp 4) requiert plus d'explications. La séquence des angles des brins d'un sommet est croissante lorsque l'on commence par le brin x_0 d'angle minimal. Cela signifie que, lorsque les arêtes d'un sommet donné sont examinées en suivant les 1-liens, les segments associés tournent dans le sens inverse des aiguilles d'une montre autour du sommet (sens trigonométrique). Un sommet qui satisfait cette condition est dit *trié*. Intuitivement, la condition (cbp 4) assure une bonne orientation de la carte et implique que les faces ne se replient pas sur elles-mêmes.
Enfin, la condition (cbp 5) n'est pas absolument indispensable : on l'ajoute simplement pour obtenir une représentation minimale en termes de nombre de brins.

4.3.3 Un algorithme pour le raffinement

Les définitions mathématiques que nous avons données des cartes et du raffinement 2D sont limitées car elles décrivent seulement les conditions statiques que doit satisfaire une carte raffinée mais n'indiquent pas comment obtenir une telle carte. Il importe donc de spécifier la construction de ces cartes et l'ensemble d'opérations nécessaire pour les raffiner : c'est là qu'interviennent les spécifications formelles. Nous nous contenterons ici d'en décrire les grandes lignes et renvoyons le lecteur à [23] [24] [55] pour plus de précisions, notamment pour y trouver les spécifications elles-mêmes, dans un langage proche de OBJ3, ou pour découvrir un prototype en langage fonctionnel (Ocaml).

Les cartes peuvent être définies à l'aide de trois générateurs (l'un crée la carte vide, le second insère deux brins 0-liés et leurs plongements, et le troisième 1-lie deux brins), et détruites par le biais de destructeurs. On définit également un ensemble de sélecteurs fonctionnels topologiques et géométriques (qui vérifient par exemple si le brin z existe dans la carte M, si les plongements de sommets ou d'arêtes sont égaux, etc.), et des constructeurs qui modifient la topologie et le plongement de cartes (fusion de deux sommets ou de deux arêtes, découpage d'une arête en deux sous-arêtes...). Tous les opérateurs spécifiés sont contraints par des préconditions qui assurent que les termes construits vérifient toujours la définition mathématique des cartes combinatoires. La logique équationnelle sous-jacente peut être utilisée pour prouver ces propriétés, principalement par induction structurelle sur la forme des termes qui représentent les cartes.

Le raffinement d'une carte consiste alors à la transformer à l'aide des opérateurs mentionnés ci-dessus jusqu'à ce qu'elle satisfasse les conditions de bon plongement. Une bonne façon de décrire formellement et intégralement le raffinement sans être gêné par des contraintes d'efficacité ou d'implémentation est d'utiliser les techniques de réécriture. Chaque transformation, et les conditions selon lesquelles elle peut être employée, peuvent être décrites élégamment par une règle de réécriture conditionnelle. Intuitivement, les conditions de ces règles testent l'existence de brins qui ne vérifient pas l'une des cinq conditions de bon plongement. Si cela se produit, un opérateur géométrique est appliqué pour corriger localement le plongement et pour adapter la topologie. Cela conduit à une définition simple et générique d'un processus de raffinement a priori complexe.

Plutôt que de présenter les équations du système de réécriture (équations que l'on peut trouver dans [23]), on se contentera ici d'en reproduire une illustration graphique commentée (cf. Fig. 4.3).

La règle R_1 détecte les arêtes nulles (violation de la condition de bon plongement (cbp 5)), qu'elle supprime. Si deux arêtes sont superposées, ce qui contredit (cbp 3), R_2 supprime la seconde. La règle R_3 s'occupe de (cbp 2) et coupe ainsi une arête en deux s'il existe un sommet qui lui est incident. La règle R_4 est aussi liée à (cbp 3) et réalise le découpage d'arêtes lié à une intersection. La règle R_5 fusionne deux sommets qui ne vérifient pas (cbp 1). Et enfin, la règle R_6 détruit les sommets non-triés (violation de (cbp 4)) : les brins qui appartiennent à un sommet non trié sont successivement supprimés de ce sommet, jusqu'à ce qu'il soit trié ; ensuite, ces brins sont correctement réinsérés par la règle R_5.

L'un des éléments intéressants dans cette approche formelle est qu'elle permet de prouver certaines propriétés logiques dans le processus de raffinement. D'abord, le système de réécriture est *à terminaison finie*, autrement dit, il n'existe pas de séquence infinie de règles. On peut également montrer sa *confluence* : le résultat du raffinement ainsi effectué ne dépend pas de l'ordre dans lequel les règles sont appliquées. Le système est ainsi *convergent*, c'est-à-dire que pour chaque carte, la réécriture conduit à une forme normale unique (modulo la congruence entre cartes, liée notamment à un renommage des brins : deux cartes peuvent être topologiquement et géométriquement identiques sans que les noms des brins correspondent). Cette propriété permet de choisir n'importe quelle stratégie efficace et pratique pour l'application des règles, ce qui est crucial dans la dérivation d'algorithmes concrets. Ces preuves sont fournies dans [23], et recourent notamment à l'induction structurelle et à la logique équationnelle.

4.3.4 Une implémentation efficace

Pour la manipulation de cartes

Dans une implémentation efficace, en langage C par exemple, une carte plongée est un pointeur vers une structure de carte, et un brin est un pointeur vers une structure de brin. La structure de carte contient principalement les caractéristiques courantes de la carte et un pointeur vers le premier élément d'une liste chaînée (dans un seul sens) de toutes les structures de brins de ses brins. Notons que l'on ne divise pas la liste suivant les composantes connexes de la carte, ce qui serait nuisible [55].

Chaque structure de brin contient essentiellement un pointeur vers son successeur dans la liste, et les brins images dans la carte par les permutations α_0, α_1, et α_1^{-1}. Elle possède également des pointeurs vers un point, un arc (segment), attributs qui plongent le brin correspondant, et certains marqueurs, booléens ou brins, qui aident à mémoriser les chemins durant les parcours de cartes.

Dans une telle implémentation pour les cartes, la complexité en temps peut être considérée pour les diverses opérations en fonction du nombre n de brins. Ainsi, les constructeurs sont en $O(1)$ et les autres opérateurs, tels que les parcours ou les équivalences sont en $O(n)$ dans le pire des cas. Ainsi, calculer des cartes est assez similaire à calculer dans des graphes, dans des cas plutôt favorables.

Enfin, concrètement, il faut aussi ajouter quelques opérations permettant de manipuler les cartes en usage courant. Ainsi, il importe par exemple de définir des opérations permettant de les stocker dans des fichiers, et de les charger ensuite en mémoire, voire de les visualiser.

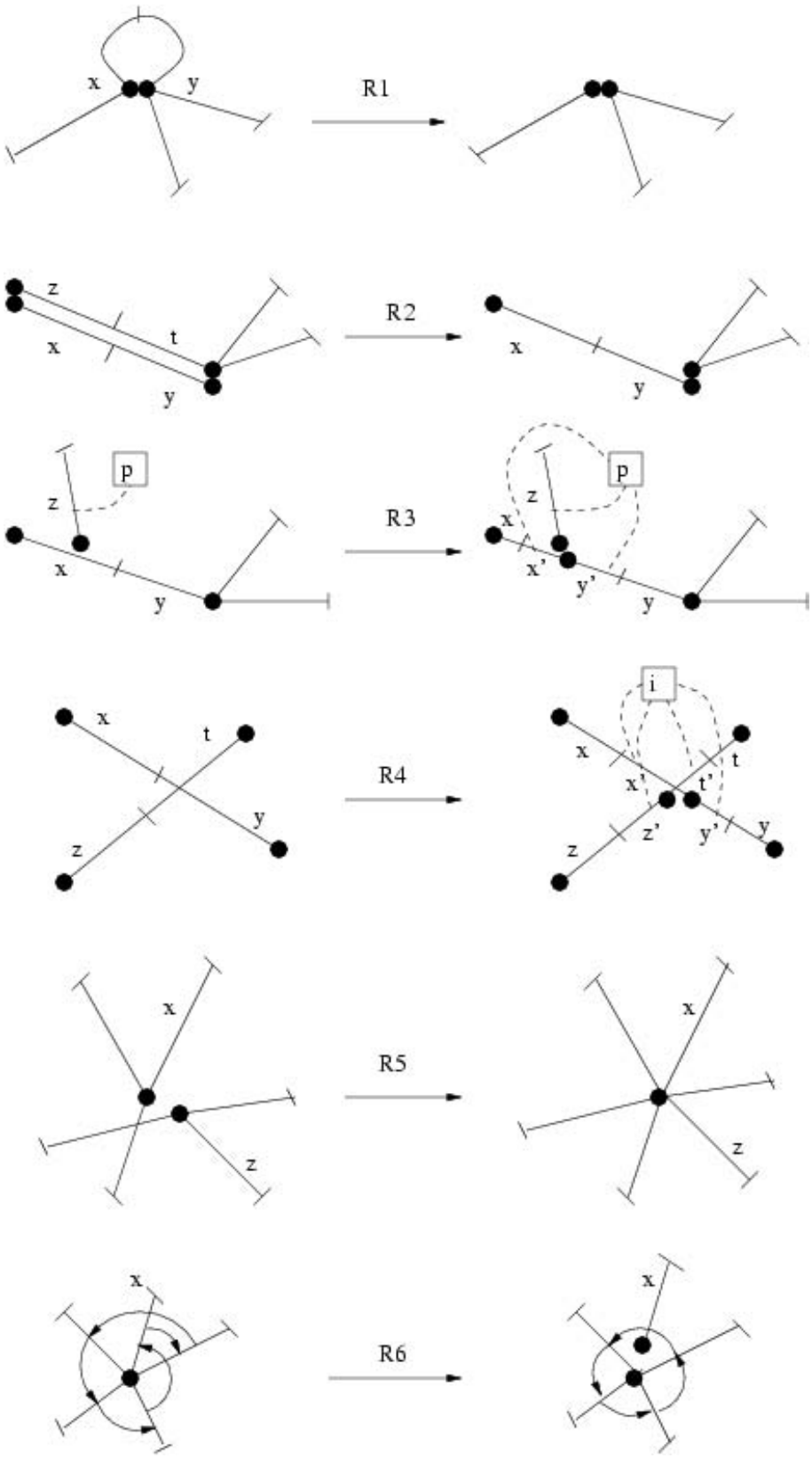


FIG. *4.3: Illustration graphique des règles de raffinement. Les plongements π_0 sont représentés par des boîtes avec des pointillés vers les brins concernés.*

Pour le raffinement

Une utilisation naïve du système de réécriture décrit ci-dessus est de tester pour chaque paire de brins si une règle peut être exécutée. Cela correspond à l'algorithme suivant, où x et z sont des brins :

Répéter
  Choisir x dans M
  Essayer d'exécuter les règles 1 et 6 avec x
  Répéter
    Choisir z dans M
    Essayer d'exécuter les règles 2 à 5 avec (x, z)
  Jusqu'à ce qu'aucune règle ne puisse être exécutée avec x
Jusqu'à ce qu'aucune règle ne puisse être exécutée.

Un algorithme aussi abstrait est indéterministe car les brins sont choisis aléatoirement. Pour décrire un algorithme réel et efficace, il faut envisager une stratégie pour le choix des brins. On peut parvenir à ce but par l'adjonction de structures de contrôle au système de réécriture.

Il est intéressant d'exploiter certaines propriétés géométriques pour éviter d'examiner des paires de brins qui ne peuvent pas interagir. On applique en fait dans cette optique une stratégie classique de balayage de plan [150], qui classe et parcourt les arêtes de gauche à droite, en maintenant à jour des structures telles que des files de priorité et des dictionnaires contenant les brins actifs à un moment donné. Ainsi, concrètement, cela revient à ajouter au modèle de cartes de nouvelles structures de contrôle, des opérateurs permettant de les manipuler, et des règles de réécriture supplémentaires destinées à les maintenir à jour (cf. [24], [23]). Notons enfin que la complexité de cet algorithme optimal par balayage est en $O((n+i)\ln(n))$, n étant le nombre de brins de la carte et i le nombre de brins créés (lors de l'appalication des règles de règles de réécriture R3 et R4), ce qui permet d'obtenir des temps de calcul très raisonnables. D'après [23], des algorithmes plus sophistiqués pourraient cependant atteindre des complexités en $O(n\ln(n)+i)$.

4.3.5 La complétion de labels

Cette opération est décrite plus en détail dans [23].

Présentation

Après une opération de raffinement, si deux objets de la carte de départ se recouvraient, leur intersection a été implicitement calculée et modélisée en un ensemble de cellules. Il peut être intéressant, notamment dans le cadre de l'évaluation d'expressions booléennes entre objets modélisés par une carte, de savoir à quel(s) objet(s) initial(initiaux) correspondent les cellules de son raffinement. Cette information est donnée à travers des labels attachés aux brins : un label $\{A, B\}$ attaché à un brin x signifie que la face qui se trouve à droite du brin appartient à la fois à l'objet A et à l'objet B. Dans notre stratégie de construction de cartes d'environnement, une telle opération nous servira à calculer les labels des faces de notre subdivision, après fusion des cartes globale et locale. Elle nous permettra également de corriger efficacement les positions des arêtes dans les phases de correction de cartes. En outre, sa complexité est limitée puisqu'elle peut être réalisée en $O(np)$ où n est le nombre de brins de la carte et p le nombre d'objets modélisés par la carte [23].

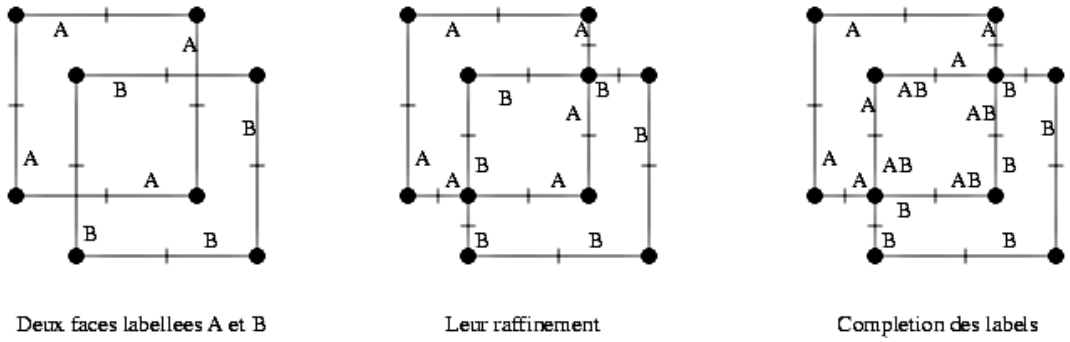


FIG. *4.4: Complétion des labels pour l'intersection de deux faces.*

Exemple (issu de [23]) :
Considérons deux faces orientées sécantes appartenant à deux objets, d'identificateurs A et B, représentées dans la figure 4.4. Avant le raffinement, les 2-labels (le i-label d'un brin est l'ensemble des identificateurs des objets auxquels appartient la cellule de dimension i de ce brin : un 2-label correspond donc à une face interne) des brins de ces deux faces valaient respectivement les ensembles $\{A\}$ et $\{B\}$, notés A et B. Après le raffinement, la face correspondant à leur intersection est formée de morceaux d'arêtes des deux faces initiales. Les brins de ces arêtes ont des 2-labels valant A ou B. On a donc une face dont les brins ont des labels différents. On peut en déduire qu'elle appartient à l'intersection recherchée. Les 2-labels des brins de cette face doivent donc être remplacés par l'union des 2-labels, soit ici l'ensemble $\{A, B\}$, noté AB. De plus, les arêtes qui appartenaient à la face B, avant le raffinement, et dont les 2-labels ont été mis à jour, sont à l'intérieur de la face A. Le 1-label de ces arêtes doit donc également être modifié. Le même raisonnement peut être effectué pour les sommets et leur 0-label.

Mise en œuvre

Traditionnellement, la classification des cellules était réalisée au moyen de tests géométriques, faisant intervenir la localisation de points ou des normales indiquant l'intérieur d'une cellule. La modélisation par cartes combinatoires permet cependant de n'utiliser que les informations topologiques qui existent dans le raffinement, et en particulier les informations d'incidence et d'adjacence.

Il faut d'abord apporter de légères modifications aux règles de raffinement, de manière, d'une part, à ne pas créer de trous (lorsqu'on insère de nouveaux brins, par découpage d'arête par exemple, les nouveaux brins héritent de labels cohérents avec ceux de leurs voisins), et d'autre part, à ne pas perdre d'information (lorsque deux arêtes sont fusionnées, l'arête résultante hérite des deux labels). Ensuite, une fois le raffinement terminé, il s'agit de modifier les labels résultants de manière à restaurer la cohérence de la carte, entre brins d'une même face notamment. Pour cela, il faut parcourir l'ensemble des faces pour harmoniser les labels au moyen d'une règle unique de complétion. Cette règle n'est appliquée qu'aux couples de brins adjacents d'une même face, dont les 2-labels diffèrent. Finalement, cette opération globale de complétion de labels est linéaire en le nombre de brins.

4.3.6 Une opération de « segmentation »

[56] mentionne une opération de segmentation, qui s'apparente plutôt à une fusion de régions en fait. Il s'agit en effet de fusionner des faces directes adjacentes possédant le même label, par suppression de leurs arêtes communes. Lorsque les labels sont des couleurs ou des niveaux de gris, une telle opération

peut être considérée comme un cas très simple de segmentation d'images. Nous verrons que ce type d'opération peut nous servir de manière à simplifier nos cartes.

4.4 Une version discrète des cartes combinatoires

L'algorithme défini par Cazier pour le raffinement des cartes combinatoires [23] est destiné à fonctionner sur des cartes décrites en nombres réels, c'est-à-dire des cartes pour lesquelles les sommets sont plongés sur des points du plan définis par des coordonnées appartenant à $\mathbb{R}^2$. Toutefois, l'ordinateur ne gère pas parfaitement les nombres réels puisqu'il ne peut considérer qu'un nombre limité de chiffres après la virgule. Concrètement, ces erreurs d'arrondis liées aux nombres flottants, classiques en géométrie algorithmique, peuvent s'avérer problématiques dans les opérations de raffinement : elles risquent d'invalider le plongement (par exemple l'ordre de succession des brins autour d'un sommet risque de ne plus être conforme au sens trigonométrique) et de rendre difficile l'application des règles de raffinement (cf. Fig. 4.5).

Pour y remédier, nous pourrions recourir aux méthodes suggérées par Cazier [24] telles que les systèmes de raisonnement symbolique, les intervalles flottants, les rationnels, les entiers multiprécision, l'arithmétique rationnelle paresseuse, etc. Une version en arithmétique exacte de l'algorithme initial a d'ailleurs été développée récemment par Cazier. On peut également noter qu'il existe des librairies de géométrie algorithmique telles que CGAL [31] [71], qui permet de définir des cartes topologiques sur le modèle des DCEL (vues au chapitre précédent) et qui gère correctement les problèmes d'arrondis (même lorsque les arêtes de la carte sont plongées sur des courbes, notamment des arcs de cercle, plutôt que sur des segments du plan). Toutefois, d'après le manuel de référence [31], il n'existe pas encore d'algorithme de complétion de labels dans cette librairie, ni de structures de données ou d'opérations permettant de gérer les situations transitoires comme nous le verrons au chapitre 7. De plus, il ne semble pas évident, au premier abord, d'étendre les types de base disponibles (par exemple pour ajouter les informations d'incertitude sur la position des sommets) et de définir des variantes du raffinement classique (cf. raffinement coloré intervenant au chapitre 7) à partir de l'opération existante de « map overlay ». Enfin, la définition des DCEL (qui correspondent plutôt à des structures de données) nous a paru légèrement moins bien formalisée que celle des cartes combinatoires (qui constituent un véritable outil algébrique) et donc moins favorable à la mise en œuvre ultérieure de preuves formelles sur les opérations de manipulation utilisées, telles que le raffinement.

Finalement, nous avons donc plutôt choisi de conserver les cartes combinatoires, mais de passer à une représentation discrète de cet outil car, comme nous le préciserons plus loin, une telle discrétisation apporte une richesse supplémentaire à notre modèle de carte. Ainsi, nous introduisons ci-après les cartes combinatoires discrètes et l'algorithme de raffinement par balayage qui s'y rattache.

4.4.1 Définition des cartes combinatoires discrètes

Intuitivement, pour passer des cartes combinatoires décrites en nombres réels aux cartes décrites en nombres entiers, il suffit de plonger les sommets de la carte sur $\mathbb{Z}^2$ plutôt que sur $\mathbb{R}^2$. Cependant, on constate rapidement que cette contrainte n'est pas suffisante. En effet, deux arêtes discrètes ainsi définies par leurs extrémités discrètes peuvent se couper en un point de coordonnées non entières. Pour définir un point de coordonnées discrètes, il faudrait alors effectuer une approximation qui modifierait la géométrie, voire la topologie de la carte. Comme l'illustre la figure 4.6, cette approximation pourrait entraîner le décalage de certains segments dans n'importe quelle direction (par effets de bord successifs), de sorte que l'on ne pourrait plus appliquer l'algorithme de raffinement par balayage (qui ne peut pas a priori revenir

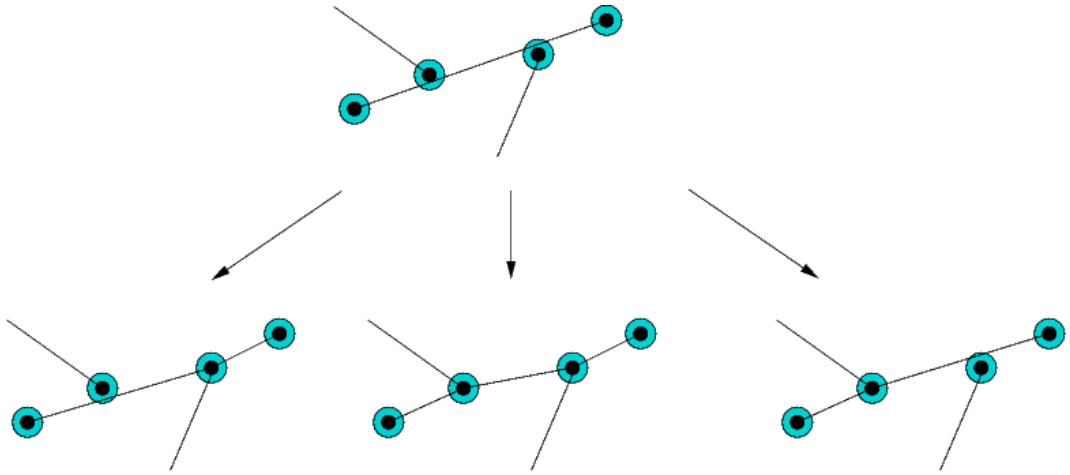


FIG. *4.5: Exemple de perturbations du raffinement du fait des erreurs d'arrondis. Le résultat du raffinement peut dépendre de l'ordre d'application des règles de réécriture.*

en arrière, en direction inverse du balayage, même si des éléments ont été déplacés). On se condamnerait ainsi à des temps de calcul prohibitifs pour le raffinement.

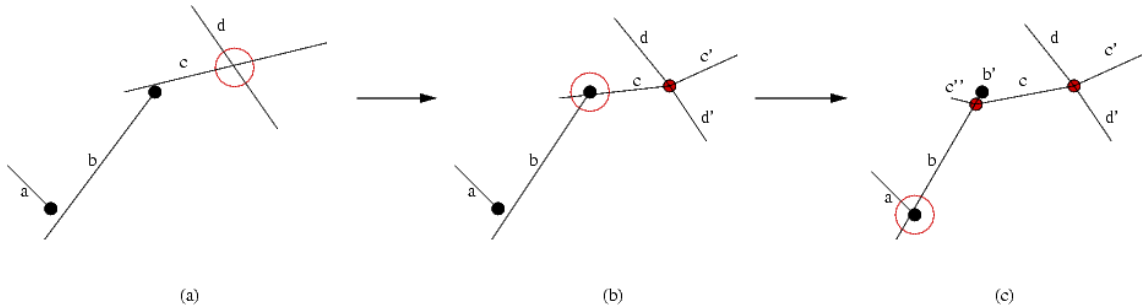


FIG. *4.6: Exemple d'effet de bord. (a) Le cercle rouge indique une intersection entre les deux arêtes c et d à extrémités discrètes. (b) Cette intersection est plongée sur un point discret, ce qui entraîne un léger décalage de l'arête c, de sorte qu'il apparaît une nouvelle intersection entre b et c (au niveau du cercle rouge). (c) Cette nouvelle intersection est plongée sur un autre point discret, ce qui entraîne un déplacement de l'arête b et une troisième intersection entre a et b, encore plus à gauche (sur une arête désactivée, qui ne sera donc plus examinée lors du balayage, qui s'effectue de gauche à droite).*

Nous avons donc défini les arêtes discrètes d'orientation quelconque, ou grandes arêtes discrètes, comme étant composées d'un ensemble de petites arêtes discrètes, qui sont soit horizontales, soit verticales (cf. Fig. 4.7). Les grandes arêtes sont formées de grands brins et correspondent à de grands segments. Les petites arêtes sont formées de petits brins et correspondent à de petits segments. Ces petits segments sont définis lors de la création des grands brins. Ils occupent dans le plan discret les points sur lesquels passe le tracé du grand segment réel correspondant, la discrétisation du segment étant effectuée par un algorithme classique de type « midpoint » ou « Bresenham »[72].

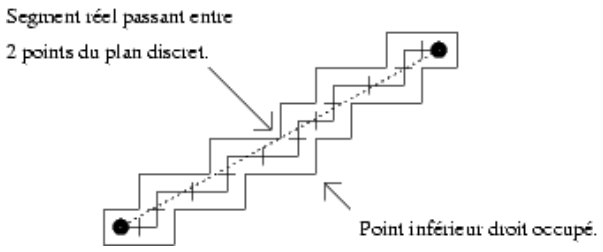


FIG. *4.7: Discrétisation d'un segment dans la carte combinatoire discrète.*

Cet algorithme classique a été légèrement modifié de manière à assurer une 4-connexité entre ces points, de sorte que les petits segments ne puissent avoir que deux orientations possibles dans le plan : verticale ou horizontale. Ainsi, pour éviter de modifier la géométrie de la carte lors du raffinement (et donc de risquer des effets de bord similaires à ceux évoqués plus haut dans le cas des plongements en nombres flottants), on garde la discrétisation des anciens grands segments lorsqu'on les scinde en deux (suite à une intersection ou à une incidence) : les nouveaux grands segments ainsi créés occupent les points discrets sur lesquels passait le grand segment initial qui a été scindé (l'algorithme de discrétisation n'est donc pas réappliqué).

Concernant la structure de carte combinatoire proprement dite, les grands brins et les petits brins sont placés dans deux cartes différentes, respectivement appelées grande carte et petite carte, chacune contenant une liste de ses brins. On appelle z-carte (« z » car les petits segments « zig-zaguent ») l'ensemble formé par une petite carte et par la grande carte qui lui correspond. Deux types de pointeurs relient les grands brins et les petits brins d'une même grande arête (cf. Fig. 4.8). Le premier type de pointeurs part des grands brins vers les petits brins : il indique pour chacun des deux grands brins celui des petits brins qui est plongé sur le même point que lui. Le deuxième type de pointeurs va des petits brins aux grands brins. Pour chacun des deux petits brins extrémités, un pointeur indique le grand brin qui est plongé sur le même point. De plus, lorsque l'on parcourt l'arête dans l'ordre de succession des petits brins, les pointeurs indiquent alternativement l'un ou l'autre des grands brins.

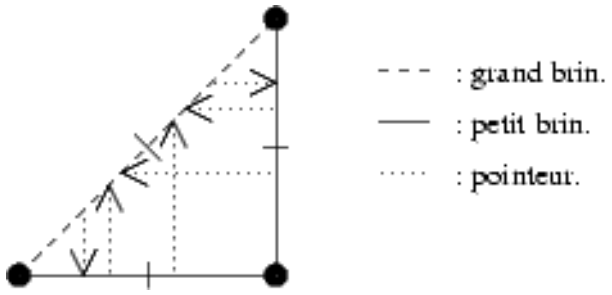


FIG. *4.8: Définition des pointeurs entre grands brins et petits brins.*

On peut remarquer qu'une telle représentation discrète permet de passer directement à une grille d'occupation classique, dont les cellules correspondent à un carré de taille unité : il suffit de parcourir l'ensemble des brins et de marquer les cellules correspondant aux arêtes « obstacles » comme occupées. Si l'on ajoute des degrés d'occupation, le « remplissage » des régions peut se faire en parcourant l'ensemble des faces de la carte : on applique sur chacune de ces faces un algorithme de « scan-conversion » (cf. [72]) pour indiquer le degré d'occupation de cette face.

Cette transposition rapide permet d'envisager l'emploi d'algorithmes basés sur ces grilles d'occupation

pour l'appariement de cartes, les stratégies d'exploration, etc. Par ailleurs, si la discrétisation « en dur » des segments alourdit la représentation, on note que dans le cas d'environnements intérieurs structurés courants où les murs sont perpendiculaires entre eux, le nombre de petits segments sera relativement réduit (à condition que le repère de la carte soit aligné avec les directions principales de l'environnement). En outre, nous verrons que le raffinement des z-cartes s'effectue sur les petits segments horizontaux et verticaux, ce qui simplifie le processus. Quant aux opérations de plus haut niveau sur les cartes combinatoires visant à construire la représentation de l'environnement (appariement de carte locale et de carte globale et gestion des incertitudes géométriques et topologiques notamment), elles s'effectueront en principe directement sur les grands brins.

4.4.2 Transposition de l'algorithme de raffinement par balayage

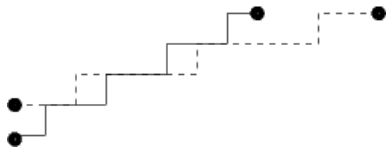


FIG. *4.9: Illustration des intersections multiples.*

Dans le cas discret, deux grandes arêtes peuvent se chevaucher sur plusieurs petites arêtes de sorte qu'il peut y avoir plusieurs points d'intersection. De plus, dans l'algorithme par balayage défini par Cazier [23], l'ordre sur les arêtes est appliqué à des arêtes d'orientation quelconque, aussi bien horizontales que verticales ou en biais. Cependant, il est difficile (impossible ?) d'établir un ordre sur les grandes arêtes qui permette de les comparer en un temps constant ou mieux que linéaire (suivant le nombre de petites arêtes). En effet, pour pouvoir déterminer si deux grandes arêtes interagissent (si elles sont sécantes, incidentes ou si elles se superposent), il ne suffit pas de connaître leurs extrémités, mais la position exacte des petits segments qui les composent. Ces difficultés nous ont conduit à effectuer un raffinement sur les petits brins, à partir desquels on reconstruit les grands brins après le raffinement. Il suffit pour cela de maintenir les pointeurs adéquats entre petits brins et grands brins lors de l'application des règles de raffinement sur les petits brins.

Par rapport au raffinement en nombres réels, le raffinement sur des petits brins verticaux et horizontaux permet un certain nombre de simplifications. D'abord, il n'y a plus que quatre angles possibles pour la direction des brins au lieu d'une infinité. Ensuite, l'ordre sur les brins est plus facile à définir. Enfin, le raffinement sur des brins verticaux et horizontaux évite le problème des « arêtes-écrans » indiqué par Cazier [23] [24], et permet d'éviter certains tests. La complexité du raffinement sur petits brins est la même que pour le cas réel, si l'on considère que n correspond dans le cas discret au nombre de petits brins : l'algorithme par balayage est en $O((n+i)*\ln(n))$. Si l'on considère que N est le nombre de grands segments mis en jeu initialement, I le nombre de grands segments ajoutés, et l une dimension caractéristique de la z-carte à raffiner (la longueur discrète du rectangle englobant par exemple, ce qui permet de borner le nombre de petits brins servant à discrétiser tout grand segment par $l\sqrt{2}$), alors on peut identifier $n+i$ à $(N+I)*l$ dans les calculs de complexité. On a donc un algorithme en $O((N+I)*l*\ln(N*l))$. Enfin, la reconstitution de la carte des grands brins après raffinement sur les petits brins s'effectue en un temps proportionnel au nombre de petits brins : on garde donc une complexité globale en $O((N+I)*l*\ln(N*l))$. Les figures 4.10 et 4.11 fournissent un exemple de raffinement.

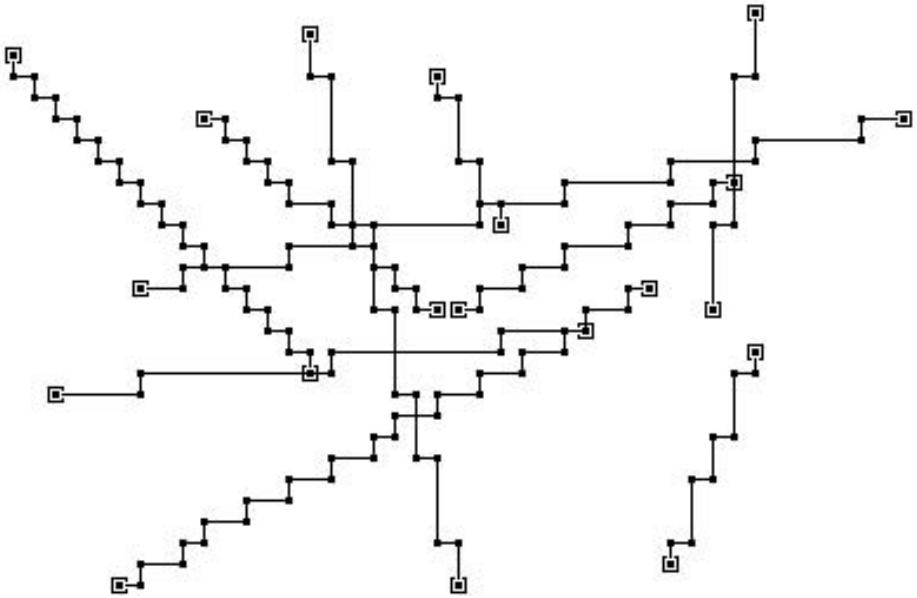


FIG. *4.10: Exemple de raffinement discret (avant).*

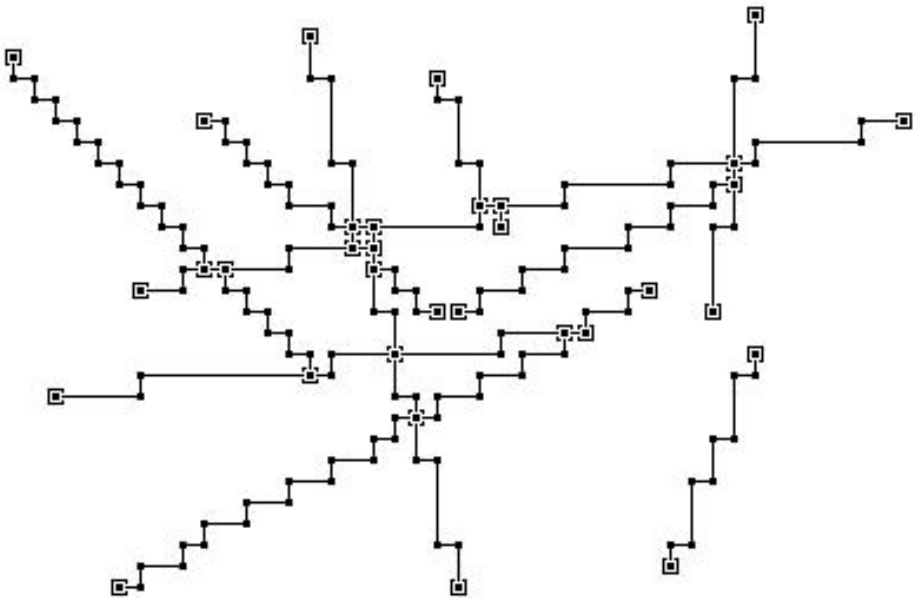


FIG. *4.11: Exemple de raffinement discret (après).*

4.5 Description de notre modèle

Nous choisissons de représenter l'environnement selon une partition polygonale en zones libres ou occupées par un obstacle, d'où le recours au modèle de **cartes combinatoires**. Ce modèle nous permet notamment de prendre en compte certaines relations particulières entre primitives géométriques constitutives de la carte. Ainsi, **les arêtes adjacentes** (murs contigus par exemple) **sont explicitement reliées entre elles** au sein des lignes polygonales (via les relations α_1 des cartes combinatoires sur les brins correspondants).

Pour rendre ce modèle de cartes combinatoires efficace dans le cadre du SLAM, nous introduisons en outre quelques éléments spécifiques qui s'intègrent facilement dans le formalisme des cartes combinatoires :

- **les arêtes de la carte sont étiquetées « obstacle »** ou **« non obstacle »** (via l'emploi d'un label sur les arêtes de la couche topologique des cartes combinatoires), selon qu'elles correspondent aux frontières d'obstacles ou aux limites de champ de perception du capteur (limites de portée ou lignes de visée liées aux occultations) ;
- **chaque face présente un degré d'occupation histogrammique** (matérialisé par un label sur chaque face de la carte) : cet indice indique ainsi le nombre de fois où la cellule correspondante a été observée comme libre d'obstacle.

Pour construire la carte globale, nous procèdons par **fusion successive de polygones d'espace libre locaux**, à l'instar Moutarlier [139] ou Gonzales et al. [153], ce qui permet notamment de conserver les informations d'adjacence entre arêtes issues des polygones locaux et de calculer les degrés d'occupation.

Plus précisément, comme le souligne Howard [87], il s'agit d'une intégration progressive des polygones d'espace libre (étoilés et sans « trou ») dans un « polysoïde » (union de polygones, pouvant contenir des trous). Toutefois, à la différence de ces travaux, **nous maintenons différents degrés d'occupation** (cf. Fig. 4.12) au lieu des trois valeurs classiques « libre », « occupé » ou « inconnu » : chaque face de la carte porte un label qui précise le nombre de fois où elle a été observée comme libre. Comme nous le verrons par la suite, **cet étiquetage permet de détecter des incohérences**, notamment sur les mises en correspondance, lorsque les appariements peu sûrs sont prudemment écartés afin d'éviter la divergence du filtre en cas d'erreur. Ce système permet également de **faire des choix de correction**, en comparant les valeurs histogrammiques de part et d'autre d'une arête par exemple. D'une certaine manière, on retarde ainsi les décisions d'appariement et de fusion.

On peut remarquer que la plupart des représentations surfaciques actuelles recourent à des degrés d'occupation probabilistes (voire à des valeurs déduites de la théorie de l'évidence ou de la logique floue) plutôt qu'à des degrés d'occupation histogrammiques. En effet, le cadre histogrammique conduit généralement à des représentations de moins bonne qualité, même s'il est bien adapté à l'évitement d'obstacles à grande vitesse par exemple [13]. Toutefois, dans notre approche, le but est de détecter des incohérences plutôt que de raffiner progressivement la position des obstacles : l'estimation de position des obstacles est gérée par ailleurs, via la mise en correspondance de primitives. En outre, notre système est susceptible d'intégrer moins de cartes locales que dans le cas d'une construction de grille d'occupation à partir de données sonar par exemple : un cadre probabiliste risquerait donc de conduire à des difficultés de convergence.

Pour construire la carte en ligne, il est indispensable de modéliser et de gérer les incertitudes de différentes natures :

- **incertitudes géométriques** : pour prendre en compte les imprécisions sur la position des primitives géométriques de la carte, on peut modéliser ces erreurs dans un cadre probabiliste (variables gaussiennes par exemple), dans un cadre de logique floue (ensembles flous) ou par une approche inspirée de la théorie de l'évidence (masses de croyance). Nous préciserons au chapitre suivant

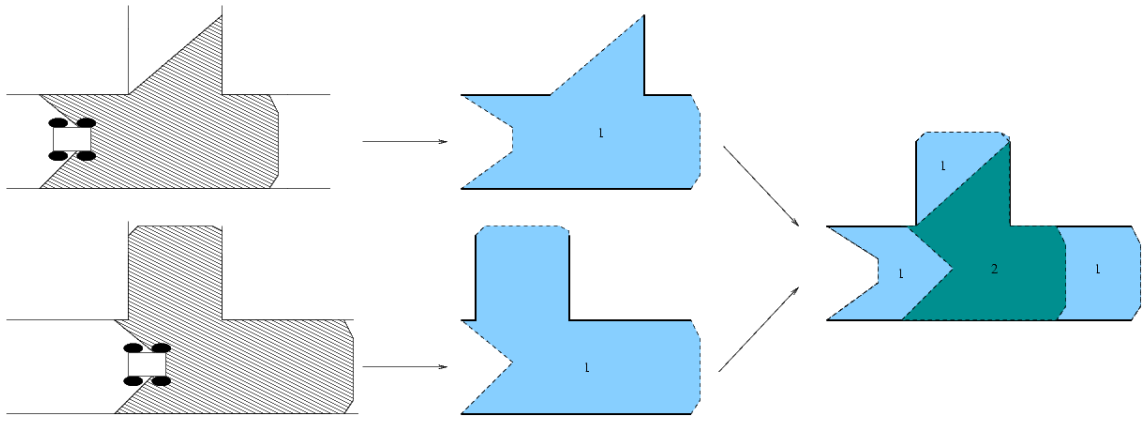


FIG. *4.12: Fusion de polygones d'espace libre locaux avec mise à jour des degrés d'occupation histogrammiques. Chaque cellule porte un label qui indique le nombre de fois où elle a été observée comme libre. Les pointillés correspondent à des segments « non obstacles » , c'est-à-dire aux limites de champ de perception du capteur (limites de portée ou lignes de visée correspondant aux occultations).*

pourquoi nous avons opté pour un cadre probabiliste ;
- **incertitudes topologiques** : nous avons proposé les degrés d'occupation histogrammiques qui renseignent sur la forme de l'espace libre (modèle surfacique) et induisent ses propriétés topologiques (nombre de trous, etc.).

4.6 Discussion

4.6.1 Position de cette représentation

Le modèle de cartes proposé mélange donc différents types de cartes métriques et topologiques :
- **carte métrique basée sur des primitives géométriques** (segments), et même sur des combinaisons de primitives géométriques (lignes polygonales qui relient les segments adjacents) ; la notion d'objet plus complexe (polygones ou ensembles de polygones notamment) n'est pas loin : elle est facile à modéliser via l'adjonction d'étiquettes sur les entités topologiques de la carte (dans la représentation proposée, on peut aisément détecter des objets comme des obstacles fermés tels que des piliers, jamais observés comme libres et entourés de segments « obstacles ») ;
- **représentation surfacique** constituée d'une grille d'occupation dont les cellules sont de taille et de forme irrégulières (pour les deux versions des cartes combinatoires : réelles ou discrètes), avec possibilité de transformation rapide en grille d'occupation régulière (via un algorithme de « scan conversion » par exemple [72]) ;
- **informations topologiques d'adjacence** entre éléments de la carte (sommets, arêtes et faces), et notamment entre cellules de la partition (faces) : on obtient ainsi un graphe topologique sous-jacent de l'espace libre, dont les sommets correspondent aux cellules et dont les arêtes correspondent aux informations d'adjacence entre cellules. On peut facilement extraire de ce graphe le nombre de trous dans l'espace libre (zones connexes inconnues ou constituées d'obstacles), et y chercher plusieurs chemins alternatifs pour se rendre à un même lieu (dans un espace discrétisé via la partition induite par la carte combinatoire).

En outre, des modèles d'incertitudes géométriques et topologiques facilitent la construction incrémentale du modèle.

Ainsi, cette représentation est hybride au sens où elle mêle plusieurs types de représentations métriques (cartes d'amers et modèle surfacique) ainsi que des informations topologiques d'adjacence. Selon la classification proposée au chapitre précédent, **l'hybridation métrique / topologique correspond en quelque sorte à un réseau de cartes locales issues d'un partitionnement**. Ces cartes locales sont très simples puisqu'il s'agit de polygones de degré d'occupation uniforme. On peut cependant remarquer que l'on dispose d'une information topologique qui n'existe pas en général dans les autres représentations employées pour le SLAM : l'adjacence entre arêtes et faces ou entre sommets et faces (en plus de l'adjacence entre cellules qui existe dans les représentations topologiques classiques et de l'adjacence entre arêtes ou entre arêtes et sommets qui existe dans les représentations métriques à base de combinaisons de primitives géométriques [18] [21]). Cette information supplémentaire permet notamment de vérifier des contraintes d'intégrité (par exemple pour éviter les « slivers » que nous avons vus au chapitre précédent, bien connus en géographie, et qui résultent d'une mauvaise superposition des primitives appariées). Ainsi, la représentation résultante favorise la détection et la correction des incohérences : on peut réellement combiner le modèle surfacique et la carte d'amers via une fusion valide des polygones d'espace libre locaux.

En fait, comme dans les modèles géographiques classiques, **on dispose d'une véritable couche topologique** et le système de labélisation associé aux cartes combinatoires permet de réaliser des raisonnements spatiaux élaborés ainsi que des opérations de manipulation efficaces (grâce aux précalculs) lors de la mise à jour de la carte et de son exploitation. En particulier, comme nous l'avons indiqué au chapitre précédent, cette couche facilite les opérations booléennes qui sont à la base de nombreuses manipulations et requêtes réalisées sur les cartes dans les systèmes d'information géographiques, et qui sont susceptibles d'être employées dans un cadre robotique.

4.6.2 Discussion sur cette représentation

La représentation choisie apparaît donc très structurée par rapport aux cartes classiques existant dans la littérature du SLAM, d'autant que l'objectif est de la construire de façon incrémentale (en ligne), et non a posteriori.

Toutefois, par rapport aux critères de comparaison des modèles de cartes proposés au chapitre 2, cette représentation présente a priori quelques limitations :

- Sa construction en ligne nécessite en principe une algorithmique plus complexe, du fait des contraintes induites sur le processus de cartographie pour maintenir la structure de la carte. Le temps réel n'est donc pas acquis. En revanche, la représentation n'induit pas de trajectoire particulière pour le robot et rien n'interdit a priori une construction véritablement « passive », ce qui favorise la souplesse des modes de construction de la carte.

- Le modèle polygonal utilisé paraît plutôt adapté à un capteur de distance précis tel qu'un télémètre laser à balayage. La généricité par rapport aux capteurs n'est donc pas évidente.

- La représentation topologique proposée est assez différente des représentations classiques et n'offre pas nécessairement tous les avantages que l'on pourrait en attendre. La partition de l'espace libre utilisée ne correspond pas à des lieux caractéristiques et n'épouse pas la structure particulière du bâtiment (pièces, couloirs...). Elle est plutôt liée au trajet suivi par le robot et aux occultations successives qui en résultent (d'où le découpage de l'espace libre et les degrés d'ocupation variables d'une cellule à l'autre). Ainsi, l'adjonction de sémantique (nom des pièces, désignation des pas-

sages,...) pourrait nécessiter une restructuration de cette partition. En outre, on ne dispose pas non plus d'un réseau de déplacement canonique à travers l'environnement, tel qu'un diagramme de Voronoï. Toutefois, l'intégration a posteriori de ces réseaux est envisageable (par raffinement de cartes par exemple), tout en gardant une structure de carte cohérente : cela augmenterait ainsi les possibilités d'exploitation de la représentation par le robot).

– Concernant la généricité par rapport à l'environnement, si le modèle permet de représenter des formes polygonales quelconques, il paraît mal adapté aux objets fins et isolés tels que les pieds de table, du fait de la multiplication des lignes de visée (arêtes « non obstacles ») qui risque d'en découler. Toutefois, la représentation est capable de prendre en compte ces petits objets, et une gestion particulière reste envisageable pour améliorer l'efficacité du système (détection des objets fins et insertion de ces objets dans la carte via une « arête virtuelle » unique - cf. chapitre 7). Par ailleurs, la version discrète des cartes combinatoires induit un découpage de l'espace selon une grille rigide, ce qui facilite la gestion de la carte mais limite les possibilités d'adaptation automatique à l'échelle des objets. Toutefois, l'utilisation des nombres flottants reste possible moyennant certaines précautions (calculs en arithmétique exacte par exemple). Enfin, le recours au modèle polygonal peut paraître restrictif, notamment si l'environnement présente des formes courbes. Nous verrons que les algorithmes employés s'attachent à limiter les problèmes induits par ces formes courbes (en particulier la multiplicité des approximations polygonales qui en résulte). En outre, la structure de cartes combinatoires est extensible aux modèles courbes (plongement courbe des arêtes), même si l'opération de raffinement correspondante reste à étudier.

En contrepartie de ces possibles limitations, la représentation proposée promet d'importants avantages :

– Même dans sa version discrète, le modèle est plus compact qu'une grille d'occupation classique, bien qu'il propose une représentation surfacique similaire.

– Concernant l'exploitation de la carte par le robot, celui-ci dispose à tout moment d'une représentation structurée, à la différence des systèmes qui introduisent cette structuration a posteriori au lieu de la construire de manière incrémentale. En outre, le modèle choisi impose une certaine cohérence entre toutes les couches de représentation : en particulier, les frontières de faces coïncident intrinsèquement avec les arêtes de la carte, ce qui assure la validité des raisonnements spatiaux. Cette propriété facilite également le « nettoyage » de la carte (suppression de microfaces, etc.). Plus généralement, la couche topologique et le système d'étiquetage associé permettent de réaliser efficacement des opérations spatiales de haut niveau telles que les requêtes appliquées aux systèmes d'information géographique. La richesse du modèle offre une certaine généricité : elle permet de mettre en œuvre des algorithmes variés (pour la planification de trajectoires par exemple), voire des combinaisons d'algorithmes opérant sur des couches distinctes du modèle, dans un souci d'efficacité et de robustesse. Enfin, le niveau de structuration proposé permet d'approcher la notion d'objet, ce qui ouvre la voie vers la gestion des éléments dynamiques, l'analyse de scènes et l'ajout d'informations sémantiques.

– Concernant l'exploitation par l'homme, elle s'avère assez naturelle puisque le modèle s'apparente à un plan d'architecte avec les informations de plein et de vide délimitées par des lignes polygonales. La différence essentielle concerne les informations d'incertitudes (degrés d'occupation et imprécisions sur la position des sommets) qui peuvent aisément être masquées à l'affichage ou apparaître de façon ergonomique.

– Comme nous le verrons, la fermeture de cycles de l'environnement peut être réalisée de manière transparente par les techniques de filtrage de Kalman par exemple. La richesse du modèle peut en outre contribuer à la robustesse du système, en particulier durant la phase de mise en correspondance entre observation locale et carte globale ou lors de la détection de fermeture de cycles. La structuration facilite également la détection et la correction explicite des incohérences.

4.6.3 Applications possibles

Nous avons vu au chapitre précédent, dans la section consacrée aux modèles géographiques, que la couche topologique se prêtait bien aux opérations booléennes élémentaires (union, intersection, différence...) entre entités topologiques de la carte, et aux opérations plus élaborées qui reposent sur des combinaisons d'opérations booléennes. Dans un cadre robotique, on peut notamment songer aux applications suivantes :

– **localisation** : possibilité de combiner plusieurs couches du modèle (représentation surfacique, primitives géométriques...) afin de vérifier les hypothèses de positionnement du robot dans la carte à partir de ses obervations courantes ;

– **planification de trajectoires** : possibilité d'utiliser les informations topologiques pour réaliser une planification grossière et les informations métriques pour la définition plus fine des trajectoires. Pour la définition géométrique du trajet à suivre, la grille d'occupation sous-jacente permet la mise en œuvre des algorithmes classiques de transformation en distance ou de champs de potentiel, tandis que la représentation à base de primitives géométriques peut améliorer l'efficacité de cette planification (moyennant l'épaississement des obstacles par le diamètre du robot [119]). On peut également envisager des techniques à base de partitions [134] exploitant directement la subdivision modélisée par la carte combinatoire ;

– **exploration** : extraction efficace des frontières de l'espace libre restant à explorer (via le parcours rapide des arêtes associées aux degrés d'occupation définis de part et d'autre), mise en œuvre d'algorithmes de « Next Best View » [153], etc. ;

– **raisonnements spatiaux plus élaborés** exploitant les opérations booléennes (via l'opération de raffinement notamment), voire des informations sémantiques : application d'opérations classiques dans les systèmes d'information géographique telles que le fenêtrage (recherche des objets de carte situés à l'intérieur d'une zone donnée - une zone contaminée par exemple) ou la superposition de cartes (recherche des intersections avec une autre couche thématique représentant par exemple les frontières de bâtiments issues d'une carte géographique plus macroscopique, des régions couvertes par les moyens de communication, des zones dangereuses,...). En outre, l'exploitation de l'information sémantique attachée (manuellement ou automatiquement) aux entités topologiques telles que l'orientation (points cardinaux) ou le type de pièces (« couloir » pour une pièce allongée...) permettrait des requêtes du type : « trouver tous les chemins qui mènent à ce point de ralliement à travers la partie nord du couloir ».

4.7 Quelle méthode de construction de carte sur ce modèle ?

Comme nous l'avons expliqué précédemment, l'algorithme de construction de cartes proposé procède par fusion successive des polygones d'espace libre locaux. A chaque étape de fusion, il s'agit donc de mettre en œuvre les opérations suivantes :

- mise en correspondance des primitives géométriques du polygone local avec les primitives de la carte globale ;
- mise à jour géométrique de la carte globale en fonction des nouvelles observations induites par la mise en correspondance ;
- mise à jour topologique de la carte globale afin de corriger les degrés d'occupation histogrammiques et de maintenir une structure de carte combinatoire cohérente.

Concernant la mise à jour géométrique de la carte, nous optons pour une approche probabiliste classique à base de filtrage de Kalman étendu. En effet, nous avons vu dans les chapitres d'état de l'art que ces techniques ont fait l'objet de nombreuses études et améliorations au cours des dernières années : preuves de convergence dans le cas linéaire, gestion des problèmes de linéarisation, réduction du coût de calcul, etc. En outre, elles offrent des possibilités d'évolution vers des approches alternatives telles que le FastSLAM, qui les simplifie en les combinant à un filtrage particulaire. Ainsi, lors de cette phase de mise à jour géométrique de la carte globale, il s'agit à la fois de préciser les positions des diverses primitives géométriques constitutives de la carte, en fonction des observations, mais aussi de compléter cette carte par les éléments nouveaux qui sont observés pour la première fois. C'est à ce stade qu'intervient le filtrage de Kalman proprement dit. Le robot observe un certain nombre de primitives géométriques, dont certaines sont appariées à des primitives connues. Ces observations relatives entre le robot et l'environnement lui permettent de corriger à la fois sa propre position et la position des primitives de la carte (ainsi que les incertitudes gaussiennes associées). Notons que dans ce processus, la localisation des primitives non observées est également raffinée, à travers les corrélations existant entre les erreurs d'estimation des divers éléments géométriques.

Par rapport à l'essentiel des représentations géométriques fondées sur les frontières, la difficulté pour la mise à jour de notre carte vient du fait que les primitives géométriques ne « flottent » pas dans l'espace les unes par rapport aux autres (comme dans les représentations « spaghettis » évoquées dans le cadre des systèmes géographiques), mais sont reliées entre elles de façon très précise. En particulier, les segments consécutifs d'une même face sont reliés entre eux par l'intermédiaire de sommets. Quelques autres représentations présentent également cette particularité [139] [18] [153] [21], mais rares sont celles qui combinent ce type de modèle à une représentation de l'espace libre [139] [153]. De plus, dans ces dernières approches, il n'existe que deux degrés doccupation possibles : libre ou occupé. Ainsi, par construction, les sommets ne sont en principe incidents qu'à deux arêtes au maximum (si la carte est exempte d'incohérences tels que des croisements de frontières obstacles et si l'on interdit les cas dégénérés tels que les obstacles d'épaisseur nulle ou les frontières d'obstacles qui se touchent de manière ponctuelle). Dans notre représentation, en revanche, certains sommets peuvent être incidents à plus de deux arêtes (quatre au maximum dans le cas des cartes discrètes). En conséquence, si la mise à jour consiste à corriger la position des segments (comme dans le cas de [139] par exemple), des incohérences risquent d'apparaître au niveau des sommets incidents à plus de deux arêtes, puisque les intersections deux à deux des nouveaux segments (ou plutôt de leurs nouvelles droites porteuses) ne coïncideront plus (cf. Fig. 4.13).

Ainsi, afin de prévenir ou de corriger ce type d'incohérences, il convient de définir précisément sur quel type de primitives nous devons travailler pour mettre à jour l'état (quels paramètres apparaissent dans le vecteur d'état du filtre de Kalman), quelles observations nous pouvons faire sur ces primitives

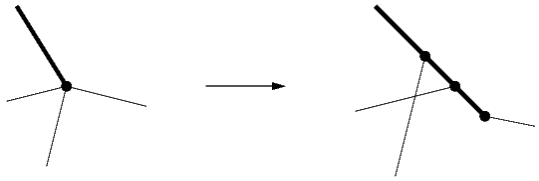


FIG. *4.13: Incohérences liées au déplacement d'un segment (segment en gras) dans le cas des sommets incidents à plus de deux segments. Ce déplacement induit l'apparition de deux nouveaux points d'intersection entre droites porteuses.*

(ce qui est lié au processus de mise en correspondance), et comment, à partir de la mise à jour de l'état, nous pouvons remettre à jour la carte combinatoire globale. Ces différentes questions seront abordées en détail dans les chapitres suivants, qui traitent chacun de l'une des étapes du processus de cartographie.

Chapitre 5

Mise en correspondance entre carte locale et carte globale

« Etre éveillé c'est pouvoir percevoir certains points de repère. Le même travail que fait le navigateur selon le sextant, l'heure du bord et les tables - tu le fais mystérieusement selon tes instruments et tes documents vivants, à l'égard des choses, qui deviennent repères, planètes, chronomètres, mesures. »
P. Valéry (« Cahiers 1894-1914 »)

5.1 Position du problème

Dans le cadre de la construction automatique de cartes d'environnement, la mise en correspondance entre carte globale et observation locale est une étape cruciale : elle permet d'assurer la cohérence de la représentation (en évitant les redondances et les faux appariements) et de recaler la position du robot. Elle est d'autant plus importante dans les méthodes à base de filtrage de Kalman qu'un mauvais appariement peut rapidement conduire à une divergence du filtre.

5.1.1 Mise en correspondance de primitives géométriques

Dans le cadre du SLAM, la technique de mise en correspondance employée dépend bien sûr de la représentation utilisée et de la méthode d'estimation mise en œuvre. Par exemple, pour les représentations à base de primitives géométriques générées par filtrage de Kalman, on recourt souvent à un test de Mahalanobis qui permet de prendre en compte les covariances dans la mesure de distance entre un élément de la carte globale et un élément de la carte locale. Pour les représentations construites par superposition de scans bruts, il s'agit généralement d'apparier des scans : il existe des techniques variées, fondées sur des appariements point à point ou exploitant l'extraction de primitives géométriques. Pour les grilles d'occupation, certaines techniques procèdent par corrélation entre deux grilles discrètes [168] [81], tandis que d'autres extraient des segments [81]. Enfin, pour les représentations topologiques, la mise en correspondance s'effectue en général sur les sommets en comparant le nombre d'arêtes incidentes dans les graphe ainsi que des attributs attachés aux nœuds et aux arcs.

Dans notre cas, on peut envisager des combinaisons de ces techniques puisque l'on dispose de différentes couches de représentations. En particulier, grâce à la représentation discrète, on peut envisager des techniques de corrélation. Nous nous focalisons toutefois dans un premier temps sur les primitives

géométriques car ces informations d'appariement entre primitives sont primordiales pour la mise à jour de la carte par filtrage de Kalman. En outre, cette approche limite a priori la combinatoire car le nombre de primitives est réduit par rapport au nombre de cellules d'une grille d'occupation ou le nombre de points dans un scan brut par exemple. Dès lors, il existe différentes options pour réaliser la mise en correspondance : nous les détaillons ci-après.

*** Possibilité d'exploitation de divers types de primitives géométriques :**

Les méthodes d'appariement exploitent divers types de primitives géométriques. Les plus courantes sont basées sur les points (ou éventuellement des coins, c'est-à-dire des « points orientés ») et les segments [139]. Quelques-unes concernent les cercles [206] ou des objets de plus haut niveau [18] [21]. Pour les segments, la correspondance est plus difficile à établir que pour les points : en effet, il faut assurer à la fois l'appariement des droites porteuses et la correspondance de la position de ces segments sur la droite porteuse. Il convient donc de vérifier que chaque segment se trouve « en face » de l'autre, en utilisant par exemple des informations sur la position des extrémités du segment sur la droite porteuse. Souvent, cette vérification est réalisée au moyen de projections orthogonales [139], mais elle suppose alors que l'on dispose d'une bonne estimation initiale des positions relatives des deux segments. Par ailleurs, il est possible de réaliser des appariements « homogènes » entre primitives de même type (point sur point ou segment sur segment) ou des appariements « hétérogènes » entre différents types de primitives (point sur segment par exemple [21] [139]).

*** Mise en correspondance dans l'espace de la carte ou dans l'espace des positions :**

Globalement, la mise en correspondance peut être réalisée dans deux espaces distincts : l'espace des positions ou l'espace des cartes. Pour le premier espace, il s'agit de déplacer la carte locale de manière à ce qu'elle se superpose au mieux à la carte globale : on recherche la meilleure position d'appariement. Les méthodes de localisation de robot mobile à partir de cartes existantes fonctionnent de cette manière. Pour le second espace, il s'agit de définir des liens de correspondance entre primitives locales et globales. Certaines approches combinent ces deux types d'appariement. Par exemple, dans le cadre de la reconnaissance d'objets dans des images, Beveridge [11] ou Loader [125] commencent par définir des hypothèses d'appariement entre primitives (segments), puis pour chacune de ces hypothèses, ils recherchent via des techniques d'optimisation (descentes de gradient, recherche taboue...) la meilleure estimée de position de l'objet local par rapport à la scène globale. Cette optimisation nécessite auparavant la définition d'une distance entre deux ensembles de segments.

Dans notre cas, nous privilégions une mise en correspondance dans l'espace des cartes puisque c'est le filtrage de Kalman qui exploite ces données d'appariement (à travers la définition des observations) pour calculer la correspondance dans l'espace des positions (la nouvelle position du robot en réalité). Toutefois, nous verrons qu'un premier appariement dans l'espace des positions peut s'avérer utile pour faciliter la mise en correspondance des primitives géométriques.

*** Appariement local ou global :**

La mise en correspondance de la carte locale avec la carte globale peut se faire à un niveau global ou seulement local. Au niveau global, on n'utilise aucune information a priori sur la position de la carte locale au sein de la représentation globale : on peut ainsi résoudre le problème bien connu du « robot kidnappé », c'est-à-dire du robot que l'on a déplacé à son insu, sans lui indiquer dans quelle zone il se trouve à l'issue de cette opération. Pour un appariement au niveau local, on dispose d'une estimation a priori (entachée

d'incertitude) sur la position du robot : il suffit donc de rechercher des mises en correspondances dans un espace limité autour des positions incertaines des éléments observés (déduites de la position incertaine du robot). Les techniques de cartographie par filtrage de Kalman fonctionnent généralement au niveau local, avec une position incertaine du robot modélisée par une gaussienne unimodale : il n'est pas possible de maintenir plusieurs hypothèses de position du robot et on ne traite pas le cas du robot kidnappé. On parle alors de « suivi de position ». Comme nous avons opté pour ce type de filtrage, nous nous orientons donc également vers un simple suivi de position. Une extension au filtrage particulaire telle que le FastSLAM pourrait permettre de passer à une mise en correspondance plus globale.

*** Méthodes gloutonnes ou optimales :**

Dans le cadre de la mise en correspondance de primitives, pour limiter la complexité des algorithmes, on recourt souvent à des méthodes gloutonnes, sous-optimales, qui ne garantissent pas de trouver la meilleure solution. En particulier, l'appariement par « recherche de plus proche voisin » est très répandu : pour chaque primitive locale, il s'agit de rechercher la primitive globale qui lui correspond le mieux (au sens où elle minimise une certaine distance ou un certain coût d'appariement). Or la meilleure solution globale n'implique pas que chaque primitive soit appariée à la plus proche dans l'autre carte. Certaines techniques légèrement plus élaborées considèrent au moins des couples de primitives à apparier, par exemple des coins constitués de deux segments [153].

Enfin, d'autres méthodes, plus optimales, s'attachent à définir une correspondance globale entre ensembles de primitives. Par exemple, Neira et Tardos ont développé un test de compatibilité global entre couples de primitives appariées [145]. Ce test est fondé sur la distance de Mahalanobis : les auteurs montrent qu'on peut calculer cette distance globale sur un ensemble d'appariements de manière incrémentale. Ce calcul incrémental est ensuite exploité dans un algorithme de recherche de type « branch and bound » (qui procède par élagage dans un arbre de recherche). L'objectif est de trouver la meilleure hypothèse globale de mise en correspondance, au sens de celle qui maximise le nombre d'appariements. Le « branching » utilise une technique de plus proches voisins pour essayer d'explorer en premier les meilleurs appariements dans l'arbre de recherche. Pour la partie « bound » (calcul de borne sur les sous-arbres restants), les auteurs définissent la qualité d'un nœud de l'arbre comme le nombre d'appariements potentiels restant.

Nous nous attacherons de même à développer une méthode aussi globale et optimale que possible.

*** Une possible exploitation de la sémantique :**

Enfin, on peut se demander comment sont traitées les mises en correspondance de représentations géographiques, où le format des cartes vecteurs exploitant une couche topologique est proche du nôtre. On s'aperçoit qu'il est souvent fait usage de la sémantique afin d'apparier les ponctuels, puis les réseaux linéiques et les éléments surfaciques. Même si cela est envisageable à l'avenir, nous tâcherons toutefois de réaliser la mise en correspondance sans l'aide d'informations sémantiques (notamment parce que nous utilisons des données laser qui sont moins adaptées à l'extraction de sémantique que des capteurs très riches tels que la vision), et plus généralement sans information de haut niveau (même avec des données laser, on peut extraire des objets de haut niveau, tels que des configurations particulières d'obstacles [18], auxquels on n'attache pas nécessairement de véritable information sémantique).

5.1.2 Spécificités induites par notre représentation et par notre approche

Nous nous plaçons dans le cadre d'un filtrage de Kalman et nous cherchons à fusionner les frontières d'obstacles qui se correspondent. Nous avons vu que la mise en correspondance qui nous intéresse en premier lieu se situe dans l'espace des cartes (appariement de primitives) plutôt que dans l'espace des positions (puisque le filtrage de Kalman se charge de cette correction de position). Comme nous disposons en outre d'estimations gaussiennes sur la position des primitives géométriques, nous nous orientons naturellement vers un test de Mahalanobis.

Il importe ensuite de déterminer quelles primitives géométriques doivent être employées dans cette mise en correspondance. Les sommets de la carte ne sont pas toujours significatifs, ni facilement observables. Par exemple, certaines intersections entre segment « obstacle » et segment « non obstacle » correspondent seulement à une limite d'occultation sur un mur : lorsque le robot se déplace vers une nouvelle position, il y a peu de chances pour que ce point se détache à nouveau sur le mur observé (cf. Fig. 5.1).

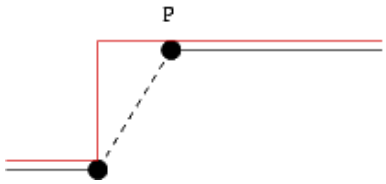


FIG. *5.1: Le point P sur la carte globale (en noir) a peu de chances d'être réobservé directement. Les lignes rouges représentent les frontières d'obstacle réelles.*

De même, lorsque l'on observe une surface au loin, l'échantillonnage sur cette surface devient lâche et on risque de ne pas voir certains décrochements qui se détacheraient mieux de près (cf. Fig. 5.2) : les sommets résultants diffèrent suivant la distance à laquelle se tient le robot.

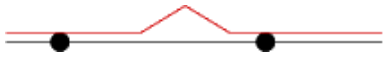


FIG. *5.2: Le décrochement du mur réel (en rouge) risque de ne pas être perçu de loin : les points noirs correspondent à des points de mesure télémétriques successifs très espacés car acquis à grande distance.*

En outre, dans un environnement qui présente des lignes courbes, les sommets issus de sa polygonalisation sont susceptibles d'évoluer car l'approximation polygonale n'est pas unique (5.3).

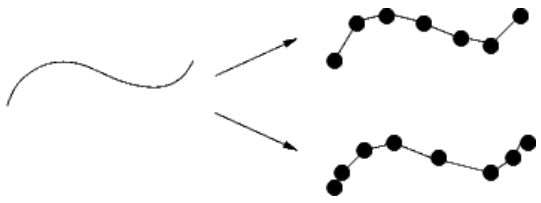


FIG. *5.3: Deux approximations polygonales différentes de la même courbe.*

Ainsi, nous nous orientons plutôt vers un appariement de segments, plutôt que vers un appariement de points. A la différence de certaines approches telles que celle d'Einsele [60], nous ne travaillerons pas sur le polygone initial constitué des segments élémentaires entre deux points de mesure : pour accélérer l'appariement et préparer une représentation plus compacte, nous chercherons à regrouper des segments

consécutifs alignés. Nous commençons donc par une phase de polygonalisation.

En outre, comme nous l'avons déjà souligné, nous souhaitons réaliser une mise en correspondance aussi globale que possible, en évitant les algorithmes gloutons de type « plus proches voisins ». Ainsi, plutôt que d'apparier indépendamment chaque segment de la carte locale, nous nous efforçons de mettre en correspondance des chaînes polygonales : il s'agit d'exploiter les informations d'adjacence dont nous disposons dans la structure de carte combinatoire. Nous pouvons envisager un test de compatibilité global. Toutefois, si l'algorithme décrit par Neira et Tardos [145] paraît particulièrement efficace dans le cadre d'un appariement de sommets, il semble plus délicat à mettre en œuvre dans le cas de segments.

Tout d'abord, la mise en correspondance des droites porteuses des segments (test de Mahalanobis sur leurs coordonnéees polaires) ne suffit pas. Il faut tenir compte des extrémités des segments pour vérifier qu'ils se trouvent bien l'un « en face » de l'autre sur ces droites : on ajoute donc une vérification sur les projections orthogonales, comme expliqué ci-dessus (cf. Fig. 5.4).

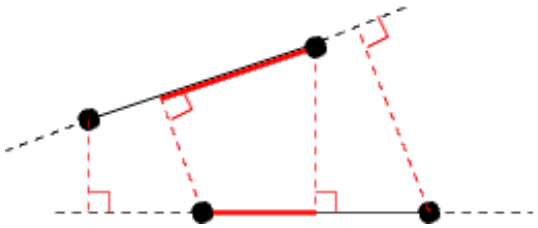


FIG. *5.4: Vérification que les segments se trouvent bien « l'un en face de l'autre », via les projections orthogonales. Ici, la vérification est positive puisque la projection orthogonale de chaque segment sur la droite porteuse de l'autre segment intersecte cet autre segment (portions rouges).*

Cette vérification nécessite donc une bonne estimation initiale des positions relatives des segments, c'est-à-dire une bonne estimation initiale de la position de la carte locale (de la position du robot en fait) par rapport à carte globale. Ainsi, une correction de position initiale (avant la mise en correspondance des primitives dans l'espace des cartes) peut s'avérer nécessaire. On vérifie également que l'observation se fait du bon côté du segment par rapport à la position de l'espace libre : en effet, une frontière d'obstacle est toujours observée du même côté, celui qui correspond à l'espace libre et non à l'espace occupé par l'obstacle (cette information est maintenue dans la carte globale via les degrés d'occupation des faces). Cela constitue une sécurité supplémentaire.

Par ailleurs, avec un test de compatibilité globale basé sur la distance de Mahalanobis [145], on risque de favoriser l'appariement de petits segments (qui ont généralement une variance plus élevée) au détriment des grands segments. Or les petits segments sont en principe moins significatifs et induisent des risques d'erreur plus importants. De plus, la méthode de « branch and bound » proposée maximise le nombre d'appariements sans distinguer les grands segments des plus petits. Dès lors, on peut envisager d'introduire explicitement la longueur de l'appariement dans le processus de mise en correspondance, à la manière des distances d'appariement proposées par Beveridge par exemple [11].

Enfin, il faut également gérer les cas d'appariements multiples de segments : certains sont autorisés et d'autres doivent être interdits. Ces interdictions peuvent être déterminées sur la base des projections orthogonales par exemple. Ainsi, le test de compatibilité globale de Neira et Tardos ne paraît pas directement applicable. On cherchera donc une méthode alternative basée sur la mise en correspondance de segments qui soit aussi globale que possible et qui gère les appariements multiples. Par ailleurs, comme la mise en correspondance de segments nécessite une bonne estimation de position initiale, on commence

par une mise en correspondance dans l'espace des positions. La figure 5.5 schématise le fonctionnement de notre algorithme d'appariement, dont nous détaillons à présent les différents modules.

5.2 Approximation polygonale

5.2.1 Méthodes existantes

Il existe de multiples méthodes d'approximation polygonale de contours échantillonnés : nombre d'entre elles sont utilisées pour les besoins de l'analyse d'images.

Par exemple, le livre édité par Cocquerez et Philipp cite plusieurs approches possibles [29] : nous nous inspirons de leur classification pour décrire les techniques rencontrées dans la littérature.

*** Reconnaître les segments de droite discrets :**

Ce type d'approche consiste à extraire individuellement des segments à partir des données brutes disponibles : elle ne fournit pas nécessairement des lignes polygonales mais plutôt des segments individuels. De plus, elle n'exploite pas nécessairement l'information de chaînage séquentiel des points de mesure.

Par exemple, lorsque l'on emploie un capteur de vision (caméra), on utilise souvent la transformée de Hough pour identifier les contours rectilignes de l'image. Nous avons d'ailleurs exploité cette approche dans une version antérieure et très préliminaire de notre algorithme de cartographie, basée sur des données visuelles [50] (cf. Fig. 5.6, 5.7 et 5.8). La transformée de Hough peut également être utilisée pour extraire des cellules obstacles alignées dans les grilles d'occupation [168].

Lorsque l'on opte pour des capteurs de distance comme les télémètres, on effectue parfois une étape d'agrégation pour extraire les points de données qui correspondent à des segments individuels ou à des lignes polygonales distinctes : par exemple, chez Castellanos et Tardos, cette étape est réalisée par filtrage de Kalman en suivant dans l'ordre les points de mesure du scan [21]. Certains points de mesure ne font partie d'aucun segment : ils sont rejetés de la polygonalisation. Ensuite, on peut estimer les segments par moindres carrés à partir des points de mesure bruts qui les constituent [170].

*** Approximer au mieux le contour pour un nombre de segments fixé**

Certains algorithmes fixent à l'avance le nombre de segments à obtenir à partir de l'observation. Chaque point de mesure est alors attribué à un segment donné (ou éventuellement à plusieurs segments avec pour chacun un certain indice de confiance). Par exemple, l'algorithme classique de classification floue par prototype décrit par Borges [14] définit C classes d'appartenance pour les points de mesures, qui correspondent chacun à un segment de prototype α_i, ρ_i (il s'agit des coordonnées polaires dans ce cas). Cet algorithme définit également des degrés d'appartenance flous $u_{i,j}$ compris entre 0 et 1, qui correspondent à l'appartenance du point de mesure P_j à la classe C_i. Ensuite, une optimisation sous contraintes permet de trouver à la fois les valeurs des prototypes α_i, ρ_i et les degrés d'appartenance floue $u_{i,j}$ des points en minimisant une fonction de coût (qui correspond à une somme de distances entre les points et les prototypes), sous contrainte que pour tout point P_j, $\Sigma_{i=1}^{C} u_{i,j} = 1$.

Un tel algorithme ne permet cependant pas d'adapter le nombre de segments au type d'environnement. De plus, comme le précédent, il n'assure pas la conservation de l'ordre des points de mesures le long du scan car l'attribution des degrés d'appartenance ne prend pas en compte cette information.

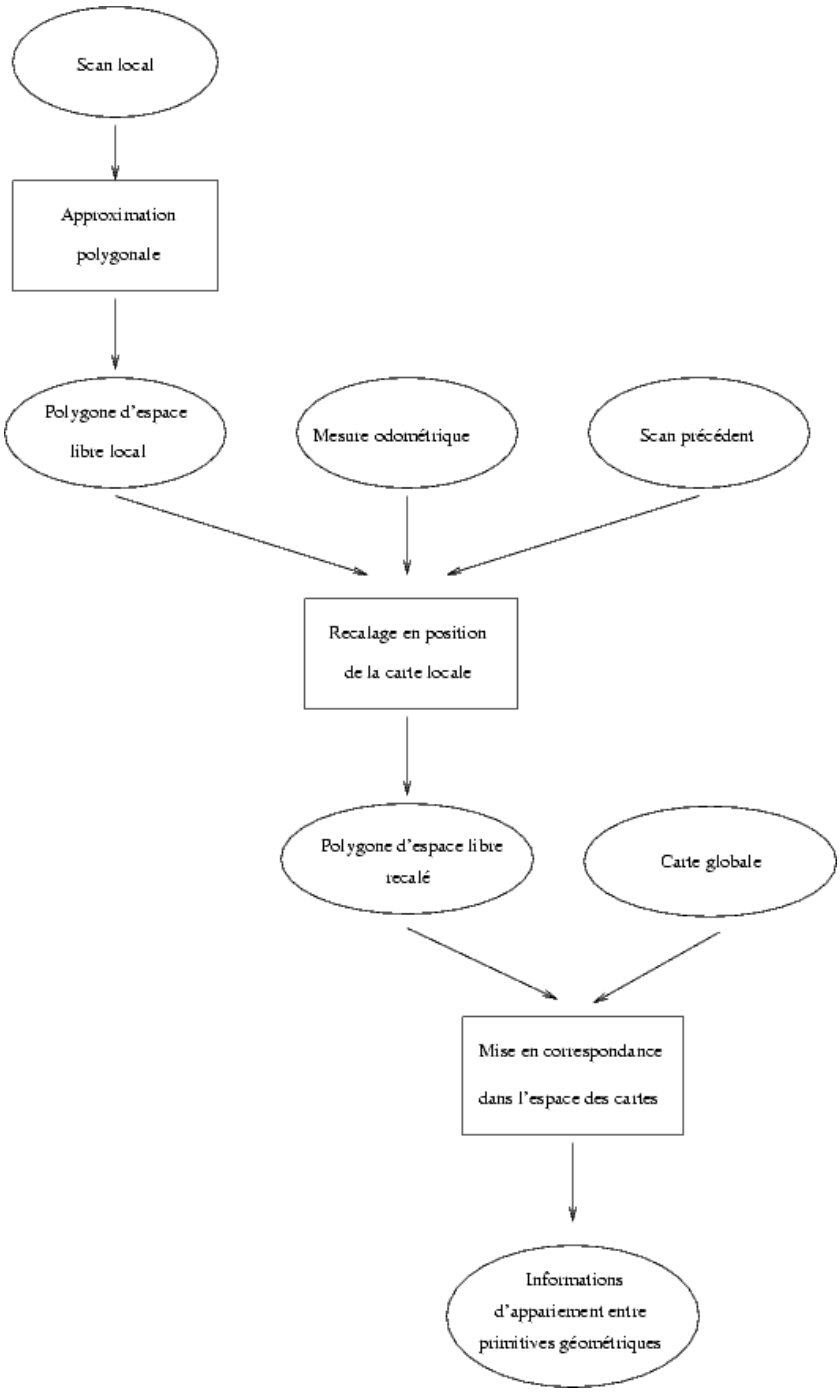


FIG. *5.5: Schéma de fonctionnement de notre algorithme d'appariement.*

FIG. *5.6: Images initiales acquises par le robot dans notre couloir (on distingue le sol marron, une double porte et des parois jaune pâle, ainsi qu'un meuble beige à portes brunes.*

FIG. *5.7: Polygones d'espace libre correspondants à ces images, reprojetés sur le plan du sol.*

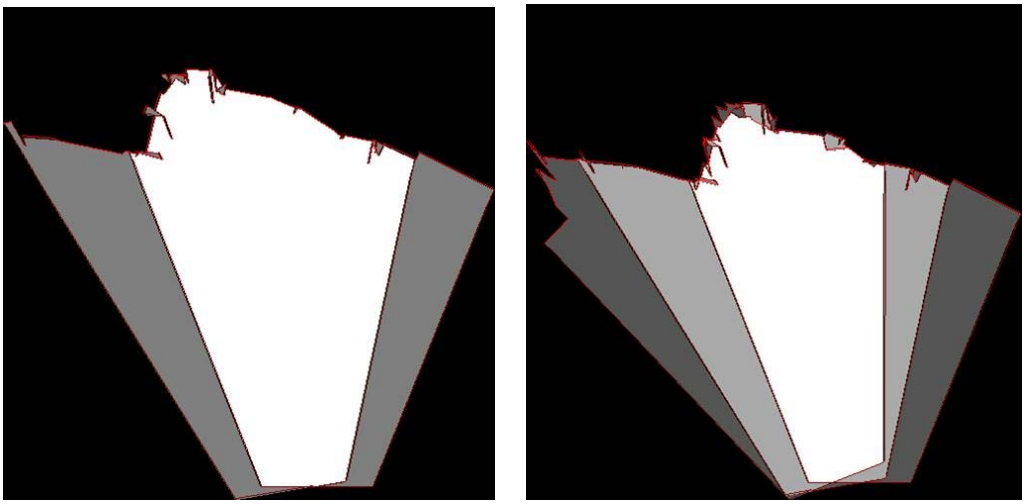


FIG. *5.8: Cartes combinatoires globales résultant de la fusion des deux premiers polygones de la figure précédente (à gauche), eux-mêmes fusionnés au troisième (à droite). Une zone de la carte apparaît d'autant plus claire qu'elle a souvent été observée comme libre.*

*** Approximer au mieux le contour pour une erreur maximale fixée**

Une autre technique classique consiste à découper un contour échantillonné de manière récursive : on parle souvent d'algorithmes de partition/fusion (« split and merge »). L'approximation polygonale peut alors résulter d'un découpage « dichotomique » ou « séquentiel » (on parle souvent dans la littérature de méthode « récursive » ou « itérative »). Elle assure en principe que les sommets du polygone et l'agrégation des points dans les segments conserve l'ordre séquentiel des points du contour. Dans le cas dichotomique (« récursif »), on commence par relier les extrémités du contour à polygonaliser par un segment unique. Ensuite, ce segment est découpé récursivement si certains critères sont vérifiés : la distance entre ce segment et les points réels qui lui correspondent doit être supérieure à un certain seuil mais sa longueur ne doit pas être inférieure à un autre seuil. Le découpage étant très sensible au choix des points extrémités, on ajoute souvent une phase de regroupement des segments adjacents alignés, qui réduit cette sensibilité [29]. Dans le cas séquentiel (« itératif »), les points sont traités en « flot de données » le long du contour : des points successifs sont regroupés tant qu'un certain critère, mis à jour à chaque ajout de point, est vérifié. Ce critère correspond généralement à une distance entre le segment en cours de construction et les points utilisés pour le construire. Ce type d'algorithme est toutefois très sensible au choix des seuils [14]. Expérimentalement, on constate également que les coins ont parfois tendance à être « rognés ».

*** Chercher des points caractéristiques pour décrire un contour :**

Pour segmenter le contour, on peut également rechercher des points spécifiques qui se démarquent au sein du contour. Par exemple, les points détectés par découpage itératif ou récursif peuvent être considérés comme tels. Les points présentant une forte courbure au sein du contour peuvent également entrer dans cette catégorie. Par exemple, Charbonnier procède en premier lieu par extraction de « points dominants » [25] : il s'agit de points de décrochements (à l'extrémité d'une ligne de visée « non obstacle » notamment) ou de points anguleux (en particulier les coins de murs). Les points de décrochement correspondent à des discontinuités dans les mesures de distance du scan : ils sont estimés par filtrage de Kalman et par logique floue. Quant aux points anguleux, ils correspondent à un maximum de courbure. Ensuite, une étape de fusion permet de retirer les points de décrochement et les points anguleux erronés.

5.2.2 Choix d'une technique

L'objectif de notre phase de polygonalisation est d'obtenir une segmentation du contour plus compacte que la liste des segments élémentaires reliant deux points de mesure successifs : il s'agit donc de regrouper les points de mesure alignés. En outre, nous cherchons à calculer les incertitudes sur les primitives géométriques extraites, afin de mettre en œuvre une technique d'estimation par filtrage de Kalman. Enfin, nous souhaitons maintenir les liens d'adjacence entre segments successifs pour mettre en place la structure de carte combinatoire : les segments extraits doivent être reliés les uns aux autres dans une même boucle polygonale. Ainsi, l'ordre des points du scan doit être conservé lors de la construction des frontières du polygone.

En conséquence, nous préférons éviter les algorithmes qui définissent précisément la position des segments par régression linéaire à partir des points de mesure. En effet, si deux segments adjacents sont estimés de cette manière, cela peut conduire à des configurations pathologiques (cf. Fig. 5.9) :

- cas de segments adjacents presque parallèles dont l'intersection est rejetée au loin ;
- cas où les points de mesure sont interclassés entre plusieurs segments différents qui ne vérifient plus l'ordre séquentiel du scan (risque signalé ci-dessus dans le cas de la classification floue ou de

l'extraction individuelle de segments) : comment définir un ordre d'adjacence entre ces segments ?
– risque d'apparition d'un polygone croisé.

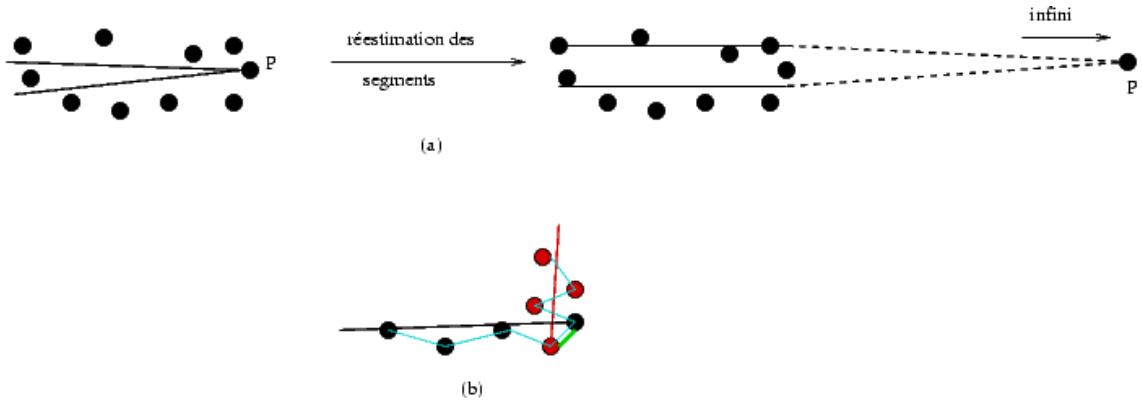


FIG. *5.9: Configurations pathologiques. (a) Cas de segments adjacents dont les nouvelles estimations sont parallèles et dont l'intersection est rejetée à l'infini. (b) Cas où les points de mesure sont interclassés entre plusieurs segments différents (ici un noir et un rouge), qui ne vérifient plus l'ordre séquentiel du scan (en bleu). Cela crée en outre un polygone croisé (les segments sont reliés en vert sur le contour du polygone résultant).*

Nous avons donc choisi d'appuyer la segmentation sur des points du scan uniquement, en conservant leur ordre initial : ainsi, certains points du scan sont sélectionnés pour constituer les sommets du polygone d'espace libre final.

Nous nous sommes donc naturellement orientés vers une méthode de découpage dichotomique (« récursif ») du scan, ou du moins des frontières « obstacles » : les frontières « non obstacles » (lignes de visée et portions de scan correspondant aux limites de portée du capteur) ayant été préalablement filtrées sur des critères de distance entre points successifs et d'orientation des segments élémentaires par rapport à la droite de visée. Toutefois, pour remédier au problème de « rognage » des coins, nous recherchons au préalable des points anguleux en comparant pour chaque point la direction du segment de droite et celle du segment de gauche (segments estimés par régression linéaire sur quelques points voisins). L'algorithme de découpage fonctionne donc sur les portions de contour « obstacle » délimitées par les points anguleux.

5.2.3 Illustration sur des données simulées

Pour les besoins de tests de nos algorithmes, nous avons développé au CEP Arcueil un petit utilitaire permettant de dessiner des environnements polygonaux et de simuler l'acquisition de scans dans ces environnements. Ces scans peuvent en outre être artificiellement bruités selon des paramètres gaussiens modifiables. La figure 5.10 montre ainsi un scan simulé (artificiellement bruité) à polygonaliser et la figure 5.11 le résultat de la polygonalisation. Les affichages se font ici à travers une interface générique que nous avons développée sous Qt permettant d'afficher des scans et des polygones intervenant dans nos algorithmes. On constate ici que les résultats de la polygonalisation se révèlent assez prévisibles et « naturels ». Ils correspondent notamment au polygone de départ ayant servi à générer le scan.

5.2.4 Illustration sur des données réelles

La figure 5.12 montre un scan réel aquis dans nos locaux (dans un couloir) au moyen de notre robot Pioneer et la figure 5.13 présente le résultat de la polygonalisation. De même, cette étape de

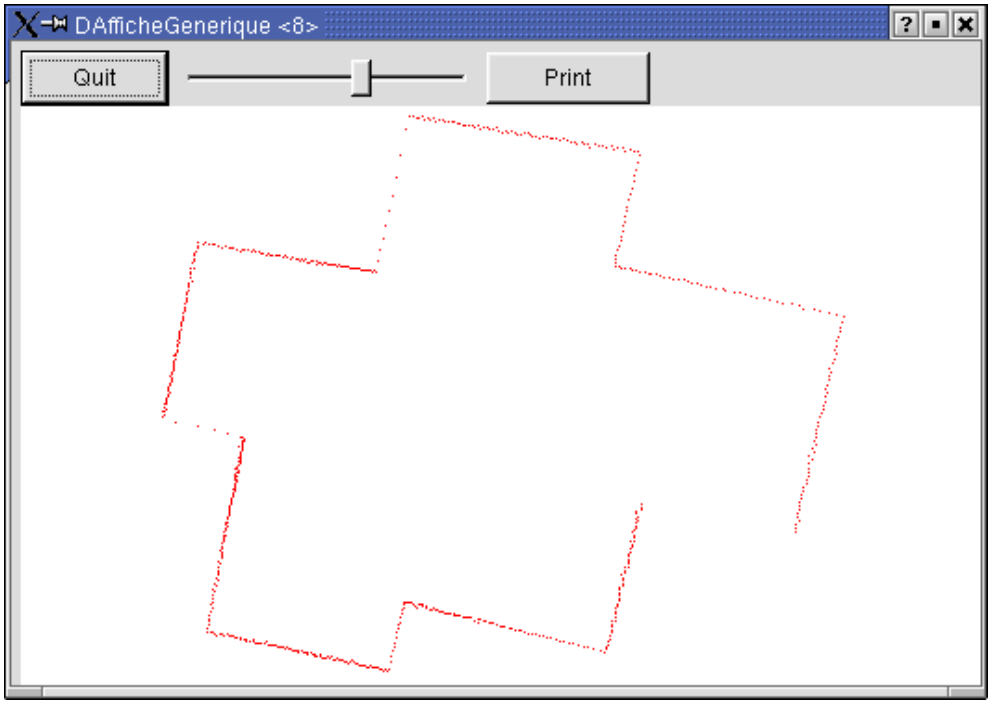


FIG. *5.10: Scan simulé à polygonaliser.*

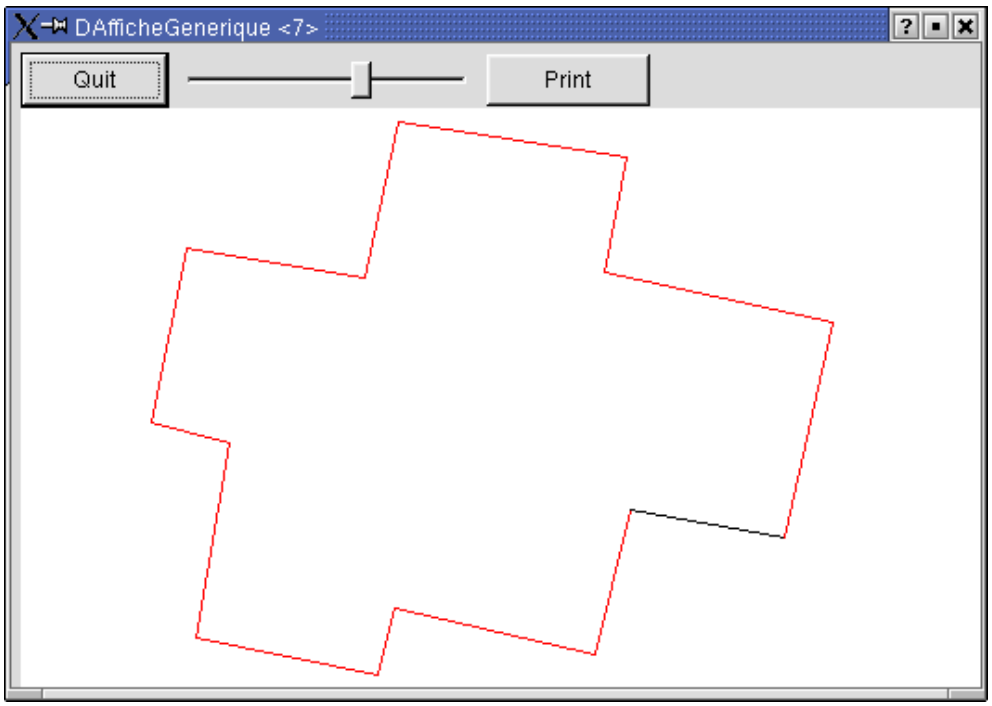


FIG. *5.11: Résultat de la polygonalisation : les segments « obstacles » apparaissent en rouge tandis que les segments « non obstacles » sont tracés en noir.*

structuration a permis de réduire de façon importante le volume de données et correspond effectivement à l'environnement initial.

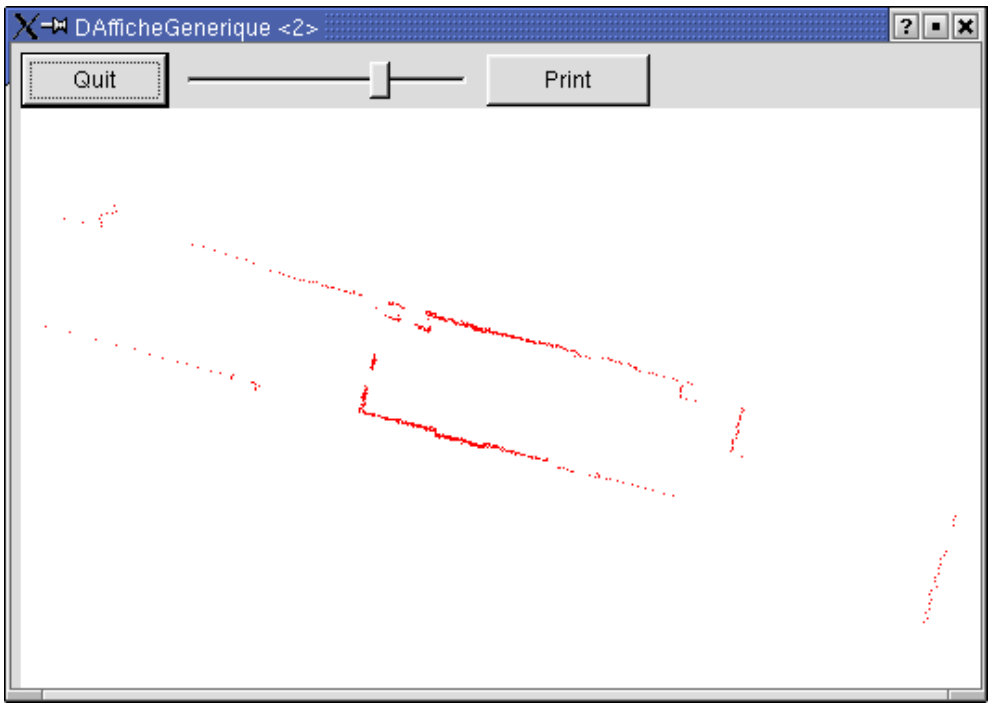


FIG. *5.12: Scan réel à polygonaliser.*

5.2.5 Perspectives d'amélioration

Les expérimentations montrent que cette méthode fournit généralement des résultats corrects. Celle-ci est cependant perfectible : par exemple, on peut tâcher de réduire sa sensibilité au bruit (dans la détection des points anguleux), en utilisant par exemple les techniques plus sophistiquées de Castellanos et Tardos (recherche des points de cassure par filtrage de Kalman et tests sur les résidus des approximations linéaires résultantes pour vérifier que la cassure les améliore). De plus, les résultats dépendent directement des seuils choisis pour le découpage récursif : ceux-ci ont été réglés expérimentalement pour limiter le nombre de segments résultant sans trop sacrifier les détails du contour. En outre, l'incertitude sur la position des extrémités de segments (les sommets du polygone) est peu précise : nous utilisons directement celle des points bruts correspondant. Les méthodes d'estimation de segments par régression linéaire [170] ou filtrage de Kalman [21] permettraient de fournir des estimations plus fines, calculées à partir de l'ensemble des points constituant le segment.

Nous n'avons cependant pas trouvé dans la littérature de méthode « sur étagère » plus élaborée qui réponde précisément aux critères que nous nous sommes fixés ci-dessus (en particulier le calcul de régression linéaire compatible avec le chaînage des segments et la gestion des configurations pathologiques citées ci-dessus). On peut également noter qu'une technique d'extraction de lignes polygonales (et donc d'un enchaînement de segments) par algorithme « EM » à partir de représentations de type « superposition de scans bruts » a été récemment proposée [190] : c'est une piste intéressante mais l'article ne précise pas comment calculer les incertitudes associées (il s'agit plutôt de la structuration a posteriori d'une carte existante).

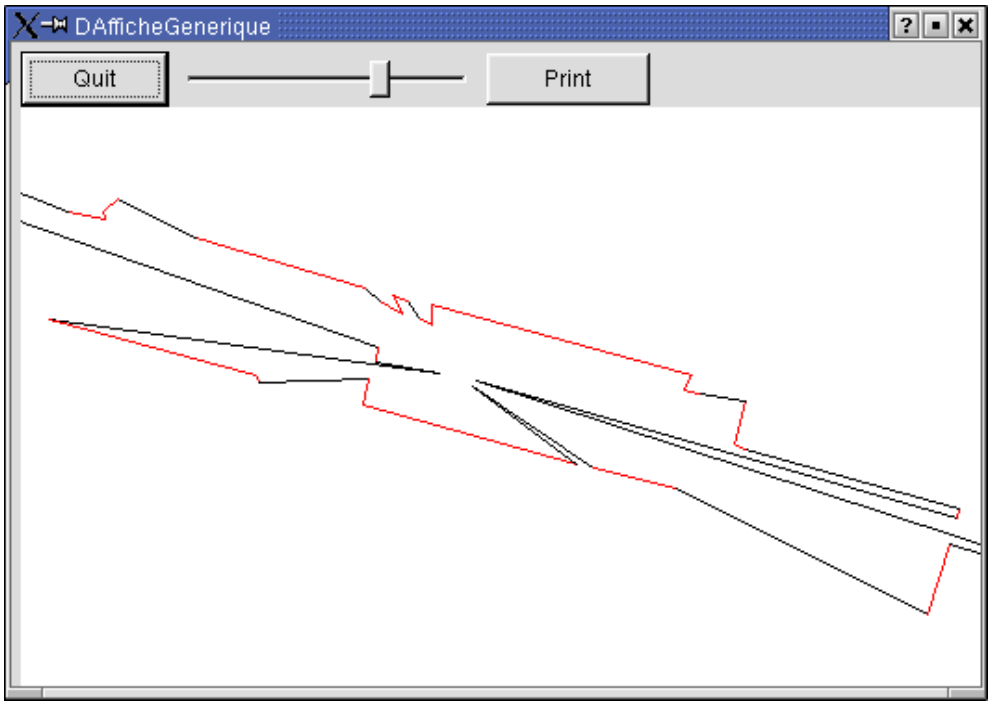


FIG. *5.13: Résultat de la polygonalisation : les segments « obstacles » apparaissent en rouge tandis que les segments « non obstacles » sont tracés en noir.*

5.3 Recalage en position de la carte locale

5.3.1 Problématique

Avant d'effectuer la mise en correspondance des primitives géométriques dans l'espace des cartes, nous réalisons un recalage en position de la carte locale (et donc de la position courante du robot) par rapport à la carte globale. Nous avons vu que ce recalage préalable est nécessaire pour assurer la validité des tests de comparaison de segments, qui vérifient en particulier que les segments local et global candidats à l'appariement se trouvent bien l'un « en face » de l'autre (au sens des projections orthogonales). L'odométrie fournit déjà une estimation du déplacement du robot par rapport à sa position précédente (position d'acquisition du scan précédent). Toutefois, cette estimation est souvent peu précise, notamment en raison des glissements qui se produisent lors des rotations du robot (certains virages sont effectués en pur glissement par blocage d'une roue) et qui biaisent les odomètres. Nous cherchons donc à compenser cette imprécision par le biais d'un « odomètre visuel » basé sur les données laser. L'exploitation de l'information de recalage dans la mise en correspondance de primitives nécessite en outre une estimation de l'erreur de localisation, puisque le test d'association de segments exploite les covariances, via un test de Mahalanobis.

Il existe différents types d'algorithmes susceptibles de répondre à nos besoins (cf. par exemple le panorama dressé dès 1996 par Borenstein et al. [12]) : la plupart correspondent en réalité à des algorithmes de localisation :

– **Recalage par corrélation de grilles d'occupation** : ce type de recalage a par exemple été proposé par Schiele et Crowley (pour la localisation d'un robot) [168] ou par Gutmann et Konolige (lors du bouclage de cycle) [81]. L'incertitude sur la position estimée peut être directement déduite des valeurs de corrélation autour de cette position, en supposant que la distribution des

valeurs de corrélation est gaussienne [168]. Les expérimentations menées utilisent toutefois des grilles probabilistes et non histogrammiques comme dans notre cas : la transposition ne paraît donc pas immédiate (la justification de la corrélation est probabiliste chez Gutmann et Konolige), d'autant que les frontières d'obstacles n'apparaissent pas de la même manière (elles ont une certaine épaisseur dans les grilles d'occupation, avec des variations de degré d'occupation, au contraire de notre représentation). Une variante consiste chez Schiele et Crowley à extraire dans l'une des cartes les segments correspondant aux frontières d'obstacles : on produit alors pour chaque segment un masque de corrélation qui correspond aux valeurs de la grille le long du segment. Expérimentalement, cette variante a toutefois donné de moins bons résultats que la corrélation avec la carte locale complète [168].

– **Recalage par appariement de scans bruts** : ce type de recalage a donné lieu à des algorithmes variés.
Certains fonctionnent directement sur les points de mesure bruts (on parle de méthodes « iconiques ») : il s'agit par exemple des algorithmes d'ICP (« Iterative Closest Point ») de Lu et Milios [127] qui recherchent pour chaque point du premier scan le meilleur correspondant parmi les points du second scan. Ces informations d'association de données permettent de réestimer la position du robot par moindres carrés, puis le processus se répète de manière itérative pour raffiner progressivement cette estimation. Chez Einsele, l'appariement entre deux scans se fait par programmation dynamique sur les séquences de segments élémentaires formés par les points successifs des scans [60]. D'autres algorithmes exploitent l'approximation polygonale de l'un des scans : chez Cox par exemple, les points de mesure du second scan sont appariés avec les segments les plus proches dans le premier scan [34]. Ensuite, comme dans le cas de l'ICP, l'estimation de position peut être fournie par moindres carrés à partir de ces informations d'association de données. Pour calculer l'incertitude correspondante, on peut utiliser comme Borges [14] la propagation des covariances à travers l'application d'un algorithme de moindres carrés [85].

D'autres chercheurs proposent une estimation d'erreurs statistique, réalisée par échantillonnage. Par exemple, Wang et Thorpe [194] appliquent le même algorithme d'appariement par ICP sur plusieurs hypothèses (échantillons) de position initiale du robot. Ils calculent ensuite la distribution d'erreur sur les résultats obtenus à partir de ces positions initiales. Toutefois, une telle technique ne prend pas en compte les erreurs de mesure ni la qualité d'appariement. Pour y remédier, ces erreurs sont intégrées dans l'estimation par le biais de grilles d'occupation : les mesures brutes des scans sont transformées en grilles d'occupation probabilistes locales correspondant chacune à un seul segment. Ensuite, pour chaque hypothèse de position initiale, on peut calculer la corrélation entre les grilles d'occupation locales issues des deux scans : les auteurs en déduisent une pondération des échantillons de l'algorithme d'appariement, qui modifie la distribution d'erreur.

Enfin, certains algorithmes recherchent les invariants des scans : ils procèdent par exemple par corrélation d'histogrammes [196] [166]. Il s'agit dans un premier temps de calculer l'histogramme d'angles de chaque scan, invariant par rotation et par translation (moyennant un décalage de phase) : ces angles correspondent à l'orientation absolue des segments élémentaires reliant deux points de mesure successifs (cf. Fig. 5.14). La recherche du pic de corrélation maximale entre les histogrammes des deux scans permet leur recalage en rotation. On extrait également de l'un des histogrammes les deux directions principales du scan correspondant : ces directions permettent de définir le repère dans lequel on calcule l'histogramme en x (projection sur l'axe des abscisses) et l'histogramme en y (projection sur l'axe des ordonnées) de chaque scan recalé : le maximum de corrélation fournit alors le recalage en translation des scans. Pour estimer l'incertitude de ces

recalages, on peut calculer comme précédemment la variance du pic de corrélation [82].

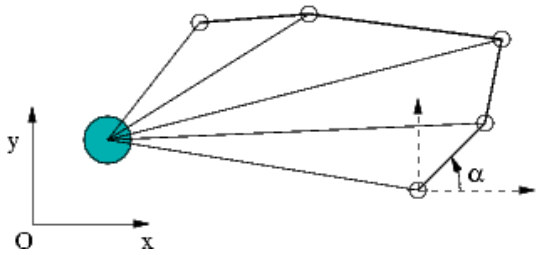


FIG. *5.14: Calcul des angles de l'histogramme pour le recalage en rotation des scans. Le télémètre laser se situe au niveau du disque bleu et les points de mesures correspondent aux petits cercles.*

Gutmann et Schlegel ont réalisé une comparaison de trois de ces approches d'appariement de scans [82] : ICP [127], association de points de mesure et de segments [34], et corrélation d'histogrammes [196]. Il en ressort que les deux dernières méthodes fournissent les meilleurs résultats en environnements polygonaux tandis que les appariements points/points fonctionnent mieux si l'hypothèse d'environnement polygonal est mise à mal. Gutmann et Schlegel combinent alors un algorithme de chaque catégorie pour réaliser l'appariement : ils utilisent l'algorithme de Cox dès que l'environnement apparaît suffisamment polygonal.

– **Mise en correspondance de primitives géométriques** : nous n'utiliserons pas cette option pour le simple recalage en position puisque nous y avons recours dans la phase suivante de l'algorithme (mise en correspondance dans l'espace des cartes).

5.3.2 Description de notre méthode

Pour estimer le déplacement du robot, nous avons opté pour l'appariement de scans, dont l'application nous a paru plus directe que la mise en correspondance de grilles d'occupation : il suffit pour cela de conserver le scan précédent. Ainsi, notre méthode se décompose de la manière suivante :

– **Prétraitement des scans** : avant d'effectuer l'association de données, nous retirons du scan les points aberrants qui se situent en-deçà de la portée minimale ou au-delà de la portée maximale du scan. Ensuite, nous appliquons sur les scans un filtre de projection comme chez Gutmann et Schlegel [82] ou chez Lu et Milios [126] : ce filtre a pour but de retirer les points de chaque scan qui n'ont (a priori) pas de correspondant dans l'autre scan du fait des occultations. On utilise pour cela les estimations de position initiale fournies par l'odométrie : ce type de filtre est donc risqué si l'odométrie est trop incertaine (dans ce cas, soit on effectue des recalages de scans à grande fréquence pour éviter les dérives de l'odométrie, soit on annule purement et simplement ce filtrage).

– **Recalage par corrélation d'histogrammes** : comme l'estimation odométrique en rotation de notre robot est peu précise (du fait des possibilités de virage par blocage de roue), nous avons opté pour une méthode de recalage qui ne nécessite pas une bonne estimation initiale de l'orientation du robot. Expérimentalement, par rapport à d'autres méthodes exploitant les invariants par rotation (telles que la méthode de programmation dynamique d'Einsele [60]), le recalage par corrélation d'histogrammes a donné les meilleurs résultats en simulation et dans différents types d'environnements réels. L'incertitude du recalage peut directement être extraite des courbes de corrélation, en

supposant que la distribution d'erreur est gaussienne autour du pic.

– **Ajout éventuel d'une étape d'ICP** : suivant la puissance de calcul disponible, on peut éventuellement ajouter une étape supplémentaire fondée sur l'algorithme d'ICP. Une telle opération présente une sécurité supplémentaire pour assurer un bon appariement. En effet, on constate expérimentalement que la corrélation d'histogrammes fonctionne souvent mieux en rotation qu'en translation (notamment lorsque l'une des deux directions principales est peu représentée dans l'histogramme d'angle, comme dans les couloirs). En outre, elle ne nécessite pas une bonne estimation de la pose initiale du robot. En contrepartie, parmi les algorithmes de mise en correspondance point à point, l'ICP est souvent plus efficace en translation qu'en rotation [127], et a plus de chances de fonctionner correctement si l'estimation de pose initiale n'est pas trop éloignée de la pose réelle. Enfin, comme nous l'évoquions ci-dessus, Gutmann et Schlegel ont montré que les méthodes de recalage par histogrammes et d'ICP sont complémentaires puisque la première fonctionne mieux en environnement polygonal, au contraire de la seconde. Si l'on ajoute cette nouvelle étape de recalage, l'estimation d'erreur de recalage est alors réalisée par propagation des incertitudes à travers l'optimisation par moindres carrés, selon la méthode proposée par Haralick [85], reprise par Borges [14].

5.3.3 Résultats expérimentaux

Un exemple sur des données simulés

La figure 5.15 montre deux scans simulés à recaler et la figure 5.16 présente le résultat du recalage, qui montre une bonne superposition des points de mesure entre les deux scans.

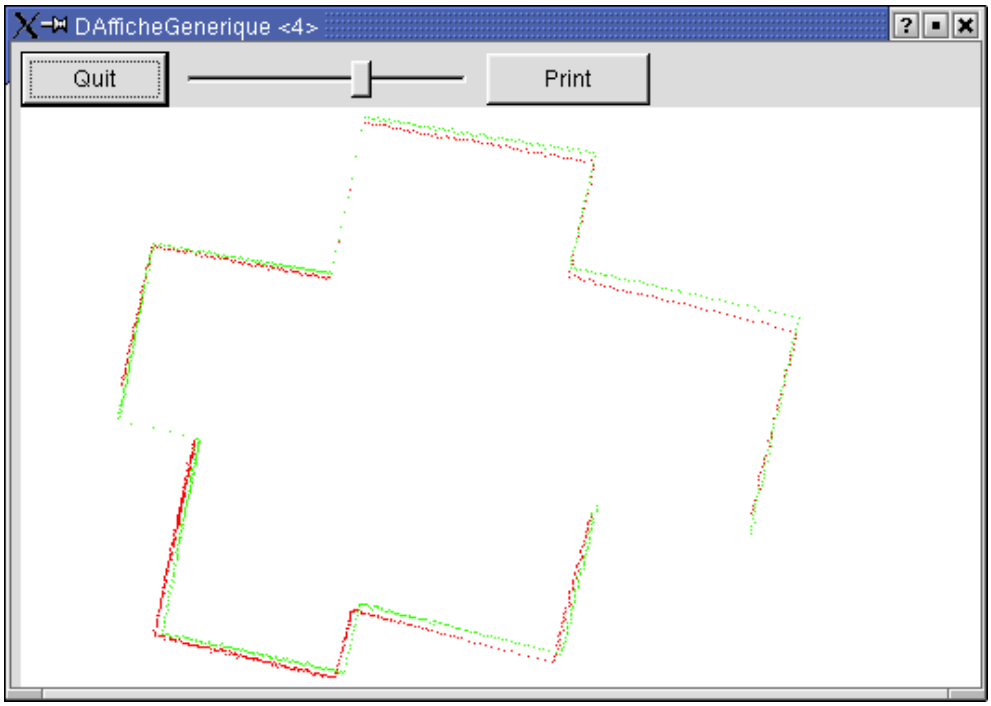


FIG. *5.15: Le premier scan apparaît en rouge, le second en vert selon une position volontairement erronée.*

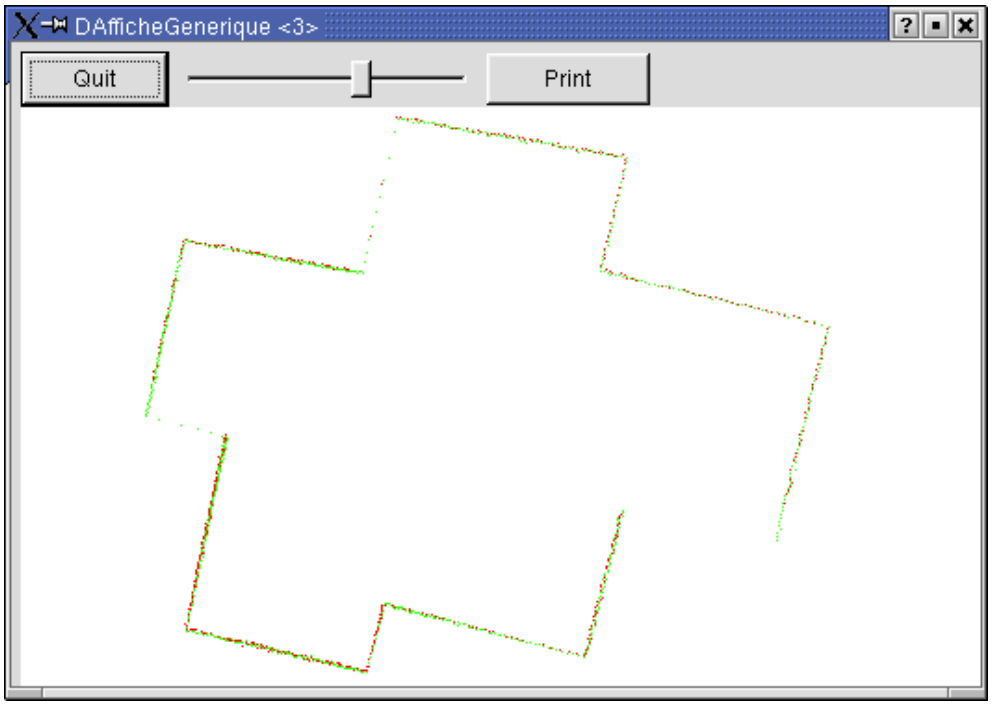


FIG. *5.16: Recalage de scans simulés. Le premier scan apparaît en rouge et le second en vert.*

Application à des données réelles

La figure 5.17 montre les deux scans réels à recaler, acquis dans notre couloir à des positions successives du robot : la position initiale du second scan est particulièrement éloignée du premier, du fait de l'odométrie très approximative de notre robot (il s'agit toutefois d'un exemple extrême de dérive en translation : cette dérive peut être limitée si l'on effectue un recalibrage à chaque utilisation du robot, de manière à corriger le dégonflage des pneus par exemple). La figure 5.18 présente le résultat du recalage, avec un zoom sur la partie centrale dans la figure 5.19. On constate là aussi *in fine* une bonne superposition des points de mesure entre les deux scans recalés.

Plus généralement, la carte réalisée en ligne à l'IFMA (Institut Français de Mécanique Avancée) lors des JNNR'03 (Journées Nationales de Recherche en Robotique) constitue un exemple de recalages multiples de scans selon la même technique (cf. Fig. 5.20). Dans ce cadre, le robot avait parcouru en téléopération un espace d'environ 60m $\times$ 60m au milieu de machines-outils et en présence de personnes mobiles. L'appariement des scans s'est donc révélé robuste (même si l'on constate de légers décalages, en haut à droite notamment) et ce malgré un certain nombre de difficultés : virages brusques du robot (dérapages susceptibles de perturber l'odométrie), existence de surfaces vitrées (qui risquent de gêner les mesures télémétriques), hypothèse d'environnement statique et structuré non vérifiée.

5.3.4 Quelques améliorations possibles

Expérimentalement, nous avons constaté que lors du recalage en translation, la corrélation d'histogramme avait logiquement tendance à favoriser les zones où l'échantillonnage est dense (du fait de la proximité du télémètre) au détriment des zones lointaines. Par exemple, dans un long couloir, il arrive que les quelques points de mesure aux extrémités, sur des murs orthogonaux à l'axe du couloir, ne soient quasiment pas pris en compte : le recalage dans l'axe du couloir s'avère donc problématique. Ainsi, l'ap-

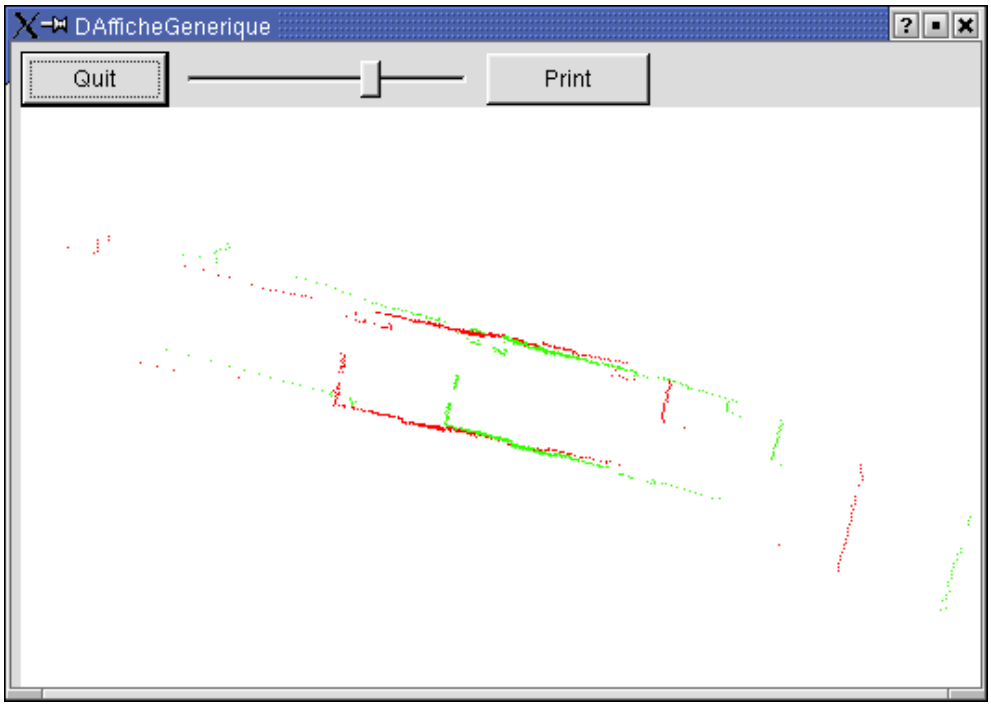


FIG. *5.17: Le premier scan apparaît en rouge, le second en vert selon une position estimée par odométrie (très dégradée).*

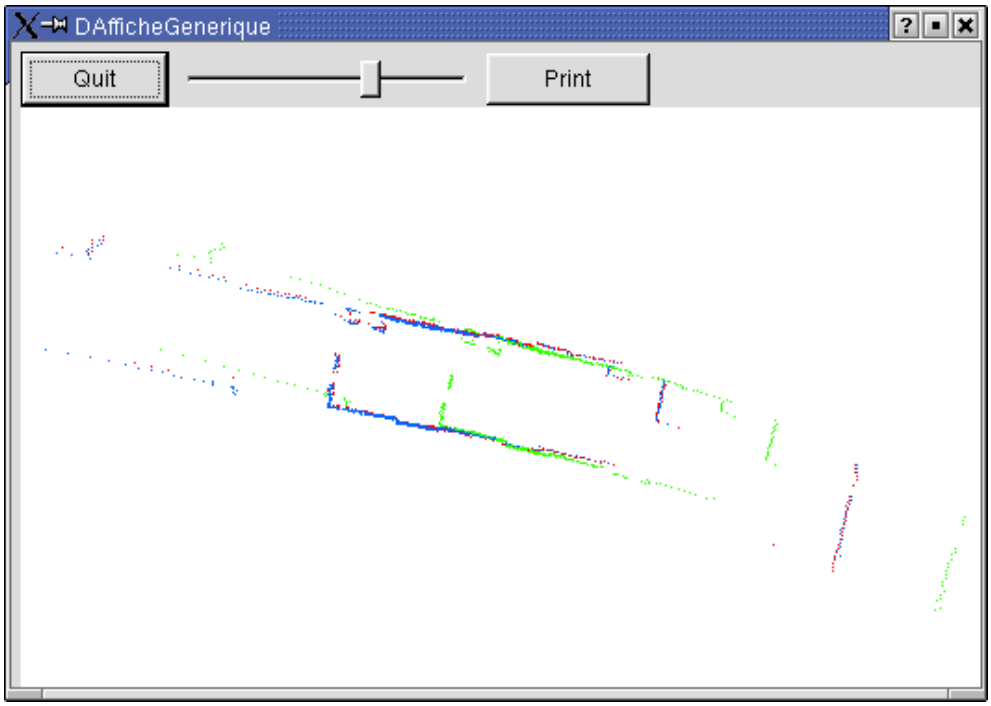


FIG. *5.18: Recalage de scans réels. Le premier scan apparaît en rouge, le second en vert selon une position estimée par odométrie (très dégradée) et le résulat du recalage du second scan par rapport au premier est tracé en bleu.*

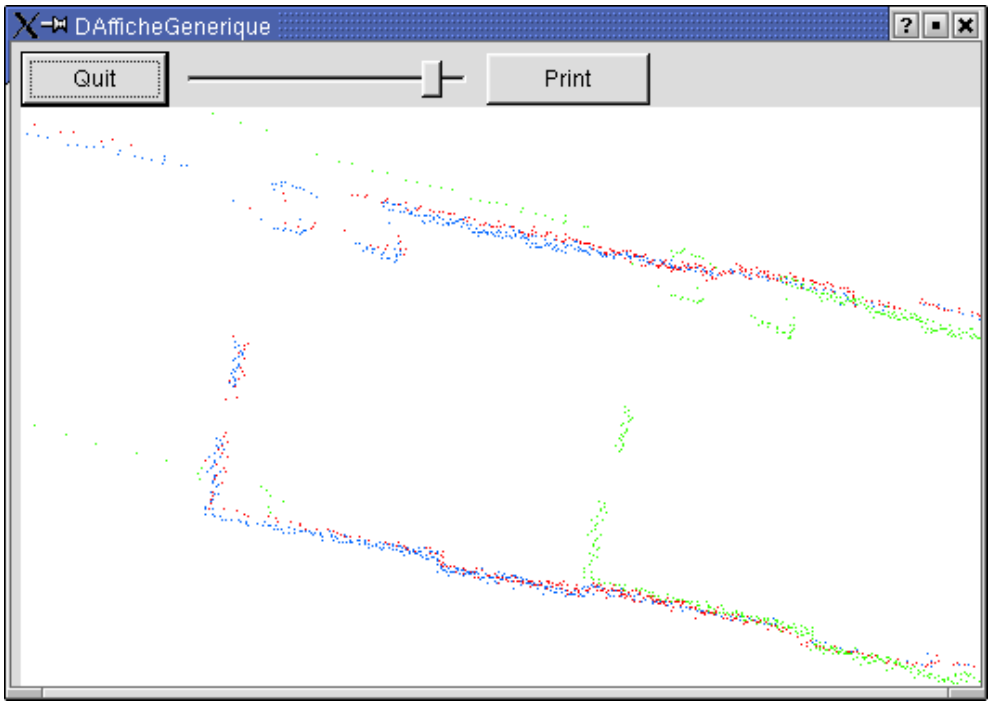


FIG. *5.19: Zoom sur le recalage des scans réels. Le premier scan apparaît en rouge, le second en vert selon une position estimée par odométrie (très dégradée) et le résulat du recalage du second scan par rapport au premier est tracé en bleu : globalement, la superposition des scans rouge et bleu est correcte.*

plication d'un ré-échantillonnage a parfois permis d'améliorer les résultats. Ce prétraitement consiste à disposer de nouveaux points de mesure de manière plus uniforme sur l'approximation polygonale des scans : le recalage en translation est alors réalisé sur ces nouveaux points de mesure.

D'autres améliorations sont bien sûr possibles : on peut par exemple envisager d'ajouter une vérification par corrélation de grilles d'occupation. La recherche des variantes les plus efficaces peut passer par des tests expérimentaux extensifs en simulation et en environnement réel, avec évaluation quantitative des résulats de recalage par rapport à une vérité terrain [82]. Dans la lignée de nos travaux sur l'évaluation d'algorithmes de traitement d'images [52], nous avons commencé à mettre en œuvre de tels tests au sein de notre département [38] sur les algorithmes proposés par Lu et Milios [127] (cf. Fig. 5.21 et 5.22).

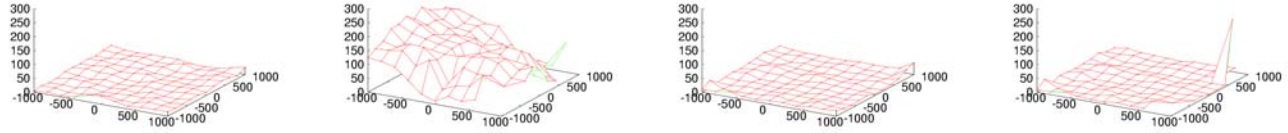
5.4 Mise en correspondance des primitives

5.4.1 Position du problème

Comme nous connaissons les liens d'adjacence entre segments, nous souhaitons réaliser une mise en correspondance de lignes polygonales « obstacles » (nous rappelons que lors de la polygonalisation, conformément au modèle défini au chapitre 4, les arêtes de la carte sont étiquetées « obstacles » ou « non obstacles », selon qu'elles correspondent aux frontières d'obstacles ou aux limites de champ de perception du capteur) plutôt qu'une simple mise en correspondance de segments individuels. Ainsi, ces liens d'adjacence interviendront explicitement dans l'algorithme d'association de données.

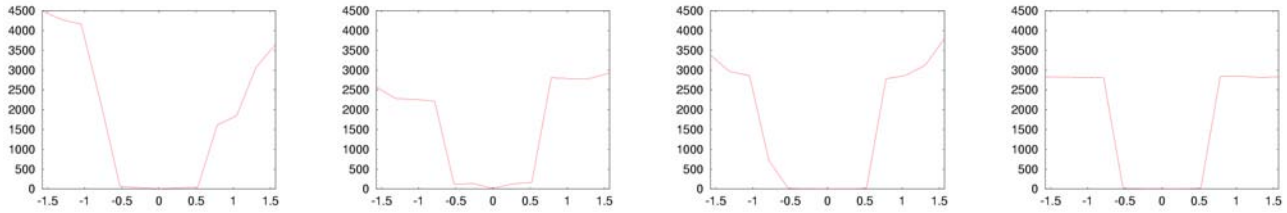
Plus précisément, nous disposons d'une part une séquence ordonnée de m segments « obstacles » (par exemple dans le sens trigonométrique autour du « centre du scan » c'est-à-dire la position du robot

FIG. *5.20: Carte d'environnement réalisée à l'IFMA lors des JNRR'03, selon la technique d'appariement de scans que nous avons développée. La trajectoire du robot apparaît en noir.*



ICP IMRP IDC IMRP-IDC

FIG. *5.21: Illustration de la comparaison de performances entre quatre algorithmes de « scan-matching » (« Iterative Closest Point » ou ICP, « Iterative Matching Range Point » ou IMRP, une hybridation des deux appelée IDC et l'application successive de l'IMRP et de l'IDC, appelée IMRP-IDC [127]). Ces comparaisons sont réalisées selon des déplacements en translation pure dans un environnement donné, à partir d'une position de référence. En abscisse et en ordonnée sont indiqués les paramètres de translation réelle et en profondeur l'erreur commise par les algorithmes (le tout en millimètres). Conformément à l'analyse théorique [127], l'IMRP apparaît bien plus sensible aux translations que les autres algorithmes.*



ICP IMRP IDC IMRP-IDC

FIG. *5.22: Comparaison de performances entre les quatre algorithmes de « scan-matching » précédents selon des déplacements en rotation pure dans un environnement donné, à partir d'une position de référence. L'angle de rotation est indiqué en abscisse et l'erreur de localisation en ordonnée (en radians). Cette fois, les meilleurs algorithmes semblent être l'IMRP et l'IMRP-IDC.*

au moment de l'acquisition du scan) qui correspond aux arêtes « obstacles » du polygone local d'espace libre : s_1,..., s_m (cf. Fig. 5.23).

Nous disposons d'autre part de l'ensemble des n segments « obstacles » appartenant à la carte globale : S_1,..., S_n. Il n'est pas évident de définir un ordre sur ces arêtes puisque celles-ci ne sont pas nécessairement organisées en séquence comme dans le cas du polygone local (cf. Fig. 5.24). En particulier, certaines arêtes peuvent être adjacentes à plusieurs autres arêtes : comme nous le verrons, de telles situations adviennent lorsqu'un appariement est manqué par exemple.

Dès lors, notre objectif est de trouver une suite de couples (s_{i_k}, S_{j_k}) avec i_k croissant (mais pas strictement, pour prendre en compte d'éventuels appariements multiples) correspondant au « meilleur » appariement global entre les segments locaux et globaux. Autrement dit, il s'agit de trouver le meilleur « chemin d'appariement » entre la carte locale et la carte globale en suivant l'ordre du polygone local. La notion de « meilleur appariement » ou de « meilleur chemin » n'est pas immédiate. Comme nous l'avons vu en introduction, nous souhaitons introduire une mise en correspondance des droites porteuses des segments via un test de Mahalanobis, une vérification du sens d'observation ainsi qu'un test de projection orthogonale pour vérifier la correspondance des segments (en tant que portions de droite porteuse) ainsi qu'une notion de longueur d'appariement.

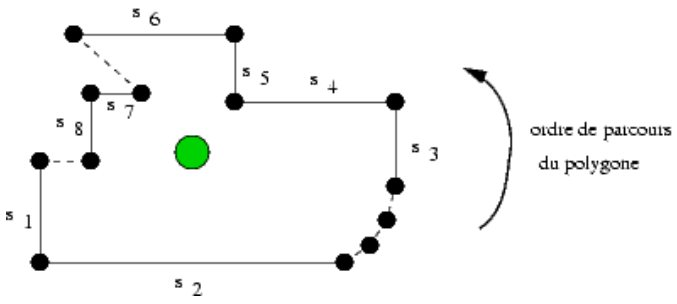


FIG. *5.23: Séquence ordonnée de $m = 8$ segments « obstacles » s_1,..., s_m d'un polygone d'espace libre. Le centre du scan (position du robot au moment de l'acquisition) apparaît en vert et les segments « non obstacles » en pointillés.*

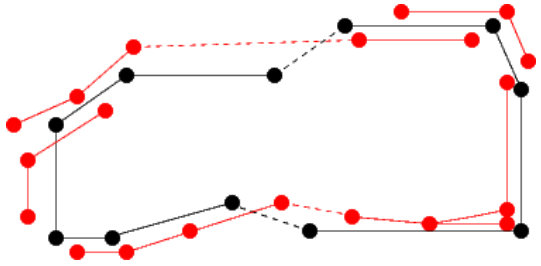


FIG. *5.24: Recherche d'appariement entre le polygone local (en noir) et la carte globale (en rouge). Les segments « obstacles » apparaissent en traits pleins et les segments non obstacles en pointillés.*

Plus formellement, nous pouvons construire un **graphe orienté** dont les sommets correspondent aux couples de segments (local et global) candidats à l'appariement. Les arcs matérialisent des transitions possibles d'un couple à l'autre en suivant l'ordre des segments locaux autour du polygone d'espace libre. **La recherche de « meilleur chemin » pourra ainsi se faire dans ce graphe d'appariement.**

*** Définition des sommets du graphe d'appariement :**

Pour définir les sommets du graphe, nous devons définir l'ensemble des couples (s_i, S_j) de segments candidats à l'appariement où s_i est un segment du polygone local (un « segment local ») et S_j un segment de la carte globale (« segment global »). Bien entendu, ces couples concernent uniquement les segments « obstacles » des deux cartes puisque les autres ne sont pas directement réobservables (sauf si l'on est sûr que le robot est revenu à une position d'observation antérieure, avec les mêmes lignes de visée et les mêmes limites de portée du télémètre : en pratique, nous n'avons pas prévu de vérifier précisément la correspondance d'une pose courante avec une pose antérieure).

Dans un premier temps, nous devons vérifier si les segments ont été observés « du même côté », c'est-à-dire si l'espace libre se trouve bien « du même côté » (cf. Fig. 5.25) : pour cela, nous orientons les segments de telle manière que l'espace libre se trouve à gauche (d'après les degrés d'occupation). Nous obtenons ainsi un vecteur $\mathbf{V_l}$ pour le segment local et un vecteur $\mathbf{V_g}$ pour le segment global (ces vecteurs sont calculés à partir des positions moyennes des sommets, c'est-à-dire que l'on utilise la moyenne des distributions gaussiennes qui modélisent leur position). Nous vérifions alors que l'on a bien $\mathbf{V_l}.\mathbf{V_g} > \mathbf{0}$. On constate ainsi que par construction, il n'existe pas d'ambiguïté sur le sens d'observation des segments de la carte globale puisque des segments appariés et fusionnés dans cette carte ont nécessairement été observés du même côté. Ce point sera précisé au chapitre 7 avec la notion de « degré d'occupation noir ».

FIG. *5.25: Vérification que les segments candidats à l'appariement ont été observés « du même côté » : la condition est satisfaite pour la configuration (a) mais pas pour la configuration (b). L'orientation des segments est indiquée par des flèches.*

Nous faisons ensuite intervenir le test de Mahalanobis pour vérifier la correspondance des droites porteuses imprécises des deux segments. Le principe de ce test est le suivant [139] [21]. Soit $\mathbf{z}$ un vecteur aléatoire gaussien de dimension n. Soit $\hat{\mathbf{z}}$ sa moyenne et $\mathbf{C}$ sa variance. Alors la variable aléatoire scalaire suivante :

$$q = (\mathbf{z} - \hat{\mathbf{z}})^{\mathbf{T}} \mathbf{C}^{-\mathbf{1}} (\mathbf{z} - \hat{\mathbf{z}})$$

est la somme des carrés de n variables aléatoires gaussiennes indépendantes, de moyenne nulle et de variance unitaire. q correspond à la « distance de Mahalanobis » au carré, et on dit qu'elle suit une distribution du χ^2 avec n degrés de liberté. Le test de Mahalanobis, fondé sur la distance du même nom, permet de vérifier des associations de données : le vecteur $\mathbf{z}$ considéré est alors l'erreur de mesure. Le test est vérifié (l'appariement est considéré comme valide) si :

$$q \leq \chi^2_{r,\alpha}$$

où $\chi^2_{r,\alpha}$ est un seuil déduit de la distribution du χ^2 pour r correspondant à la dimension n du vecteur $\mathbf{z}$ et α à la probabilité de rejeter un bon appariement.

Dans le cas de l'appariement des droites porteuses globale $\mathbf{D_{glob}}$ et locale $\mathbf{D_{loc}}$, on applique donc ce test à la variable :

$$\mathbf{z} = \mathbf{Tr}(\mathbf{D_{loc}}, \mathbf{P_r}) - \mathbf{D_{glob}}$$

où $\mathbf{P_r} = (X_r, Y_r, \theta_r)$ correspond à la pose du robot et où $\mathbf{Tr}$ est l'opérateur de projection qui transpose les coordonnées polaires (d_l, α_l) de la droite locale dans le repère du robot vers ses coordonnées polaires dans le repère de référence de la carte globale. Plus précisément, $\mathbf{z} = (z_1, z_2)^T$ avec :

$$z_1 = d_l + X\cos(\theta_r + \alpha_l) + Y\sin(\theta_r + \alpha_l) - d_g$$

$$z_2 = \theta_r + \alpha_l - \alpha_g$$

où (d_g, α_g) sont les coordonnées polaires de la droite $\mathbf{D_{glob}}$.

Par ailleurs, soient $\mathbf{J_r}$ et $\mathbf{J_l}$ les matrices jacobiennes de $\mathbf{Tr}$ respectivement par rapport à $\mathbf{P_r}$ et par rapport à $\mathbf{D_{loc}}$. Alors l'erreur de mesure s'écrit :

$$\delta = \mathbf{Tr}(\hat{\mathbf{D}}_{\mathbf{loc}}, \hat{\mathbf{P}}_{\mathbf{r}}) - \hat{\mathbf{D}}_{\mathbf{glob}} = \mathbf{J_r}\epsilon_{\mathbf{r}} + \mathbf{J_l}\epsilon_{\mathbf{l}} - \epsilon_{\mathbf{d}}$$

où $\epsilon_{\mathbf{r}}$ représente l'erreur d'estimation sur $\mathbf{P_r}$, $\epsilon_{\mathbf{l}}$ l'erreur d'estimation sur $\mathbf{D_{loc}}$ et $\epsilon_{\mathbf{g}}$ l'erreur d'estimation sur $\mathbf{D_{glob}}$.

Alors la distance de Mahalanobis au carré s'écrit :

$$q = D^2 = \mathbf{J_r}\mathbf{C_r}\mathbf{J_r^T} + \mathbf{J_l}\mathbf{C_l}\mathbf{J_l^T} + \mathbf{C_g}$$

où $\mathbf{C_r}$ représente la matrice de covariance de $\mathbf{P_r}$, $\mathbf{C_l}$ la matrice de covariance de $\mathbf{D_{loc}}$ et $\mathbf{C_g}$ la matrice de covariance de $\mathbf{D_{glob}}$. Par ailleurs, les incertitudes sur les droites $\mathbf{D_{glob}}$ et $\mathbf{D_{loc}}$ sont déduites si besoin de celles de leurs éléments constitutifs par une classique propagation des incertitudes, en recourant éventuellement à une linéarisation [113] (cf. chapitre 6).

Nous devons par ailleurs nous assurer que ces segments se trouvent bien « l'un en face de l'autre » : pour cela, comme nous l'avons expliqué précédemment, nous utilisons des projections orthogonales. Ainsi, la projection orthogonale de S_j sur la droite porteuse de s_i doit intersecter le segment s_i. De même, la projection orthogonale de s_i sur la droite porteuse de S_j doit intersecter le segment S_J.

On remarque que certains couples possèdent le même segment local ou le même segment global : on peut donc modéliser les appariements multiples. Par ailleurs, il nous faut ajouter un dernier type de sommet, qui correspond au fait qu'un segment local peut ne pas être apparié dans le « meilleur chemin d'appariement ». Ainsi, on définit le segment global $S*$ qui modélise en fait l'absence de segment global dans le couple d'appariement : on ajoute donc au graphe les sommets $(s_i, S*)$.

*** Définition des arcs du graphe d'appariement :**

Avant d'indiquer comment construire les arcs du graphe, nous devons préciser deux notions : la première concerne la compatibilité des appariements multiples et la seconde concerne l'ordre des segments globaux par rapport à un segment local donné.

Soient S_j et S_l des segments globaux et soit s_i un segment local candidat à l'appariement avec ces deux segments globaux. On dit que S_j et S_l sont *en situation de compatibilité verticale* par rapport à s_i si les projections orthogonales de S_j et S_l sur s_i ne se superposent pas (cf. Fig. 5.26). De façon symétrique, soient s_i et s_k des segments locaux candidats à l'appariement avec le segment global S_j. On dit que s_i et s_k sont *en situation de compatibilité horizontale* par rapport à S_j si les projections orthogonales de s_i et s_j sur S_j ne se superposent pas. Ainsi, en quelque sorte, des appariements multiples sont autorisés s'ils ne portent pas sur les mêmes portions du segment commun. Nous verrons plus loin que les adjectifs « horizontal » et « vertical » se reportent à la mise en œuvre d'un algorithme de programmation dynamique.

FIG. *5.26: Vérification de la compatibilité verticale des segments S_j et S_l par rapport à s_i. (a) Les segments globaux sont bien en situation de compatibilité verticale. (b) Ces segments ne sont pas en situation de compatibilité verticale (du fait du recouvrement en rouge).*

Soient S_j et S_l deux segments globaux candidats à l'appariement avec le segment local s_i et soient S'_j et S'_l leurs projections orthogonales sur la droite porteuse de s_i. On suppose tous ces segments non nuls (dans le cas contraire, ces segments sont considérés comme « non obstacles » et ne sont pas examinés). On oriente ces trois segments de telle sorte que l'espace libre se trouve à leur gauche : on appelle alors « extrémité gauche » celle qui correspond à la première extrémité selon cette orientation et « extrémité droite » l'autre extrémité. On dit que S_j est *supérieur à S_l par rapport à s_i* si et seulement si l'extrémité

gauche de S'_j se trouve après l'extrémité droite de S'_l selon l'orientation de s_i. Comme l'indique la figure 5.27, il s'agit d'un ordre partiel sur les segments globaux (et relatif au segment local s_i) puisque deux segments globaux donnés peuvent n'être ni supérieurs ni inférieurs l'un à l'autre (s'ils sont en situation d'incompatibilité verticale avec s_i).

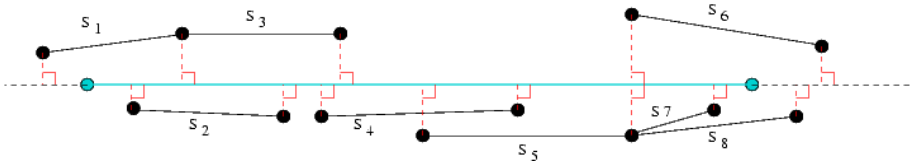


FIG. *5.27: Définition d'un ordre partiel sur les segments globaux (en noir) appariés à un même segment local (en bleu). Par exemple, S_1 est inférieur à S_2 et à S_4. En revanche, S_1 et S_3 ne sont pas comparables car ils sont en situation d'incompatibilité verticale.*

Ainsi, avec ces définitions, il existe un arc du couple (s_i, S_j) vers le couple (s_k, S_l) si et seulement si :

- $S_j = S*$ et $S_l = S*$ et $k = i + 1$: cas de deux segments locaux successifs non appariés (on parle de liaison « $S* - S*$ ») ;
- ou $S_j = S*$, $S_l \neq S*$ et $k = i + 1$: cas d'une liaison depuis un segment local non apparié (on parle de liaison « $S* - S$ ») ;
- ou $S_j \neq S*$, $S_l = S*$ et $k = i + 1$: cas d'une liaison vers un segment local non apparié (on parle de liaison « $S - S*$ ») ;
- ou $S_j \neq S*$, $S_l \neq S*$, $j \neq l$, $k = i$, S_j et S_l sont en situation de « compatibilité verticale » par rapport à s_i et S_l est « supérieur » à S_j d'après la référence s_i : cas d'une liaison multiple sur le segment local s_i (la liaison est dite « verticale ») ;
- ou $S_j \neq S*$, $j = l$ (donc nécessairement $S_l \neq S*$), $k = i + 1$, et s_i et s_k sont en situation de « compatibilité horizontale » par rapport à S_j : cas d'une liaison multiple sur le segment global S_j (la liaison est dite « horizontale ») ;
- ou $S_j \neq S*$, $S_l \neq S*$, $j \neq l$, $k = i + 1$: cas d'une transition d'un couple candidat vers un autre couple comportant le segment local suivant (la liaison est dite « oblique »).

Intuitivement, ces arcs indiquent une possible adjacence dans le chemin d'appariement entre les couples (s_i, S_j) et (s_k, S_l). Alors, en pondérant ces arcs, le problème se ramène à une recherche de plus court chemin, avec des heuristiques que nous allons détailler.

*** Définition de la pondération des arcs du graphe d'appariement :**

Pour préciser la notion de « meilleur chemin », nous devons définir des coûts sur les arcs du graphe. Intuitivement, il s'agit de définir des scores d'appariement sur les sommets du graphe et des coûts de transition sur les arcs. Les scores d'appariement que nous souhaitons utiliser doivent être liés à la distance de Mahalanobis entre les coordonnées polaires des droites porteuses des deux segments, ainsi qu'à une notion plus heuristique de « longueur d'appariement ». Quant aux coûts de transition entre sommets du

graphe, ils sont a priori liés à certaines configurations que l'on souhaite pénaliser.

Plus précisément, nous définissons le score d'appariement d'un sommet (s_i, S_j) comme la « longueur » de l'appariement pondérée par la « qualité » d'appariement déduite du test de Mahalanobis. Pour définir la longueur d'appariement, nous utilisons les projections orthogonales (cf. Fig. 5.28). Soit l la longueur de superposition entre le segment s_i et la projection orthogonale S'_j de S_j sur s_i. Soit L la longueur de superposition entre le segment S_j et la projection orthogonale s'_i de s_i sur S_j. Alors la longueur d'appariement est définie par la moyenne $(l + L)/2$ entre l et L.

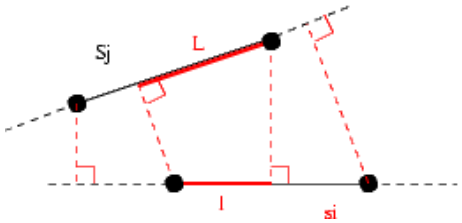


FIG. *5.28: Définition de la longueur d'appariement* $(L + l)/2$.

Concernant la qualité de l'appariement, nous avons cherché une quantité liée à la distance de Mahalanobis, mais qui soit sans dimension, de manière à pouvoir l'utiliser pour pondérer la distance d'appariement. Or le seuil du test de Mahalanobis correspond justement à un seuil de probabilité au-delà duquel l'association est peu vraisemblable. En fait, la distance de Mahalanobis suit une loi du χ^2 : il est possible de retrouver le seuil de probabilité correspondant à une certaine valeur de cette distance (utilisation de la fonction Gamma dont le calcul est indiqué dans [162]). On obtient ainsi un poids compris entre 0 et 1 (en fait, sa limite basse correspond au seuil utilisé pour valider la possibilité d'appariement, à la création du sommet du graphe) et qui augmente lorsque la distance de Mahalanobis décroît, c'est-à-dire lorsque l'appariement est meilleur. Pour accélérer les calculs, il est possible de tabuler les valeurs du seuil de probabilité en fonction de la distance de Mahalanobis.

Les coûts de transition correspondent quant à eux à une pénalisation de configurations peu souhaitables : « dépassements », « décrochements » et « écarts angulaires ».

Les « dépassements » peuvent intervenir dans les liaisons obliques, dans le cas où les segments locaux et/ou les segments globaux concernés sont adjacents. Supposons que les segments locaux s_i et s_k soient adjacents alors que les segments globaux S_j et S_l correspondants ne le sont pas forcément. On oriente les segments de gauche à droite dans l'ordre de parcours du polygone local, de s_i vers s_k (et donc de S_j vers S_l). Alors, par souci d'homogénéité avec les autres définitions, nous définissons les dépassements par rapport à la projection orthogonale P' du sommet commun aux segments locaux sur les autres segments (cf. Fig. 5.29). Ainsi, le dépassement de S_j par rapport à P' correspond à la longueur de la portion éventuelle de S_j à droite de P'. De même, le dépassement de S_l par rapport à P' correspond à la longueur de la portion éventuelle de S_l à gauche de P'.

Les dépassements des segments locaux par rapport aux segments globaux se définissent de façon symétrique. En revanche, si aucune paire de segments (locaux ou globaux) n'est adjacente, on considère qu'il existe une certaine indépendance entre ces segments : on ne sait pas quelle est la forme de l'environnement entre ces segments, donc on ne considère aucune pénalité. C'est aussi le cas des liaisons impliquant $S*$. Par ailleurs, dans le cas des liaisons horizontales ou verticales, la notion de « dépassement » pourrait correspondre aux incompatibilités (le dépassement est alors considéré entre segments locaux ou entre segments globaux puisque le sommet à projeter de l'autre paire de segments disparaît).

On peut raffiner cette définition des dépassements dans le cas où certains sommets locaux ou glo-

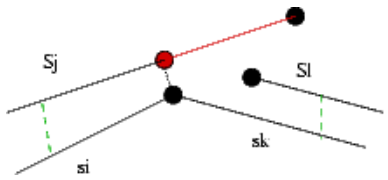


FIG. *5.29: Configuration de « dépassement » du segment S_j. Les liens d'appariement entre segments sont indiqués en pointillés verts. La longueur de dépassement est indiquée en rouge.*

baux sont appariables (suivant un test de Mahalanobis). Par exemple, dans le cas où les deux paires de segments (locaux et globaux) sont adjacentes, si les deux sommets communs (local et global) sont appariables, alors on ne considère aucun coût de dépassement. Enfin, on note que l'on pourrait interdire ces dépassements. Nous préférons les accepter en les pénalisant, afin de gérer les erreurs de recalage entre cartes locale et globale (ces erreurs peuvent générer des dépassements qui n'auraient pas lieu avec un recalage parfait).

Les « décrochements » peuvent intervenir dans les liaisons horizontales, verticales ou les liaisons obliques dans lesquelles un seul couple est adjacent (cf. Fig. 5.30). Par exemple, en reprenant les notations de définition du dépassement dans le cas d'une liaison oblique, il s'agit de l'écart entre la projection orthogonale sur la bissectrice de (s_i, s_k) de l'extrémité droite de S_j et de l'extrémité gauche de S_l. On pourrait également envisager une définition à partir des directions orthogonales au lieu de la bissectrice, comme dans le cas « dégénéré » des liaisons horizontales et verticales (où les segments adjacents sont alignés et où le sommet commun disparaît). Toutefois, on se rend compte que cet écart correspond en quelque sorte à un écart sur ρ dans les coordonnées polaires ρ, θ des droites porteuses. Plus précisément, en notant respectivement (ρ_i, θ_i), (ρ_j, θ_j), (ρ_k, θ_k) et (ρ_l, θ_l) les coordonnées polaires des droites porteuses de s_i, S_j, s_k et S_l, on peut définir le décrochement comme $\|(\rho_i - \rho_j) - (\rho_k - \rho_l)\|$. Dans le cas vertical, on a $s_i = s_k$. Pour les configurations restantes, il faut inverser les rôles entre segments locaux et globaux.

FIG. *5.30: Configuration de « décrochement » dans le cas d'une liaison oblique à gauche et d'une liaison horizontale à droite. Les liens d'appariement entre segments sont indiqués en pointillés verts. La longueur de décrochement est indiquée en rouge.*

Enfin, les « écarts angulaires » interviennent dans les mêmes types de liaisons (cf. Fig. 5.31). Avec les mêmes notations, on peut les définir de la manière suivante : $\|(\theta_i - \theta_j) - (\theta_k - \theta_l)\|$.

En fait, décrochements et écarts angulaires peuvent être détectés via un test de Mahalanobis. On peut donc avoir l'idée de regrouper les tests d'appariement de segments (intervenant dans le calcul du score d'appariement des sommets) et les tests de conformité des liaisons en termes de décrochement et d'écart angulaire (intervenant dans le calcul des coûts de liaison) en une seule étape. Ainsi, les longueurs de dépassement pourraient être soustraites aux longueurs d'appariement. Quant aux tests de qualité d'appariement et de conformité entre les segments locaux et globaux, ils peuvent être réalisés en une seule fois, via un test de Mahalanobis global sur les différences de coordonnées polaires. De plus, on peut considérer uniquement un coût de liaison (sur les arcs) en utilisant la demi-longueur d'appariement pondérée à gauche et la demi-longueur d'appariement pondérée à droite.

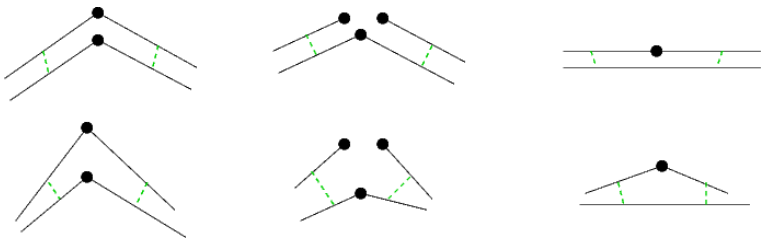


FIG. *5.31: Ecarts angulaires entre couples de segments. En haut, il sont quasiment nuls, au contraire des configuration de la ligne du bas. Les liens d'appariement entre segments sont indiqués en pointillés verts.*

Plus formellement, nous allons détailler ces calculs suivant le type de liaison et selon que les segments locaux ou globaux sont adjacents ou non :

- liaison $S* - S*$: le score de transition est nul ;
- liaison $S* - (s_k, S_l)$: le score de transition correspond à la demi-longueur d'appariement à droite, pondérée par la probabilité associée à la distance de Mahalanobis sur le vecteur d'observation $(\rho_k - \rho_l, \theta_k - \theta_l)$;
- liaison $(s_i, S_j) - S*$: le score de transition correspond à la demi-longueur d'appariement à gauche, pondérée par la probabilité associée à la distance de Mahalanobis sur le vecteur d'observation $(\rho_i - \rho_j, \theta_i - \theta_j)$;
- liaison horizontale $(s_i, S_j) - (s_k, S_j)$: le score de transition correspond à la somme de la demi-longueur d'appariement à gauche et de la demi-longueur d'appariement à droite, pondérée par la probabilité associée à la distance de Mahalanobis sur le vecteur d'observation $(\rho_i - \rho_j, \theta_i - \theta_j, \rho_k - \rho_j, \theta_k - \theta_j)$;
- liaison verticale $(s_i, S_j) - (s_i, S_l)$: le score de transition correspond à la somme de la demi-longueur d'appariement à gauche et de la demi-longueur d'appariement à droite, pondérée par la probabilité associée à la distance de Mahalanobis sur le vecteur d'observation $(\rho_i - \rho_j, \theta_i - \theta_j, \rho_i - \rho_l, \theta_i - \theta_l)$;
- liaison oblique $(s_i, S_j) - (s_k, S_j)$ avec au moins l'un des couples (local ou global) adjacents : le score de transition correspond à la somme de la demi-longueur d'appariement à gauche (à laquelle on a éventuellement soustrait la longueur de dépassement) et de la demi-longueur d'appariement à droite (à laquelle on a éventuellement soustrait la longueur de dépassement), pondérée par la probabilité associée à la distance de Mahalanobis sur le vecteur d'observation $(\rho_i - \rho_j, \theta_i - \theta_j, \rho_k - \rho_l, \theta_k - \theta_l)$;
- liaison oblique $(s_i, S_j) - (s_k, S_j)$ avec aucun couple adjacent : on considère qu'il y a une certaine indépendance entre les segments locaux et entre les segments globaux (on ne sait pas ce qu'il y a entre). Ainsi, le score de transition correspond à la somme de la demi-longueur d'appariement à gauche, pondérée par la probabilité associée à la distance de Mahalanobis sur le vecteur d'observation $(\rho_i - \rho_j, \theta_i - \theta_j)$ et de la demi-longueur d'appariement à droite, pondérée par la probabilité associée à la distance de Mahalanobis sur le vecteur d'observation $(\rho_k - \rho_l, \theta_k - \theta_l)$.

Le problème revient alors à chercher le meilleur chemin d'appariement dans le graphe

au sens où il faut maximiser une somme de scores de transition associés aux arcs. Comme nous allons le préciser, cette recherche de plus court chemin présente cependant des particularités par rapport aux problèmes classiques :

- Certaines incompatibilités horizontales peuvent se produire « à distance » : en effet, dans certaines configurations, il se peut que les droites porteuses de deux segments locaux s_i et s_k soient compatibles avec celle du segment global S_j alors que s_i et s_k ne sont pas consécutifs dans l'ordre de parcours des segments « obstacle » du polygone. Ainsi, il faut réaliser une vérification de ces incompatibilités « à distance » éventuelles pour valider un chemin d'appariement.
- Il ne s'agit pas exactement d'une recherche de meilleur chemin mais plutôt de meilleur cycle car la séquence de segments locaux reboucle sur elle-même.

5.4.2 Recherche du plus court chemin

Il est clair que l'on ne peut pas tester un par un tous les chemins (cycles) possibles car cela conduirait à une explosion combinatoire. En effet, si l'on considère qu'il existe m segments locaux et que chaque segment local est apparié à p segments globaux, avec des transitions possibles entre tous les couples de segments, on obtient p^m chemins à tester (sans tenir compte des appariements multiples associés à un même segment local). Nous devons donc nous orienter vers une méthode de recherche plus efficace.

Nous avons donc cherché parmi les algorithmes de plus court chemin existants une méthode optimale, qui fournisse la meilleure solution possible au problème posé (sous certaines conditions que nous allons voir plus loin) : en effet, nous souhaitons éviter au maximum les erreurs d'appariement, qui risqueraient de faire diverger le filtre de Kalman. Toutefois, nous n'avons pas trouvé dans la littérature une méthode qui réponde exactement aux spécificités mentionnées ci-dessus. La plupart des algorithmes classiques concernent la recherche de plus court chemin et non de plus court cycle (excepté dans le cas particulier du voyageur de commerce où le cycle doit passer par l'ensemble des sommets). En outre, la présence d'incompatibilités horizontales « à distance », qui revient à exclure que certains sommets du graphe (non nécessairement adjacents) puissent se retrouver ensemble dans le chemin final, n'est pas prévue dans les algorithmes usuels.

Le fait que l'on dispose d'une séquence ordonnée de segments pour le polygone local et de liens d'adjacence pour la carte globale rappelle cependant des problèmes d'appariement de séquences, que l'on résout classiquement par programmation dynamique. En outre, dans le cadre du SLAM, les travaux d'Einsele sur la mise en correspondance de deux scans reposent sur l'appariement de séquences de segments élémentaires par programmation dynamique [60]. Dans le domaine de l'analyse d'image, certains travaux ont également exploité ce type d'algorithme pour la mise en correspondance de chaînes polygonales : on peut par exemple citer les travaux de Collin sur l'analyse des déformations de structures magnétiques solaires de forme filamentaire [30]. Nous avons donc cherché à adapter ce type de méthode à notre problématique.

*** Une technique basée sur la programmation dynamique :**

Un problème pouvant être traité par programmation dynamique présente les caractéristiques suivantes [185] :

- (1) Le problème peut se diviser en plusieurs étapes, chacune de ces étapes requérant une prise de décision.
- (2) Chacune de ces étapes est associée à un certain nombre d'états.
- (3) La décision prise à une étape donnée transforme un état en un autre état associé à l'étape

suivante.
- (4) Etant donné l'état courant, la décision optimale ne dépend pas des états ou des décisions précédentes (hypothèse markovienne).
- (5) Il existe une relation récursive qui identifie la décision optimale à l'étape j, à condition que l'étape $j+1$ ait déjà été résolue.
- (6) L'état final doit être soluble par lui-même.

Dans notre cas, seules certaines de ces conditions sont totalement remplies. Plus précisément :

- Condition (1) : notre problème peut effectivement se diviser en différentes étapes, correspondant chacune à un segment local, pour lequel une décision de transition vers l'étape suivante (concernant le segment local suivant) doit être prise. Ces transitions se déroulent entre couples de segments local et global appariés. En réalité, la situation se complique avec les possibilités d'appariement multiple : plusieurs décisions de transition peuvent être prises pour un même segment local, si elles sont compatibles « verticalement ». Nous verrons que la définition ci-dessus d'un ordre partiel sur les segments globaux associés à un segment local donné permet de découper plus finement les étapes associées à chaque segment local : la condition est alors vérifiée.

- Condition (2) : avec le découpage plus fin proposé ci-dessus, nous pouvons effectivement associer à chaque étape un certain nombre d'états, qui correspondent chacun à un appariement donné pour le segment local correspondant avec un segment global (ou $S*$). Ces états correspondent aux sommets du graphe dans lequel s'effectue la recherche de meilleur chemin (cycle).

- Condition (3) : toujours avec ce découpage fin, la décision de transition transforme en effet un état (un sommet du graphe) en un autre concernant l'étape suivante.

- Condition (4) : en général, les décisions de transition ne dépendent pas des états et des décisions passés, puisque les scores de transition se calculent localement. Toutefois, les rares situations d'incompatibilité horizontale « à distance » mettent à mal cette hypothèse : en théorie, il ne faut pas appliquer une décision si elle entraîne une incompatibilité avec une décision passée. Nous verrons comment traiter ce problème a posteriori, à l'issue de la programmation dynamique.

- Condition (5) : cette relation récursive existe en effet, puisque grâce aux scores de transition, nous pouvons définir la décision optimale à l'étape j, si l'étape $j+1$ a été résolue (« backtracking » ou « recherche arrière » classique). La question des incompatibilités horizontales « à distance » perturbe toutefois cette hypothèse.

- Condition (6) : a priori, en raison du rebouclage de la séquence, l'état final ne peut être résolu en lui-même. Toutefois, nous verrons que dans certains cas, ce bouclage peut être correctement résolu.

La mise en œuvre concrète de cet algorithme, ainsi que les adaptations proposées sont détaillées ci-dessous.

*** Mise en œuvre et définition plus fine des étapes de décision**

Pour mettre en œuvre l'algorithme de programmation dynamique, on définit classiquement une matrice pour laquelle les colonnes correspondent aux segments locaux dans l'ordre de parcours du polygone et les lignes correspondent aux segments globaux (dans un ordre quelconque). En pratique, comme il

n'existe qu'un nombre limité de segments globaux appariables à un segment local donné, on définit pour chaque segment local une liste de ces segments globaux appariables (incluant le segment $S*$), qui correspondent chacun à un sommet du graphe d'appariement (cf. Fig. 5.32).

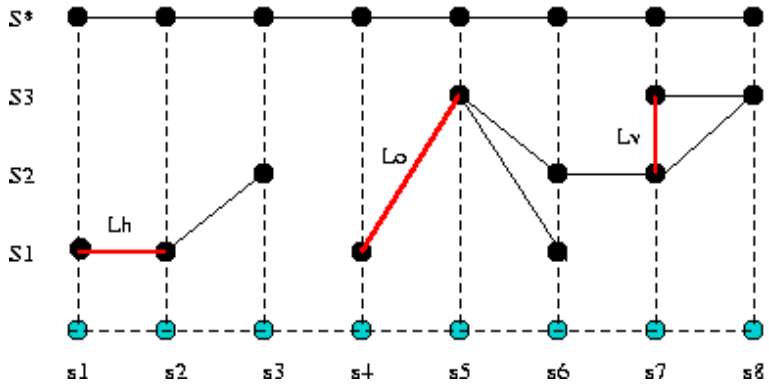


FIG. *5.32: Structure matricielle de la programmation dynamique. Les liaisons Lh, Lo et Lv (en rouge) représentent respectivement des transitions horizontale, oblique et verticale.*

On peut alors définir les liaisons autorisées entre sommets de deux colonnes successives : il s'agit des liaisons obliques du graphe et des liaisons horizontales (compatibles). Il existe également des liaisons autorisées entre sommets d'une même colonne : celles-ci correspondent aux liaisons verticales compatibles. On voit ainsi apparaître dans la représentation graphique de la matrice la signification des adjectifs « horizontal », « vertical » et « oblique » appliqués aux liaisons.

Pour mettre en œuvre la programmation dynamique, on effectue une propagation des scores cumulés d'appariement de gauche à droite : chaque sommet porte donc un tel score cumulé. A l'étape correspondant à la colonne c_i, pour chaque sommet de cette colonne, on recherche le « meilleur » antécédent parmi les sommets de la colonne précédente c_{i-1} (pour lesquels la transition vers le sommet courant est autorisée), au sens du score cumulé de l'antécédent ajouté au score de transition vers le sommet courant. On obtient alors le score cumulé du sommet courant et on définit un pointeur vers le sommet précédent pour la recherche arrière (« backtracking »). Cette recherche arrière consiste à retrouver le meilleur chemin en le parcourant de droite à gauche à partir du sommet de la colonne de droite portant le meilleur score cumulé.

*** Gestion des liaisons verticales :**

Toutefois, nous avons vu que l'existence des liaisons verticales perturbe les conditions d'application de la programmation dynamique. En conséquence, nous devons découper plus finement chaque étape de l'algorithme, qui correspond a priori à une colonne c_i de la matrice (associée au segment local s_i). Pour cela, nous utilisons l'ordre partiel défini plus haut sur les segments globaux $S_{j_1}, ..., S_{j_p}$ associés au segment local s_i (le segment $S*$ est exclu de l'ensemble ordonné). Les segments globaux qui ne sont pas comparables par rapport à cet ordre partiel correspondent aux segments en situation d'incompatibilité verticale. Nous pouvons alors réaliser une « propagation » des transitions en suivant l'ordre partiel.

Plus précisément, tout se passe comme si l'on avait découpé le segment local s_i en r petites portions délimitées par les projections orthogonales des extrémités des segments S_{j_k} (cf. Fig. 5.33). Ensuite, dans la structure matricielle de la programmation dynamique, on crée à droite de la colonne c_i de s_i (et à gauche de celle de s_{i+1} si s_{i+1} existe) r copies $c_{i_l}, l \in 1, .., r$ de c_i. Chacune de ces colonnes correspond à une portion de s_i, dans l'ordre de parcours du polygone local. On définit comme nuls les scores de

transition (horizontale) entre les sommets correspondants des colonnes c_{i_l} et $c_{i_{l+1}}$. Ensuite, pour chacune des colonnes c_{i_l}, on regarde quels sont les segments globaux dont l'extrémité gauche se projette à l'extrémité droite de la portion de s_i correspondante. Pour chacun de ces sommets globaux, on crée un lien de transition depuis les segments globaux inférieurs (au sens de l'ordre partiel défini plus haut) de la colonne précédente vers le segment global courant. Le score de transition correspondant est calculé comme une transition verticale.

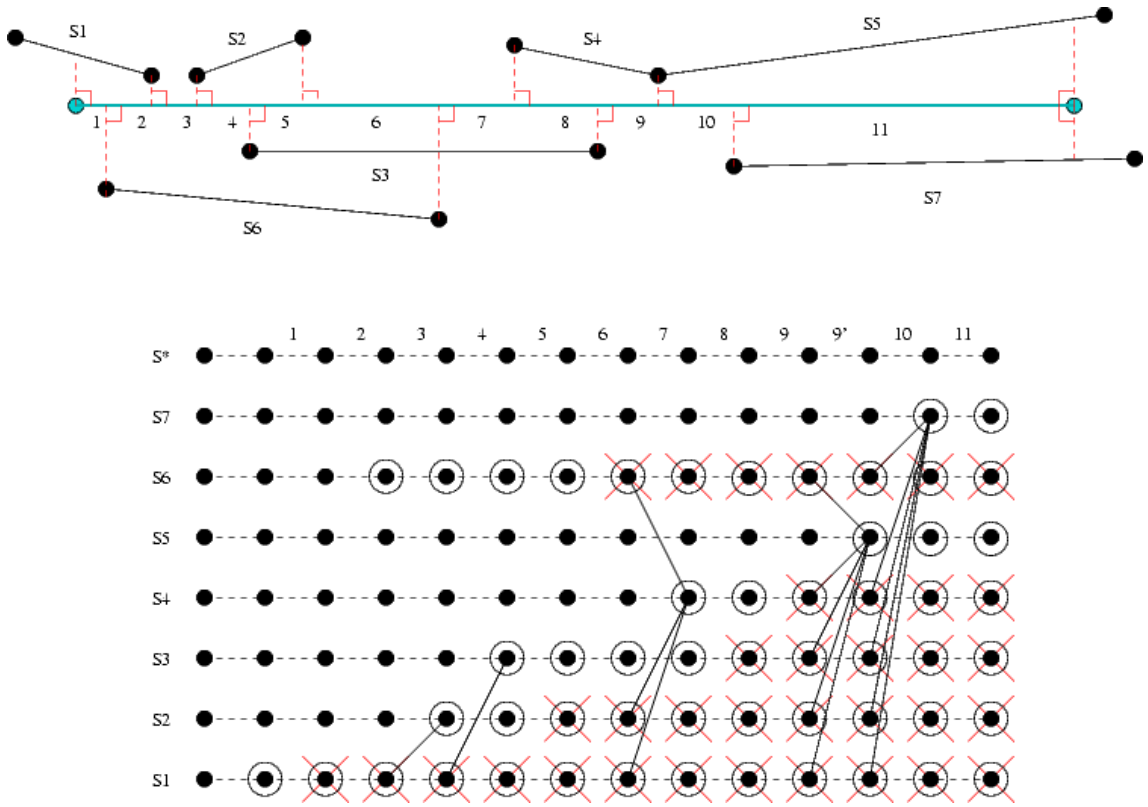


FIG. *5.33: Dépliement de la structure matricielle de la programmation dynamique pour les transitions verticales. La figure du haut indique la configuration des segments globaux par rapport au segment local apparié. En bas apparaît la structure de la matrice dépliée correspondante.*

En procédant ainsi, nous avons en quelque sorte « déplié » la structure matricielle qui supporte l'algorithme de programmation dynamique. Cela nous permet également de vérifier que l'on évite les petits cycles qui étaient à craindre du fait des liaisons verticales au sein d'une même colonne : de tels cycles n'auraient pas été compatibles avec la technique de programmation dynamique. Nous avons également défini des étapes plus « fines » dans le processus de programmation dynamique, chaque étape (colonne de la matrice) correspondant à une portion de segment local selon la description ci-dessus.

En pratique, nous utilisons un mécanisme d'activation et de désactivation des segments globaux (des couples du graphe d'appariement en fait), analogue à celui qui est mis en œuvre dans l'algorithme de raffinement par balayage des cartes combinatoires [24]. Pour cela, nous déplaçons virtuellement une « limite d'activation » le long du segment local dans le sens de parcours du polygone. Au début, l'ensemble des segments désactivés est nul. Quand cette limite d'activation dépasse l'extrémité « gauche » d'un segment global S_{j_k}, celui-ci est activé et nous pouvons procéder aux transitions de l'ensemble des segments désactivés vers ce segment S_{j_k}. Quand cette limite d'activation dépasse une extrémité « droite » d'un

segment global, celui-ci est ajouté à l'ensemble des segments désactivés.

*** Gestion des incompatibilités horizontales « à distance » :**

Si en principe, les cas d'incompatibilité horizontale « à distance » sont rares, ils peuvent néanmoins se produire, et d'autant plus si la carte est très incertaine (du fait des capteurs ou de l'odométrie par exemple). Il faut donc être en mesure de les gérer : à défaut, la mise à jour de la carte globale à partir des observations locales risque de présenter des incohérences. Nous proposons de traiter ces incompatibilités a posteriori, une fois un chemin d'appariement trouvé. Deux méthodes sont envisageables : une méthode optimale qui garantit le meilleur résultat d'appariement, mais qui peut s'avérer coûteuse, et une méthode sous-optimale qui ne conduit pas nécessairement à cette solution optimale.

Dans le cas optimal, on recherche l'une (quelconque) des incompatibilités horizontales. Pour cela, on compare les projections orthogonales des segments locaux appariés à chaque segment global S_j du chemin d'appariement, jusqu'à trouver une incompatibilité. Cette incompatibilité se présente comme un ensemble de deux couples (s_i, S_j) et (s_k, S_j) pour lesquels la projection orthogonale de s_i sur S_j « empiète » sur celle de s_k. Ainsi, le véritable chemin optimal ne peut contenir à la fois le sommet (s_i, S_j) et le sommet (s_k, S_j) : il contient soit l'un, soit l'autre, soit aucun des deux. En conséquence, si l'on recommence la recherche de plus court chemin d'abord sans le sommet (s_i, S_j) mais en conservant le sommet (s_k, S_j), puis sans le sommet (s_k, S_j) mais en conservant le sommet (s_i, S_j), on est sûr de tomber sur le meilleur chemin dépourvu de cette incompatibilité particulière. Toutefois, d'autres incompatibilités horizontales « à distance » peuvent survenir lors de ces nouvelles recherches de chemin : il faut donc recommencer cette procédure récursivement, jusqu'à ce qu'il n'y ait plus aucune incompatibilité. On peut accélérer ce processus en mémorisant les scores de transition d'un sommet à l'autre. Cependant, si le nombre d'incompatibilités est trop important, cet algorithme peut devenir très coûteux car il est probable qu'une seule incompatibilité disparaisse à chaque appel récursif. Il est donc préférable dans ce cas de passer à un algorithme sous-optimal.

Dans le cas sous-optimal, on commence par recenser l'ensemble des incompatibilités horizontales. Ainsi, pour chaque segment global du meilleur chemin trouvé, on recherche l'ensemble des segments locaux appariés. Un test portant sur les projections orthogonales de ces segments locaux sur le segment global permet de trouver les incompatibilités : il peut y avoir plusieurs ensembles d'incompatibilités pour un même segment global. Soit q le nombre total de ces ensembles, que l'on désigne par $e_j, j \in 1, ..., q$: chaque ensemble est constitué de sommets du graphe d'appariement qui comportent le même segment global S_j. Pour chaque ensemble e_j, on choisit alors le sommet qui présente le meilleur score d'appariement, au sens de la « distance d'appariement » pondérée par la « qualité de cet appariement » (notions définies ci-dessus) : les autres sommets sont retirés du graphe. On recherche alors le meilleur chemin dans ce nouveau graphe d'appariement. On recherche ensuite les incompatibilités horizontales « à distance » éventuelles dans ce nouveau chemin : s'il en existe, on recommence la procédure. Avec cet algorithme, le nombre d'itérations a des chances d'être moins élevé qu'avec l'algorithme optimal. En revanche, l'obtention du meilleur chemin n'est plus garantie puisque certains chemins ne sont plus testés. En outre, en supprimant d'emblée plusieurs sommets, on risque de limiter artificiellement le nombre global d'appariements : la qualité de la représentation finale pourrait donc s'en ressentir.

*** Gestion du rebouclage de la séquence de segments du polygone :**

Le problème du rebouclage de la séquence de segments locaux a déjà été évoqué par Einsele dans le cas du SLAM [60]. Dans sa méthode, Einsele imposait que le sommet de départ soit le même que

le sommet d'arrivée (la première colonne de la matrice de programmation dynamique est recopiée en dernière position). Toutefois, il n'est pas garanti que le chemin optimal vérifie cette condition : dans ce cas, l'algorithme préfère un autre chemin (sous-optimal) satisfaisant cette condition, au détriment du chemin optimal.

Nous avons également choisi de traiter le problème de recherche de cycle comme un problème de recherche de chemin, en fixant le segment local de départ. Soit m le nombre de segments « obstacles » $s_i, i \in 1, ..., m$ du polygone local. Si cela est possible, le segment de départ s_1 est choisi de telle sorte que les segments s_m et s_1 ne soient pas adjacents et que parmi les segments globaux appariés à s_m, aucun ne soit adjacent à l'un des segments globaux appariés à s_1. L'intérêt de cette condition est que si elle est satisfaite, notre algorithme fournit le meilleur cycle (et non uniquement le meilleur chemin) car les deux extrémités s_1 et s_m du polygone deviennent en quelque sorte indépendantes. En effet, le score de transition entre les sommets associés à s_1 et ceux associés à s_m est alors constitué des seuls scores d'appariement, et les pénalités de transition (dépassements et écarts traités par un test de Mahalanobis commun) n'interviennent pas. Ainsi, même si la recherche de chemin débutait ailleurs sur le polygone, tous les sommets associés à s_1 auraient le même antécédent parmi les sommets associés à s_m : celui qui présente le meilleur score d'appariement (individuel). Ainsi, tout se passe comme si on recommençait une nouvelle recherche de chemin en ce point : on est assuré que les scores calculés correspondent bien aux scores de cycles et non simplement de chemins.

S'il n'est pas possible de satisfaire cette condition, le segment de départ s_1 est au moins choisi de telle sorte que les segments s_m et s_1 ne soient pas adjacents, de manière à introduire autant d'indépendance que possible. En pratique, un tel segment s_1 peut généralement être trouvé car le champ de visibilité d'un télémètre laser est souvent inférieur à 360° (180° ou 270° pour les plus courants). Sur notre robot, cette propriété est garantie en raison d'une légère occultation systématique portant sur les premiers points du scan, du fait d'une pièce de métal servant à supporter un autre capteur. On peut donc toujours trouver deux segments « obstacles » du scan séparés par une chaîne polygonale « non obstacle ».

Dans tous les cas, les scores cumulés des sommets de la première colonne sont initialisés avec le score de transition correspondant à une liaison $S* - (s_i, S_j)$. De même, sur la dernière colonne, les scores cumulés sont incrémentés du score de transition correspondant à une liaison $(s_i, S_j) - S*$.

5.4.3 Complexité

La question de la complexité de l'algorithme global de mise en correspondance porte essentiellement sur la programmation dynamique. En effet, la polygonalisation a un coût borné puisque le nombre n_s de points du scan est fixé : les filtres appliqués (élimination de mesures aberrantes, distinction des segments obstacles et non obstacles) et la recherche de points anguleux ont un coût proportionnel à n_s et le nombre de découpages est borné par n_s. De même, l'appariement de scans est borné pour les mêmes raisons : les filtres (élimination de mesures aberrantes, filtres de projection) sont en $O(n_s)$, la construction d'un histogramme coûte n_s (il en faut 6 en tout), la corrélation p^2 (p étant le pas d'échantillonnage de l'histogramme) et la détection de pic est linéaire en p (avec une extraction de la variance locale en temps borné). Le coût d'une étape éventuelle d'ICP est en $O(n_s^2 \times n_i)$, n_i étant le nombre d'itérations considérées (que l'on peut borner).

Concernant la programmation dynamique, supposons que le polygone local contienne n_l segments locaux « obstacles ». Dans le pire des cas, chacun de ces segments locaux est appariable à l'ensemble des n_g segments de la carte globale (tests d'appariements en $O(n_l \times n_g)$, cette complexité pouvant être réduite

si l'on considère uniquement des candidats « locaux » à l'appariement). En pratique, ce nombre est plus réduit (à moins que les cartes locale ou globale ne présentent une très grande incertitude) car seuls les segments vérifiant les conditions énoncées ci-dessus (association valide des droites porteuses, superposition des projections orthogonales et sens d'observation identique) peuvent être appariés : supposons que chaque segment local soit apparié à n_m segments globaux en moyenne (en comptant le segment $S*$). Le coût de calcul sur une colonne de la matrice de programmation dynamique est en $O(n_m^2)$ pour les transitions horizontales et obliques depuis la colonne précédente (la vérification locale de compatibilité horizontale et le calcul des coûts de transition sont bornés) et également en $O(n_m^2)$ pour les transitions verticales (puisque pour chaque segment global, on a au maximum n_m prédécesseurs désactivés). Le calcul global des transitions s'effectue donc en $O(n_m^2 \times n_l)$ puisqu'il faut considérer n_l colonnes. Quant à la recherche arrière, son coût est négligeable par rapport à celui des transitions puisqu'il est en $O(n_m \times n_l)$ (si tous les couples font partie du chemin).

Il faut cependant multiplier cette complexité globale en $O(n_m^2 \times n_l)$ par les itérations supplémentaires de suite aux détections d'incompatibilités horizontales « éloignées » (dont la vérification se fait en $O(n_g \times n_l^2)$). Dans le cas optimal, on multiplie cette complexité par 3^i, i étant le nombre d'incompatibilités horizontales « à distance » trouvées (a priori, celles-ci sont très limitées, mais au pire, elles peuvent être de $O(n_m \times n_l)$ c'est-à-dire de la longueur du chemin). Dans le cas sous-optimal, on multiplie au mieux par 2, et sinon par un nombre plus élevé si les nouvelles configurations génèrent de nouvelles incompatibilités.

Finalement, si l'on considère que les cas d'incompatibilités horizontales « à distance » sont limités (en supposant que leur nombre est faible du fait du caractère improbable de telles configurations ou que le nombre d'itérations est très réduit dans le cas sous-optimal), on obtient donc une complexité globale en $O(n_m^2 \times n_l)$. En pratique, n_l est borné puisque le nombre de segments du polygone est limité par le nombre de points de mesure (à diviser par le seuil correspondant au nombre minimal de points constituant un segment - on arrive généralement à un nombre n_l de l'ordre de la centaine). Quant à n_m, il est généralement limité à quelques unités mais dans un cas extrême, il pourrait atteindre n_g. Dans les cas que nous avons traités jusqu'à présent, nous n'avons pas eu de problèmes particuliers lors de l'exécution, liés à la complexité (par exemple, l'appariement entre deux polygones se fait en moins d'une seconde, avec un algorithme non optimisé implanté sur un processeur Pentium IV).

5.4.4 Résultats expérimentaux

Un exemple simple en simulation

La figure 5.34 montre un exemple simple d'appariement de segments à partir des données simulées précédentes.

Exemples sur des données réelles

La figure 5.35 montre tous les segments candidats à l'appariement dans les polygones précédents (issus de scans réels) tandis que la figure 5.36 indique le meilleur chemin d'appariement trouvé par programmation dynamique.

La figure 5.37 fournit un exemple d'appariement impliquant une approximation polygonale non unique de l'environnement (cf. Fig. 5.37).

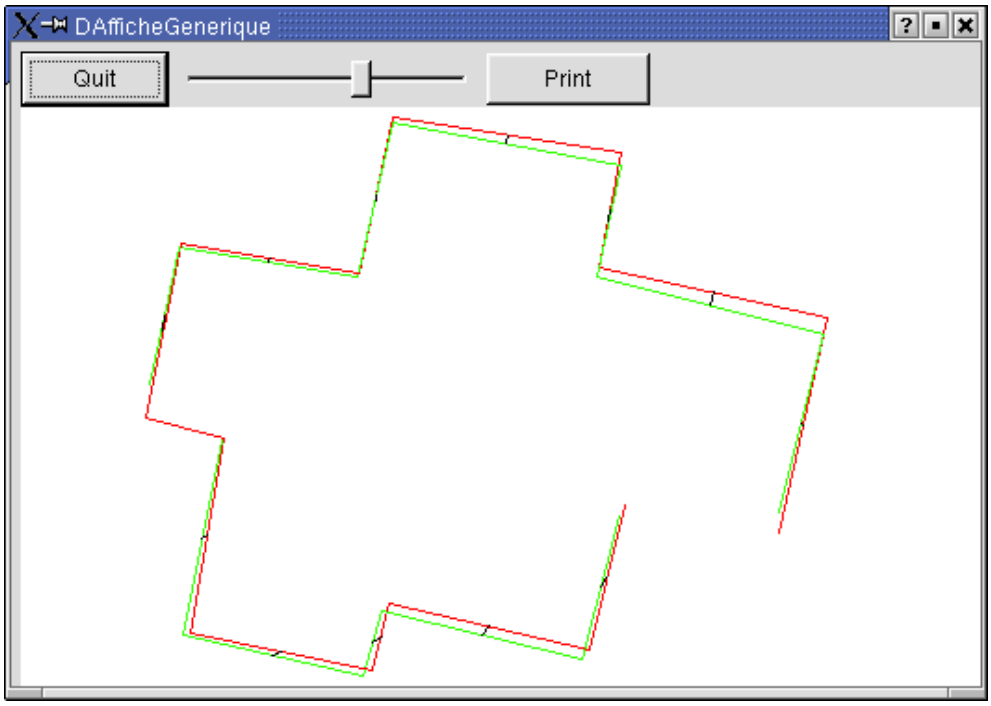


FIG. *5.34: Résultat d'appariement de polygones sur des données simulées. Le premier polygone apparît en rouge, le second en vert et les liens d'appariements sont matérialisés par des segments noirs qui relient les milieux de segments concernés. Les segments non obstacles ne sont pas dessinés.*

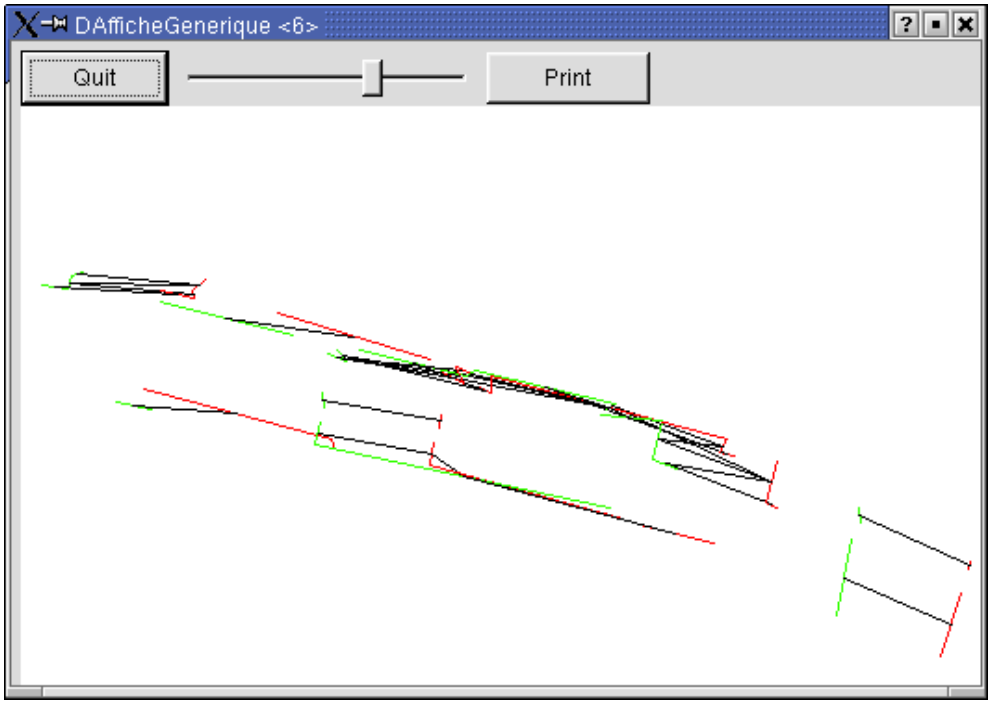


FIG. *5.35: Segments candidats à l'appariement. Le premier polygone apparaît en rouge, le second en vert et les hypothèses d'appariements sont matérialisés par des segments noirs qui relient les milieux de segments concernés.*

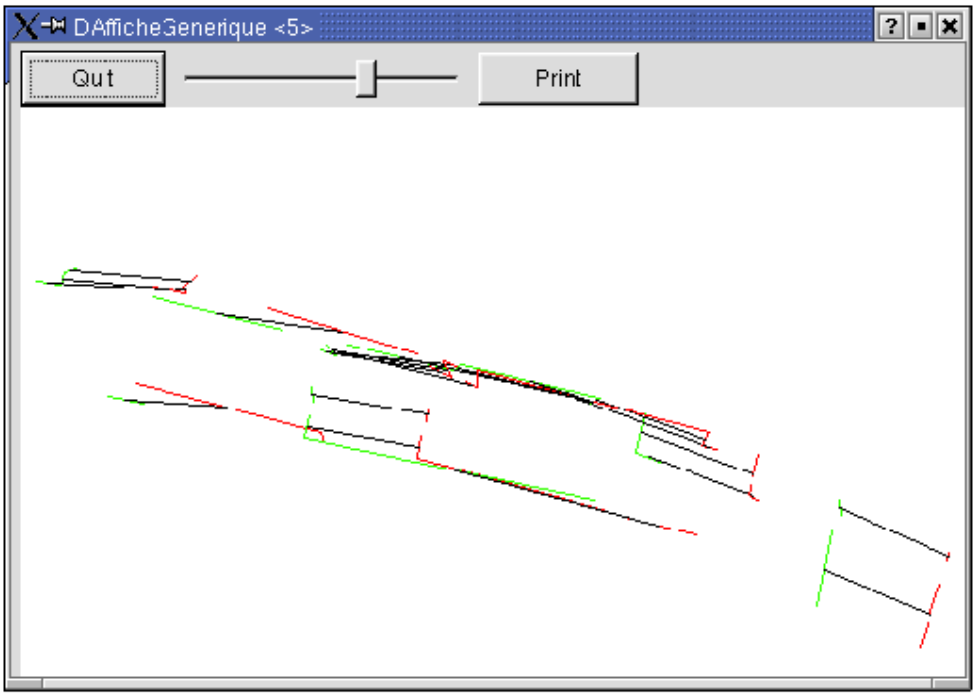


FIG. *5.36: Chemin d'appariement généré par programmation dynamique. Le premier polygone apparaît en rouge, le second en vert et les appariements sont matérialisés par des segments noirs qui relient les milieux de segments concernés.*

5.5 Conclusion

Nous avons donc défini un algorithme de mise en correspondance de cartes globale et locale qui se déroule en plusieurs étapes : polygonalisation du scan courant, correction de l'odométrie par recalage du scan local par rapport au scan précédent et mise en correspondance de segments par adaptation d'une stratégie de programmation dynamique. La stratégie de programmation cherche à maximiser les longueurs d'appariement des chaînes polygonales des deux cartes, pondérées par la qualité du test de Mahalanobis (permettant de valider l'appariement entre primitives incertaines). Cette stratégie est en principe plus satisfaisante qu'une méthode classique d'appariement glouton par plus proches voisins : en effet, elle considère systématiquement les primitives adjacentes et en outre, la maximisation de la longueur appariée est globale (la correction d'odométrie est également une étape d'appariement global). On remarque qu'en théorie, dans des configurations extrêmement défavorables (et improbables), la complexité de notre algorithme peut devenir très élevée (en particulier si le nombre d'incompatibilités horizontales « à distance » est important). Toutefois, comme nous l'avons expliqué ci-dessus, elle s'avère a priori limitée en pratique.

Le problème de la mise en correspondance de contours (et de chaînes polygonales) peut être considéré comme difficile, en particulier parce qu'il n'est pas réellement bien posé (par exemple, quels sont les bons critères d'appariements, quelles configurations sont à privilégier ?) : il a donné lieu à de nombreuses publications, que ce soit dans le domaine de la robotique mais aussi du traitement d'images notamment. De plus, nous avons ajouté des contraintes qui ne permettent pas d'utiliser des algorithmes classiques : en particulier, nous imposons une gestion des appariements multiples et des incompatibilités et la prise en compte des informations d'incertitude. Ainsi, nous pensons que notre méthode peut encore être améliorée

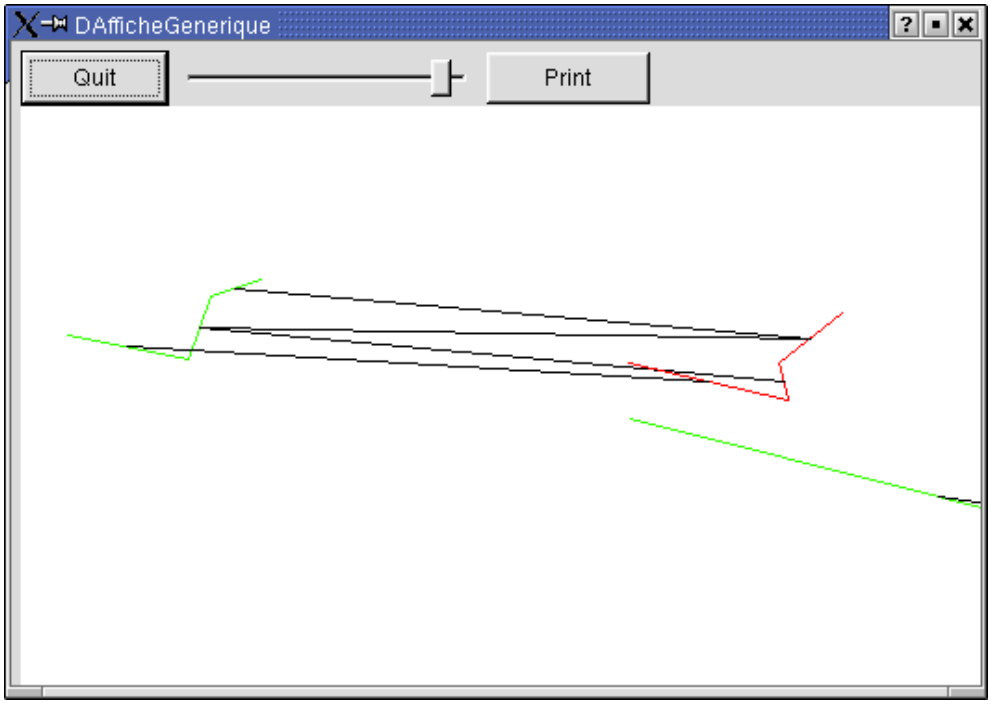


FIG. *5.37: Zoom sur des appariements multiples obtenus en faisant varier artificiellement les paramètres d'incertitude. Comme sur les figures précédentes, le premier polygone apparaît en rouge, le second en vert et les appariements sont matérialisés par des segments noirs qui relient les milieux de segments concernés. On constate que l'appariement peut ainsi gérer des approximations polygonales différentes d'un même environnement.*

selon différents aspects :

*** Améliorations du calcul de l'approximation polygonale :**

Notre technique d'approximation polygonale présente encore parfois des problèmes de « coins rognés » ou des aberrations locales liées aux erreurs de mesures. Il semble possible de l'améliorer en s'inspirant d'algorithmes plus sophistiqués, tels que ceux proposés par Castellanos et Tardos [21] ou par Veeck et Burgard [190] (algorithmes évoqués ci-dessus). De tels algorithmes permettent en principe de mieux gérer le bruit de mesure tout en ajustant correctement les points anguleux. Il faut toutefois s'assurer que l'on n'introduira pas de problèmes de chaînage (maintien de l'ordre séquentiel des segments du polygone) et que l'on pourra calculer les incertitudes des primitives.

*** Améliorations du recalage :**

Il paraît possible de rendre plus robuste la technique de recalage de carte locale en l'associant à des vérifications concernant d'autres couches de représentation. Par exemple, on peut vérifier que les grilles d'occupation correspondant d'une part au polygone local et d'autre part à la carte globale (ou dans un premier temps au polygone précédent) se superposent correctement [168] [81].

*** Améliorations de la mise en correspondance de segments :**

Il pourrait être utile d'accélérer la recherche de couples candidats à l'appariement en éliminant les couples de segments trop éloignés l'un de l'autre, comme le suggère Moutarlier (par exemple [139]).

Par ailleurs, il serait intéressant de revoir certains tests de compatibilité entre segments (par exemple les vérifications que les segments se trouvent bien « l'un en face de l'autre » et qu'ils ont été observés « du même côté ») en introduisant leur incertitude en position. En effet, les tests actuels paraissent trop dépendants des projections orthogonales sur leurs positions moyennes : les résultats dépendent notamment de la qualité du recalage de l'odométrie et de la précision courante des cartes.

En outre, il semble indispensable de passer à une représentation de type « SP-Map » afin de définir les incertitudes dans des repères locaux [21]. En effet, avec notre représentation, les valeurs d'incertitude dépendent de la position du repère de référence absolu, ce qui peut fausser les tests d'appariement (en particulier dans le cas où on utilise des coordonnées polaires). Comme nous le verrons au chapitre suivant, une représentation proche des « SP-Maps » paraît compatible avec notre modèle de carte.

*** Améliorations de la méthode générale :**

Enfin, nous pouvons envisager des améliorations de la technique générale de mise en correspondance de primitives, actuellement basée sur une stratégie de programmation dynamique. En particulier, nous pourrions considérer des méthodes d'optimisation approchées, ou tenter de poser le problème différemment, en s'inspirant par exemple des problèmes d'ordonnancement (les tâches pourraient correspondre aux segments globaux et l'échelle de temps à la séquence de segments locaux) qui peuvent être abordés via des techniques de résolution sous contraintes (contraintes liées aux incompatibilités notamment).

Nous pourrions également introduire certaines sécurités permettant de valider les associations de données proposées. Par exemple, le test de « Joint compatibility » pourrait ête appliqué sur le chemin d'appariement trouvé [145] (il faudrait toutefois définir la stratégie à adapter en cas d'échec de compatibilité globale). Nous verrons au chapitre 7 que les informations topologiques peuvent également favoriser la détection d'incohérences liées à un mauvais appariement.

Les résultats de la mise en correspondance sont destinés à raffiner la position des sommets de la carte globale. Pour cela, nous avons choisi de recourir à un filtre de Kalman. Il s'agit donc, dans un premier temps, d'extraire du chemin d'appariement un certain nombre d'observations permettant d'alimenter le filtre, puis de corriger le vecteur d'état du système en appliquant les équations de mise à jour adéquates. Le chapitre suivant est consacré à la description de ces différentes opérations.

Chapitre 6

Mise en œuvre du filtre de Kalman

« L'allégorie bouddhiste du 'filet d'Indra' parle d'un réseau de fils infinis étendu sur l'univers, dont les fils horizontaux traversent l'espace et les fils verticaux le temps. A chaque intersection des fils se trouve un individu, et chaque individu est une perle de cristal. La grande lumière de l' 'Être absolu' éclaire et pénètre chaque perle, qui reflète non seulement la lumière de toutes les autres perles du réseau, mais aussi le reflet de chacun des reflets de l'univers. »
D. Hofstadter (« Gödel, Escher, Bach : les brins d'une guirlande éternelle »).

6.1 Formalisme du filtre de Kalman

Le filtre de Kalman est un algorithme optimal récursif, qui permet d'estimer un vecteur d'état à partir de mesures indépendantes, reliées à ce vecteur par une équation linéaire. Dans le cas où cette équation n'est pas linéaire, nous devons la linéariser, ce qui conduit au filtre de Kalman étendu. Nous avons déjà vu dans l'état de l'art qu'il existe beaucoup de variantes et d'améliorations du filtre de Kalman étendu appliqué au problème du SLAM. Nous avons opté dans un premier temps pour la version du filtre proposée par Moutarlier, qui est un peu plus générique que le filtre de Kalman proprement dit puisqu'elle gère les bruits corrélés et colorés [139] [113]. Le filtre proposé par Moutarlier peut être appliqué de deux manières différentes. La méthode de base permet de mettre à jour simultanément la position du robot et celle des primitives de la carte : elle est en principe optimale mais on constate que le système résultant peut manquer de stabilité en présence de biais sur la position des entités. La seconde méthode, appelée recalage - fusion, néglige certaines corrélations : elle est donc a priori sous-optimale. Toutefois, l'analyse de son comportement montre qu'elle permet de réduire considérablement la propagation aux objets de l'environnement d'un biais sur l'odométrie du robot.

Ainsi, notre système est modélisé par :

– un vecteur d'état : $\mathbf{x} = [x_1 ... x_n]$;

– son estimée $\hat{\mathbf{x}}$ et la variance $\sigma_{\mathbf{xx}}$ de l'erreur d'estimation $\epsilon_{\mathbf{x}}$:

$$\mathbf{x} = \hat{\mathbf{x}} + \epsilon_{\mathbf{x}}$$

et

$$x_i = \hat{x_i} + \epsilon_{x_i}, (1 \leq i \leq n)$$

$$\sigma_{\mathbf{xx}} = E[\epsilon_{\mathbf{x}} \epsilon_{\mathbf{x}}^T]$$

avec

$$\sigma_{ij} = E[\epsilon_{x_i} \epsilon_{x_j}^T]$$

Dans le cas du SLAM, le vecteur d'état est constitué de la position du robot et de la position des primitives géométriques qui composent la carte.

Les observations des éléments constitutifs du vecteur d'état sont introduites dans le processus d'estimation via une équation de mesure (on appelle mesure une relation aléatoire que l'on sait vérifiée pour les vraies valeurs x_i de l'état) :

$$\mathbf{z} = \mathbf{h}(x_1, ..., x_n)$$

Cette mesure peut être prédite à partir de l'estimée courante $\hat{\mathbf{x}}$ de l'état :

$$\hat{\mathbf{z}} = \mathbf{h}(\hat{x}_1, ..., \hat{x}_n)$$

L'ancienne estimée du vecteur $\hat{\mathbf{x}}$ est alors corrigée d'une valeur proportionnelle à l'écart de mesure δ entre la prédiction $\hat{\mathbf{z}}$ et la valeur mesurée $\mathbf{z}$.

Soit $\mathbf{H_x} = (\mathbf{H_1}, ..., \mathbf{H_n})$ la jacobienne de $\mathbf{h}$ par rapport à $\mathbf{x}$:

– La nouvelle estimée s'écrit :

$$\hat{\mathbf{x}}' = \hat{\mathbf{x}} + \Gamma_{\mathbf{xz}} \Gamma_{\mathbf{zz}}^{-1} (\mathbf{z} - \hat{\mathbf{z}})$$

où

$$\Gamma_{\mathbf{xz}} = E[\epsilon_{\mathbf{x}}.(\mathbf{z} - \hat{\mathbf{z}})^{\mathbf{T}}] \text{ avec } (\mathbf{\Gamma_{xz}})_{\mathbf{ij}} = \mathbf{\Sigma_{k=1}^{n}} \sigma_{\mathbf{ik}} \mathbf{H_{kj}^{T}}$$

$$\Gamma_{\mathbf{zz}} = E[(\mathbf{z} - \hat{\mathbf{z}}).(\mathbf{z} - \hat{\mathbf{z}})^T] = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{H_i} \sigma_{\mathbf{ij}} \mathbf{H_j^T}$$

.

– La variance de cette nouvelle estimée s'écrit :

$$\sigma_{\mathbf{x'x'}} = \sigma_{\mathbf{xx}} - \Gamma_{\mathbf{xz}} \Gamma_{\mathbf{zz}}^{-1} \Gamma_{\mathbf{xz}}^T$$

.

6.2 Choix des composantes du vecteur d'état

Pour appliquer cette technique de filtrage à notre représentation, nous devons tout d'abord choisir les éléments constitutifs du vecteur d'état. Plusieurs options s'offrent à nous : nous pouvons notamment utiliser les arêtes de la carte combinatoire, ses sommets ou bien directement les brins.

Nous avons vu au chapitre 4 que dans le cas où les primitives du vecteur d'état sont des segments (ou leurs droites porteuses), nous risquons de voir apparaître des incohérences au niveau des sommets. Par exemple, si la mise à jour consiste à corriger la position des segments, des aberrations risquent de se

produire au niveau des sommets incidents à plus de deux arêtes, puisque les intersections deux à deux des nouveaux segments (ou plutôt de leurs nouvelles droites porteuses) ne coïncideront plus.

En outre, on peut d'ores et déjà entrevoir d'autres types d'incohérences. Par exemple, la correction de deux segments incidents peut conduire à de nouvelles droites porteuses parallèles : que faire du sommet qui disparaît et comment restaurer la cohérence topologique ? Dans le cas où les éléments de base du vecteur d'état sont les sommets de la carte, dautres difficultés apparaissent. En particulier, dans la configuration de la figure 6.1, il faut créer un nouveau sommet sur le segment de la carte globale, afin de modéliser la cassure du segment initial. Mais où placer ce nouveau sommet sur le segment, et quelles variances et covariances lui associer ?

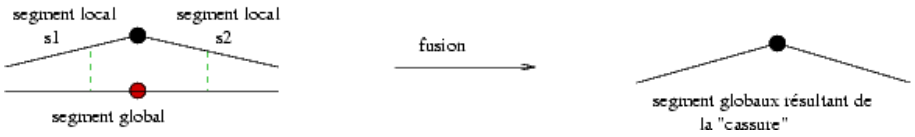


FIG. *6.1: Observation d'une cassure dans le segment global, résultant de l'appariement (en pointillés verts) du segment global avec deux segments locaux adjacents.*

Alternativement, nous pourrions envisager de prendre comme éléments de base les brins. Chaque brin comporterait par exemple trois paramètres : les coordonnées du point 2D sur lequel il est plongé et un angle, correspondant à l'orientation du segment associé. Les 0-coutures correspondraient à des corrélations fortes entre les orientations des deux brins d'une même arête, et les 1-coutures à des corrélations fortes entre les positions des plongements (points 2D) des brins d'un même sommet. Cependant, une telle représentation comporterait des redondances, au niveau des plongements des sommets et des brins. De plus, la cohérence des plongements de plusieurs brins associés à un même sommet ou à un même segment ne serait pas forcément assurée durant le processus de filtrage (par exemple en raison d'arrondis ou d'approximations linéaires dans le cas du filtre de Kalman étendu).

Finalement, nous nous orientons plutôt vers un vecteur d'état constitué des sommets de la carte, représentés avec leurs incertitudes en coordonnées cartésiennes (cf. Fig. 6.2). Certains de ces sommets correspondent à des balises, et donc à des objets véritablement observables (par exemple l'intersection de deux murs), et d'autres correspondent à des objets non observables qui sont mis à jour uniquement par le biais des corrélations (par exemple un sommet reliant deux arêtes non obstacles). La principale difficulté concerne alors les configurations du type de celle présentée ci-dessus, avec la création d'un sommet sur un segment de la carte globale : c'est ce que nous étudions dans la section suivante.

On peut remarquer que pour réaliser les mises en correspondance étudiées au chapitre précédent, nous devons calculer les incertitudes sur les positions des droites porteuses des segments (en coordonnées polaires) à partir des incertitudes sur les positions des sommets (en coordonnées cartésiennes). Pour cela, nous utilisons classiquement une linéarisation avec propagation des incertitudes [139] [113].

Soit $\mathbf{x}$ un vecteur composé de n variables aléatoires $(x_i)_{i=1...n}$ et caractérisé par son estimation $\hat{\mathbf{x}}$ et la variance $\sigma_{\mathbf{xx}}$ de l'erreur d'estimation. Soit $\mathbf{f}$ une fonction vectorielle applicable sur $\mathbf{x}$ (fonction non nécessairement linéaire) et soit $\mathbf{F_x}$ la matrice jacobienne de $\mathbf{f}$ par rapport à $\mathbf{x}$. La propagation de l'incertitude de $\mathbf{x}$ sur $\mathbf{f}(\mathbf{x})$ consiste à estimer les deux premiers moments du vecteur aléatoire $\mathbf{y} = \mathbf{f}(\mathbf{x})$. Un développement en série de Taylor fournit les approximations suivantes :

$$\hat{\mathbf{y}} \simeq \mathbf{f}(\hat{\mathbf{x}})$$

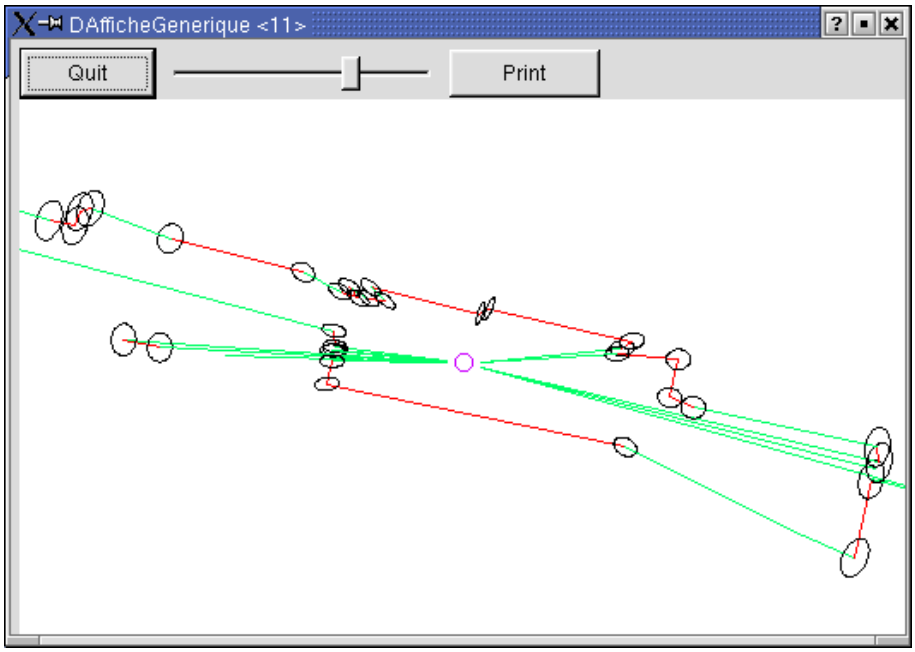


FIG. *6.2: Affichage des ellipses d'incertitudes associées aux sommets de la carte correspondant au polygone de la figure* 5.13. *Les segments « obstacles » apparaissent en rouge et les segments « non obstacles » en vert. La position du robot est indiquée par un cercle magenta. Seules les ellipses situées aux extrémités des segments « obstacles » sont indiquées sur cette figure.*

$$\sigma_{\mathbf{yy}} \simeq \mathbf{F_x} \sigma_{\mathbf{xx}} \mathbf{F_x^T}$$

Dans le cas qui nous intéresse, $\mathbf{x}$ contient les coordonnées cartésiennes (X_1, Y_1) et (X_2, Y_2) des deux extrémités du segment considéré et $\mathbf{y}$ correspond aux coordonnées polaires r, θ de la droite à estimer. Ainsi, $\theta = \arctan(\frac{X_2 - X_1}{Y_1 - Y_2})$ et $R = X_1 \cos(\theta) + Y_1 \sin(\theta)$. On note cependant que cette équation implique l'utilisation de la fonction arctan, qui n'est pas continue (et donc non dérivable) en tout point. Cette difficulté peut être levée en utilisant systématiquement les changements de repère (rotations) adéquats, de manière à placer le segment sur l'axe des ordonnées. Par ailleurs, l'estimation n'est pas réalisée si les deux extrémités du segment se superposent (le segment, de longueur nulle, n'a pas besoin d'être apparié).

6.3 Création de sommets

Pour initialiser la carte globale à partir du premier polygone d'espace libre et pour insérer de nouveaux points (jamais observés auparavant) dans la carte globale, nous avons besoin d'une opération de création de sommets.

Soit la fonction $\mathbf{Tr}$ qui projette un objet local $\mathbf{x_n}$ (défini dans le repère du robot, et dont l'erreur d'estimation est notée $\epsilon_\mathbf{n}$) dans le repère absolu au travers de la position $\mathbf{X_r}$ du robot, et soient $\mathbf{J_n}$ et $\mathbf{J_r}$ ses jacobiennes, respectivement en fonction de l'objet et du robot. On note $\mathbf{X_n}$ les coordonnées de l'objet dans le repère absolu, $\mathbf{C_{nn}}$ la variance de $\mathbf{X_n}$, $\mathbf{C_{rr}}$ celle de $\mathbf{X_r}$ et $\mathbf{C_{ni}}$ celle d'un objet $\mathbf{X_i}$ de la carte globale. On peut alors intégrer l'objet (supposé être le n-ième) dans l'état en calculant :

$$\mathbf{X_n} = \mathbf{Tr}(\mathbf{x_n}, \mathbf{X_r})$$

$$\mathbf{C_{nn}} = \mathbf{J_n} \mathbf{E}[\epsilon_\mathbf{n} \epsilon_\mathbf{n}^\mathbf{T}] \mathbf{J_n^T} + \mathbf{J_r} \mathbf{C_{rr}} \mathbf{J_r^T}$$

et pour chaque élément i de l'état :

$$\mathbf{C_{ni}} = \mathbf{J_r C_{ri}}$$

6.4 Définition des observations

Dans les méthodes SLAM à base de filtrage de Kalman, les appariements multiples ne sont généralement pas traités. Les primitives de la carte globale sont associées individuellement à des primitives de la carte locale. Avec la représentation que nous utilisons et les appariements que nous venons de définir, nous devons cependant extraire les observations à partir d'appariements de lignes polygonales. Cela nécessite donc de traiter de nouveaux types d'observations, telles que les « cassures ».

*** Gestion des « cassures » :**

Lorsque l'on observe une cassure (cf. Fig. 6.1), il faut créer un nouveau sommet S_c sur le segment A_1A_2 sur lequel se produit cette cassure. On pourrait envisager d'initialiser ce nouveau sommet de la même manière que ceux qui n'ont jamais été observés auparavant, avec les équations indiquées ci-dessus. Pourtant, un sommet de cassure ne correspond pas vraiment à un nouveau sommet, au même titre que les points jamais observés auparavant. En effet, un tel sommet est lié aux extrémités de son arête par des corrélations plus fortes qu'un nouveau sommet quelconque : intuitivement, son observation devrait entraîner toute l'arête dans sa direction (cf. Fig. 6.3).

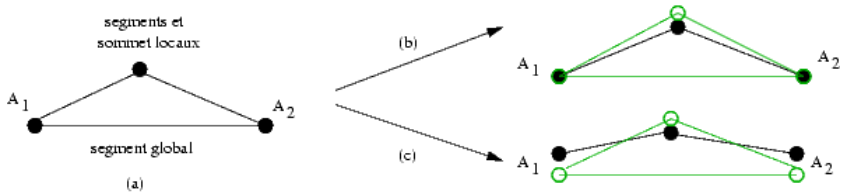


FIG. *6.3: (a) Appariement entre un segment global et deux segments locaux : il faut créer une cassure sur le segment global. (b) Création simple d'un nouveau sommet sur le segment global : les extrémités A_1 et A_2 sont peu affectées (le résultat apparaît en noir tandis que les données initiales sont indiquées en vert). (c) Effet d'entraînement d'une cassure sur A_1 et A_2 du fait de corrélations entre le sommet de cassure et les extrémités du segment global.*

En réalité, ce point de cassure a déjà été observé implicitement à chaque fois que l'on a observé l'arête globale A_1A_2 à laquelle il appartient : il est alors observé avec les sommets extrémités de cette arête globale comme situé sur l'arête locale appariée. Ainsi, à chacune de ces observations, les liens de corrélation (dans la direction du segment global A_1A_2) ont tendance à augmenter entre ce sommet implicite et les extrémités A_1 et A_2 de l'arête globale à laquelle il appartient. Ensuite, à l'instant où l'on observe une cassure, on observe en fait pour la première fois explicitement ce sommet implicite.

Nous avons donc songé à créer sur chaque nouvelle arête globale A_1A_2 un point « virtuel » (ou « implicite ») P_v dont la position présente une grande incertitude longitudinale (dans la direction de l'arête) et une incertitude orthogonale similaire celle de l'arête (cf. Fig. 6.4). La grande incertitude longitudinale indique que l'on ne sait pas à l'avance où se trouve le sommet de cassure qui se trouve sur cette arête. A chaque fois que l'arête est observée, ce point P_v est mis à jour au même titre que les extrémités A_1 et A_2 de l'arête : cette mise à jour est liée à l'observation de P_v sur l'arête locale appariée

et aux liens de corrélation avec les autres éléments de la carte. Si l'on observe une cassure sur A_1A_2, P_v est observé directement sur le point de cassure local : il s'agit de sa première observation explicite.

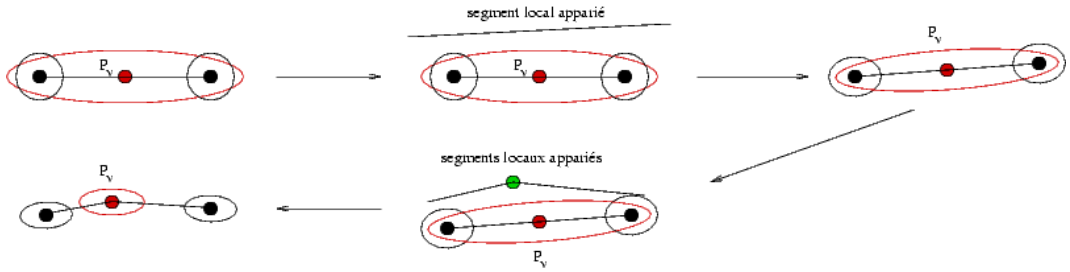


FIG. *6.4: (a) Initialisation du point virtuel P_v. (b) Observation implicite de ce point sur l'arête locale appariée. (c) Configuration résultante. (d) Observation explicite de P_v au niveau de la cassure. (e) Configuration résultante.*

Initialisation et observation du point virtuel :

L'initialisation d'un point virtuel P_v a lieu lors de l'initialisation du segment $[A_1A_2]$ sur lequel il est placé. On le définit de la manière suivante (cf. Fig. 6.5) :

- Il est placé au milieu du segment $[A_1A_2]$.
- L'ellipse de confiance à 95% de ce point (en coordonnées cartésiennes) est orientée dans la direction de $[A_1A_2]$. La longueur de son axe parallèle à $[A_1A_2]$ correspond à $2*L$ avec $L = A_1A_2$. La longueur de son axe orthogonal correspond à l'intervalle de confiance à 95% sur le rayon r des coordonnées polaires (r, θ) de la droite (A_1A_2).

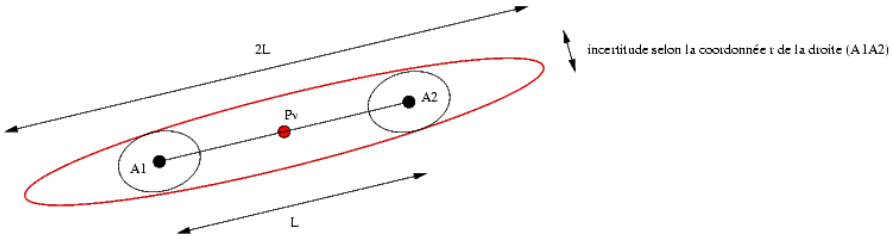


FIG. *6.5: Initialisation du point virtuel P_v et de son ellipse d'incertitude.*

On note Δ la valeur de Δ_{χ^2} pour un niveau de confiance à 95% et une variable à deux dimensions. De même, on note δ la valeur de Δ_{χ^2} pour un niveau de confiance à 95% et une variable à une dimension. La matrice de covariance associée à P_v s'écrit alors :

$$C = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

avec

$$a = \Delta[\frac{\cos^2\theta}{(2L)^2} + \frac{\sin^2\theta}{\sigma_r^2\delta}]$$

$$b = c = \Delta[\frac{-\cos\theta\sin\theta}{(2L)^2} + \frac{\sin^2\theta}{\sigma_r^2\delta}]$$

$$d = \Delta[\frac{\sin^2\theta}{(2L)^2} + \frac{\cos^2\theta}{\sigma_r^2\delta}]$$

Lorsque le segment global A_1A_2 est apparié à un segment local, P_v est observé (implicitement) comme situé sur ce segment local : il s'agit d'une observation « point sur droite ». Lors d'une cassure, l'observation de ce point virtuel correspond à une observation explicite des deux coordonnées cartésiennes de ce point : il s'agit d'une observation « point sur point ». Si l'on observe simultanément plusieurs cassures, P_v est démultiplié : tout se passe comme si les différents points de cassure étaient initialisés de la même manière et faisaient systématiquement l'objet des mêmes observations à chaque appariement de l'arête A_1A_2. Les covariances entre points virtuels d'une même arête correspondent donc à leur variance individuelle. Ils ne se différencient que lors de leur observation explicite sur un point de cassure. Il faut également dupliquer les points virtuels pour en doter chaque sous-segment issu de la cassure du segment initial. La gestion complète des points virtuels est précisée ci-dessous, lors de l'opération d'extraction des observations.

Sélection des types d'observation :

Au final, nous pouvons utiliser trois modèles d'observation qui, comme nous allons le voir, couvrent tous les cas de figure, y compris les observations de points virtuels (cf. Fig. 6.6) :
- observation d'un point global à l'emplacement d'un point local ;
- observation d'un point global sur une droite de la carte locale ;
- observation d'un point virtuel global à l'emplacement d'un point local (cassure).

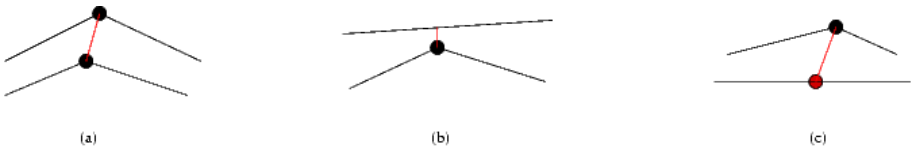


FIG. *6.6: Définition des trois types d'observation. (a) Obervation d'un point global à l'emplacement d'un point local. (b) Observation d'un point global sur une droite de la carte locale. (c) Observation d'un point virtuel global (en rouge) à l'emplacement d'un point local (cassure).*

Il nous faut toutefois préciser comment ces observations sont définies à partir du chemin d'appariement entre carte globale et polygone local.

6.5 Extraction des observations à partir du chemin d'appariement

L'extraction des observations est réalisée en suivant pas à pas le chemin d'appariement : à chaque transition, on définit les équations de mesures de certains éléments du vecteur d'état. En parallèle, on gère également l'observation (implicite ou explicite) des points virtuels et leur démultiplication éventuelle. Il faut également traiter les possibles appariements multiples des extrémités de segments. Ainsi, nous détaillons cette opération suivant le type de transition du chemin d'appariement. Nous verrons que les observations des points virtuels initiaux sont systématiquement réalisées à droite des segments globaux (avec la convention utilisée ci-dessous que le sens de parcours du polygone correspond localement à un parcours de « gauche à droite »). Sur les parties gauche ou centrale, on se contente si besoin de copie de ces points virtuels initiaux.

On note également que dans les schémas explicatifs accompagnant cette description, les sommets noirs correspondent aux sommets initiaux de la carte globale et du polygone local. Les sommets rouges désignent les sommets virtuels initiaux associés aux segments globaux, et les sommets bleus les copies de ces points virtuels. Les sommets blancs correspondent à des cassures sur les segments locaux, qui doivent

être créés en vue de la fusion des cartes locale et globale (cf. chapitre 7).

Transition $S* - S*$:

Pour ce type de transition, il n'y a rien à faire puisqu'aucun segment n'est impliqué.

Transition $(s_i, S_j) - S*$:

Les différentes configurations correspondant à cette transition sont décrites dans la figure 6.7.

On vérifie tout d'abord par un test de Mahalanobis si les deux sommets concernés (sommets droits de s_i et de S_j) sont appariables. Si c'est le cas (cf. configuration (a) de la figure 6.7), ces deux sommets sont appariés (observation « point sur point »). Sinon, on examine la projection orthogonale sur S_j du sommet droit de s_i. Si elle tombe sur le segment S_j (cf. configuration (b) de la figure 6.7), alors on observe une cassure sur S_j : on effectue deux copies du point virtuel de S_j, l'une de ces copies servant de nouveau point virtuel pour le sous-segment gauche de S_j et l'autre copie étant observée sur le sommet droit de s_i. Sinon, si la projection orthogonale sur S_j du sommet droit de s_i tombe au-delà de S_j (cf. configuration (c) de la figure 6.7), le sommet droit de S_j est observé sur le segment s_i.

De plus, dans tous les cas, on observe implicitement le point virtuel de S_j (ou sa copie sur le sous-segment gauche pour la configuration (b)) sur s_i (observation « point sur droite »).

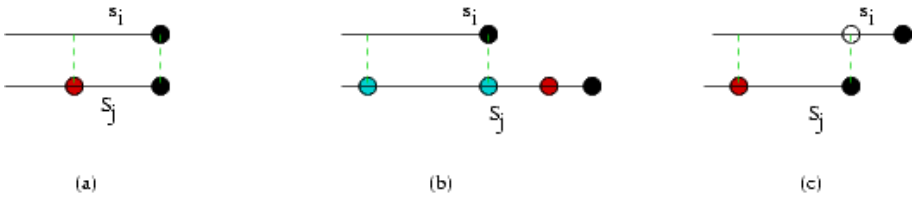


FIG. *6.7: Observations liées à une transition $(s_i, S_j) - S*$. (a) Les sommets droits sont appariés. (b) La projection orthogonale sur S_j du sommet droit de s_i se situe sur le segment S_j. (c) La projection orthogonale sur S_j du sommet droit de s_i se situe au-delà du segment S_j.*

Transition $S* - (s_k, S_l)$:

Les configurations à considérer sont symétriques par rapport à la transition $(s_i, S_j) - S*$ (cf. Fig. 6.8). La seule différence est que l'on ne traite pas l'observation implicite du point virtuel de S_l : cette observation sera extraite lors des transitions ultérieures, impliquant la partie droite de S_l.

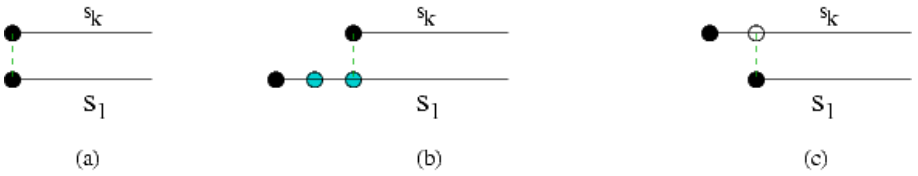


FIG. *6.8: Observations liées à une transition $S* - (s_k, S_l)$. (a) Les sommets gauches sont appariés. (b) La projection orthogonale sur S_l du sommet gauche de s_k se situe sur le segment S_l. (c) La projection orthogonale sur S_l du sommet gauche de s_k se situe avant le segment S_l.*

Transition horizontale $(s_i, S_j) - (s_k, S_j)$ **:**

Dans le cas où les segments locaux s_i et s_k sont adjacents (cf. configuration (a) de la figure 6.9), on observe une seule cassure. On effectue deux copies du point virtuel de S_j : l'une de ces copies est observée sur s_i et l'autre sur le sommet commun à s_i et s_k.

Dans le cas où les segments locaux ne sont pas adjacents (cf. configuration (b) de la figure 6.9), on observe deux cassures. On effectue quatre copies du point virtuel de S_j : la première est observée sur s_i, la seconde sur le sommet droit de s_i, la troisième est placée sur le sous-segment délimité par les deux cassures et la quatrième est observée sur le sommet gauche de s_k.

FIG. *6.9: Observations liées à une transition horizontale* $(s_i, S_j) - (s_k, S_j)$*. (a) Les segments locaux* s_i *et* s_k *sont adjacents. (b) Les segments locaux ne sont pas adjacents.*

Transition verticale $(s_i, S_j) - (s_i, S_l)$ **:**

Dans le cas où les segments globaux S_j et S_l sont adjacents (cf. configuration (a) de la figure 6.10), on observe leur sommet commun sur le segment s_i. Dans le cas contraire (cf. configuration (b) de la figure 6.10), on observe le sommet droit de S_j et le sommet gauche de S_l sur s_i.

Dans tous les cas, on observe le point virtuel de S_j sur s_i.

FIG. *6.10: Observations liées à une transition verticale* $(s_i, S_j) - (s_i, S_l)$*. (a) Les segments globaux* S_j *et* S_l *sont adjacents. (b) Les segments globaux ne sont pas adjacents.*

Transition oblique $(s_i, S_j) - (s_k, S_l)$ **:**

Dans le cas où les deux paires de segments (locaux et globaux) sont adjacentes, on vérifie tout d'abord si les sommets communs de chaque paire (locale et globale) peuvent être appariés (via un test de Mahalanobis).

Si cette condition est satisfaite (cf. configuration (a) de la figure 6.11), alors le sommet commun A_g à S_j et à S_l est observé sur le sommet commun A_l à s_i et à s_k et le point virtuel de S_j est observé sur s_i. Sinon, on traite les observations liées auxsegments s_i et S_j comme celles d'une transition $(s_i, S_j) - S*$ et les observations liées aux segments s_k et S_l comme celles d'une transition $S* - (s_k, S_l)$.

Lorsque A_l et A_g ne sont pas appariables, on remarque qu'il existe des cas où A_g est apparié deux fois, une fois, ou au contraire des cas où il n'est jamais apparié (cf. configurations (b), (c), (d) et (e)

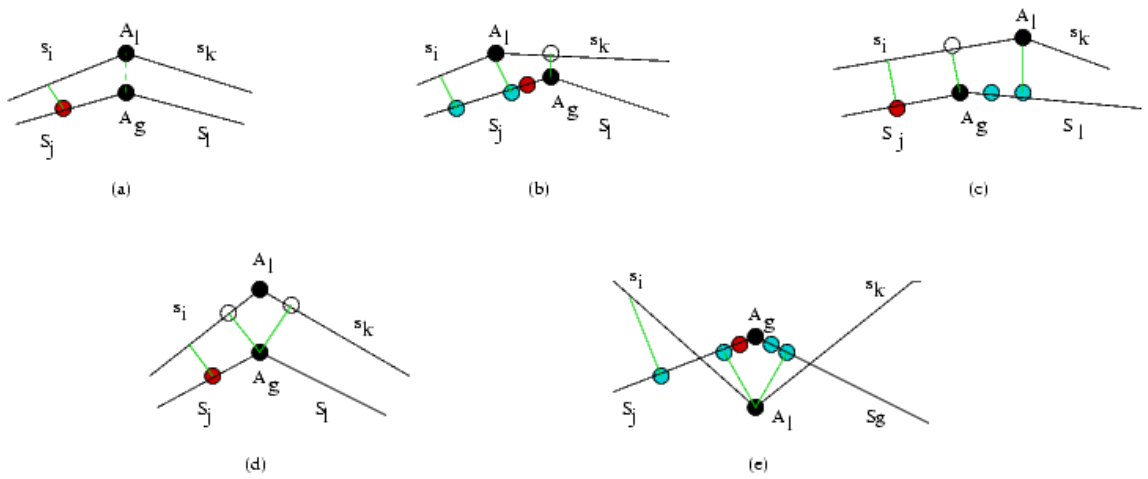


FIG. *6.11: Observations liées à une transition oblique* $(s_i, S_j) - (s_k, S_l)$*,* s_i *et* s_k *étant adjacents, de même que* S_j *et* S_l*. (a) Les sommets* A_l *et* A_g *sont appariés. (b)* A_g *est observé sur* s_k*. (c)* A_g *est observé sur* s_i*. (d)* A_g *est observé à la fois sur* s_i *et sur* s_k*. (e)* A_g *n'est observé nulle part.*

de la figure 6.11). Il en est de même pour A_l. Ces situations sont dues au fait que l'on se trouve dans une situation incohérente : en principe, il ne devrait pas y avoir de transition oblique directe entre deux paires de segments adjacents si leur sommet commun n'est pas appariable. Dans le cas des appariements uniques de A_g, on devrait passer auparavant par des transitions horizontales et verticales (par exemple, dans la configuration (b), il faudrait passer par une liaison horizontale $(s_i, S_j) - (s_k, S_j)$ puis par une liaison verticale $(s_k, S_j) - (s_k, S_l)$). A priori, les cas d'appariements inexistants ou d'appariement doubles de A_g sont aussi liés à des erreurs d'appariement entre les paires de segments. Nous avons toutefois choisi de les conserver du fait des difficultés à réaliser un appariement correct sur des données incertaines. Ces incohérences pourront cependant être détectées et traitées ultérieurement lors de la mise en forme de la carte (cf. chapitre 7). Dans le cas où ils sont appariés deux fois, ces sommets sont en réalité dédoublés et reliés par un segment « obstacle ». Pour le sommet global, tout se passe comme si deux sommets globaux distincts avaient été initialisés comme infiniment proches et avaient systématiquement été appariés aux mêmes sommets, jusqu'à ce que de nouvelles observations les distinguent enfin. Pour le sommet local, tout se passe comme s'il existait deux sommets infiniment proches dans le polygone local.

De même, dans le cas où seule une paire de segments (locaux ou globaux) est adjacente, ou dans le cas où aucune ne l'est, on procède pour les segments de gauche comme dans le cas d'une transition $(s_i, S_j) - S*$ et pour les segments de droite comme dans le cas d'une transition $S* - (s_k, S_l)$. Il peut aussi se produire des situations d'appariements multiples de sommets, qui sont traités de la même manière que dans le cas où les deux paires de segments sont adjacentes.

6.6 Mise à jour des positions des sommets

Comme le souligne Dissanayake [46], les équations de mesure correspondent à des observations relatives entre positon du robot et position des éléments de la carte. Plus précisément, les observations à l'instant t font intervenir l'estimation de position du robot $\mathbf{P}_r = (X_r, Y_r, \theta_r)^T$ (à l'instant $t-1$), l'estimation de déplacement du robot $\mathbf{P}_\Delta = (\Delta X_r, \Delta Y_r, \Delta\theta_r)^T$ (entre les instants $t-1$ et t) corrigée grâce au recalage de la carte locale par rapport à la carte globale, l'estimation de position $\mathbf{p}_{glob}$ de la primitive globale (calculée à l'instant $t-1$) et l'estimation de position $\mathbf{p}_{loc}$ de la primitive locale observée (à

l'instant t) dans le repère local du robot. On obtient ainsi l'équation générique suivante pour la mesure :

$$\mathbf{T}(\mathbf{p}_{loc}, \mathbf{P}_r + \mathbf{P}_\Delta) - \mathbf{p}_{glob} = 0$$

où l'opérateur de projection $\mathbf{T}$ permet de passer des coordonnées locales dans le repère du robot aux coordonnées globales dans le repère de la carte en cours de construction. On suppose ici que la position du robot coïncide avec celle du capteur : en pratique il faut effectuer un second changement de repère si le centre du robot ne coïncide pas avec celui du capteur.

Avec les instances d'observations sélectionnées plus haut, nous pouvons nous ramener à deux types d'équations : les mesures de coïncidence d'un point (global) sur un autre point (local) ou d'un point (global) sur une droite (locale). Ces équations s'appliquent également aux points virtuels.

*** Observation « point sur point » :**

Le point local $\mathbf{P}_l$ est représenté (dans le repère local) en coordonnées polaires (r_l, θ_l), déduites directement des mesures brutes du scan. Quant au point global $\mathbf{P}_g$, il est décrit en coordonnées cartésiennes (X, Y) (dans le repère de référence de la carte globale) : ces coordonnées sont stockées dans le vecteur d'état.

On note respectivement $\mathbf{C}_l$, $\mathbf{C}_g$, $\mathbf{C}_r$ et $\mathbf{C}_\Delta$ les matrices de covariance de $\mathbf{P}_l = (r_l, \theta_l)^T$, $\mathbf{P}_g = (X, Y)^T$, $\mathbf{P}_r = (X_r, Y_r, \theta_r)^T$ et $\mathbf{P}_\Delta = (\Delta X_r, \Delta Y_r, \Delta\theta_r)^T$.

La mesure à 2 dimensions s'écrit :

$$\mathbf{z} = (z_1, z_2)^T = \mathbf{f}(r_l, \theta_l, X, Y, X_r, Y_r, \theta_r, \Delta X_r, \Delta Y_r, \Delta\theta_r)$$

avec :

$$z_1 = r_l * \cos(\theta_l + \Delta\theta_r + \theta_r) + X_r + \Delta X_r - X$$

$$z_2 = r_l * \sin(\theta_l + \Delta\theta_r + \theta_r) + Y_r + \Delta Y_r - Y$$

La matrice de covariance de la mesure s'écrit de la manière suivante :

$$\mathbf{C_z} = \mathbf{J_l C_l J_l^T} + \mathbf{J_g C_g J_g^T} + \mathbf{J_r C_r J_r^T} + \mathbf{J_\Delta C_\Delta J_\Delta^T}$$

où $\mathbf{J_l}$, $\mathbf{J_g}$, $\mathbf{J_r}$ et $\mathbf{J_\Delta}$ représentent respectivement les matrices jacobiennes de $\mathbf{f}$ par rapport aux coordonnées (r_l, θ_l) du point local, aux coordonnées (X, Y) du point global, à la pose (X_r, Y_r, θ_r) du robot et au déplacement $(\Delta X_r, \Delta Y_r, \Delta\theta_r)$ depuis la dernière estimation.

*** Observation « point sur droite » :**

La droite locale est représentée (dans le repère local) en coordonnées polaires (d_l, α_l). Ces coordonnées sont déduites de celles des points locaux $P_1(\rho_1, \theta_1)$ et $P_2(\rho_2, \theta_2)$ extrémités du segment de droite par les équations suivantes :

$$\alpha_l = arctan\left(\frac{\rho_2 \cos\theta_2 - \rho_1 \cos\theta_1}{\rho_1 \sin\theta_1 - \rho_2 \sin\theta_2}\right)$$

$$d_l = \rho_1(\cos\theta_1 \cos\alpha + \sin\theta_1 \sin\alpha)$$

Pour calculer la matrice de covariance de ces coordonnées, on peut utiliser une propagation d'incertitude comme expliqué ci-dessus.

Le point global est décrit comme dans l'observation précédente en coordonnées cartésiennes (X, Y) dans le repère de référence de la carte globale.

On obtient ainsi la mesure à 1 dimension suivante :

$$\mathbf{z} = (z) = f(\alpha_l, d_l, X, Y, X_r, Y_r, \theta_r, \Delta X_r, \Delta Y_r, \Delta\theta_r)$$

avec :

$$z = (X - \Delta X_r - X_r)\cos(\alpha_l + \Delta\theta_r + \theta_r) + (Y - \Delta Y_r - Y_r)\sin(\alpha_l + \Delta\theta_r + \theta_r) - d_l$$

La variance de la mesure s'écrit de la manière suivante :

$$\sigma_z^2 = \mathbf{J_l C_l J_l^T} + \mathbf{J_g C_g J_g^T} + \mathbf{J_r C_r J_r^T} + \mathbf{J_\Delta C_\Delta J_\Delta^T}$$

où J_l, J_g, J_r et J_Δ représentent respectivement les matrices jacobiennes de $\mathbf{f}$ par rapport aux coordonnées (d_l, α_l) de la droite locale, aux coordonnées (X, Y) du point global, à la pose (X_r, Y_r, θ_r) du robot et au déplacement $(\Delta X_r, \Delta Y_r, \Delta\theta_r)$ depuis la dernière estimation.

*** Mise à jour par filtrage de Kalman :**

Ces équations de mesure sont introduites dans le formalisme décrit ci-dessus, qui permet de mettre à jour le vecteur d'état $\mathbf{x}$.

Plus concrètement, $\mathbf{x}$ est composé pour ses trois premières lignes de la pose du robot. Chaque plongement de sommet de la carte combinatoire contient donc les numéros de ligne du vecteur d'état correspondant à ses coordonnées. Au début, le vecteur ne fait que croître : tous les sommets existants sont conservés et les coordonnées des nouveaux sommets sont ajoutées à la fin du vecteur. Toutefois, si la mise en forme de la carte exige la suppression de certains sommets (cf. chapitre suivant), les numéros de lignes correspondant aux sommets conservés risquent d'être modifiés (la pose du robot occupe toutefois indéfiniment les trois premières lignes de $\mathbf{x}$). Ainsi, en pratique, nous pouvons utiliser une deuxième structure de données qui indique pour chaque numéro de ligne à quel plongement de sommet et à quelle coordonnée (X ou Y) de ce plongement elle correspond.

6.7 Réduction de la taille du vecteur d'état

Par rapport à une représentation qui ne comporterait que des segments indépendants (représentation « spaghettis » où les segments ne sont pas reliés entre eux par des relations d'adjacence), notre représentation implique un vecteur d'état de taille différente.

6.7.1 Bilan sur les arêtes « obstacles »

Supposons que la carte globale soit composée de n segments « obstacles ». Dans le cas d'une représentation « spaghettis », il faut en général quatre paramètres pour modéliser un segment individuel (par exemple les quatre coordonnées cartésiennes des extrémités, ou bien les coordonnées polaires de la droite porteuse, la direction de la demi-droite qui relie l'origine du repère au milieu du segment et la demi-longueur du segment). On aboutit donc à un vecteur d'état de taille $4n + 3$ (on ajoute le nombre 3 pour la pose du robot).

Avec notre représentation, il faut également quatre paramètres pour représenter un segment (deux coordonnées cartésiennes par sommet extrémité) mais certains sommets sont partagés entre segments adjacents : on économise donc certains paramètres. En réalité, il faut aussi ajouter deux paramètres supplémentaires par segment pour modéliser le point virtuel. On oscille donc, selon le nombre de relations d'adjacence existant, entre un vecteur de taille $4n + 3$ (cas où toutes les extrémités de segments sont reliées à des segments voisins) et un vecteur de taille $6n + 3$ (cas où aucun segment n'est relié à un segment voisin). On peut même descendre en-dessous de $4n + 3$ si l'on autorise des sommets incidents à plus de deux arêtes « obstacles » (mais ces configurations sont en général liées à des erreurs d'association).

A priori, il n'y a donc pas de grande différence quantitative entre les deux types de représentations pour le nombre de segments « obstacles », si ce n'est que dans notre modèle de carte, on ne peut pas en « oublier » (par exemple les plus petits ou ceux qui n'apportent pas grand chose à la localisation du robot dans les stratégies de « map management » [45]) à la différence des représentations « spaghettis » : de tels oublis risqueraient de générer des incohérences dans la modélisation de l'espace libre. On pourrait aussi objecter que dans certains cas, la fusion d'un segment obstacle et d'un segment non obstacle résulte en un seul segment dans les cartes « spaghettis » alors que dans notre représentation, on en obtient entre un et trois, suivant les mises en correspondance réalisées. Avec notre modèle de carte, il est en outre plus difficile de supprimer des sommets entre segments adjacents alignés car ces sommets peuvent être reliés à des segments « non obstacles ». Cependant, ces segmentations plus fines sont parfois plus représentatives de l'environnement. De plus, comme nous le verrons au chapitre suivant (dans le cadre de la mise en forme de la carte), il est possible de retirer des arêtes « non obstacles » et éventuellement leurs sommets extrémités si les faces adjacentes à ces arêtes ont été suffisamment observées comme étant libres.

6.7.2 Modélisation des sommets « non obstacles »

Toutefois, comme nous modélisons également l'espace libre, nous devons aussi maintenir dans le vecteur d'état des sommets « non obstacles », à l'intersection des arêtes « non obstacles ». A priori, ces sommets ne peuvent pas être réobservés car ils se situent sur des lignes de visée ou des limites de portée des capteurs : il faudrait que le robot repasse exactement au même endroit (et que l'on soit capable de le détecter) pour obtenir des configurations identiques. Il arrive que ces sommets « non obstacles » soient relativement nombreux : c'est le cas notamment si le capteur arrive en limite de portée dans une pièce de grandes dimensions ou s'il présente une zone aveugle. Dans ce cas, il serait plus efficace de conserver « en dur » les positions relatives des sommets « non obstacles » par rapport à la pose courante du robot et de n'introduire dans le vecteur d'état que les coordonnées du robot (c'est-à-dire celles du « centre » du scan courant). Si le nombre local de sommets « non obstacles » est de i, on passe ainsi de $2i$ paramètres à seulement 3.

6.8 Convergence du filtre

Nous utilisons une formulation du filtre de Kalman très proche de celle employée par Dissanayake et al. lors de leur démonstration de convergence [46]. Nous pourrions facilement nous ramener à une formulation totalement identique (sauf dans le cas de la procédure de « recalage-fusion ») et satisfaire ainsi aux hypothèses de la démonstration : Moutarlier montre d'ailleurs que sa formulation est équivalente à celle du filtre de Kalman dans le cas où les bruits sur l'état et la mesure sont considérés comme blancs et indépendants [139]. En particulier, les points virtuels de notre représentation peuvent être considérés comme des sommets classiques avec des observations de même nature lors des cassures ou lors des observations sur des droites locales. De même, les appariements multiples de sommets peuvent être

traités comme deux appariements uniques, correspondant chacun à l'un des sommets dédoublés. Ainsi, sous réserve des problèmes d'approximations numériques et de linéarisation du filtre de Kalman étendu, on a en principe les propriétés de convergence suivantes :

- la covariance de chaque objet de la carte décroît de façon monotone ;
- les estimations de balises tendent à devenir complètement corrélées ;
- leur covariance tend vers une borne inférieure liée à la covariance initiale du véhicule.

6.9 Conclusion et perspectives

Nous avons adapté les techniques de SLAM par filtrage de Kalman à notre représentation : en particulier, pour modéliser les « cassures », nous recourons à des points virtuels placés sur les segments globaux et présentant une grande incertitude dans l'axe du segment correspondant. Ces points virtuels sont observés à chaque fois que le segment global correspondant est vu par le robot (à condition qu'il soit bien apparié avec un segment local), ce qui permet de maintenir les covariances avec les autres éléments de la carte. Une cassure correspond alors à une observation explicite de ce point virtuel sur un point de mesure. Ainsi, les observations peuvent être classées selon trois catégories qui permettent de traiter tous les cas : observation d'un point global sur une droite locale, observation d'un point global sur un point local, cassure (observation d'un point virtuel sur un point local). Nous avons également défini un algorithme permettant d'extraire ces observations à partir du chemin d'appariement extrait par la technique décrite au chapitre précédent. Enfin, comme les éléments du vecteur d'état et les équations de mesure restent similaires à celles des techniques classiques, nous conservons les mêmes propriétés que dans le cas du SLAM basé sur l'EKF.

Parmi les perspectives envisageables figurent bien évidemment les améliorations au cadre classique du SLAM par filtrage de Kalman, en particulier un découpage en sous-cartes et une réduction des erreurs de linéarisation. Nous pouvons également envisager de passer à d'autres méthodes d'estimation, telles que le FastSLAM qui est a priori tout-à-fait compatible avec notre représentation : les positions successives du robot seraient modélisées par des particules et les positions des points (y compris des points virtuels) pourraient être traitées via des filtres de Kalman individuels.

Par ailleurs, nous pouvons envisager d'autres manières de gérer les points virtuels, qu'il serait intéressant de comparer à la méthode proposée ci-dessus :

- Une première idée consiste à s'inspirer du formalisme proposé par Leonard et Rikoski [118] sur les décisons retardées (« delayed decision »). Au lieu de maintenir dans le vecteur d'état un point virtuel par segment global S_j, il s'agirait de garder en mémoire toutes les observations (les segments locaux) associées à S_j (ainsi que l'instant auquel cette observation a été faite). Au moment d'observer une cassure, on crée un nouveau point à l'endroit indiqué par l'observation, corrélé à la position courante du robot. On l'observe ensuite selon les mesures attachées à S_j. Une telle observation fait intervenir les variables suivantes : pose du robot à l'instant de l'observation (cette pose passée apparaît déjà dans le vecteur d'état) et position courante du point de cassure. On met ainsi à jour les positions passées du robot, on raffine la position courante du point de cassure et on précise la position de tous les éléments corrélés dans le vecteur d'état.

- Une seconde idée consiste à s'inspirer des techniques de SLAM angulaire (« bearing only SLAM »), exploitant généralement des données de vision sans connaissance de la profondeur des objets observés. Le problème d'initialisation de la profondeur des objets ressemble à notre problème d'initialisation de la cassure, sauf que dans notre cas, cette initialisation ne résulte pas nécessairement de

plusieurs observations du même objet (la cassure peut n'être observée qu'une seule fois). On peut notamment considérer les travaux de Kwok et Dissanayake [109], qui proposent d'utiliser plusieurs gaussiennes le long du segment pour modéliser les profondeurs possibles du point observé (ce qui revient à un filtre de Kalman multi-hypothèse) : l'hypothèse la plus probable sera ensuite sélectionnée selon les appariements réalisés. Dans notre représentation, l'utilisation de plusieurs points virtuels sur les grands segments permettrait de limiter d'éventuels problèmes de linéarisation. En effet, les observations implicites sur les segments locaux appariés peuvent conduire à des valeurs d'incertitude (et de déplacement) différentes suivant la position du point de cassure sur son segment, en particulier si le segment local apparié présente un écart angulaire non négligeable avec le segment global. De plus, l'utilisation de points virtuels présentant une incertitude moindre pourrait prévenir d'éventuels problèmes liés aux approximations numériques, dans les calculs d'inversion de matrices notamment. La mise en œuvre de plusieurs points virtuels nécessiterait toutefois de modifier légèrement l'algorithme d'extraction d'observations pour en tenir compte (au niveau de la dupplication de ces points lors des cassures notamment).

Le nouveau vecteur d'état calculé par application du filtre de Kalman doit contribuer à la mise à jour de la carte globale. Il s'agit d'abord de déplacer les sommets selon les nouvelles valeurs indiquées par le vecteur d'état. Il faut également réaliser une restructuration topologique de la carte globale suite à la fusion avec le polygone d'espace libre local, selon les résultats de la phase de mise en correspondance. Comme nous allons le voir dans le chapitre suivant, ces deux étapes sont en fait étroitement liées.

Chapitre 7

Mise à jour de la carte globale

« L'oeil doit 'brouter' la surface, l'absorber partie après partie, et remettre celle-ci au cerveau qui emmagasine les impressions et les constitue en un tout. »
P. Klee.

7.1 Position du problème

La mise à jour de la carte globale suite à l'acquisition et au traitement d'un nouveau polygone d'espace libre concerne deux aspects :

- **la mise à jour géométrique**, dont l'objectif est de corriger la position géométrique des éléments de la carte globale selon les nouvelles estimations résultant du filtrage de Kalman ;
- **la mise à jour topologique**, qui concerne la fusion de la carte locale dans la carte globale : cette opération assure l'intégration des nouveaux éléments apparaissant dans la carte locale (jamais observés jusqu'alors) ainsi que l'incrémentation des degrés d'occupation histogrammiques liés à la nouvelle observation.

Toutefois, l'étape de mise à jour géométrique ne peut être réalisée indépendamment de toute considération topologique. En effet, comme l'illustre la figure 7.1, une simple modification des plongements géométriques risque d'aboutir à des incohérences topologiques (retournements de faces, polygones croisés, superpositions...) voire même à une perte d'information sur les niveaux d'occupation. De telles configurations sont souvent évitées dans les travaux classiques de modélisation géométrique, par exemple dans le cas où l'on modélise la déformation d'un tissu, et où les contraintes mécaniques empêchent généralement les cellules du maillage de se recouper ou de se retourner. Dans notre cas, les contraintes ne sont pas mécaniques, mais proviennent des corrélations entre estimations de position des éléments de la carte, et nous ne pouvons exclure a priori que des situations pathologiques apparaissent. Il importe donc de gérer ces incohérences (qui peuvent n'être que temporaires et disparaître lors de la prochaine mise à jour géométrique) et d'assurer en permanence le « bon plongement » de la carte combinatoire globale.

Comme nous l'avons vu dans les chapitres concernant l'état de l'art, rares sont les travaux de SLAM qui s'attachent à reconstruire dans un cadre unifié une représentation à la fois géométrique (représentation polygonale des frontières d'obstacles) et surfacique de l'environnement. A notre connaissance, seuls les travaux de Moutarlier [139] [141] ont détaillé la fusion incrémentale et cohérente des polygones d'espace libre dans une représentation unique, maintenant également la meilleure estimation des positions des frontières d'obstacles. Toutefois, la représentation choisie ne présente que deux niveaux d'occupation (« libre » et « inconnu ») et ne maintient pas toutes les arêtes des polygones initiaux : cette limita-

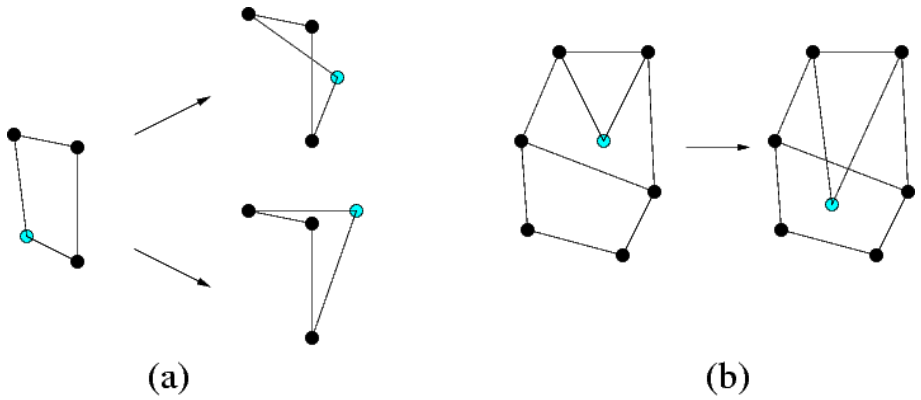


FIG. *7.1: Déplacement de sommet. (a) Le déplacement du point bleu conduit à un retournement de la cellule puis à un polygone croisé. (b) Le déplacement du point bleu conduit à la superposition de deux faces de la carte.*

tion empêche de gérer certains phénomènes. Par exemple, comme l'illustre la figure 7.2, il peut arriver qu'au gré des mise à jour géométriques, deux polygones initialement en superposition partielle soient finalement séparés. Si les arêtes « non obstacle » de la zone de recouvrement des polygones ne sont pas gardées en mémoire, la fusion devient irréversible : il est impossible de reconstruire les arêtes effacées. De même, lorsqu'un polygone devient croisé, Moutarlier propose de supprimer l'un des appendices, qui aurait pourtant pu disparaître de lui-même à la prochaine mise à jour géométrique. Enfin, lorsqu'un robot s'approche d'un mur et découvre un coin ou un renfoncement qui lui avait échappé jusqu'alors (suite à une erreur de mesure par exemple), il lui est difficile de trancher entre les deux hypothèses de mur lisse et de mur découpé : un test sur des degrés d'occupation histogrammiques permettrait au moins de maintenir l'ambiguïté durant un certain temps, puis de choisir l'hypothèse correspondant à l'observation la plus fréquente par exemple (cf. Fig. 7.3).

Plus généralement, il est apparu que les incohérences de type « polygones croisés » ou « retournement » étaient difficiles à gérer et nécessitaient de recourir à certaines conventions (par exemple la définition arbitraire de l'intérieur et de l'extérieur de l'espace libre), parfois décorrélées des processus réels d'acquisition de données et d'estimation de position sous-jacents. Notre objectif consiste donc à limiter ces problèmes grâce au modèle de carte choisi, en utilisant notamment la structure de cartes combinatoires et les opérations de manipulation associées.

FIG. *7.2: Exemple de séparation de deux polygones : après mise à jour de la position des sommets bleus, les deux polygones ne se recouvrent plus.*

Avant de décrire les différentes stratégies de mise à jour de la carte globale, nous définissons un nouveau modèle de cartes, un peu plus sophistiqué que celui qui a été proposé dans le chapitre de choix du modèle, afin de prendre en compte les configurations provisoires (et parfois incohérentes) induites par les déplacements géométriques de sommets. Nous introduisons ainsi le concept de **« carte combinatoire colorée »** (la couleur s'appliquant ici aux brins et aux sommets, à la différence des problèmes habituels de coloriage de cartes qui s'intéressent aux couleurs de régions). Cette représentation affinée

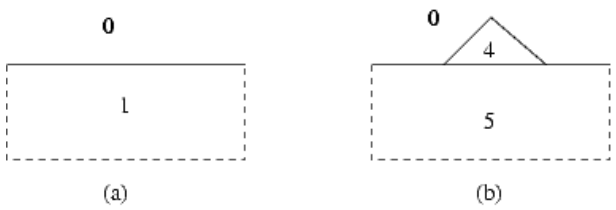


FIG. *7.3: Exemple d'observations variables suivant la distance à l'obstacle. (a) De loin, le mur est perçu comme rectiligne. (b) De près, les points de mesures étant plus denses, on découvre un renfoncement : cette observation se superpose à la précédente d'où l'apparition d'un segment noir incohérent entre deux zones libres. Si une majorité d'observations incluent le renfoncement (comme c'est le cas ici d'après les degrés d'occupation), on peut pencher en faveur de cette modélisation et supprimer le segment incohérent.*

contient d'une part une composante monochrome « glissante », susceptible de se déplacer (au sens où les plongements de sommets peuvent évoluer) mais pérenne dans la mesure où sa structure topologique est fixée une fois pour toutes (au risque que cette carte soit mal plongée, ne permettant donc pas de modéliser correctement les degrés d'occupation), et d'autre part une partie colorée « fixe », bien plongée (et modélisant correctement les degrés d'occupation), qui n'a pas vocation à être déplacée et n'a qu'une validité provisoire.

Ensuite, nous détaillons et comparons deux stratégies de mise à jour :
- **une stratégie de superposition de polygones**, qui conserve explicitement dans la carte globale la structure de chaque polygone local d'espace libre : la nouvelle carte globale est obtenue en recommençant à chaque étape la superposition puis la fusion de tous ces polygones. Nous verrons toutefois que cette stratégie limite les possibilités ultérieures de mise en forme et de simplification de la carte, par suppression d'arête notamment ;
- **une stratégie de fusion incrémentale des polygones** d'espace libre, qui ne nécessite pas le maintien explicite de la structure de chaque polygone, et permet de procéder par modifications incrémentales à chaque étape. Nous verrons que cette stratégie facilite la modification ultérieure de la carte et allège la représentation.

Chacune de ces stratégies est décrite dans le cas des cartes combinatoires plongées dans $\mathbb{R}^2$ puis dans le cas des cartes combinatoires discrètes. Par ailleurs, nous verrons qu'elles sont toutes deux basées sur une nouvelle opération fondamentale pour le maintien de la cohérence de la structure : **le déplacement « sûr » des sommets**, qui permet de maintenir des degrés d'occupation corrects même dans le cas des configurations de retournement ou de croisement de polygones (cf. figure 7.1), assurant ainsi la réversibilité de ces situations pathologiques.

7.2 Définition de cartes combinatoires « colorées »

Les cartes combinatoires colorées ont vocation à gérer une structure de carte combinatoire à plongement géométrique évolutif, susceptible de générer des intersections, des superpositions et des retournements d'objets potentiellement provisoires dans le processus de construction de cartes. La différence entre la partie évolutive et la partie figée est matérialisée dans la carte selon un code de couleur. La représentation comprend ainsi :
- **une partie pérenne monochrome (de couleur noire)**, qui correspond d'une part aux liens d'adjacence initiaux entre arêtes (définis lors de la polygonalisation des scans) et aux sommets des polygones fusionnés et d'autre part aux liens induits par les appariements successifs entre carte

locale et carte globale ;
– **une partie éphémère « colorée » (notamment en rouge)**, qui correspond à la gestion des intersections, incidences et superpositions induites par le plongement géométrique provisoire des éléments de la partie noire. Ces configurations provisoires doivent être prises en compte pour définir un plongement correct de la carte et pour calculer les degrés d'occupation réels. Cependant, lors des prochaines mises à jour de la carte globale, ces configurations peuvent être amenées à disparaître.

7.2.1 Extension colorée aux cartes combinatoires classiques

Plus précisément, par rapport à la définition initiale des cartes combinatoires (noires), on réalise les modifications suivantes (cf. Fig. 7.4) :

– On ajoute à l'ensemble des sommets noirs initiaux (définis lors de la polygonalisation des scans) des sommets rouges, qui correspondent à des intersections ou à des incidences non matérialisées sur la carte noire mal plongée. Ces sommets induisent un découpage des arêtes noires initiales (définies lors de la polygonalisation des scans).

– Les brins sont colorés. Ainsi, les brins noirs initiaux restent noirs. Les brins ajoutés suite au découpage des arêtes noires et plongés sur les sommets rouges du découpage sont rouges. Les brins qui seraient provisoirement supprimés (parce que l'arête correspondante est de longueur nulle ou parce qu'elle est superposée à une autre) sont « grisés » afin de ne plus être pris en compte dans les tests utilisés dans les opérations de raffinement : un brin noir devient gris et un brin rouge devient rose. En outre, pour des raisons de commodité, les brins peuvent également être « bleutés » pour indiquer qu'ils sont superposés à des brins grisés : dans ce cas, un brin noir devient bleu et un brin rouge devient violet (cf. Tab. 7.1). Pour gérer ces liens de superposition (et parcourir les différents brins superposés), nous introduisons également une nouvelle permutation appelée β.

foncé	**grisé**	**bleuté**
noir	gris	bleu
rouge	rose	violet

TAB. *7.1: Tableau des colorations de brins : couleurs résultant des opérations qui consistent à « griser » ou à « bleuter » selon que le brin est initialement noir ou rouge.*

– En plus des 0-coutures et des 1-coutures noires définies à l'initialisation des polygones et suite aux appariements, on ajoute des 0-coutures et des 1-coutures rouges provisoires, qui matérialisent les liens courants entre tous les éléments de la carte courante bien plongée. On note que les 0-coutures noires des brins rouges renvoient au brin noir 0-lié avec le brin noir dont ils sont issus. Quant aux 1-coutures noires des brins rouges, elles renvoient au brin rouge issu de la même arête noire. Nous verrons que ces coutures noires sur brins rouges servent notamment lors de l'opération de déplacement de sommet définie plus loin.

– Pour gérer d'éventuelles fusions de sommets, chaque brin possède outre son plongement noir classique (le point sur lequel le brin est plongé lors de sa création) un plongement rouge courant sur un éventuel sommet avec lequel le plongement noir serait provisoirement fusionné. On note que la notion de sommet rouge (sommet provisoire) est indépendante de la notion de plongement rouge (plongement provisoire) : un brin peut avoir un plongement rouge sur un sommet noir et inverse-

ment, un brin peut avoir un plongement noir sur un sommet rouge.

– Les brins portent des labels de face qui correspondent à des degrés d'occupation noirs et rouges. Les degrés d'occupation noirs indiquent le nombre d'arêtes fusionnées par appariement sur l'arête courante (ainsi que le côté où se trouve l'espace libre puisque seul un brin de l'arête porte ce degré d'occupation) : ils ne correspondent pas à de véritables degrés d'occupation puisqu'ils ne tiennent pas compte de la restructuration topologique de la carte. Les degrés d'occupation rouges en revanche représentent les véritables degrés d'occupation issus de la superposition des polygones d'espace libre (chacun de degré d'occupation 1) dans la carte colorée bien plongée.

– Pour la stratégie de mise à jour par superposition de polygones uniquement, chaque brin porte également la liste des numéros de polygones dont il est issu.

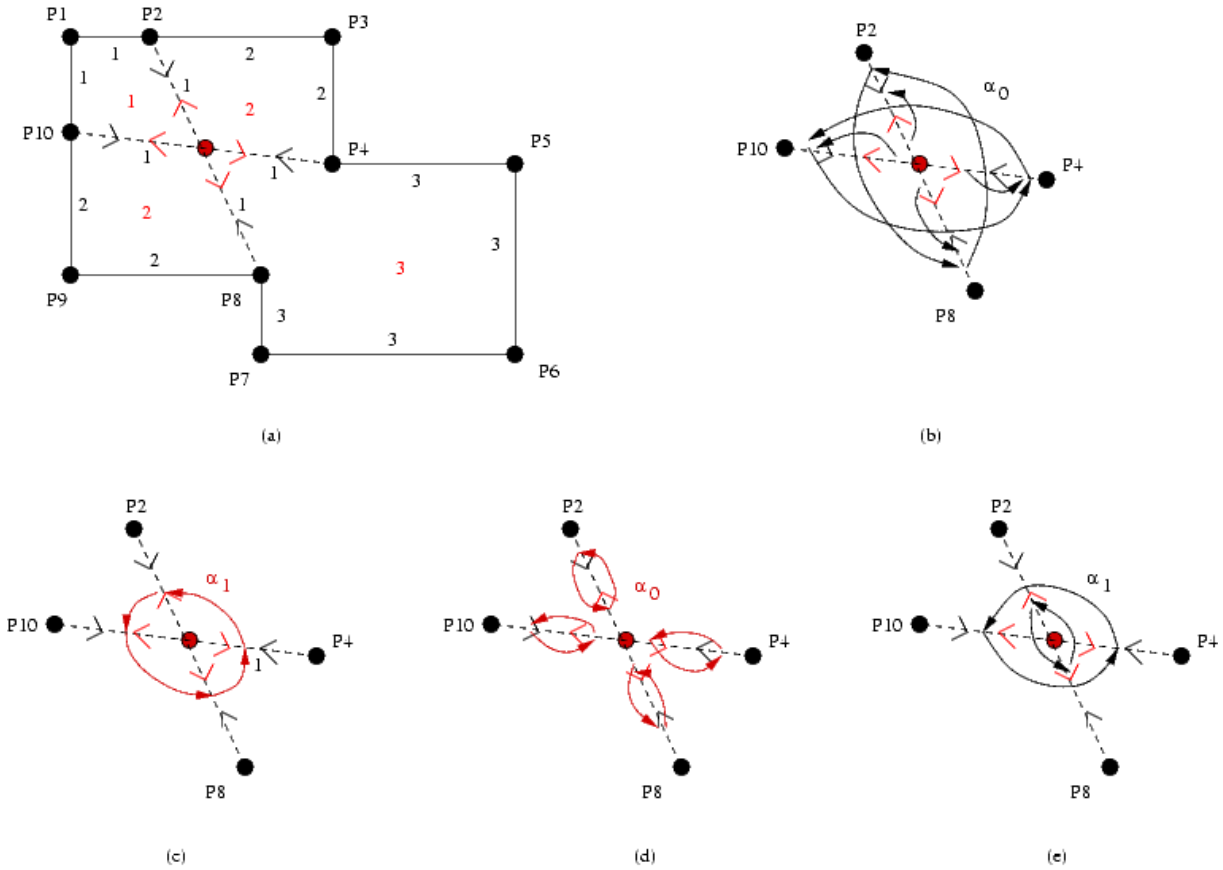


FIG. *7.4: Exemple de carte colorée constituée de la superposition de trois polygones d'espace libre : $P_1P_2P_3P_4P_5P_6P_7P_8P_9P_{10}$, $P_2P_3P_4P_5P_6P_7P_8$ et $P_4P_5P_6P_7P_8P_9P_{10}$. (a) L'intersection des segments « non obstacles » (en pointillés) a donné lieu à la création d'un sommet rouge et de quatre nouveaux brins (flèches rouges) associés. Les nombres en noir correspondent aux degrés d'occupation noirs associés aux brins noirs et les nombres en rouge correspondent aux degrés d'occupation rouges. (b) Définition des 0-coutures noires. (c) Définition des 1-coutures rouges. (d) Définition des 0-coutures rouges. (e) Définition des 1-coutures noires.*

7.2.2 Définition plus formelle

Il est en principe possible de définir de façon formelle ces modifications aux cartes combinatoires classiques, à travers des spécifications algébriques. La spécification complète de cette nouvelle représentation (à la manière des spécifications fournies dans la thèse de Cazier par exemple [23]) nous a paru sortir du cadre de ce mémoire de thèse. Nous pouvons cependant indiquer quelques éléments qui faciliteront la mise en place de ces spéfications, ainsi que la validation théorique des opérations de manipulation que nous décrivons dans les sections suivantes.

Ainsi, on peut définir la couche topologique d'une *carte combinatoire colorée* comme constituée cette fois d'un 9-uplet $M = (D_n, D_{ng}, D_r, D_{rg}, \alpha_0, \alpha_1, \alpha_{0_r}, \alpha_{1_r}, \beta)$ où :

- D_n, D_{ng}, D_r et D_{rg} sont des ensembles finis disjoints d'éléments appelés respectivement *brins noirs*, *brins gris*, *brins rouges* et *brins roses*. $D = D_n \bigcup D_{ng} \bigcup D_r \bigcup D_{rg}$ est appelé ensemble des *brins colorés* ou simplement ensemble des *brins*. $D_f = D_n \bigcup D_r$ est appelé ensemble des *brins foncés* et $D_g = D_{ng} \bigcup D_{rg}$ ensemble des *brins grisés* ;
- α_{0_r} (α_0 rouge) est une involution dans D ;
- α_0 est une fonction de D vers $D_n \bigcup D_{ng}$ et sa restriction α_{0_n} à $D_n \bigcup D_{ng}$ est une involution ;
- α_1 et α_{1_r} (α_1 rouge) sont des permutations dans D ;
- β est une permutation dans D, qui sert à gérer les liens entre arêtes superposées.

Soit $z \in D$. Les brins $\alpha_1 z$ et $\alpha_1^{-1} z$ sont respectivement les *1-successeur noir* et *1-prédécesseur noir* de z. Ils sont dits *1-liés* ou *1-cousus* par *1-liaison* ou *1-couture noire*. Le brin $\alpha_0 z$ est le *0-successeur noir* de z. Si de plus $z \in D_n \bigcup D_{ng}$, le brin $\alpha_{0_n}^{-1} z$ est le *0-prédécesseur noir* de z : ils sont dits *0-liés* ou *0-cousus* par *0-couture noire*.

Soit k égal à 0 ou 1. Les brins $\alpha_{k_r} z$ et $\alpha_{k_r}^{-1} z$ sont respectivement les *k-successeur rouge* et *k-prédécesseur rouge* de z. Ils sont dits *k-liés* ou *k-cousus* par *k-liaison rouge* ou *k-couture rouge*. Les brins βz et $\beta^{-1} z$ sont respectivement les *β-successeur* et *β-prédécesseur* de z. Enfin, un brin est dit *bleuté* s'il appartient à l'ensemble D_f et s'il n'est pas son propre *β-successeur* (c'est-à-dire s'il existe des brins superposés). Ainsi, un brin bleuté sera appelé *brin bleu* s'il appartient à D_n et *brin violet* s'il appartient à D_r (cf. 7.1).

Il faut également définir le plongement planaire de ces cartes colorées. Pour cela, on ajoute au 9-uplet précédent une fonction de *plongement rouge* ainsi qu'une fonction de *plongement noir*. Les numéros de polygones utilisés dans la stratégie de mise à jour par superposition de polygones sont également des 2-labels, de même nature que les degrés d'occupation noirs.

Par ailleurs, certaines propriétés doivent être respectées dans la construction des cartes colorées et lors de leur manipulation, afin de maintenir une structure cohérente :

- Pour gérer les liens entre brins superposés, la permutation β relie un brin bleu au premier brin gris de la liste des brins superposés, puis chaque brin gris au suivant dans la liste (cf. Fig. 7.5). Le dernier brin gris de la liste peut remonter au brin bleu. Quant aux brins qui ne sont ni bleutés, ni grisés, ils sont leur propre successeur par β. Ainsi, une orbite de β contient un et un seul brin foncé, et tous les brins de cette orbite ont des plongements rouges superposés. Quant aux 1-liaisons rouges des brins grisés, on peut définir par convention que ces brins constituent leurs propres 1-successeurs rouges.

- Les brins d'une orbite de α_{0_r} sont nécessairement tous inclus soit dans D_f (*arête foncée*) soit dans $D_{rg} \bigcup D_{ng}$ (*arête grisée*). En outre, si un brin est bleuté, le brin 0-lié en rouge l'est également par

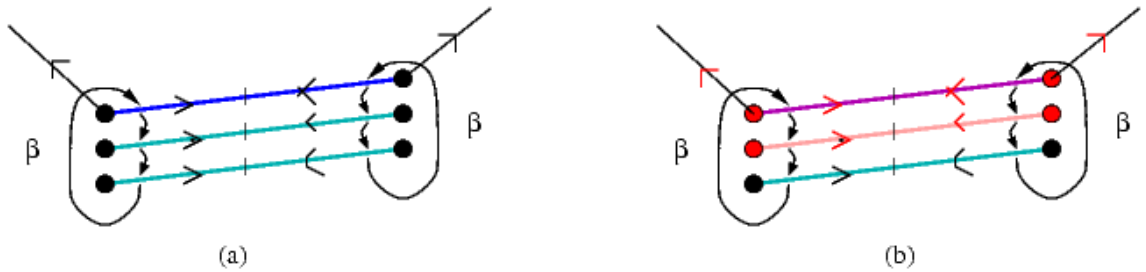


FIG. *7.5: (a) Liaison β entre brins bleus et brins gris (cyan sur la figure) superposés. (b) Liaison β entre brins violets, brins roses et brins gris superposés.*

construction (*arête bleutée*).

- Les brins d'une orbite de α_1 sont nécessairement tous inclus soit dans $D_r \bigcup D_{rg}$ (*sommet rouge*) soit dans $D_n \bigcup D_{ng}$ (*sommet noir*). En outre, les plongements noirs des brins d'une même 1-orbite noire (sommets noirs ou sommets rouges) sont identiques.

- Les plongements rouges des brins d'une même 1-orbite rouge sont identiques.

- Les plongements rouge et noir d'un même brin sont superposés géométriquement.

- Le triplet $(D_n \bigcup D_{ng}, \alpha_0, \alpha_1)$ (il s'agit en fait des restrictions de α_0 et α_1 à $D_n \bigcup D_{ng}$) muni de la fonction de plongement noir (sa restriction à $D_n \bigcup D_{ng}$ en fait) est une carte combinatoire monochrome classique (*carte noire*), a priori mal plongée. On peut remarquer que ses sommets sont noirs.

- Le triplet $(D_f, \alpha_{0_r}, \alpha_{1_r})$ (il s'agit en fait des restrictions de α_{0_r} et α_{1_r} à D_f) muni de la fonction de plongement rouge (sa restriction à D_f en fait) est une carte combinatoire monochrome (*carte rouge*) classique (bien plongée suite au raffinement coloré que nous décrirons plus loin). On peut remarquer que ses sommets sont noirs ou rouges.

En outre, les relations entre ces deux cartes rouge et noire (relations qui ne sont pas prises en compte dans ces cartes combinatoires monochromes) sont régies par :

- les liaisons rouges et le plongement rouge des brins foncés : ces liaisons assurent la restructuration topologique de la carte noire mal plongée via la carte rouge ;

- les liaisons noires des brins rouges et roses : ces liaisons permettent de remonter au brin noir initial dont les brins rouges et roses sont issus (suite au découpage des arêtes de la carte noire via des sommets rouges). Ainsi, les 1-liaisons noires des brins rouges (ou roses) relient entre eux les brins rouges (ou roses) adjacents le long d'une même arête de la carte noire. Quant aux 0-liaisons noires de brins rouges (ou roses), elles renvoient au brin noir de la même arête de carte noire, dans le sens opposé au brin rouge (ou rose) courant. Ces propriétés doivent également être respectées lors de la création et de la manipulation des cartes colorées. Elles serviront notamment lors des opérations de suppression d'arêtes ;

- la permutation β : nous avons vu que cette permutation permet de conserver dans la carte colorée les brins grisés, sans les supprimer réellement. Comme nous l'expliquons dans les sections suivantes,

les brins grisés n'interviennent plus dans les règles de déclenchement du raffinement coloré, mais ils doivent être considérés dans les opérations de déplacement et d'ajout de sommets.

En outre, on peut remarquer qu'une carte monochrome bien plongée peut facilement être transformée en carte colorée. Dans ce cas, il n'existe que des brins noirs : les ensembles D_r et D_g sont vides et il n'existe pas d'arêtes bleutées. Les liaisons noires par α_0 et α_1 sont déduites de la carte monochrome initiale. Les liaisons rouges par α_{0_r} et α_{1_r} sont respectivement des copies des 0- et 1-liaisons noires. La permutation β est réduite à la fonction identité. Quant aux plongements rouges, ils sont identiques aux plongements noirs. Par ailleurs, nous verrons plus loin qu'il peut être utile de modifier la couleur d'un brin. En théorie, il faut créer pour cela un nouveau brin d'une autre couleur, et copier tous les plongements, labels et liaisons du brin original dans le nouveau. En pratique, il suffira de modifier l'attribut « couleur » du brin.

Enfin, les cartes colorées peuvent être étiquetées, de la même manière que les cartes monochromes classiques. En particulier, on définit des labels de faces correspondant à des degrés d'occupation : un *degré d'occupation rouge* (véritables degrés d'occupation de la partie rouge bien plongée de la carte) et un *degré d'occupation noir* (en fait le nombre d'arêtes de polygones fusionnées sur l'arête courante, en plaçant le label sur la face qui correspond à l'intérieur de ces polygones). Les labels noirs sont gérés directement dans les opérations de mise en correspondance, d'ajout et de déplacement de sommet. En revanche, les labels rouges, qui sont initialement déduits des labels noirs, vont être corrigés dans les opérations de complétion de label associées au raffinement.

7.3 Raffinement de cartes colorées

Dans cette section, nous détaillons les règles de raffinement qui permettent la restructuration topologique d'une superposition de cartes colorées, de manière à vérifier les propriétés énoncées ci-dessus. Pour cela, nous apportons quelques modifications aux six règles initialement proposées par Cazier [23]. Notre description reste textuelle. Toutefois, dans un souci de validation, il devrait être possible de la formaliser ultérieurement au moyen de règles de réécriture conditionnelles, comme chez Cazier [23].

Avant de décrire ces règles, précisons que les éléments constituant la carte noire ne sont pas modifiés lors du raffinement coloré, si ce n'est que certains brins noirs peuvent être grisés ou bleutés. Seuls les plongements rouges, les liaisons rouges et la fonction β vont subir des changements. Certains brins rouges, roses et violets pourront également être ajoutés. Ainsi, les liaisons et plongements noirs ne seront pas considérés durant cette opération : ce sont les liaisons et plongements rouges qui serviront à déclencher les règles et qui en seront affectés. Par ailleurs, les arêtes grisées (grises et roses) sont considérées comme supprimées durant toute la phase de raffinement : elles sont également ignorées dans les conditions de déclenchement des règles.

Enfin, pour décrire ces règles, nous introduisons les fonctions de liaison $l_0(x,y)$, $l_1(x,y)$, $l_{0_r}(x,y)$, $l_{1_r}(x,y)$ et $l_\beta(x,y)$ qui lient les brins x et y respectivement par 0-liaison noire, 1-liaison noire, 0-liaison rouge, 1-liaison rouge et β-liaison. Dans le cas de $l_1(x,y)$, de $l_{1_r}(x,y)$ et de $l_0(x,y)$, y devient le k-successeur de x (le 1-successeur dans le cas des 1-liaisons et le 0-successeur dans le cas de la 0-liaison noire) et x devient le k-prédécesseur de y (mais à l'inverse, x ne devient pas nécessairement le k-successeur de y). De même, pour $l_\beta(x,y)$, y devient le β-successeur de x et x devient le β-prédécesseur de y (mais pas nécessairement l'inverse). En revanche, dans le cas de $l_{0_r}(x,y)$, y devient le 0-successeur et le 0-prédécesseur rouges de x, x devient le 0-successeur et le 0-prédécesseur rouges de y, de manière à reconstituer une involution.

Notation : l'arête composée des brins x et y 0-liés par liaison rouge est notée $(x, y)_r$.

7.3.1 Règle 1

Condition de déclenchement : l'arête $(x, y)_r$ est de longueur nulle.

Résultat : l'arête $(x, y)_r$ est grisée. Plus précisément :

* Les deux brins x et y sont grisés. Ainsi, un brin noir (ou bleu) devient gris et un brin rouge (ou violet) devient rose.

* Les 1-liaisons rouges sont mises à jour sur les brins grisés et sur les brins adjacents. Ainsi, x et y sont court-circuités dans leurs 1-orbites rouges respectives (on effectue $l_{1_r}(\alpha_{1_r}^{-1}x, \alpha_{1_r}(x))$ et $l_{1_r}(\alpha_{1_r}^{-1}y, \alpha_{1_r}(y))$) puis il deviennent chacun leur propre 1-successeur rouge ($l_{1_r}(x, x)$ et $l_{1_r}(y, y)$).

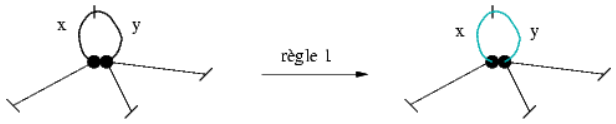


FIG. *7.6: Règle 1 du raffinement coloré*

7.3.2 Règle 2

Condition de déclenchement : l'arête $(x, y)_r$ est superposée avec l'arête $(z, t)_r$, c'est-à-dire que le plongement (noir ou rouge) de x est superposé avec celui de z et que le plongement (noir ou rouge) de y est superposé avec celui de t.

Résultat : l'arête $(x, y)_r$ est grisée, l'arête $(z, t)_r$ est bleutée et une β-liaison entre les deux arêtes est créée. Plus précisément :

* Les deux brins x et y sont grisés. Ainsi, un brin noir (ou bleu) devient gris et un brin rouge (ou violet) devient rose.

* Les 1-liaisons rouges sont mises à jour sur les brins grisés et sur les brins adjacents. Ainsi, x et y sont court-circuités dans leurs 1-orbites rouges respectives ($l_{1_r}(\alpha_{1_r}^{-1}(x), \alpha_{1_r}(x))$ et $l_{1_r}(\alpha_{1_r}^{-1}(y), \alpha_{1_r}(y))$) puis ils deviennent chacun leur propre 1-successeur rouge ($l_{1_r}(x, x)$ et $l_{1_r}(y, y)$).

* Les nouveaux brins grisés sont insérés (avec les autres brins de leur propre β-orbite) dans les β-orbites des brins bleutés. Ainsi, si $\beta(x) = x$ (x n'était pas bleu au départ) alors on applique $l_\beta(x, \beta(z))$ et $l_\beta(z, x)$. Sinon, on effectue $l_\beta(\beta^{-1}(x), \beta(z))$ et $l_\beta(z, x)$.

De même, si $\beta(y) = y$ (y n'était pas bleu au départ) alors on applique $l_\beta(y, \beta(t))$ et $l_\beta(t, y)$. Sinon, on effectue $l_\beta(\beta^{-1}(y), \beta(t))$ et $l_\beta(t, y)$.

* Les deux brins z et t sont donc bleutés. Ainsi, un brin noir devient bleu, un brin rouge devient violet et un brin bleu ou violet reste de la même couleur.

FIG. *7.7: Règle 2 du raffinement coloré*

7.3.3 Règle 3

Condition de déclenchement : le brin x est incident à l'arête $(z,t)_r$ de la carte rouge.

Résultat : création d'un nouveau sommet rouge et des brins rouges associés sur l'arête (z,t). Plus précisément :

* Création d'un sommet (rouge) S superposé au plongement de x.

* Création de deux brins rouges a et b plongés en noir et en rouge sur S. On note que si $(x,y)_r$ est une arête bleutée, alors a et b sont en fait des brins violets.

* Définition des liaisons rouges des nouveaux brins rouges (ou violets) :
$l_{0_r}(a,z)$, $l_{0_r}(z,a)$, $l_{0_r}(b,t)$ et $l_{0_r}(t,b)$;
$l_{1_r}(a,b)$ et $l_{1_r}(b,a)$.

* Définition des liaisons noires des nouveaux brins rouges (ou violets) :
$l_0(a,\alpha_0(t))$ et $l_0(b,\alpha_0(z))$;
$l_1(a,b)$ et $l_1(b,a)$.

* Définition des liaisons β des nouveaux brins rouges (ou violets) :
$l_\beta(a,a)$ et $l_\beta(b,b)$

* Gestion des arêtes grisées éventuellement superposées à l'arête $(z,t)_r$: il s'agit de découper ces arêtes de la même manière que $(z,t)_r$.
Si $\beta(z)$ est différent de z, cela signifie que l'arête $(z,t)_r$ est superposée à au moins une arête grisée ($(z,t)_r$ est une arête bleutée). Dans ce cas, on parcourt l'ensemble des brins grisés $z_i, i \in 1..n$ superposés à z en suivant sa β-orbite jusqu'à revenir à z lui-même. Pour chacun de ces brins grisés z_i, on définit un sommet (rouge) S_i superposé à S. Soit $t_i = \alpha_{0_r}(z_i)$. On définit deux nouveaux brins roses a_i et b_i plongés en noir et en rouge sur S_i, puis les liaisons rouges de ces nouveaux brins :
$l_{0_r}(a_i,z_i)$, $l_{0_r}(z_i,a_i)$, $l_{0_r}(b_i,t_i)$ et $l_{0_r}(t_i,b_i)$;
$l_{1_r}(a_i,b_i)$ et $l_{1_r}(b_i,a_i)$.
On définit également les liaisons noires de ces nouveaux brins roses :
$l_0(a_i,\alpha_0(t_i))$ et $l_0(b_i,\alpha_0(z_i))$;
$l_1(a_i,b_i)$ et $l_1(b_i,a_i)$.
Enfin, on insère ces deux nouveaux brins roses dans la β-orbite des brins rouges a et b créés ci-dessus. On pose $a_0 = a$ et $b_0 = b$. Alors pour tout $i \geq 1$, $l_\beta(a_i,\beta(a_{i-1}))$ et $l_\beta(a_{i-1},a_i)$. De même, pour tout $i \geq 1$, $l_\beta(b_i,\beta(b_{i-1}))$ et $l_\beta(b_{i-1},b_i)$.

7.3.4 Règle 4

Condition de déclenchement : les arêtes $(x,y)_r$ et $(z,t)_r$ de la carte rouge s'intersectent.

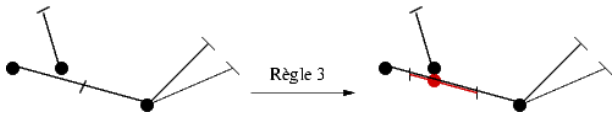


FIG. *7.8: Règle 3 du raffinement coloré*

Résultat : création d'un nouveau sommet rouge sur l'intersection calculée et des brins rouges associés sur l'arête (z,t). Plus précisément :

* Création de deux sommets (rouges) S et S' superposés sur l'intersection calculée.

* Création de deux brins rouges a et b plongés en noir et en rouge sur S et de deux brins rouges a' et b' plongés en noir et en rouge sur S'. On note que si $(x,y)_r$ est une arête bleutée, alors a et b sont en fait des brins violets. De même, si $(z,t)_r$ est une arête bleutée, alors a' et b' sont en fait des brins violets.

* Définition des liaisons rouges des nouveaux brins rouges (ou violets) :
$l_{0_r}(a,z)$, $l_{0_r}(z,a)$, $l_{0_r}(b,t)$ et $l_{0_r}(t,b)$;
$l_{1_r}(a,b)$ et $l_{1_r}(b,a)$.
$l_{0_r}(a',x)$, $l_{0_r}(x,a')$, $l_{0_r}(b',y)$ et $l_{0_r}(y,b')$;
$l_{1_r}(a',b')$ et $l_{1_r}(b',a')$.

* Définition des liaisons noires des nouveaux brins rouges (ou violets) :
$l_0(a,\alpha_0(y))$ et $l_0(b,\alpha_0(x))$;
$l_1(a,b$ et $\alpha_1(b,a)$.
$l_0(a',\alpha_0(t))$ et $l_0(b',\alpha_0(z))$;
$l_1(a',b')$ et $l_1(b',a')$.

* Définition des liaisons β des nouveaux brins rouges (ou violets) : $l_\beta(a,a)$ et $l_\beta(b,b)$ $l_\beta(a',a')$ et $l_\beta(b',b')$

* Gestion des arêtes grisées éventuellement superposées à l'arête $(x,y)_r$: il s'agit de découper ces arêtes de la même manière que $(x,y)_r$.
Si $\beta(x)$ est différent de x, cela signifie que l'arête $(x,y)_r$ est superposée à au moins une arête grisée ($(x,y)_r$ est une arête bleue). Dans ce cas, on parcourt l'ensemble de ces brins grisés $x_i, i \in 1..n$ superposés à x en suivant sa β-orbite jusqu'à revenir à x lui-même. Pour chacun des brins grisés x_i, on définit un sommet (rouge) S_i superposé à S. Soit $y_i = \alpha_{0_r}(x_i)$. On définit deux nouveaux brins roses a_i et b_i plongés en noir et en rouge sur S_i, puis les liaisons rouges de ces nouveaux brins :
$l_{0_r}(a_i,x_i)$, $l_{0_r}(x_i,a_i)$, $l_{0_r}(b_i,y_i)$ et $l_{0_r}(y_i,b_i)$;
$l_{1_r}(a_i,b_i)$ et $l_{1_r}(b_i,a_i)$.
On définit également les liaisons noires de ces nouveaux brins roses :
$l_0(a_i,\alpha_0(y_i))$ et $l_0(b_i,\alpha_0(x_i))$;
$l_1(a_i,b_i)$ et $l_1(b_i,a_i)$.
Enfin, on insère ces deux nouveaux brins roses dans la β-orbite des brins rouges a et b créés ci-dessus. On pose $a_0 = a$ et $b_0 = b$. Alors pour tout $i \geq 1$, $l_\beta(a_i,\beta(a_{i-1}))$ et $l_\beta(a_{i-1},a_i)$. De même, pour tout $i \geq 1$, $l_\beta(b_i,\beta(b_{i-1}))$ et $l_\beta(b_{i-1},b_i)$.

* Gestion des arêtes grisées éventuellement superposées à l'arête $(z,t)_r$: il s'agit de découper ces arêtes de la même manière que $(z,t)_r$. Il suffit de transposer à $(z,t)_r$ les règles ci-dessus, énoncées pour

$(x,y)_r$. Ainsi, si $\beta(z)$ est différent de z, cela signifie que l'arête $(z,t)_r$ est superposée à au moins une arête grisée ($(z,t)_r$ est une arête bleue). Dans ce cas, on parcourt l'ensemble de ces brins grisés $z_i, i \in 1..n$ superposés à z en suivant sa β-orbite jusqu'à revenir à z lui-même. Pour chacun des brins grisés z_i, on définit un sommet (rouge) S'_i superposé à S'. Soit $t_i = \alpha_{0_r}(z_i)$. On définit deux nouveaux brins roses a'_i et b'_i plongés en noir et en rouge sur S'_i, puis les liaisons rouges de ces nouveaux brins :
$l_{0_r}(a'_i, z_i)$, $l_{0_r}(z_i, a'_i)$, $l_{0_r}(b'_i, t_i)$ et $l_{0_r}(t_i, b'_i)$;
$l_{1_r}(a'_i, b'_i)$ et $l_{1_r}(b'_i, a'_i)$.
On définit également les liaisons noires de ces nouveaux brins roses :
$l_0(a'_i, \alpha_0(t_i))$ et $l_0(b'_i, \alpha_0(z_i))$;
$l_1(a'_i, b'_i)$ et $l_1(b'_i, a'_i)$.
Enfin, on insère ces deux nouveaux brins roses dans la β-orbite des brins rouges a' et b' créés ci-dessus. On pose $a'_0 = a'$ et $b'_0 = b'$. Alors pour tout $i \geq 1$, $l_\beta(a'_i, \beta(a'_{i-1}))$ et $l_\beta(a'_{i-1}, a'_i)$. De même, pour tout $i \geq 1$, $l_\beta(b'_i, \beta(b'_{i-1}))$ et $l_\beta(b'_{i-1}, b'_i)$.

FIG. *7.9: Règle 4 du raffinement coloré*

7.3.5 Règle 5

Condition de déclenchement : les plongements des brins x et y sont superposés tandis que x et y ne se trouvent pas dans la même 1-orbite rouge.

Résultat : on fusionne ces sommets dans la carte rouge. Plus précisément :

* Le plongement rouge du brin y et de chacun de ses brins 1-liés par liaison rouge devient le plongement rouge du brin x.

* Les brins de la 1-orbite rouge du brin y sont insérés dans la 1-orbite rouge du brin x : pour cela les brins des 1-orbites rouges des deux brins sont ordonnés dans le sens trigonométrique (de l'orientation de leur 1-plongement) et les 1-liaisons rouges sont redéfinies dans cet ordre trigonométrique.

FIG. *7.10: Règle 5 du raffinement coloré*

7.3.6 Règle 6

Condition de déclenchement : l'ordre des brins dans la 1-orbite rouge de x ne correspond pas à l'ordre trigonométrique sur leur 1-plongement.

Résultat : on réordonne la 1-orbite rouge de ces brins dans la carte rouge. Plus précisément :

* Les 1-liaisons des brins de la 1-orbite rouge du brin x sont réordonnées : pour cela les brins de la 1-orbite rouge de x sont ordonnés dans le sens trigonométrique (de l'orientation de leur 1-plongement) et les 1-liaisons rouges sont redéfinies dans cet ordre trigonométrique.

FIG. *7.11: Règle 6 du raffinement coloré*

7.3.7 Complétion de labels sur les cartes colorées

A priori, la complétion de labels associée au raffinement d'une carte combinatoire colorée concerne essentiellement les labels « rouges » (en particulier les degrés d'occupation rouges), c'est-à-dire les labels associés à chaque brin de la carte rouge bien raffinée. Cependant, lors du découpage provisoire des arêtes de la carte noire (par les règles 3 et 4), nous verrons qu'il peut être utile de transférer les labels noirs de l'arête noire initiale vers les petites arêtes résultantes. Nous nous intéressons ici aux 1-labels et 2-labels noirs, ainsi qu'aux 2-labels rouges. Les règles de complétion des cartes colorées sont largement inspirées de celles de la complétion de labels des cartes combinatoires monochromes classiques.

Ainsi, on ajoute les opérations suivantes lors de l'application des règles de raffinement coloré :
- Règle 1 : pas de modification.
- Règle 2 : ajout des labels rouges des brins grisés aux brins bleutés β-liés. En revanche, les labels noirs ne sont pas modifiés.
- Règle 3 : transfert des labels rouges et noirs des brins de l'arête $(z,t)_r$ initiale vers les brins des « sous-arêtes » résultantes. Plus précisément, selon les notations des règles de raffinement ci-dessus, transfert des labels de z vers b et de t vers a.
- Règle 4 : de la même manière, transfert des labels rouges et noirs des brins des arêtes initiales vers les brins des « sous-arêtes » résultantes. Plus précisément, selon les notations des règles de raffinement ci-dessus, transfert des labels de x vers b, de y vers a, de z vers b' et de t vers a'.
- Règle 5 : pas de modification.
- Règle 6 : pas de modification.

Ensuite, les labels rouges peuvent être uniformisés lors du parcours de faces, par application de la fonction « comp » définie chez Cazier sur les brins adjacents qui ne partagent pas les mêmes labels rouges (cf. Fig. 7.12).

7.3.8 Propriétés et complexité du raffinement de cartes colorées

En principe, l'opération de raffinement coloré hérite de propriétés similaires à celles du raffinement classique [23]. En effet, dans le processus de raffinement coloré, tout se passe comme si l'on raffinait de manière classique la partie rouge de la carte colorée : les conditions de déclenchement s'appliquent de la même manière (les segments grisés deviennent « invisibles » pour le raffinement, même s'ils ne disparaissent pas de la carte) et les règles produisent les mêmes effets sur la carte rouge qu'un raffinement classique. On vérifie donc ainsi la terminaison de cette opération. En ce qui concerne la confluence, comme dans le cas du raffinement monochrome, il semble que le résultat de l'application d'un nombre quelconque de règles ne dépende *quasiment* pas de l'ordre dans lequel ces règles ont été appliquées : a

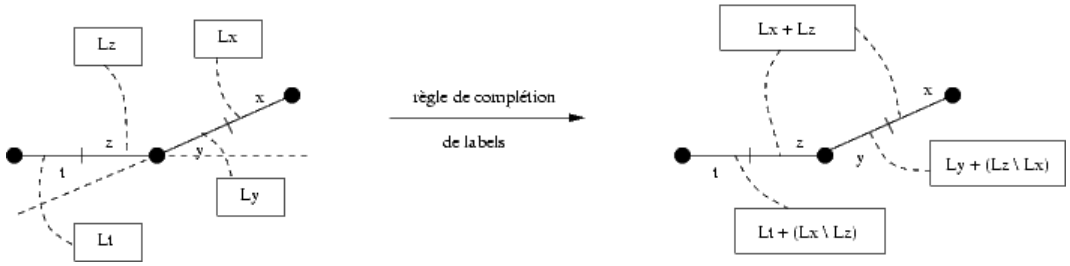


FIG. *7.12: Illustration de la règle de complétion de labels définie par Cazier [23]. Les labels sont indiqués dans les cases rectangulaires*

priori, seuls les plongements rouges (résultant de l'application de la règle 5) et les β-orbites pourraient varier (ces β-orbites pourraient se trouver dans un ordre différent). Une spécification plus formelle des cartes colorées et du raffinement permettraient toutefois de valider ces suppositions.

Cependant, si nos suppositions s'avèrent exactes, ces légères variations par rapport au cas monochrome ne nous gênent pas pour le choix d'un algorithme efficace de raffinement (nous gardons le raffinement par balayage proposé par Cazier [23]) et pour la mise en œuvre des opérations exploitant ce raffinement (ces opérations devraient donner des résultats similaires - aux mêmes variations près sur les β-orbites et les plongements rouges - quel que soit l'ordre d'application des règles).

Enfin, concernant la complexité du raffinement coloré par balayage, celui-ci demeure globalement en $O((n+i)\log n)$, n étant le nombre de brins foncés initiaux et i le nombre de brins foncés créés (proportionnel au nombre d'intersections et d'incidences). En effet, on se ramène une fois de plus au raffinement monochrome de la carte rouge : les règles sont appliquées de la même manière et dans le même nombre que dans le cas classique. Seules les opérations élémentaires réalisées dans les règles deviennent légèrement plus complexes, du fait de l'apparition de brins grisés notamment (il faut fusionner des β-orbites dans la règle 2 et découper les arêtes grisées superposées aux arêtes courantes dans les règles 3 et 4). En particulier, le coût d'application des règles 3 et 4 dépend de la taille des β-orbites. Toutefois, même dans le cas monochrome, le coût de certaines règles dépend de la configuration de la carte : le coût de la règle 5, qui effectue un tri sur les brins, dépend du nombre de brins dans les 1-orbites des brins considérés (ce coût est cependant borné dans le cas des cartes discrètes puisque les 1-orbites contiennent au maximum 4 éléments). Le calcul de complexité ne prend pas cela en compte puisqu'il reste au niveau des règles.

7.4 Un algorithme « naïf » pour la mise à jour

On suppose que l'on dispose à l'instant t d'une carte globale colorée bien raffinée de l'environnement et d'un nouveau polygone d'espace libre (sous forme de carte combinatoire monochrome, avec un degré d'occupation égal à 1 sur la face interne). Dans un premier temps, on pourrait imaginer l'algorithme naïf suivant pour la mise à jour de cette carte globale, suite aux opérations de mise en correspondance et de filtrage de Kalman (cf. Fig. 7.13) :

– On ajoute le nouveau polygone local dans la « partie noire » de la représentation (c'est-à-dire la carte noire au sein de la carte colorée). Pour cela, il faut d'abord définir les fusions d'arêtes et de sommets noirs, ainsi que l'ajout de sommets noirs sur des arêtes existantes, qui découlent de la mise en correspondance de segments obstacles entre cartes globale et polygone local. Cette opération est

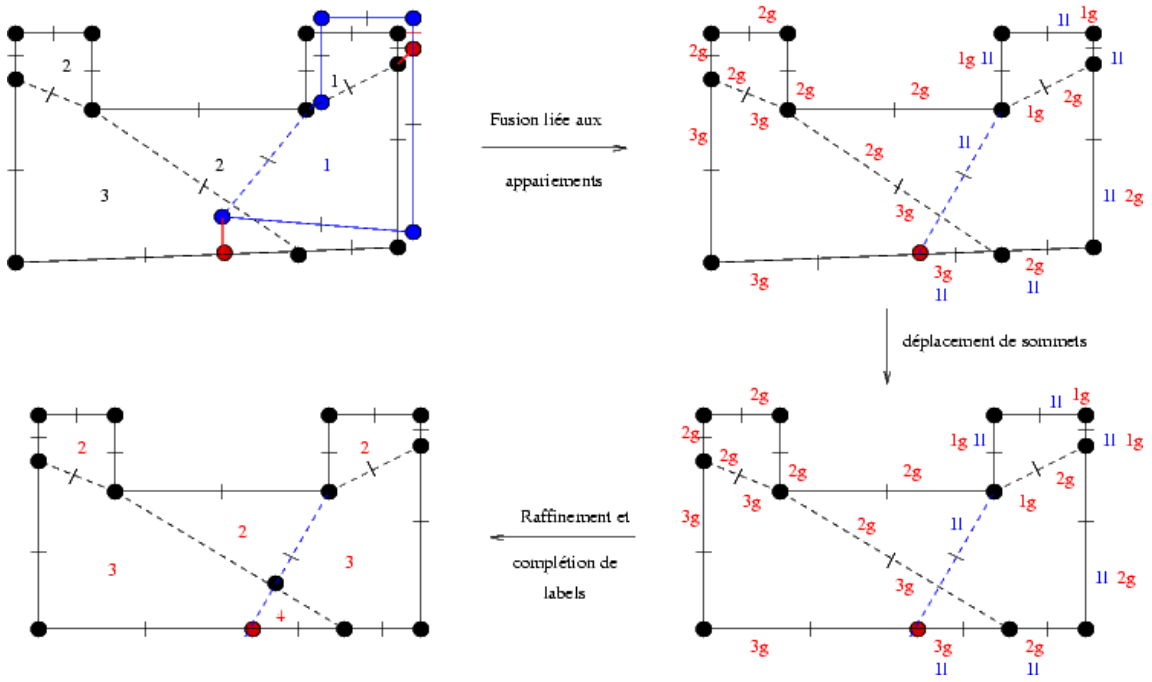


FIG. *7.13: Illustration de l'algorithme « naïf » de mise à jour : sur ce schéma, la carte locale est représentée en bleu et les cassures en rouge. Les nombres en rouge suivis de la lettre « g » correspondent aux degrés d'occupation dans la carte globale et les nombres en bleu suivis de la lettre « l » aux degrés d'occupation de la carte locale.*

réalisée à partir du chemin d'appariement entre carte locale et globale qui permet de connaître les arêtes et sommets appariés, ainsi que les cassures de segments globaux (appariements de sommets virtuels de la carte globale à des sommets locaux) et les cassures de segments locaux (observation d'un sommet local sur un segment global). Lors de cette opération, on transfère également le degré d'occupation du polygone local (égal à 1) sur les brins noirs fusionnés. Il faut aussi ajouter les arêtes non obstacles et les arêtes nouvellement observées (non appariées) du polygone local. Toutes ces modifications interviennent en réalité dans la couche topologique de la carte colorée. En particulier, les points virtuels qui sont transformés en points réels ont dans un premier temps un plongement arbitraire, situé au milieu du segment auquel ils sont rattachés par exemple.

– On déplace les sommets suivant leur nouvelle position calculée par filtrage de Kalman. A priori, il suffit pour cela de modifier leur plongement : on effectue par ce biais la mise à jour géométrique de la carte. Nous verrons cependant que cette opération est en réalité plus complexe.

– On effectue l'auto-raffinement de cette carte colorée, selon les règles définies dans la section précédente. Une complétion de labels permet en outre le calcul des nouveaux degrés d'occupation rouges.

Au premier abord, cette stratégie « naïve » peut paraître applicable. En pratique, comme nous allons le voir, la mise à jour géométrique risque cependant de générer des incohérences dans le calcul des degrés d'occupation.

7.4.1 Incohérences liées à des retournements et croisements de polygones

En fait, la stratégie « naïve » peut fonctionner si l'on est certain que le déplacement de sommets n'entraîne pas d'incohérences dans la structure des polygones fusionnés. Pourtant, le déplacement des sommets est régi par le processus d'estimation par filtrage de Kalman. A la différence des sommets de maillages physiques, qui obéissent à des contraintes mécaniques (ce qui peut empêcher les faces de se retourner ou de devenir des quadrilatères croisés), les positions des sommets d'une carte stochastique sont estimées par filtrage bayésien, avec des liens entre sommets qui correspondent à des corrélations. Ainsi, même si une observation « parfaite » de l'espace libre ne pourrait mener à de telles configurations, les erreurs de mesure et d'estimation (erreurs d'arrondis, problèmes de linéarisation, etc.) peuvent générer des incohérences liées à des retournements de faces ou à des faces croisées telles que dans les exemples fournis sur la figure 7.14. En conséquence, les degrés d'occupation deviennent difficiles à définir pour un polygone croisé ou retourné, et à plus forte raison pour une carte constituée d'une superposition de ces polygones incohérents.

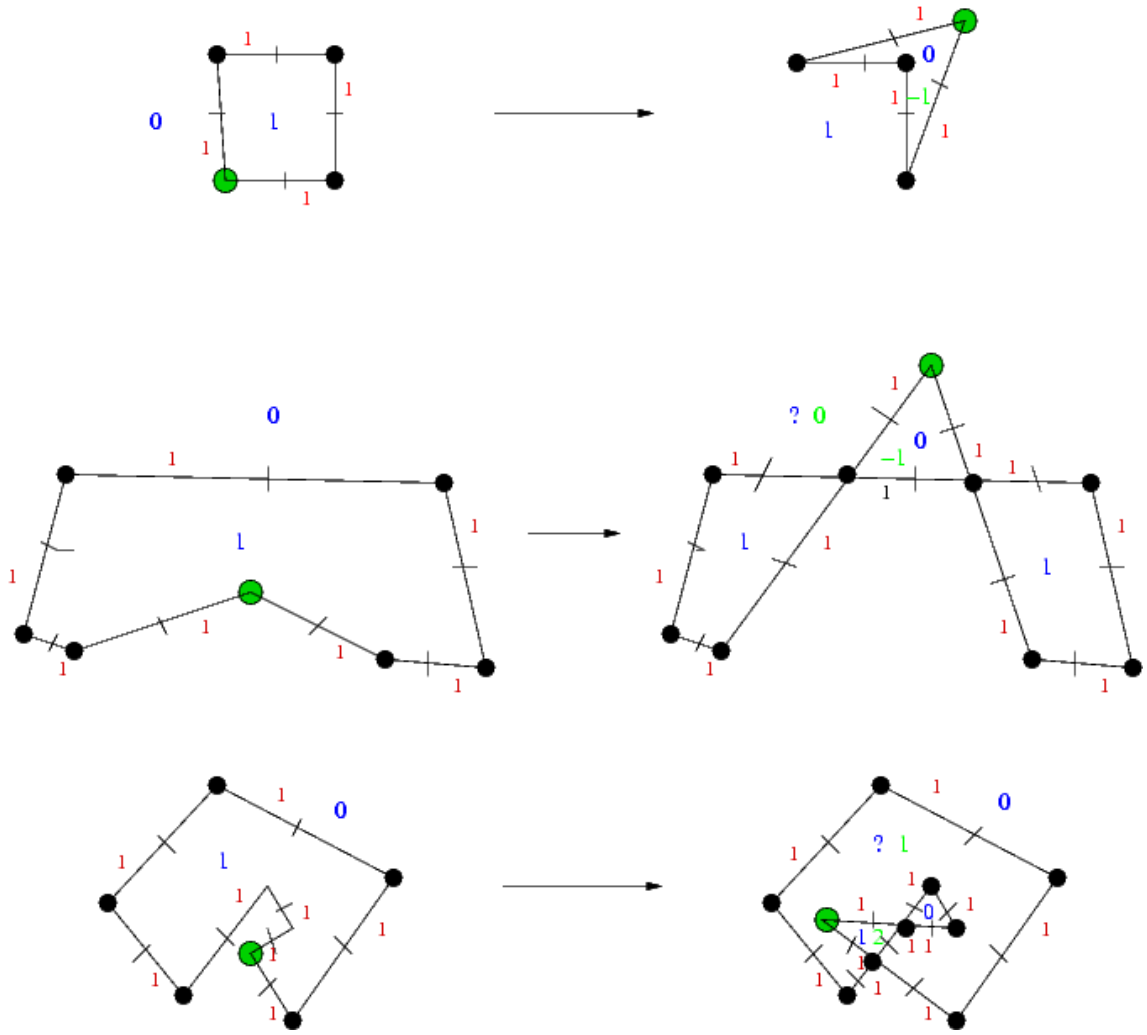


FIG. *7.14: Exemples d'incohérences liées au déplacement d'un sommet (ici le sommet vert) : retournements et croisements de polygones. Les nombres en rouge représentent les 2-labels de brins correspondant aux degrés d'occupation rouges (pour plus de clarté, les degrés nuls sont omis). Les nombres bleus représentent les degrés d'occupations déduits du parcours de faces. Les numéros verts correspondent à la convention choisie à la section 7.5.*

7.5 Gestion du déplacement de sommets dans un polygone d'espace libre

Avant d'étudier le déplacement de sommets dans les cartes colorées, nous nous concentrons sur le déplacement de sommets dans un unique polygone, dont la face interne présente un degré d'occupation unitaire. Très globalement, nous partons du principe que le déplacement d'un sommet vers l'extérieur du polygone revient à incrémenter d'une unité la zone supplémentaire couverte par ce polygone (cf. Fig. 7.15). On parle alors de « zone d'ajout ». Inversement, le déplacement d'un sommet vers l'intérieur du polygone revient à décrémenter d'une unité l'ancienne zone couverte par ce polygone : on parle alors de « zone de retrait ». On obtient en fait une carte identique à celle que l'on aurait pu créer si l'on avait directement observé le polygone avec le sommet au nouvel endroit. Cela revient également à adopter une convention qui mène par exemple aux degrés d'occupation en vert dans les cas d'incohérence (cf. Fig. 7.14). Notons que dans certains cas, cela revient à adopter des degrés d'occupation négatifs : comme nous le verrons plus tard, ces degrés d'occupation négatifs sont toutefois réversibles, et peuvent disparaître ou réapparaître au gré des déplacements de sommets successifs.

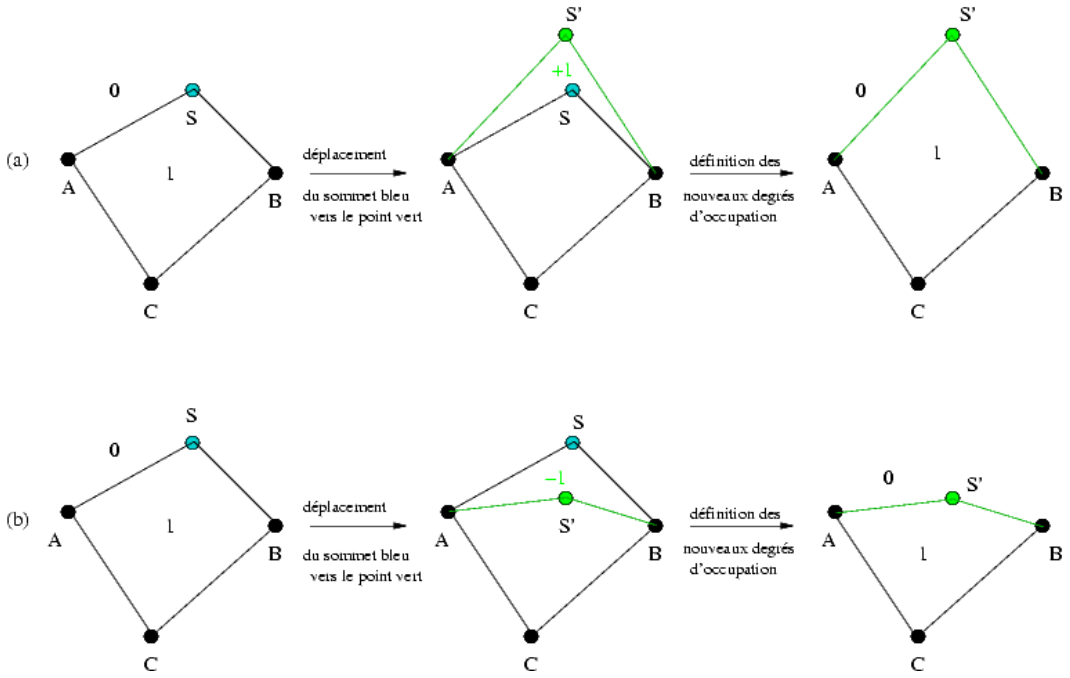


FIG. *7.15: Exemples d'application des règles de mise à jour des degrés d'occupation associées au déplacement de sommets au sein du polygone ACBS. (a) Déplacement d'un sommet vers l'extérieur du polygone avec création d'une « zone d'ajout » de degré d'occupation +1 (quadrilatère ASBS′). (b) Déplacement d'un sommet vers l'intérieur du polygone avec création d'une « zone de retrait » de degré d'occupation −1 (quadrilatère AS′BS).*

En pratique, l'application de cette convention n'est pas immédiate : dans le cas d'un déplacement du sommet vers l'extérieur du polygone, la zone d'ajout peut recouper d'autres arêtes de ce même polygone (cf. Fig. 7.14). Ces recoupements peuvent aussi exister dans le cas d'un déplacement vers l'intérieur (cf. Fig. 7.14). Ainsi, pour calculer les intersections éventuelles de ces zones d'ajout ou de retrait avec le polygone, nous choisissons d'utiliser des « mini-cartes combinatoires » qui les matérialisent. Les intersections et les nouveaux degrés d'occupation peuvent alors être calculés par raffinement et par complétion

de labels, ce qui permet de gérer les cas « pathologiques ».

Pour définir plus en détail l'opération de déplacement de sommet, on fait l'hypothèse de départ que le polygone initial est bien plongé et correct (c'est-à-dire ni croisé, ni retourné). En outre, ses degrés d'occupation noirs sont bien définis (1 pour la face interne, 0 pour la face externe). On transforme ce polygone en carte colorée selon la procédure définie plus haut : tous les brins restent noirs et toutes les liaisons rouges sont déduites des liaisons noires, de même que les degrés d'occupation rouges sont initialisés à la valeur des degrés noirs. En principe, par construction, toute carte locale est un polygone d'espace libre étoilé (dont tous les sommets peuvent être vus depuis un point intérieur au polygone, en l'occurrence la position du robot au moment de son acquisition), qui vérifie l'hypothèse. Toutefois, si pour une raison quelconque cette hypothèse n'était pas vérifiée (par exemple en raison d'approximations numériques), on peut générer un polygone plongé sur de nouveaux points en commençant par projeter à intervalle régulier sa séquence de sommets (définie par les liaisons noires) sur un cercle de grand diamètre (pour éviter tout risque de superpositions à epsilon près, ou à une cellule près dans cas discret qui sera détaillé plus loin). Puis on ramène chaque sommet à sa position réelle via l'opération de déplacement définie ci-dessous (cf. Fig. 7.16).

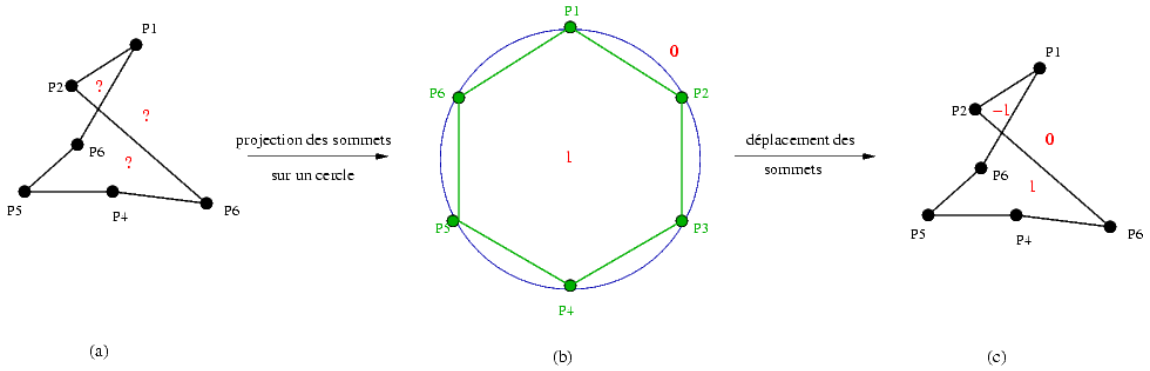


FIG. *7.16: Opération de projection d'un polygone sur un cercle pour assurer son bon plongement. (a) Le polygone initial est croisé, ce qui rend difficile la détermination des degrés d'occupation de la carte combinatoire correspondante. (b) Ce polygone est projeté sur un cercle, ce qui permet de d'initialiser correctement les degrés d'occupation. (c) Chaque sommet est ensuite déplacé au moyen d'une opération « sûre » (selon la procédure définie ci-dessous, qui garantit le calcul des degrés d'occupation selon la convention que nous avons choisie) vers sa position initiale.*

7.5.1 Définition d'une mini-carte combinatoire

Comme nous l'avons vu dans la figure 7.15, la mini-carte combinatoire créée pour le déplacement d'un sommet correspond en général à un quadrilatère (quadrilatère « d'ajout » ou « de retrait »). Il existe cependant des cas où ce quadrilatère est croisé ou des cas où il est dégénéré (cf. Fig. 7.17). En fait, on peut voir la mini-carte comme composée de deux triangles élémentaires : un triangle par arête déplacée (donc un triangle pour chacun des deux sommets $A_i, i \in \{1, 2\}$ adjacents au sommet Sa déplacé vers la nouvelle position Sn). Les figures 7.18 et 7.19 fournissent toutes les configurations possibles pour les mini-cartes.

Par définition, les sommets du triangle associé au sommet A_i sont l'ancien sommet S_a, le nouveau sommet S_n et le sommet A_i lui-même. En fait, on crée pour ce triangle de nouveaux brins superposés à ceux du polygone : il s'agit bien de copies. En revanche, les plongements noirs des sommets S_a et A_i des

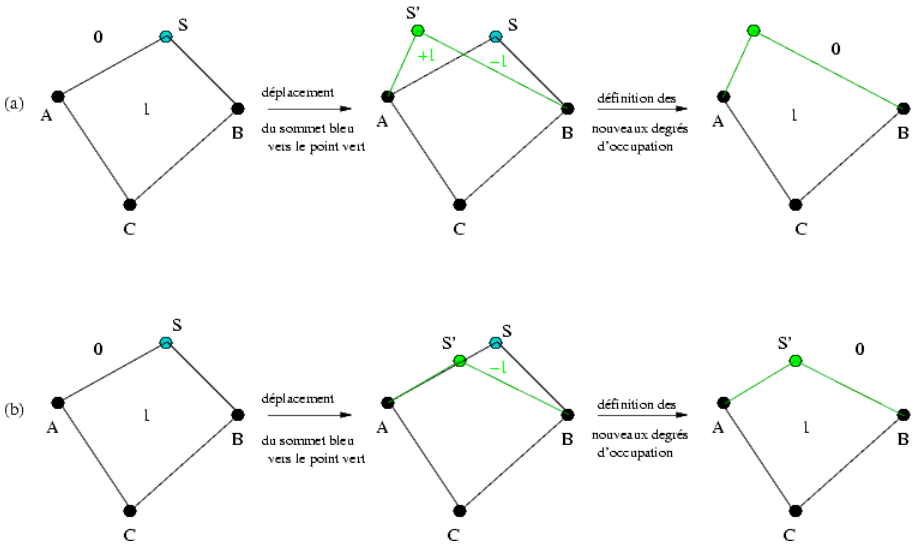


FIG. *7.17: Exemples de déplacement de sommet menant à une mini-carte composée d'un polygone $ASBS'$ croisé (a) ou dégénéré (b).*

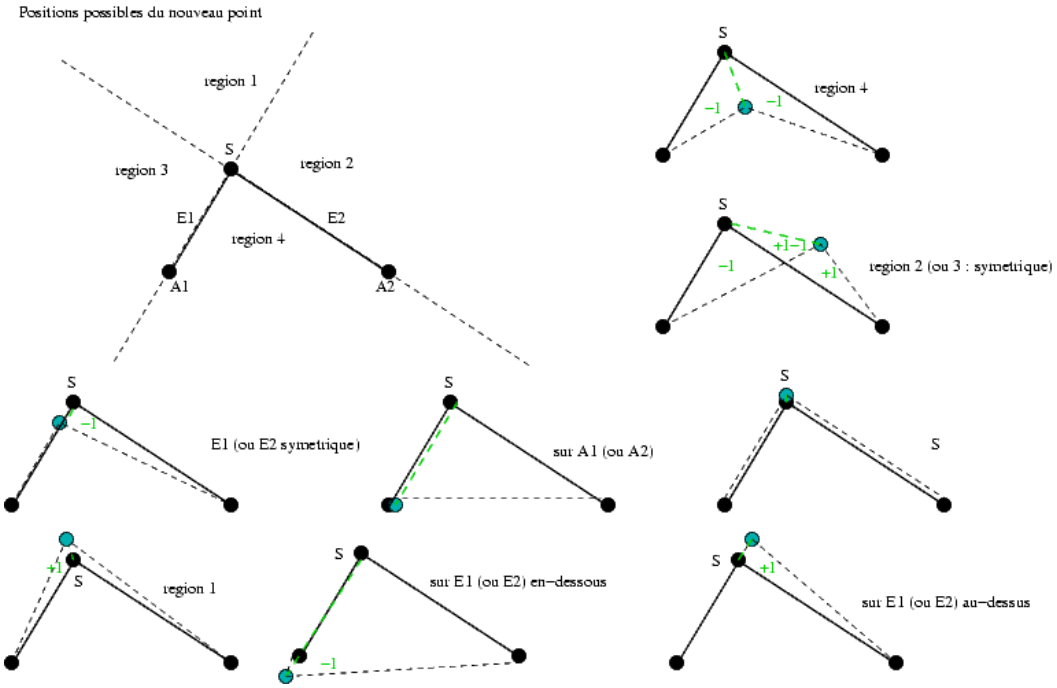


FIG. *7.18: Dénombrement des figures associées au déplacement de sommets, dans le cas où A_1, A_2 et S ne sont ni alignés, ni superposés. Ici, les pointillés noirs ne correspondent pas à des segments « non obstacles » mais indiquent les nouvelles arêtes A_1S et A_2S. Quant aux pointillés verts, ils correspondent aux segments provisoires qui relient l'ancien et le nouveau sommet.*

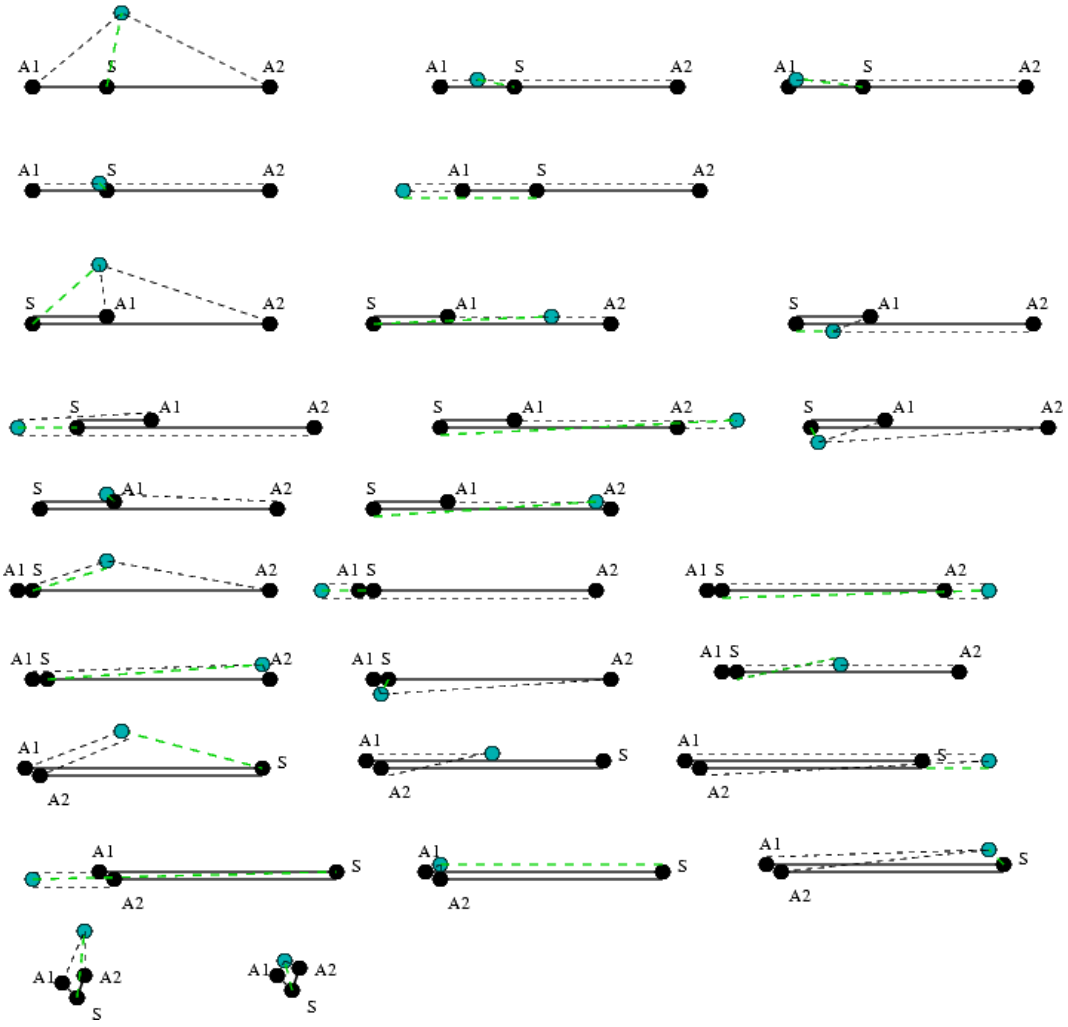


FIG. *7.19: Dénombrement des figures associées au déplacement de sommets dans les cas dégénérés où A_1, A_2 et S sont alignés ou superposés.*

triangles sont identiques à ceux du polygone. Quant aux arêtes, elles sont composées de l'ancienne arête A_iS_a (dont l'originale est à supprimer dans le polygone), d'une nouvelle arête A_iS_n (dont l'originale est à ajouter dans ce polygone) et d'une arête provisoire S_aS_n qui deviendra inutile et pourra être supprimée une fois la mini-carte complètement constituée par fusion des deux triangles. Les brins créés pour ce triangle sont reliés entre eux par 0- et 1-liaisons rouges. Les deux brins d'une même arête sont en outre reliés par 0-liaison noire. Les degrés d'occupation des faces de la mini-carte dont déduits de la superposition (raffinement et complétion de labels) des deux triangles. La face interne du triangle contient un degré d'occupation positif (+1) si A_iS_n se trouve dans le demi-plan délimité par la droite A_iS_a qui correspond à la face externe du polygone (d'après les degrés d'occupation noirs). Ce degré d'occupation est négatif (-1) dans le cas contraire, et indifférent (nul par convention) dans le cas où A_iS_n est superposé à la droite A_iS_a.

Chaque triangle est auto-raffiné avec les règles de raffinement coloré, de manière à traiter les éventuels cas pathologiques. Ensuite, les deux triangles sont superposés et raffinés avec complétion de label. A leur création, les brins de l'arête S_aS_n se voient en outre attribuer le 1-label « provisoire » et les brins des arêtes A_iS_a portent le 1-label « à supprimer » (cf. Fig. 7.20) : ces labels sont transmis aux sous-brins éventuels issus d'un découpage (par les règles 3 et 4 du raffinement). De même, les brins du sommet S_a portent le 0-label « à supprimer ». De cette manière, il devient possible de supprimer l'arête S_aS_n une fois la mini-carte construite, puis de supprimer les anciennes arêtes A_iS_a ainsi que le sommet S_a une fois la fusion de la mini-carte et du polygone réalisée. Cette opération de suppression est décrite dans la prochaine section. Par ailleurs, les sommets S_a, S_n et A_i sont plongés en noir sur les mêmes points (mêmes objets physiques) dans les deux triangles et dans le polygone, ce qui génère une fusion véritable entre ces sommets lors de l'application de la règle 5 de raffinement : exceptionnellement, les plongements noirs seront identiques, comme les plongements rouges.

Une fois la mini-carte constituée par fusion des deux triangles élémentaires et suppression de l'arête S_aS_n, il importe de définir ses 1-liaisons noires. Pour cela, chacun des quatre sommets noirs de la mini-carte finale constitue une orbite de α_1, qui comprend les deux brins plongés sur ce sommet. Par ailleurs, avant de raffiner la superposition de la mini-carte et du polygone, le polygone lui-même subit quelques modifications : ses arêtes A_iS_a et son sommet S_a sont affectés du même label « à supprimer » que dans les mini-cartes. De plus, une fois ce raffinement coloré achevé, les 1-liaisons noires du polygone sont modifiées : dans la 1-orbite noire de chaque sommet A_i, on remplace le brin de l'arête A_iS_a (du polygone) par celui de l'arête A_iS_n (issu de la mini-carte).

Enfin, les règles de complétion de labels intervenant dans chacun de ces raffinements sont un peu différentes. Lors de l'autoraffinement des triangles, il existe des cas dégénérés où le triangle est aplati. Dans ce cas, de par les règles définies ci-dessus, les deux brins d'une même arête porteront chacun deux labels : « 0 » et « +1 » (ou « -1 »). Il faut donc se ramener à un label unique : « 0 ». Lors de la fusion des deux triangles pour construire la mini-carte combinatoire, les cas dégénérés ayant été traités au niveau des triangles, il suffit de réaliser une complétion de labels normale et d'additionner ensuite les labels issus des deux triangles pour les cellules communes. De même, lorsque l'on fusionne la mini-carte avec le polygone, on réalise une complétion de labels classique, puis on additionne à la fin les labels issus des deux cartes pour les cellules concernées. On note que le nombre de cas possibles étant limité pour les mini-cartes combinatoires (cf. Fig. 7.18 et 7.19), on peut les vérifier directement.

7.5.2 Règles de suppression d'arêtes et de sommets

Avant de détailler les règles de suppression d'arêtes et de sommets, nous décrivons quelques opérations particulières qui nous serviront dans l'algorithme global : la mise à jour de la 1-orbite rouge lors de la suppression d'un brin et la recherche d'éventuels sommets rouges à éliminer lors de la suppression d'un

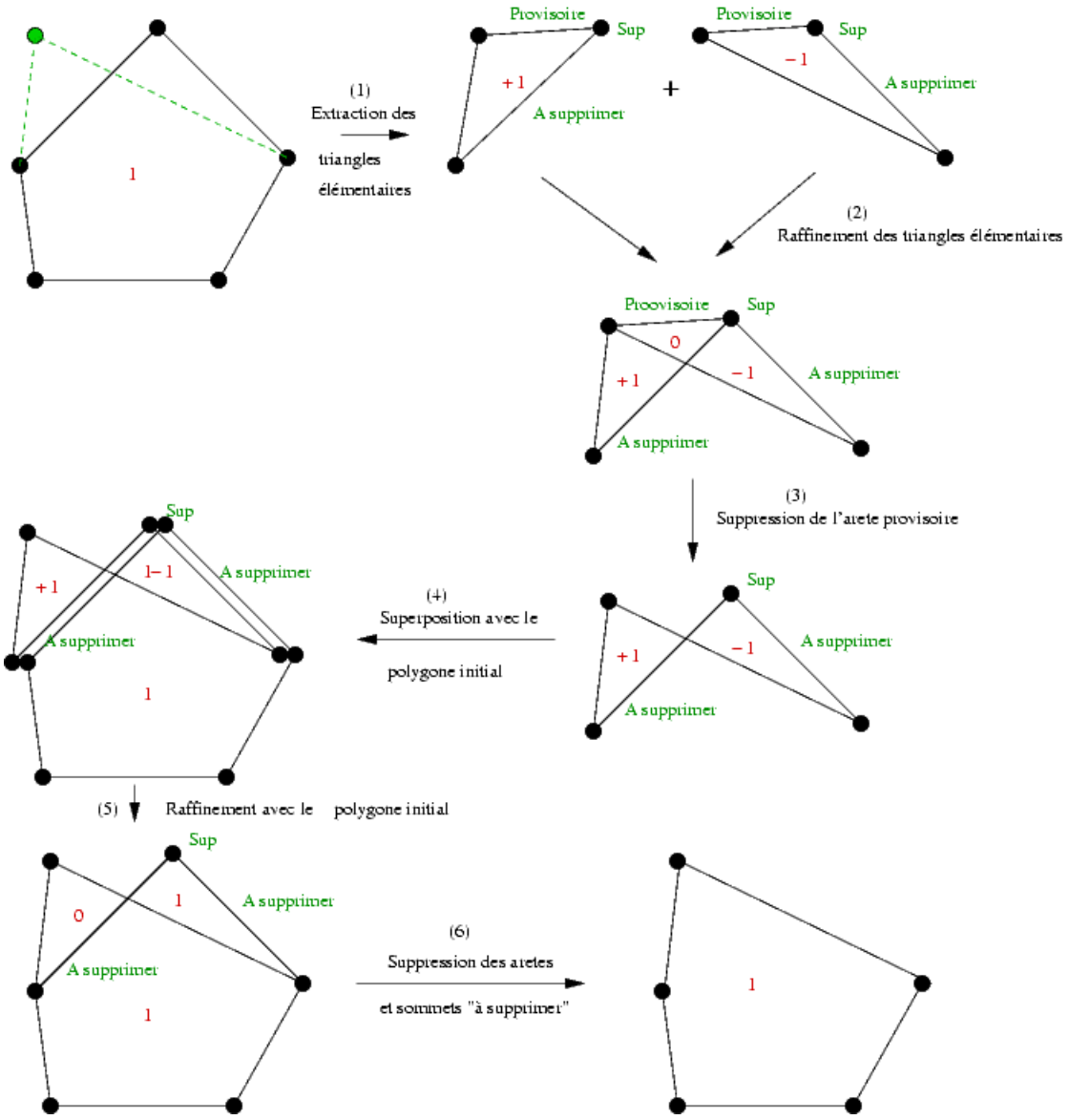


FIG. *7.20: Illustration de l'opération de déplacement de sommet.*

sommet. Nous nous intéressons également à un cas particulier plus simple que le cas général (cas d'une arête à supprimer de longueur nulle).

Mise à jour de la 1-orbite rouge lors de la suppression d'un brin

Lorsque l'on supprime le brin x, pour mettre à jour sa 1-orbite rouge, on procède de la manière suivante :

- si x est grisé : il n'y a rien à faire puisque ce brin est son propre successeur par α_{1_r} ;
- si x est bleuté : on remplace x par $\beta(x)$ dans sa 1-orbite rouge ;
- si x est son propre successeur par β (ni grisé, ni bleuté), on le supprime de sa 1-orbite rouge.

Recherche de sommets rouges à éliminer lors de la suppression d'un sommet

Il arrive que le plongement des sommets étiquetés « à supprimer » se situe sur d'autres éléments de la carte : en particulier, s'ils sont superposés à une autre arête, l'application de la règle 3 de raffinement a généré sur cette arête un nouveau sommet rouge (cf. Fig. 7.21, configuration (a)). Après l'effacement du sommet « à supprimer », ce nouveau sommet rouge n'a en principe plus lieu d'être, sauf si un troisième sommet de la carte se trouve à cet endroit. Il faut donc le supprimer, ainsi que les brins rouges de son 1-orbite rouge. De même, il arrive que les arêtes étiquetées « à supprimer » aient généré des incidences ou des intersections (application des règles 3 et 4 de raffinement) avec d'autres arêtes. Dans le cas d'une intersection avec une autre arête, deux nouveaux sommets rouges ont été créés : s'ils ne sont pas superposés à un troisième sommet de la carte, ils doivent tous deux être supprimés, ainsi que les brins rouges de son 1-orbite rouge (cf. Fig. 7.21, configuration (b)).

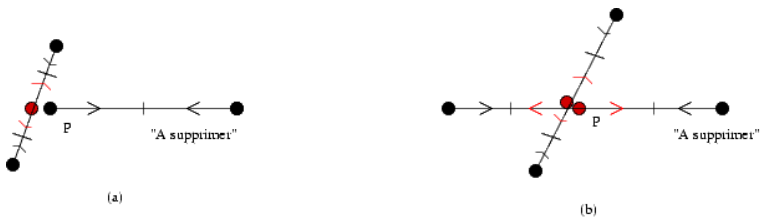


FIG. *7.21: Sommets rouges à éliminer lors de la suppression d'un sommet P sur une arête « à supprimer ». (a) Cas d'une arête incidente à un sommet P étiqueté « à supprimer ». (b) Cas d'une arête intersectant l'arête à supprimer au niveau du sommet P.*

En fait, on se rend compte qu'un sommet rouge S_r superposé avec le sommet à supprimer S_{sup} doit également être éliminé s'il est l'unique autre sommet superposé avec S_{sup} (à l'exception éventuelle d'autres sommets rouges superposés qui se trouveraient dans leurs β-orbites respectives) : quelques exemples de sommets à supprimer sont fournis dans la figure 7.22. Précisons par ailleurs que dans l'algorithme complet présenté ci-dessous, un sommet S_{sup} n'est supprimé que si son 1-orbite noire ne contient plus qu'un seul brin b_{sup}, lui-même en cours de suppression.

Ainsi, lorsque l'on supprime un sommet S_{sup}, il faut parcourir les brins de la β-orbite et de la 1-orbite rouge de b_{sup} à l'exclusion de b_{sup} lui-même (les brins traités sont marqués par un label spécial) :

- Si l'on trouve un brin noir, gris ou bleu (c'est-à-dire un sommet noir superposé à S_{sup}), alors on peut arrêter la vérification car il n'y a pas lieu de supprimer de sommet rouge ;
- Si l'on trouve un brin x rouge, rose ou violet non traité, alors les deux brins de sa 1-orbite noire sont marqués comme traités et l'on retient que l'on a trouvé un sommet rouge qui est potentiellement

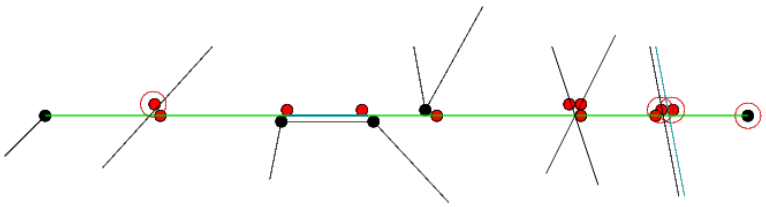


FIG. *7.22: Exemples de sommets à supprimer (cerclés en rouge) lors du parcours de l'arête « à supprimer » (en vert), qui contient une sous-arête grisée.*

à supprimer. Ainsi, si l'on découvre ultérieurement un second brin rouge, rose ou violet non traité, cela signifie qu'un second sommet rouge est superposé à S_{sup} et la vérification n'a plus lieu d'être. Par ailleurs, on parcourt également les brins de la β-orbite de x (lorsque x n'est pas dans la β-orbite de b_{sup}) : si l'on trouve un brin gris (c'est-à-dire un sommet noir superposé à S_{sup}), alors on peut arrêter la vérification qui n'a plus lieu d'être.

Enfin, si l'on n'a pas interrompu la vérification en cours de route et que l'on a trouvé un sommet rouge S_r qui est potentiellement à supprimer, cela signifie qu'il faut vraiment retirer ce sommet S_r de la carte, ainsi que ceux (nécessairement rouges) qui pourraient se trouver sur une arête superposée à celle de S_r. Soient x et y les brins de la 1-orbite noire de S_r (par construction, cette orbite contient exactement deux brins). On effectue alors les opérations suivantes :

- On relie par 0-liaison rouge $\alpha_{0_r}(x)$ et $\alpha_{0_r}(y)$.
- Pour chaque brin z de la β-orbite de x autre que x lui-même, on relie par 0-liaison rouge $\alpha_{0_r}(z)$ et $\alpha_{0_r}(\alpha_{1_r}(z))$.
- On supprime x, y et tous les brins de leur β-orbite respective (si x et y ne sont pas leur propre β-successeur).

En pratique, lorsqu'il existe des arêtes superposées, il faut également gérer les cas où $\alpha_{0_r}(z)$ et $\alpha_{0_r}(\alpha_{1_r}(z))$ ne seraient pas tous deux grisés ou tous deux bleutés. Il s'agit alors de griser et de bleuter les brins adéquats de la β-orbite de $\alpha_{0_r}(\alpha_{1_r}(z))$ de manière à ce que les couleurs de $\alpha_{0_r}(z)$ et $\alpha_{0_r}(\alpha_{1_r}(z))$ soient compatibles.

Cas particulier de la suppression d'une arête grisée de longueur nulle

Nous traitons à part le cas où l'arête à supprimer est une arête de longueur nulle (nécessairement grisée). Une telle arête peut par exemple être générée par un triangle élémentaire dégénéré. Soient x et y les deux brins constitutifs de cette arête. Nécessairement, par construction, x et y sont leurs propres 1-successeurs rouges.

On vérifie d'abord si le sommet noir de x est marqué « à supprimer » et si son 1-orbite noire est bien réduite à x (sinon, comme les autres brins de la 1-orbite noire de x ont un plongement noir identique à celui de x par construction, ce point devra être supprimé ultérieurement, lorsque cette orbite sera réduite à un seul élément en cours de suppression). Si c'est le cas, on peut effectivement supprimer le sommet (et notamment son point de plongement). Pour cela, on effectue les opérations suivantes :

- On vérifie selon la procédure décrite au paragraphe précédent s'il n'y a pas de sommets rouges superposés à supprimer et on élimine ces sommets rouges si besoin.
- On vérifie si l'application de la règle 5 de raffinement n'a pas créé dans la 1-orbite rouge de S_{sup} un autre brin z dont le plongement rouge correspond au plongement noir P_{sup} de S_{sup}. Dans ce

cas, le plongement noir de z (et de sa β-orbite) devient le plongement rouge de tous les brins de la 1-orbite rouge de z (et des brins de leur β-orbite respective) qui étaient plongés en rouge sur P_{sup}.
- On supprime le point P_{sup}.
- On supprime le brin x.

Si le sommet noir (et le plongement associé) ne doit pas être supprimé, on se contente de retirer le brin x de sa 1-orbite noire et de le supprimer.

On réalise ensuite les mêmes opérations sur le brin y.

Algorithme général de suppression d'arêtes et de sommets associés

Dans le cas des mini-cartes, l'objectif de cet algorithme est de supprimer l'arête marquée « provisoire ». Dans le cas de la fusion du polygone d'espace libre et de la mini-carte, le but est d'éliminer toutes les arêtes « à supprimer » ainsi que les sommets « à supprimer » associés (sommets noirs qui correspondent nécessairement par construction à une 1-orbite noire du brin « à supprimer »). Dans la description qui suit, on parlera indifféremment d'arête « provisoire » ou « à supprimer ». Toutefois, dans le cas de la suppression des arêtes provisoires, il ne faudra pas considérer la suppression de sommets.

Pour réaliser cette opération, on parcourt l'ensemble des brins (de toutes les couleurs possibles) jusqu'à en trouver un étiqueté « à supprimer ». Si ce brin n'est ni noir, ni gris, ni bleu, on remonte alors au brin (noir, gris ou bleu) 0-lié par lien noir : on aboutit ainsi nécessairement à un brin x plongé sur un sommet noir S à l'extrémité d'une arête « à supprimer ». La suppression de cette arête se fait du sommet S (en le supprimant si besoin) vers le sommet du brin 0-lié en noir à x (que l'on supprime également si besoin), en éliminant peu à peu chaque petite arête rouge située sur cette arête. Au passage, on supprime les sommets rouges situés sur cette arête, ainsi que les autres sommets rouges superposés qui le nécessiteraient (pour faciliter la lecture de cet algorithme, on peut se reporter à la figure 7.22).

1) La première étape consiste à traiter l'arête $(x, \alpha_{0_r}(x))_r$ (première petite arête de la carte rouge issue de la grande arête à supprimer). On pose $y = \alpha_{0_r}(x)$ et on réalise les opérations suivantes :

- Si le brin x est bleu (nécessairement, le brin y est aussi bleuté par construction), alors il faut foncer le brin $\beta(x)$. Cela signifie qu'un brin gris devient noir (bleu en fait si son β-successeur n'est pas x, c'est-à-dire s'il existe au moins une troisième arête superposée à (x, y)) et qu'un brin rose devient rouge (violet en fait si son β-successeur n'est pas x). On procède de même pour $\beta(y)$.

- Si la β-orbite de x n'est pas réduite à x, on supprime x de cette β-orbite. On procède de même pour y.

- Si le sommet S du brin x est étiqueté « à supprimer », on vérifie si son 1-orbite noire est bien réduite à x, auquel cas on peut effectivement supprimer ce sommet. Pour cela, on regarde selon la procédure décrite plus haut s'il n'y a pas de sommets rouges superposés à supprimer et on élimine ces sommets rouges si besoin. On vérifie ensuite si l'application de la règle 5 de raffinement n'a pas créé dans la 1-orbite rouge de S un autre brin z dont le plongement rouge correspond au plongement noir P de S. Dans ce cas, le plongement noir de z (et de sa β-orbite) devient le plongement rouge de tous les brins de la 1-orbite rouge de z (et des brins de leur β-orbite respective) qui étaient plongés en rouge sur P. En revanche, si le sommet S ne doit pas être supprimé, on se contente de retirer x de sa 1-orbite noire. On procède de la même manière pour le sommet S' du brin y.

- On supprime x et y de leur 1-orbite rouge respective, selon la procédure décrite plus haut.

– On supprime le brin x et le point P si le sommet S a été supprimé. On procède de même pour y et pour son plongement noir P'.

Si y appartenait à un sommet rouge, alors on passe à l'étape suivante et le brin courant devient le brin qui était 1-lié en noir avec y (le brin de la 1-orbite rouge de y qui est issu de la même grande arête « à supprimer » de la carte noire). Sinon, la suppression de l'arête est terminée.

2) Les étapes suivantes consistent à traiter une petite arête de la carte rouge issue de la grande arête à supprimer. On note x le brin courant (nécessairement rouge, rose ou violet) et y son brin 0-lié par liaison rouge. On réalise les opérations suivantes :

– Si le brin x est violet (nécessairement, le brin y est aussi bleuté par construction), alors il faut foncer le brin $\beta(x)$, comme dans la première étape. On procède de même pour $\beta(y)$.

– Si la β-orbite de x n'est pas réduite à x, on supprime x de cette β-orbite. On procède de même pour y.

– Le sommet rouge S du brin x est nécessairement à supprimer. Pour cela, on vérifie selon la procédure décrite plus haut s'il n'y a pas de sommets rouges superposés à supprimer et on élimine ces sommets rouges si besoin. On vérifie ensuite s'il existe dans la 1-orbite rouge de S un autre brin z dont le plongement rouge correspond au plongement noir P de S (une telle configuration peut avoir été générée par application de la règle 5 de raffinement). Dans ce cas, le plongement noir de z (et de sa β-orbite) devient le plongement rouge de tous les brins de la 1-orbite rouge de z (et des brins de leur β-orbite respective) qui étaient plongés en rouge sur P. Si le sommet S' du brin y est étiqueté « à supprimer », on procède de la même manière.

– On supprime x et y de leur 1-orbite rouge respective, selon la procédure décrite plus haut.

– On supprime le brin x et le point P si le sommet S a été supprimé. On procède de même pour y et pour son plongement noir P'.

Si y appartenait à un sommet rouge, alors on recommence cette étape avec le brin qui était 1-lié en noir avec y (le brin de la 1-orbite rouge de y qui est issu de la même grande arête « à supprimer » de la carte noire), et qui devient le brin courant. Sinon, la suppression de l'arête est terminée.

Ensuite, on continue le parcours des brins de la carte rouge jusqu'à en retrouver un nouveau marqué « à supprimer » : on revient à son brin 0-lié par liaison noire et on élimine la grande arête « à supprimer » correspondante. On procède ainsi jusqu'à la fin du parcours des brins de la carte rouge. On peut s'assurer que cet algorithme se termine car il n'y a pas de création de nouveaux brins durant son application (uniquement des brins qui changent de couleur, éventuellement).

7.5.3 Déplacement de tous les sommets en une seule étape

Plutôt que de déplacer les n sommets du polygone un à un en réalisant un raffinement et une complétion de labels à chaque étape, il est possible de réaliser l'opération en une seule étape (cf. Fig. 7.23).

Pour cela, on ordonne les sommets du polygone de S_1 à S_n (par exemple dans l'ordre de parcours du polygone par les liaisons noires ou dans l'ordre de balayage du raffinement). Lors de la création de la mini-carte combinatoire associée au brin x du sommet S_i, les triangles considérés sont composés des sommets suivants : ancienne position S_a de S_i, nouvelle position S_n de S_i et sommet S_j du brin 0-lié en noir à x. Si $j > i$, on utilise l'ancienne position S'_a de S_j et l'arête qui relie S_n à S'_a est étiquetée « à supprimer », de même que celle qui relie S_a à S'_a. Sinon, on utilise la nouvelle position S'_n de S_j et cette fois, l'arête qui relie S_n à S'_n n'est pas étiquetée « à supprimer », à la différence de celle qui relie S_a à S'_n.

Ensuite, une fois auto-raffinées et une fois leurs arêtes provisoires S_aS_n supprimées, toutes ces mini-cartes combinatoires peuvent être fusionnées en une seule fois avec le polygone par raffinement et complétion des n labels. On peut alors définir le 2-label d'un brin de mini-carte comme le couple composé du numéro du sommet déplacé et du degré d'occupation rouge de la mini-carte correspondante. Le 2-label d'un brin du polygone correspond à son degré d'occupation rouge. Lors de la complétion de labels, on concatène sur un brin toutes les listes de labels des brins de sa β-orbite. Ensuite, on parcourt toutes les faces de la carte raffinée et on vérifie pour tout couple de brins adjacents sur cette face s'ils portent les mêmes couples de labels concaténés. En principe, pour une face donnée, on ne peut pas trouver deux couples qui partagent le même numéro de sommet mais pas le même degré d'occupation associé : les cartes de départ étant bien raffinées avec des degrés d'occupation bien définis, de tels couples doivent appartenir à des cellules différentes. De même, on ne peut trouver deux couples de brin de la même face portant des degrés d'occupation distincts et étant issus du polygone de départ. Ces remarques permettent d'accélérer la comparaison. Si des brins adjacents portent des labels différents, alors on applique la règle « comp » proposée par Cazier [23] (cf. section 7.3.7.).

Enfin, une fois la complétion de labels effectuée, il suffit de parcourir tous les couples de labels et d'additionner pour chaque brin tous les degrés d'occupation des listes de couples concaténées associées et de l'ajouter au degré d'occupation rouge associé au polygone de départ.

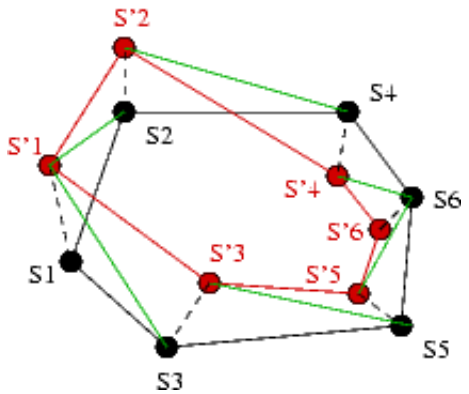


FIG. *7.23: Déplacement simultané de tous les sommets de la carte noire vers les positions rouges. Les segments en pointillés sont « provisoires » et les segments verts et noirs « à supprimer ».*

7.5.4 Complexité

Supposons qu'un polygone présente n sommets en moyenne. Dans les polygones initiaux, ce nombre est borné par construction (par le nombre de points du scan) mais ce nombre peut augmenter au fur et à mesure du fait des mises en correspondances avec les autres polygones, qui génèrent des cassures. Le

nombre de brins constituant le polygone est alors de $2n$. Par ailleurs, une mini-carte contient au maximum 16 brins (dans les cas très dégénérés, comme l'indique la figure 7.19) et peut être créée en temps constant (borné par le cas le plus défavorable) : la création de l'ensemble de ces cartes est donc linéaire en n.

Ainsi, dans le cas du déplacement de sommets un à un, chaque raffinement du polygone avec une mini-carte est de l'ordre de $(16 + n + i)log(16 + n)$ (i étant le nombre de brins ajoutés) et chaque complétion de labels est linéaire en n (plus précisément en $2 \times (16 + n)$ puique la carte à raffiner est constituée de deux objets : le polygone et la mini-carte). La suppression des arêtes « à supprimer » est a priori négligeable par rapport au raffinement : si l'on suppose les 1-orbites et β-orbites bornées, elle a un coût proportionnel au nombre de brins à supprimer. Au final, on obtient donc une complexité théorique en $O(n \times (n+i)logn)$ puisqu'il faut répéter cette opération pour chaque sommet. Cette complexité pourrait cependant être considérablement améliorée si l'on pouvait réaliser des raffinements plus locaux avec la mini-carte qui est a priori de taille réduite (un découpage régulier de la carte globale pourrait peut-être y contribuer).

Alternativement, dans le cas du déplacement simultané de tous les sommets, le raffinement de la carte avec l'ensemble des mini-cartes présente une complexité de l'ordre de $(16n + 2n + i_t)\log(16n + 2n)$ c'est-à-dire en $(18n + i_t)\log(18n)$, i_t correspondant au nombre de brins ajoutés lors du raffinement (et étant a priori plus important que la somme des i individuels car la carte raffinée présente plus de brins). Quant à la complétion de labels, elle a en principe un coût en $O(n \times (n + i_t))$ (puisque l'on considère $(18n+i_t)$ brins et $n+1$ objets). En pratique, ce coût est généralement bien plus réduit car les mini-cartes couvrent de petites zones, qui ont peu de chances de recouvrir l'ensemble des objets de la carte : chaque face appartient à un nombre réduit d'objets et il existe a priori peu de labels à comparer pour deux brins adjacents d'une même face. A l'extrême, si l'on considère que ce nombre est borné, on obtient un coût linéaire en $n + i_t$, qui est moins coûteux que le raffinement. Ainsi (même si i_t est difficile à comparer à la somme des i individuels), le déplacement de tous les sommets en une seule étape paraît plus efficace que le déplacement un à un de ces sommets.

7.6 Principe de la mise à jour par superposition de polygones

Dans cette section, nous introduisons la **première stratégie de mise à jour, qui maintient explicitement la structure de tous les polygones d'espace libre observés et recommence leur fusion à chaque étape de mise à jour**. On généralise en fait l'opération de mise à jour de polygone au cas de cartes combinatoires constituées d'une superposition de polygones d'espace libre.

7.6.1 Représentation utilisée

Pour mettre en œuvre cette stratégie, nous avons opté pour une représentation globale unique, dont la structure contient suffisamment d'informations pour extraire individuellement chacun de ses polygones constitutifs (cf. Fig. 7.24 (a) et (b)). Ainsi, cette représentation globale est une carte combinatoire colorée dont chaque brin porte un 2-label (label de face) qui indique le numéro des polygones dont il est issu. En fait, pour une arête donnée, un seul brin porte ce label : celui qui est orienté dans le sens des aiguilles d'une montre sur le périmètre du polygone. De cette manière, on précise de quel côté de l'arête se trouve la face interne du polygone concerné. Chaque brin de la carte peut donc contenir jusqu'à n labels de ce type. Par ailleurs, les liaisons noires de la carte colorée précisent l'ordre de chaînage des arêtes au sein des polygones, de façon à permettre leur extraction.

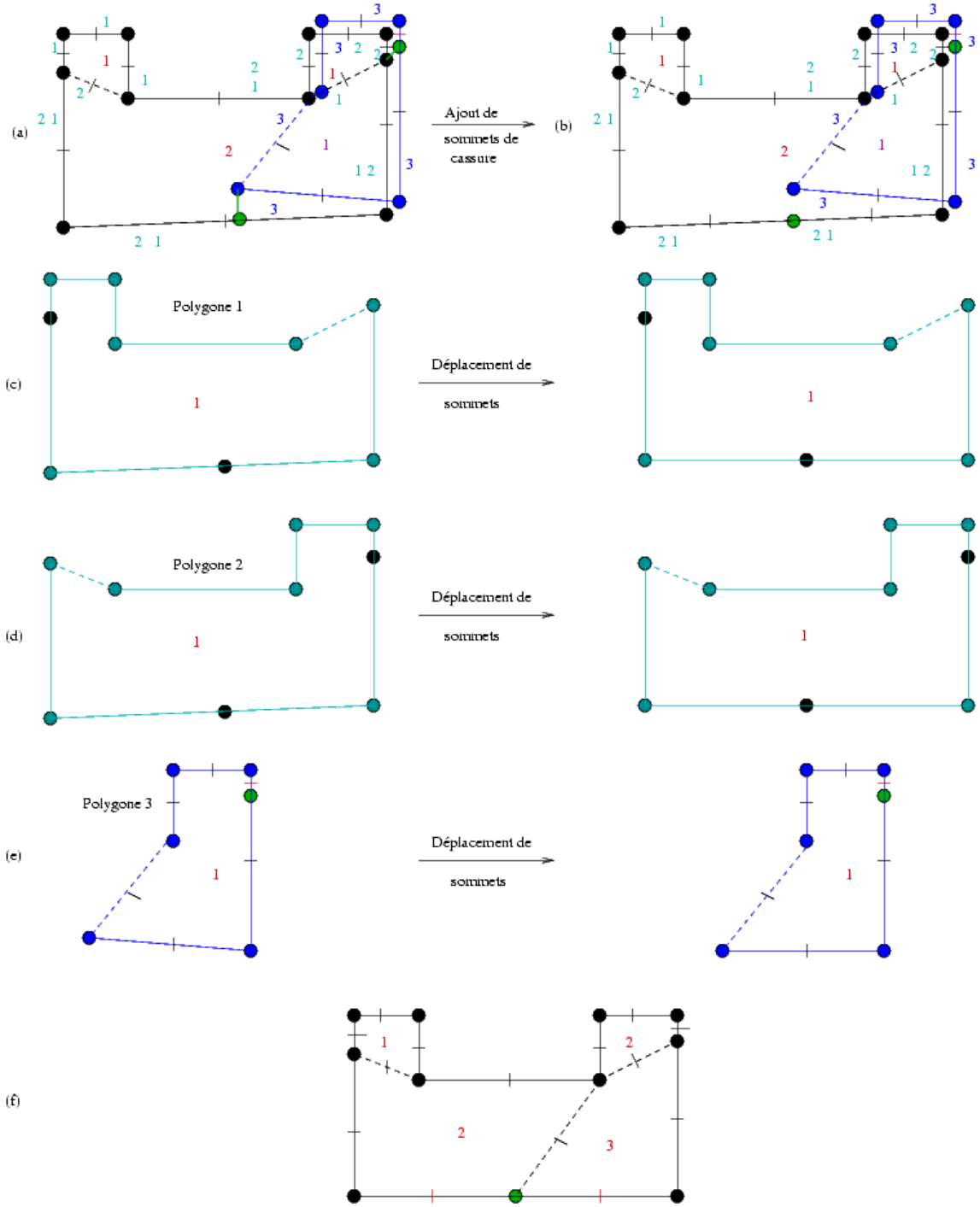


FIG. *7.24: Illustration de la stratégie de fusion de polygones. (a) Carte globale (en noir) et polygone local (en bleu) : la mise en correspondance induit des cassures en vert sur certaines arêtes. Les nombres en rouge indiquent les degrés d'occupaton des cellules (en violet pour le polygone local) et les nombres en bleu clair les numéros de polygone (en bleu foncé pour le polygone local). (b) Les points de cassure ont été intégrés dans la représentation. (c)(d)(e) Déplacement de sommets sur les polygones extraits (les sommets noirs correspondent à des sommets qui n'existaient pas dans le polygone initial). (f) Carte globale finale résultant de la fusion des trois polygones.*

Une solution alternative consisterait à maintenir individuellement la structure (par exemple la liste de chaînée des sommets) de chacun des n polygones d'espace libre observés. Ensuite, ces polygones peuvent être fusionnés dans une carte unique, dont la structure ne nécessite de garder explicitement celle de chaque polygone fusionné, puisque ceux-ci sont stockés en mémoire par ailleurs. On éviterait ainsi d'introduire dans la représentation globale des listes potentiellement longues de numéros de polygones. Toutefois, l'utilisation de ces $n + 1$ cartes (n cartes de polygones individuels et 1 carte globale) introduirait des redondances inutiles au niveau des brins et des liaisons noires, et ne réduirait pas la taille globale de la représentation. En outre, comme on l'a vu aux chapitres précédents, certaines étapes du processus de cartographie (mise en correspondance, extraction des observations...) travaillent sur une représentation globale unique. Or elles nécessiteraient parfois de propager certaines modifications de la carte globale vers les polygones individuels : c'est notamment le cas des phénomènes de « cassure » qui résultent de l'observation des points virtuels sur les segments globaux. Ces cassures sont déterminées à partir d'observations sur la carte globale, et doivent être répercutées sur les arêtes correspondantes des polygones. Pour mettre à jour la structure des polygones individuels, on doit donc impérativement maintenir un lien depuis les arêtes de la carte globale vers celles des polygones locaux.

7.6.2 Algorithme de mise à jour

Finalement, l'algorithme de mise à jour se décompose de la manière suivante :

*** Transformation du polygone local en carte combinatoire :**

Pour réaliser cette transformation, on parcourt les arêtes du polygone : pour chaque arête, on crée deux brins noirs 0-liés par liaison noire et on relie le premier brin au second de l'arête précédente par 1-liaison noire. On a ainsi créé la couche topologique de la partie noire de la carte colorée correspondant à ce polygone. Ensuite, pour s'assurer de son bon plongement, on peut la projeter sur un grand cercle comme expliqué précédemment. On la transforme ensuite en carte colorée bien plongée. Enfin, en prévision de la reconstruction de la carte globale par superposition de polygones, on ajoute à chaque brin du polygone (de la carte colorée) un label indiquant le numéro de ce polygone ($m + 1$ si la carte globale est déjà composée de m polygones). En fait, ce label peut être vu comme un label de face, qui indique en même temps de quel côté de l'arête du brin se trouve l'espace libre : parmi les deux brins d'une même arête, seul l'un d'entre eux porte ce label. On note qu'un degré d'occupation noir serait redondant avec cette information : « 1 » si le brin porte le numéro du polygone et « 0 » dans le cas contraire. Pour plus de commodité, on peut toutefois ajouter ce degré d'occupation noir.

*** Ajout de sommets dans la carte globale liés aux cassures et aux appariements multiples de sommets :**

Lors de l'observation de « cassures » (observation de points virtuels sur les segments obstacles), on ajoute des sommets sur les arêtes de la carte globale (arêtes de la carte noire). Dans un premier temps, on ne se soucie pas de leur plongement géométrique (qui pourra être initialisé arbitrairement au centre de l'arête découpée par exemple) : seule la couche topologique de la carte noire importe. Le plongement sera géré par ailleurs au niveau des polygones individuels. Lorsqu'il existe plusieurs cassures sur une arête donnée, il faut cependant s'intéresser à l'ordre de chaînage des nouveaux sommets, pour obtenir des liaisons noires correctes. Pour cela, nous avons choisi d'examiner à la fois les anciennes et les nouvelles positions géométriques des sommets, avant et après le calcul réalisé par filtrage de Kalman.

Soient A_1 et A_2 les anciens plongements des extrémités de l'arête concernée. Soient $B_j, j \in \{1..n\}$ les anciens plongements des n cassures et $P_{i_j}, i_j \in \{1..n\}$ leurs projections orthogonales sur (A_1A_2), classées dans l'ordre le long de la droite (A_1A_2), de A_1 vers A_2. A priori, cet ordre fournit celui du chaînage : $A_1, P_{i_1}, ..., P_{i_n}, A_2$. Il risque cependant d'arriver que certains P_{i_j} tombent en dehors du segment $[A_1A_2]$ une fois la correction de position effectuée. Une telle configuration peut par exemple résulter d'une erreur sur la position de la carte locale lorsque l'on met en œuvre l'étape d'appariement (cf. Fig. 7.25).

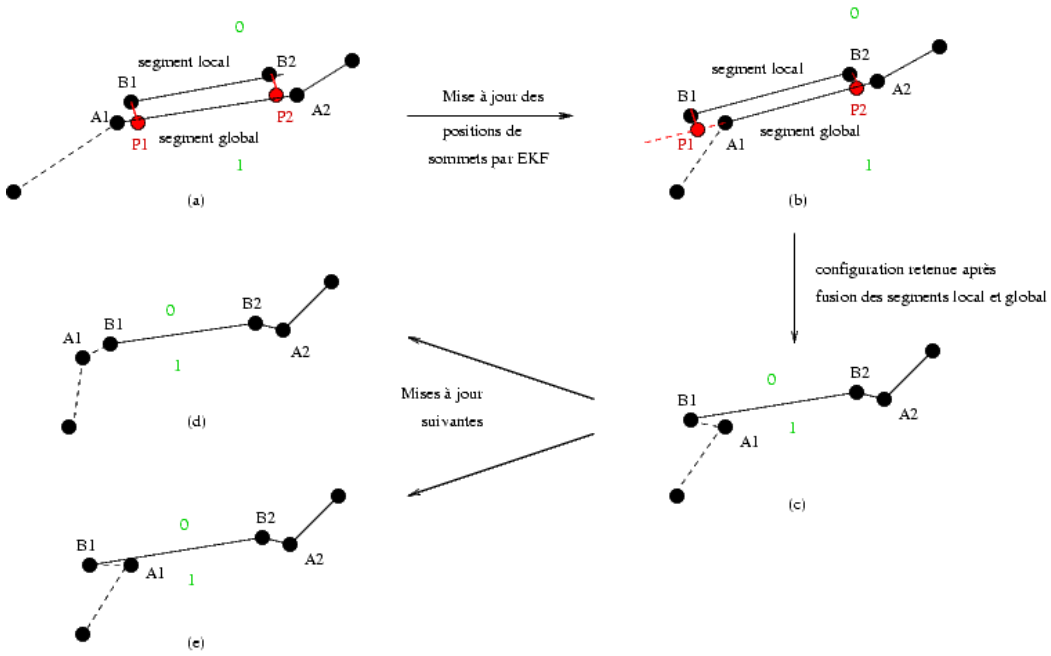


FIG. *7.25: Problème d'ordre de chaînage des cassures sur une arête de la carte globale. (a) Le point B_1 extrémité du segment local se projette entre A_1 et A_2. (b) Après calcul des nouvelles positions de sommets par filtrage de Kalman, B_1 se projette hors du segment $[A_1A_2]$. (c) On définit l'ordre de chaînage A_1, B_1, B_2, A_2 et A_1B_1 devient non obstacle. (d) Après de nouvelles mises à jour géométriques, l'ordre des projections orthogonales est rétabli dans l'ordre de chaînage. (e) Après de nouvelles mises à jour, on confirme que B_1 est « à gauche » de A_1 : les degrés d'occupation (en vert) sont corrects.*

Dans ce cas, on garde l'ordre de chaînage défini plus haut, mais si P_{i_1} est à l'extérieur de $[A_1A_2]$, le segment $[A_1B_1]$ devient « non obstacle ». Ainsi, au cours des mises à jour géométriques ultérieures, si B_1 se projette finalement entre A_1 et A_2 sur la droite (A_1A_2), le chaînage choisi et les degrés d'occupation sont corrects (par rapport à la configuration qui aurait été adoptée si les sommets avaient directement été observés à leur position courante), même si l'on a perdu une information sur le fait que A_1B_1 est obstacle. Inversement, si l'ordre P_{i_1}, A_1, A_2 est confirmé, avec des points alignés (ou presque), les degrés d'occupation restent corrects (ou presque) et comme le segment $[P_1A_1]$ est devenu « non obstacle », il ne risque pas de créer des mises en correspondance dans le mauvais sens (opposé au sens d'observation du segment $[A_1A_2]$ initial, qui indique de quel côté de ce segment se trouve l'espace libre). De même, si P_{i_n} est à l'extérieur de $[A_1A_2]$, alors le segment $[A_2B_n]$ devient « non obstacle ». Cette règle est généralisable au cas où il existe plusieurs P_{i_j} à gauche ou à droite de $[A_1A_2]$.

Les sommets ajoutés sont des sommets noirs. Chaque nouveau brin de l'arête découpée est de la couleur de l'arête initiale (noire, grise ou bleue). Il hérite des labels du brin de l'arête de départ orienté dans le même sens que lui (en particulier les numéros de polygones et le degré d'occupation noir). Quant aux nouvelles liaisons rouges, elles sont calquées sur les nouvelles liaisons noires créées pour les nouveaux brins.

*** Ajout de sommets dans la carte locale liés aux observations de points globaux sur une droite locale ou aux appariements multiples de sommets locaux :**

On procède de la même manière que dans le cas de l'ajout de sommets dans la carte globale : les chaînages sont définis de façon similaire. Par ailleurs, pour chaque arête du polygone local appariée à une arête de la carte globale, on ajoute à chaque brin local un label contenant le numéro du brin global correspondant.

Cette opération est facilitée par l'ajout de sommets réalisé dans les cartes locale et globale puisque les arêtes appariées sont matérialisées et se correspondent exactement. Comme nous le verrons ultérieurement, l'information contenue dans ce label servira lors du raffinement des différents polygones.

De plus, les sommets du polygone local qui correspondent à des sommets de la carte globale adoptent le même plongement noir.

*** Extraction de chaque polygone individuel à partir de la carte globale**

Pour extraire le polygone i, on réalise les opérations suivantes :

- On parcourt l'ensemble des brins de la carte globale jusqu'à trouver le premier qui porte le numéro i. On considère alors le brin a qui lui est 0-lié (par liaison rouge ou noire) : par construction, ce brin ne porte pas le label i. On extrait alors le brin x_0 qui lui est 0-lié par liaison noire pour être sûr qu'il s'agit d'un brin d'un sommet noir, qui porte le label i. On copie ensuite ce brin pour initialiser l'extraction du polygone i et on ajoute sur cette copie un label portant le numéro du brin x_0 dans la carte globale.

- A partir de x_0, on suit les liaisons noires pour extraire le polygone complet jusqu'à revenir au brin x_0. Ainsi, à chaque étape j, soit x_j le brin courant (à l'initialisation, il s'agit de x_0). On copie le brin $y_j = \alpha_0(x_j)$ en lui adjoignant une étiquette portant le numéro de y_j dans la carte globale, et on relie cette copie par 0-liaison noire à celle du brin courant. Puis on parcourt les brins 1-liés à y_j par liaison noire jusqu'à en trouver un, z_j, qui porte le numéro du polygone i (par construction, il en existe nécessairement un et un seul différent de y_j, y_j ne portant pas le label i). Si z_j est différent de x_0, on copie z_j en lui adjoignant une étiquette portant le numéro de z_j dans la carte globale et on relie cette copie par 1-liaison noire à celle de y_j dans le polygone reconstruit. Puis z_j devient le brin courant x_{j+1} et on effectue l'étape $j+1$. Sinon (si z_j correspond au brin x_0), l'extraction est terminée et on relie y_j à x_0 par liaison noire.

En outre, lors de la copie des brins de la carte globale, on ajoute un label indiquant le numéro de son point de plongement dans la carte originale. Ainsi, comme nous le verrons, il pourra y avoir une véritable fusion des sommets (modification des liaisons noires) lors de l'application de la règle 5 de raffinement pour reconstruire la prochaine carte globale par superposition de polygones. On ajoute également sur chaque copie de x_j le numéro du polygone i, en prévision de la reconstruction de la prochaine carte globale par superposition de polygones. Par ailleurs, à l'instar du polygone d'espace libre, l'algorithme ci-dessus n'a en réalité extrait que la couche topologique noire de chaque polygone. Pour définir le plongement de ses sommets en évitant les incohérences (superpositions, retournements...), on peut commencer par projeter le polygone sur un grand cercle en plaçant les sommets à intervalle régulier, comme décrit plus haut. Ensuite, on peut transformer cette carte monochrome en carte colorée bien plongée.

*** Déplacement de sommets sur chaque polygone issu de la carte globale et sur le polygone local**

Pour chaque polygone, on réalise un déplacement (simultané puisque c'est la méthode la plus efficace a priori) de tous ses sommets vers leur nouvelle position calculée par filtrage de Kalman. On utilise l'algorithme décrit dans la section précédente.

*** Raffinement coloré des n polygones mis à jour**

On effectue le raffinement des n polygones correspondant chacun à une carte colorée. On utilise pour cela les règles définies dans la section précédente, avec la contrainte de maintenir explicitement la structure de chaque polygone individuel pour être capable de l'extraire à nouveau à la prochaine mise à jour. Il s'agit donc de retenir pour chaque brin le numéro des polygones initiaux auquel il appartient (1-labels créés sur chaque polygone et transférés à la carte globale lors du raffinement).

Plus précisément, par rapport aux règles de raffinement coloré définies précédemment, on effectue les modifications suivantes :

- **Règle 1** : pas de modification ;

- **Règle 2** : soient $(x, y)_r$ et $(z, t)_r$ les deux arêtes à l'origine du déclenchement cette règle (le plongement de x étant superposé à celui de z). Si le plus petit des numéros de polygones attachés à $(z, t)_r$ est inférieur au plus petit de $(x, y)_r$, alors on échange les rôles des deux arêtes (la notation x s'applique à z, la notation y s'applique à t, et réciproquement) : de cette manière, comme nous le verrons ci-dessous, on s'assure de la validité des 0-successeurs noirs des brins.

  Alors pour chaque brin a de la β-orbite de z (z inclus), on vérifie si ce brin porte le même numéro de brin de carte globale que x (s'il existe un tel brin, alors il est nécessairement unique). Si c'est le cas, on fusionne réellement a et x : cela signifie que l'on transfère les numéros de polygone indiqués sur le brin supprimé a vers le brin conservé x. De plus, on supprime réellement le brin a (on ne se contente pas de le griser ou de le laisser grisé) et on le supprime non seulement de sa 1-orbite rouge mais aussi de sa 1-orbite noire. La 0-orbite rouge n'a toutefois pas besoin d'être modifiée car le brin 0-lié par couture rouge est également supprimé lors de l'application de cette même règle (cf. ci-dessous). Par ailleurs, puisque les deux brins fusionnés portent le même numéro de brin de carte globale, il n'y a pas de contradiction au niveau de ce label entre les brins fusionnés.
  En revanche, si les deux brins a et x ne portent pas le même numéro de brin global, la seconde arête est seulement grisée (du moins si elle ne l'était pas précédemment) et on ne transfère pas les numéros de polygone d'une arête à l'autre : on revient à la règle 2 classique décrite précédemment.

  On procède de la même manière pour les brins b de la β-orbite de x (x exclu puisqu'on l'a déjà traité), en s'assurant lors d'une fusion véritable que le brin conservé est le brin b.

  Enfin, on procède de la même manière pour les brins y et t (et pour leurs β-orbites respectives). Cette fois, ce sont les brins de la β-orbite de y qui sont conservés lors d'une fusion véritable.

- **Règles 3 et 4** : ces règles restent inchangées, si ce n'est que l'on transfère certains labels. Ainsi, lorsqu'une arête $(x, y)_r$ est découpée en deux arêtes $(x, a)_r$ et $(b, y)_r$ alors a hérite du numéro de

label de carte globale dont y est issu et des numéros de polygones de y et b hérite de la même manière des labels de x.

– **Règle 5** : soient x et y les brins à l'origine du déclenchement de cette règle.

Pour chaque brin a de la 1-orbite rouge de x (x inclus),
pour chacun des brins a' β-liés à a (a inclus),
pour chaque brin b de la 1-orbite rouge de y,
pour chacun des brins b' β-liés à b,
si a' et b' appartiennent à $D_n \bigcup D_{ng}$ et ne sont pas 1-liés par liaison noire, on compare leurs labels de numéros de sommets : s'ils sont identiques, on fusionne réellement ces deux sommets, c'est-à-dire qu'on insère la 1-orbite noire de a' dans la 1-orbite noire de b' (leurs plongements noirs sont identiques donc il n'y a rien à faire du côté des plongements).
Ensuite, on applique à x et y la règle 5 classique qui ne modifie que les 1-liaisons rouges et les plongements rouges.

– **Règle 6** : pas de modification.

On peut se demander pourquoi dans la règle 2, on s'attache à fusionner d'éventuelles arêtes grisées. En fait, ces arêtes grisées sont aussi candidates à l'appariement avec le polygone local d'espace libre dans la phase de mise en correspondance : elles peuvent donc subir des cassures. Ensuite, ces cassures doivent être répercutées dans les polygones individuels correspondants, au moment de leur extraction. Or si les arêtes de ces différents polygones ne sont pas fusionnées dans la carte globale, certaines risquent de ne pas être découpées par les cassures.

Par ailleurs, sans les précautions de renommage des brins suite à la comparaison des numéros de brins de la carte globale, on peut noter qu'après l'application de la règle 2, les 0-successeurs noirs des brins risqueraient de ne plus être valides (au sens des propriétés énoncées dans la définition des cartes colorées). En effet, lorsque les arêtes de deux polygones i et j sont superposées et ont été découpées par un sommet rouge, alors les petites arêtes de la carte rouge qui en résultent ne seraient pas nécessairement fusionnées de la même manière par application de la règle 2 : tantôt on peut garder la petite arête du polygone i, tantôt celle du polygone j. Ainsi, en gardant systématiquement celle du polygone qui a le plus petit numéro, on est sûr au final de garder l'arête complète du même polygone. On étend ensuite cette règle au cas d'arêtes qui portent déjà plusieurs numéros de polygones : l'arête conservée est celle qui contient le plus petit numéro de polygone parmi les deux listes. Une autre manière de résoudre cela pourrait consister à parcourir les arêtes de la carte noire via les liaisons rouges et de corriger les 0-successeurs noirs erronés.

*** Mise à jour des degrés d'occupation noirs**

La mise à jour des degrés d'occupation noirs peut se faire directement lors de l'application des règles de raffinement. En effet, comme il ne s'agit pas de labels de face qui doivent être propagés lors du découpage des cellules, il n'est pas nécessaire de réaliser de complétion de labels ultérieure.

Ainsi, on effectue les ajouts suivants aux règles de raffinement précédentes :

– **Règle 2** : lorsque deux arêtes subissent une véritable fusion, on ajoute au degré d'occupation noir de l'arête conservée celui de l'arête supprimée.
– **Règles 3 et 4** : lorsqu'une arête $(x, y)_r$ est découpée en deux arêtes $(x, a)_r$ et $(b, y)_r$ alors a hérite du degré d'occupation noir de y et b hérite de celui de x.

*** Mise à jour des degrés d'occupation rouges**

La mise à jour des degrés d'occupation rouges passe nécessairement par une complétion de labels. En effet, pour chaque cellule résultant de la restructuration topologique de la couche rouge de la carte colorée, il faut savoir à quels polygones individuels appartient cette cellule. Pour cela, on peut utiliser directement les labels de face indiquant les numéros de polygones dont les brins sont issus. Ainsi, on se retrouve dans une configuration de complétion de labels classique. Les règles de raffinement ci-dessus intègrent déjà les modifications nécessaires à cette complétion de labels (règles 2, 3 et 4).

Une fois le raffinement terminé, on parcourt les faces de la carte rouge. Sur chacune de ces faces, pour chaque couple (x, y) de brins adjacents, on concatène sur x (dans une nouvelle liste, pour ne pas perdre la liste initiale qui servira aux extractions de polygones futurs) tous les numéros de polygones issus de la β-orbite de x. On procède de même pour $\alpha_{0_r}(x)$, y et $\alpha_{0_r}(y)$. Puis on applique la règle « comp » définie par Cazier [23] (cf. Fig. 7.12) sur ces quatre brins. Ensuite, il suffit d'additionner pour chaque brin le nombre de numéros de polygones qu'il porte (puisque chacun correspond à un degré d'occupation d'une unité).

7.6.3 Complexité globale de cette approche

Soit m le nombre moyen de sommets d'un polygone : ce nombre est borné à l'initialisation mais il peut augmenter avec les cassures liées aux appariements successifs. Soit n_p le nombre de polygones constituant la carte globale. La transformation du polygone local en carte combinatoire est linéaire en m, comme l'ajout de sommets dans la carte globale (le nombre de sommets à ajouter est limité par le nombre de sommets du polygone local). L'ajout de sommets dans un polygone est linéaire en $m \times n_p$ (le nombre de sommets à ajouter est limité par le nombre de sommets de la carte globale, qui est de l'ordre de $m \times n_p$). L'extraction de polygones est linéaire en le nombre total d'arêtes des polygones (en supposant les 1-orbites bornées) : elle est en $O(m \times n_p)$. Le déplacement de sommets pour chaque polygone et la complétion de labels associée sont en $O(m \times (m + i_t))$ (i_t étant le nombre de brins ajoutés lors de cette opération) dans le pire cas et en $O((m + i_t) \log m)$ si le nombre de labels d'une cellule peut être considéré comme borné. Pour les n_p polygones, la complexité globale est donc de $O(n_p \times m \times (m + i_t))$ dans le pire cas ou en $O(n_p \times (m + i_t) \log m)$ si le nombre de labels d'une cellule peut être considéré comme borné. Le raffinement des n_p polygones est en $O(n_p \times (m + i_t) + i) \log(n_p \times (m + i_t))$ si i est le nombre de brins ajoutés lors de cette opération. Enfin, la mise à jour des degrés d'occupation rouges est en $O(m^2 \times n_p)$ (m objets dans une carte de $O(m \times (n_p + 1))$ brins) : c'est donc cette opération qui conditionne la complexité de l'ensemble si m est très supérieur à n_p. Sinon, c'est le raffinement global qui est le plus coûteux.

7.6.4 Bilan sur cette approche

Le fait de maintenir implicitement la structure de chaque polygone d'espace libre introduit des avantages et des inconvénients :

- Parmi les avantages, cette méthode permet de remonter aux incohérences de retournement et de croisement au niveau de chaque polygone individuel, ce qui peut faciliter la correction ulérieure de ces incohérences. On peut notamment envisager de revenir en arrière sur certaines opérations de fusion : si celles-ci ont été irrémédiablement prises en compte avec le formalisme de filtrage que nous utilisons, il n'en est pas de même avec des techniques de type « décisions retardées » (« delayed decisions ») par exemple.

- En revanche, cette représentation paraît coûteuse à la fois en termes d'espace mémoire et de temps de calcul pour la mise à jour. En particulier, la taille d'un brin n'est plus bornée et peut croître

indéfiniment avec le nombre de scans introduits dans la représentation. De plus, la fusion de tous les polygones à chaque itération paraît très lourde à mettre en œuvre. Enfin, ce modèle de carte peut limiter les possibilités de mise en forme de la carte puisque la suppression d'une arête risque d'empêcher l'extraction ultérieure des polygones qui la contiennent.

7.7 Mise à jour par fusion incrémentale de polygones

On s'intéresse à présent à une **seconde stratégie de mise à jour, qui ne nécessite pas de retenir explicitement la structure des polygones d'espace libre initiaux : cette stratégie fonctionne de manière incrémentale à partir de la carte globale courante et de la nouvelle carte locale**.

7.7.1 Définition de la représentation utilisée

Comme on ne cherche plus à maintenir la structure des polygones d'espace libre, la représentation employée pour la carte globale peut demeurer plus légère : en particulier, on ne conserve plus les numéros de polygone sur les arêtes. Ainsi, cette représentation est simplement une carte colorée dont les arêtes portent comme précédemment des degrés d'occupation rouges et noirs. Les degrés d'occupation noirs indiquent pour une arête donnée combien d'arêtes de polygones ont été fusionnées par le processus d'appariement. Comme les arêtes ne peuvent être appariées que si elles ont été observées dans le même sens, les degrés d'occupation noirs de part et d'autre d'une arête sont nécessairement 0 d'un côté et $i \geq 1$ de l'autre.

Quant aux degrés d'occupation rouges, ils indiquent les véritables degrés d'occupation de la carte colorée bien plongée, une fois la restructuration topologique réalisée.

7.7.2 Déplacement et ajout de sommets

Cette fois, le déplacement de sommets ne se fait plus dans un polygone où les 1-orbites noires sont composées d'exactement 2 brins mais dans une représentation où il peut y avoir plus de deux brins incidents à un même sommet. Ainsi, les mini-cartes combinatoires doivent être légèrement modifiées pour tenir compte de cette spécificité. Par ailleurs, pour éviter de surcharger la carte par de nouveaux sommets de cassure avant de réaliser la mise à jour géométrique, cet ajout de ces sommets peut se faire après le déplacement de sommets noirs de la carte vers leur nouvelle position calculée par filtrage de Kalman. Dans ce cas, les sommets ajoutés risquent de ne plus être alignés avec les extrémités de l'arête concernée : ils donnent lieu à de petites arêtes qui ne sont plus superposées aux anciennes. Pour gérer les éventuelles intersections avec d'autres éléments de la carte qui pourraient en résulter, il faut introduire une nouvelle opération, similaire au déplacement de sommets et basée sur la fusion (le raffinement) avec des mini-cartes combinatoires (cf. Fig. 7.28). On note que cet ajout de sommet a posteriori pourrait également être envisagé dans la stratégie de mise à jour par superposition de polygones.

*** Ajout de sommet :**

Les mini-cartes combinatoires considérées sont composées d'un unique triangle élémentaire dont les sommets sont confondus avec les deux extrémités A_1 et A_2 du segment à « casser » et avec la nouvelle position du sommet S à ajouter. L'arête A_1A_2 est étiquetée « à supprimer ». Cette fois, le degré d'occupation de la face interne de ce triangle n'est plus 1 ou -1 mais i ou $-i$, i étant le degré d'occupation noir du côté de l'arête où se trouve l'espace libre (cf. explication ci-dessous pour le déplacement de sommets). Ce degré d'occupation est négatif si S se trouve dans le demi-plan délimité par A_1A_2 du côté de l'espace libre et positif sinon.

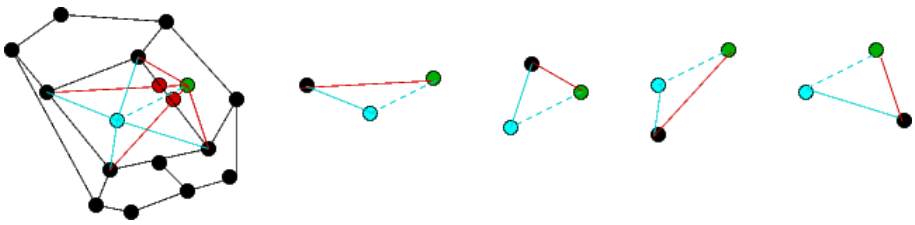


FIG. *7.26: Exemple de déplacement d'un sommet (de la position bleue vers la position verte) dans une carte. Les triangles à droite de la carte correspondent aux triangles élémentaires qui composent la mini-carte combinatoire permettant le déplacement du sommet.*

Ensuite, comme dans l'opération de déplacement de sommet (décrite ci-dessous), ce triangle est auto-raffiné, puis superposé pour raffinement avec la carte globale dont l'arête A_1A_2 a été étiquetée « à supprimer ». Enfin, on effectue la complétion de labels pour mettre à jour les degrés d'occupation rouges et on supprime dans la carte globale résultante les arêtes marquées « à supprimer ».

Si l'on souhaite ajouter plusieurs sommets sur une même arête, il suffit de répéter plusieurs fois cette opération dans l'ordre de chaînage des sommets.

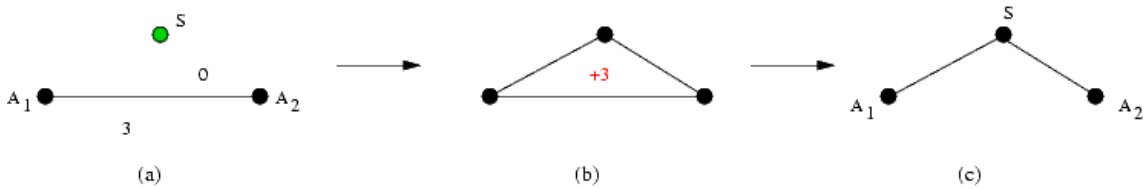


FIG. *7.27: Exemple de d'ajout de sommet S sur une arête A_1A_2. (a) Configuration initiale dans laquelle on indique les degrés d'occupation noirs pour les brins de l'arête. (b) Triangle élémentaire correspondant. (c) Résultat de l'opération.*

*** Déplacement de sommet :**

L'opération de déplacement de sommet se déroule de la même manière que dans le cas d'un polygone unique, sauf que cette fois, la mini-carte combinatoire peut être composée de plus de deux triangles élémentaires. En fait, on superpose autant de triangles élémentaires qu'il existe d'arêtes incidentes au sommet déplacé. En outre, les degrés d'occupation de ces triangles peuvent être différents de « 1 », « -1 » ou « 0 ».

Ainsi, pour réaliser le déplacement d'un sommet noir dont on dispose d'un brin x, on réalise les opérations suivantes :

– On crée les différents triangles élémentaires qui composent la mini-carte. Pour cela on parcourt les brins y de la 1-orbite noire de x (x inclus). Pour chaque brin y, on crée un triangle dont les sommets sont plongés sur l'ancien plongement de $\alpha_0(y)$, le nouveau plongement de $\alpha_0(y)$ et sur le plongement de x. A priori, on réalise l'opération de déplacement de tous les sommets en une seule étape. Ainsi, si le plongement de x est placé avant celui de $\alpha_0(y)$ dans l'ordre global des points, on choisit le nouveau plongement de x. On utilise l'ancien plongement sinon. Les conventions d'étiquetage « provisoire » et « à supprimer » sont identiques au cas du déplacement de sommet dans un polygone unique.

Le degré d'occupation interne de chaque triangle est calculé en fonction des degrés d'occupation noirs i et j de part et d'autre de l'arête $(y, \alpha_0(y))$. En fait, cette arête est composée d'une fusion de $i + j$ arêtes de polygones, dont i se trouvent d'un côté et j de l'autre (au sens de la localisation de l'espace libre). Ainsi, lorsque l'on déplace cette arête, on déplace i arêtes de polygone individuel dans un sens donné d'un côté et j polygones dans l'autre sens de l'autre côté. Le degré d'occupation résultant est donc $j - i$ ou $i - j$, suivant le sens dans lequel on effectue ce déplacement. En fait, par construction (de par les règles de mise en correspondance), l'un des deux degrés i ou j est nul. En conséquence, tout se passe comme dans le cas du polygone unique, sauf que l'on manipule des degrés d'occupation de valeur potentiellement supérieure.

Chaque triangle est ensuite auto-raffiné comme dans le cas du polygone unique : on adopte la même convention d'annulation des degrés d'occupation en cas de triangle aplati.

– On crée la mini-carte combinatoire par superposition des triangles élémentaires et auto-raffinement. On supprime ensuite l'arête provisoire qui relie l'ancienne et la nouvelle position du sommet à déplacer. Les liaisons noires sont également définies de manière similaire au cas du polygone unique.

– On superpose les mini-cartes avec la carte globale. Le raffinement, la complétion de labels et la suppression des arêtes marquées « à supprimer » se déroulent également de manière similaire au cas du polygone unique.

7.7.3 Algorithme global

Finalement, l'algorithme global de mise à jour incrémentale se décompose de la manière suivante :

– **On déplace les sommets noirs de la carte locale vers leur nouvelle position calculée par filtrage de Kalman**. Si besoin, pour s'assurer que le polygone local ne présente pas d'incohérences (croisements ou retournements), on le projette sur un cercle, puis on déplace les sommets vers leur nouvelle position selon la procédure exposée précédemment. Ce déplacement peut s'effectuer en une seule étape comme on l'a vu à la section précédente dans le cas du polygone unique.

– **On déplace les sommets noirs de la carte globale vers leur nouvelle position**. Cette opération peut également être réalisée en une seule étape, comme on l'a vu dans le cas du polygone unique.

– **On ajoute les sommets de cassure sur les arêtes de la carte globale**, en utilisant le raffinement de triangles élémentaires expliqué ci-dessus. Pour chaque arête, l'ordre de chaînage est défini de la même manière que dans la stratégie de superposition de polygones, avec les mêmes conventions de marquage des segments « obstacles » ou « non obstacles ». Comme dans le cas du déplacement de sommets, l'ajout de ces points peut être réalisé en une seule opération, en introduisant un ordre sur les sommets à rajouter, puis en utilisant à chaque création de triangle élémentaire les mêmes règles de choix entre anciens et nouveaux sommets. La stratégie de complétion de labels est également similaire.

– **On ajoute également sur la carte locale les « cassures » locales** qui correspondent en réalité à des observations de sommets de la carte globale sur des arêtes de la carte locale. On procède de la même manière que pour la carte globale, avec les mêmes conventions de chaînage.

– **On raffine la superposition des deux cartes locale et globale**.
Pour cela, il faut effectuer quelques modifications dans les règles de raffinement générales des cartes colorées, à l'instar de la stratégie de superposition de polygone.

Dans la règle 2, on fusionne réellement les deux arêtes si et seulement si elles portent le même label d'appariement, créé lors de l'extraction des observations.
Dans ce cas, on additionne les degrés d'occupation noirs des brins qui se correspondent, et on supprime physiquement la seconde arête avec modification des 1-orbites noires et rouges aux extrémités.
En revanche, si les labels de sommets ne se correspondent pas, on applique la règle 2 classique qui ne supprime pas réellement la seconde arête mais se contente de la griser.

Lors de l'application des règles 3 et 4, on transfère les labels d'appariement aux sous-brins créés, ainsi que les degrés d'occupation noirs et rouges.

Dans la règle 5, on effectue une fusion réelle des sommets uniquement si les labels d'appariement de sommets sont identiques. Dans ce cas, on modifie les 1-liaisons noires en plus des 1-liaisons rouges. Sinon, on ne modifie que les 1-liaisons rouges comme dans le cas classique.

– **On calcule les nouveaux degrés d'occupation rouges**.

Pour cela, classiquement, les degrés d'occupation rouges de la carte locale et ceux de la carte globale ont été propagés par les règles de raffinement. Puis la complétion de labels est mise en œuvre par parcours de faces avec application de l'opération « comp » (cf. Fig. 7.12) sur les brins adjacents qui ne portent pas les mêmes labels. Enfin, on parcourt à nouveau l'ensemble des brins pour additionner ces degrés d'occupation rouges.

7.7.4 Complexité

Par rapport à la stratégie de fusion de polygones, on économise notamment le déplacement d'un même sommet dans différents polygones : ce déplacement est géré en une seule fois lorsque les sommets sont fusionnés dans la couche noire de la représentation. De plus, on évite de traiter de trop nombreuses superpositions d'arêtes lors de la superposition car celles-ci restent fusionnées.

La taille d'une mini-carte est proportionnelle au nombre d'arêtes adjacentes au sommet déplacé. Si l'on suppose ce nombre d'arêtes borné, ainsi que le nombre de mini-cartes superposées sur chaque cellule, on obtient donc une complexité globale en $O((n+i)\log n)$, i étant le nombre de brins ajoutés lors du raffinement et n le nombre de brins initiaux dans les deux cartes.

7.7.5 Bilan

Ainsi, par rapport à la stratégie précédente, nous disposons d'une représentation beaucoup plus compacte, dont la mise à jour est en principe moins coûteuse en temps de calcul. En outre, il devient possible de supprimer des arêtes lors d'une mise en forme de la carte, sans perturber le calcul des degrés d'occupation : les polygones n'ont pas besoin d'être complets pour reconstituer la carte globale. Cette approche nous a donc paru plus prometteuse que la première : c'est pourquoi nous l'avons choisie pour notre implémentation. Cependant, pour ne pas risquer d'être confrontés à des problèmes d'arrondis de nombres flottants, nous avons opté pour une version discrète décrite ci-dessous.

7.8 Application aux cartes discrètes

Pour passer aux cartes combinatoires discrètes, il faut combiner le principe des brins colorés avec celui de la discrétisation, qui associe les grands brins classiques à des petits brins discrets. Il faut également garder un lien vers la représentation en nombres réels pour être en mesure de modifier la carte lorsque les positions des sommets sont réestimées par filtrage de Kalman. On remarque cependant qu'une fois la carte discrétisée, il est difficile de remonter à une représentation en nombres réels bien plongée : en effet, les sommets rouges et les degrés d'occupation (rouges) ne peuvent pas être transposés de la carte discrète vers une carte réelle, notamment du fait des intersections multiples de segments discrets (qui n'existent pas dans la représentation en nombres flottants). En revanche, la partie « évolutive » de la carte (partie noire) peut facilement être maintenue dans les deux espaces, réel et discret. Ainsi, dans une application pratique, il existe deux options : soit on choisit de travailler directement en discret avec des degrés d'occupation qui correspondent seulement au cas discret, soit on travaille en nombres flottants (avec les risques associés, à moins peut-être d'une implantation en arithmétique exacte) puis on discrétise selon les besoins (pour mettre en œuvre un algorithme de planification sur grille discrète par exemple).

7.8.1 Définition des cartes combinatoires colorées discrètes

La structure des cartes colorées discrètes est très similaire à celle des cartes colorées en nombres réels. Les grands brins forment une carte combinatoire monochrome (projetée sur $\mathbb{R}^2$) tandis que les petits brins composent une carte colorée projetée sur $\mathbb{Z}^2$ de manière à former des 1-plongements horizontaux ou verticaux. Les liens entre petits brins et grands brins sont similaires à ceux définis dans le cadre des cartes discrètes monochromes.

Une *carte colorée discrète* est donc un 11-uplet $(DG, D_n, D_{ng}, D_r, D_{rg}, \alpha_0, \alpha_1, \alpha_{0_r}, \alpha_{1_r}, \beta, \gamma)$ où :
- DG est un ensemble d'éléments appelés *grands brins* ;
- D_n, D_{ng}, D_r et D_{rg} sont des ensembles disjoints d'éléments appelés *petits brins*, respectivement *noirs, gris, rouges et roses* ; $DP = D_n \bigcup D_{ng} \bigcup D_r \bigcup D_{rg}$ est appelé ensemble des *petits brins colorés* ou simplement ensemble des *petits brins*. $D_f = D_n \bigcup D_r$ est appelé ensemble des *petits brins foncés* et $D_g = D_{ng} \bigcup D_{rg}$ ensemble des *petits brins grisés* ;
- α_{0_r} (α_0 rouge) est une involution dans DP ;
- α_0 est une fonction de $DG \bigcup DP$ vers $DG \bigcup D_n \bigcup D_{ng}$ et sa restriction α_{0_n} à $D_n \bigcup D_{ng}$ est une involution, de même que sa restriction à DG ;
- α_1 est une permutation dans $DG \bigcup DP$;
- α_{1_r} (α_1 rouge) est une permutation dans DP ;
- β est une permutation dans DP, qui sert à gérer les liens entre arêtes superposées ;
- γ est une fonction de $DG \bigcup DP$ dans $DG \bigcup DP$ qui sert à gérer les liens entre grands brins et petits brins.

En outre, les petits brins sont plongés sur des sommets discrets définis dans $\mathbb{Z}^2$ (ou *petits sommets*) et les grands brins sur des sommets réels définis dans $\mathbb{R}^2$ (ou *grands sommets*).

Une telle carte vérifie les propriétés suivantes :
- Le 9-uplet $(D_n, D_{ng}, D_r, D_{rg}, \alpha_0, \alpha_1, \alpha_{0_r}, \alpha_{1_r}, \beta)$ (où les fonctions α_0, α_1, α_{0_r} et α_{1_r} sont ici leurs restrictions à DP) a une structure de carte combinatoire colorée appelée *petite carte colorée*. On emploie pour désigner ses éléments constitutifs le même vocabulaire que dans le cas réel, les noms étant précédés de l'adjectif « petit ».
- Le triplet (DG, α_0, α_1) (où les fonctions α_0 et α_1 sont ici leurs restrictions à DG) a une structure de carte combinatoire monochrome classique appelée *grande carte noire*. On emploie pour désigner

ses éléments constitutifs le vocabulaire classique, les noms étant précédés de l'adjectif « grand ».
- Si x est un petit brin, $\gamma(x)$ représente le grand brin dont il est issu (dans le même sens que lui sur l'arête discrétisée). Inversement, si X est un grand brin, $\gamma(X)$ représente le premier petit brin de sa discrétisation.
- Les petits brins x_i et $\alpha_0(x_i)$ liés par la fonction γ aux grands brins X et $\alpha_0(X)$ correspondent au plongement discret de l'arête $(X, \alpha_0(X))$. Comme dans le cas monochrome, ce plongement vérifie une discrétisation par l'algorithme de Bresenham [72], légèrement modifié pour assurer une 4-connexité des petites arêtes : chacune de ces petites arêtes présente un 1-plongement vertical (même abscisse pour les 0-plongements de ses extrémités), ou horizontal (même ordonnée pour les plongements de ses extrémités). Initialement (au moment de la discrétisation), deux petites arêtes consécutives sur une même grande arête sont orthogonales (ensuite, une petite arête peut être coupée en deux via un petit sommet rouge). Enfin, ces petits brins x_i et $\alpha_0(x_i)$ sont reliés entre eux par les fonctions α_0 et α_1 dans l'ordre de parcours de l'arête.

On pourrait également définir de grands brins colorés, mais nous n'en avons pas vu l'utilité dans notre application. Par ailleurs, on peut remarquer une certaine redondance entre les orbites noires des petits brins appartenant à des sommets noirs et les orbites des grands brins correspondants : il apparaît inutile de définir les fonctions α_0 et α_1 pour les grands brins puisqu'elles sont identiques à celles des petits brins auxquels ils sont reliés par γ. Il nous a paru cependant plus pratique de conserver la structure de grande carte combinatoire, en particulier pour la phase de mise en correspondance entre cartes locale et globale. Enfin, il est inutile de maintenir la notion de « plongement rouge » puisqu'il n'existe plus de « micro-déplacements » de sommets fusionnés (par application de la régle 5) liés aux erreurs d'arrondis de nombres flottants : deux sommets fusionnés se trouvent nécessairement exactement sur la même cellule de la grille discrète.

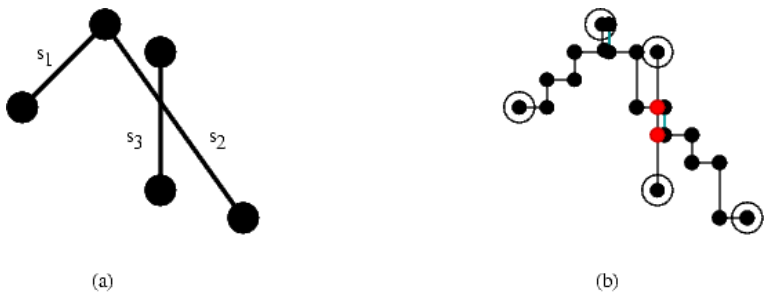


FIG. *7.28: Exemple de carte discrète. (a) Illustration des grands brins et des grands sommets. (b) Petits brins et petits sommets correspondants. On note que la discrétisation introduit une intersection entre s_1 et s_2 avec un petit segment grisé. Par ailleurs, l'intersection entre s_2 et s_3 n'est pas réduite à un point mais correspond à un petit segment : elle induit ainsi deux nouveaux petits sommets rouges sur s_3.*

7.8.2 Raffinement coloré et complétion de labels des cartes colorées discrètes

Le raffinement coloré des cartes discrètes porte uniquement sur les petits brins : la structure des grandes cartes n'est pas modifiée durant l'opération. La seule évolution concerne la fonction γ. Les nouveaux petits brins créés par application des règles 3 et 4 ont la même image par γ que les petits brins dont ils sont issus. En revanche, l'image par γ des brins existants ne change pas. De plus, contrairement au cas des cartes monochromes discrètes, il est inutile de remonter aux grands brins une fois le raffinement des petits brins terminés puisque la grande carte demeure inchangée.

7.8.3 Ajout et déplacement de sommets dans les cartes colorées discrètes

Le déplacement et l'ajout de sommets dans les cartes discrètes concerne des sommets réels (« grands » sommets) et les grandes arêtes. On crée donc les triangles élémentaires et les mini-cartes de la même manière que dans le cas réel, si ce n'est que les arêtes de ces triangles sont discrétisés. Cette discrétisation suit exactement celle de la carte combinatoire pour les anciennes arêtes : on l'obtient en copiant les petits brins de la carte. Pour les nouvelles arêtes, elle doit être créée via l'algorithme de Bresenham. Les arêtes des triangles élémentaires portent des labels « provisoire » ou « à supprimer » comme dans le cas réel. Il est inutile de les recopier sur les petits brins puisque la fonction γ relie directement les petits brins au grand brin dont ils sont issus : on peut facilement savoir pour un petit brin donné s'il doit être supprimé ou non.

Ainsi, lorsque l'on veut supprimer les arêtes et les sommets marqués « à supprimer », on procède de la même manière que dans le cas réel en parcourant dans l'ordre les petits brins composant la grande arête à éliminer. Une fois les petits brins supprimés, il faut en faire de même pour les grands brins correspondants, en les éliminant de leurs 1-orbites noires.

7.8.4 Mise à jour des cartes

Les deux stratégies de mise à jour, par superposition de polygones ou par fusion incrémentale, peuvent être appliquées aux cartes discrètes. Les informations de correspondance nécessaires aux « fusions réelles » d'arêtes et de sommets (où l'on supprime l'un des éléments superposés et où l'on modifie les liaisons noires) sont indiquées sur la grande carte. Ensuite, l'algorithme se déroule de la même manière que dans le cas des cartes réelles, en utilisant les opérations ci-dessus, adaptées aux cartes discrètes.

7.8.5 Complexité

Soit l une dimension caractéristique de la carte à raffiner (par exemple sa longueur en unités discrètes).

Concernant la stratégie de fusion de polygones, soit m le nombre moyen de sommets d'un polygone et n_p le nombre de polygones constituant la carte globale. La transformation du polygone local en carte combinatoire est linéaire en $l \times m$, comme l'ajout de sommets dans la carte globale (le nombre de sommets à ajouter est limité par le nombre de sommets du polygone local). L'ajout de sommets dans le polygone local est linéaire en $m \times n_p$ (le nombre de sommets à ajouter est limité par le nombre de grands sommets de la carte globale). L'extraction de polygones est linéaire en le nombre total d'arêtes des polygones (en supposant les 1-orbites bornées) : elle est en $O(m \times n_p \times l)$. Le déplacement de sommets pour chaque polygone et la complétion de labels associée sont en $O(m \times (m + i_t) \times l)$ dans le pire cas (i_t étant le nombre de brins ajoutés lors de l'opération) et en $O((m + i_t) \times l \log(m \times l))$ si le nombre de labels d'une cellule peut être considéré comme borné. Pour les n_p polygones, la complexité globale est donc de $O(n_p \times l \times m \times (m + i_t))$ dans le pire cas ou en $O(n_p \times (m + i_t) \times l \log(m \times l))$ si le nombre de labels d'une cellule peut être considéré comme borné. Le raffinement des n_p polygones est en $O(n_p \times (m+i_t) \times l \times log(n_p \times (m+i_t) \times l))$ $O(n_p \times (m+i_t)+i)log(n_p \times (m+i_t))$ si i est le nombre de brins ajoutés lors de cette opération. Enfin, la mise à jour des degrés d'occupation rouges est en $0(m^2 \times n_p \times l)$ (m objets dans une carte de $O(m \times (n_p+1) \times l)$ brins) : c'est donc cette opération qui conditionne la complexité de l'ensemble si m est très supérieur à n_p. Sinon, c'est le raffinement global qui est le plus coûteux.

Concernant la stratégie de mise à jour incrémentale, si l'on suppose ce nombre d'arêtes incidentes à un sommet donné borné, ainsi que le nombre de mini-cartes superposées sur chaque cellule, on obtient une complexité globale en $O(n + i) \times l \log(n \times l)$, i étant le nombre de brins ajoutés lors du raffinement

et n le nombre de brins initiaux dans les deux cartes.

La complexité réelle de ces opérations est en fait très dépendante de l'orientation de la grille de discrétisation par rapport à la carte. Par exemple, si les arêtes de l'environnement sont toutes orientées selon deux directions orthogonales et si la discrétisation se fait selon ces deux directions, alors le nombre de petits brins correspond au nombre de grands brins. En revanche, si la discrétisation est orientée selon la bissectrice de ces directions, l'effet de la longueur caractéristique l se fait largement sentir. On cherchera donc à réaliser une discrétisation selon la direction principale de la carte.

7.9 Résultats expérimentaux

L'ensemble de la chaîne algorithmique de construction de cartes a été implémentée en C++. Concernant l'étape de mise à jour, nous avons opté pour la stratégie de fusion incrémentale de polygones, qui nous a paru plus efficace que la stratégie de superposition de polygones, comme nous l'avons évoqué précédemment. Si de premiers tests ont été réalisés avec des cartes combinatoires en nombres flottants, la chaîne complète a été développée au moyen des cartes colorées discrètes, afin d'en garantir la robustesse.

Une illustration sur un cas simple

Nous commençons par illustrer les mécanismes de mise à jour sur un cas simple constitué de deux polygones avec peu de sommets (cf. Fig. 7.29 et 7.30), afin de montrer pas à pas chaque étape de l'algorithme. Nous avons volontairement choisi quelques cas dégénérés afin de vérifier la robustesse de l'approche. Dans toutes les illustrations qui suivent (**qui correspondent toutes à des images créées directement par notre programme** au format ppm, à l'exception du schéma de notations 7.37), les pointillés indiquent les arêtes « non obstacles ». Les petits sommets rouges sont indiqués explicitement par un point rouge tandis que les grands sommets sont matérialisés par un carré noir (les petits sommets noirs ne sont pas tracés). Les arêtes grisées ne sont pas dessinées : seules les arêtes bleutées correspondantes (bleues ou violettes) sont indiquées. Les arêtes composées de deux brins rouges (ou violets) sont tracées selon cette couleur. Enfin, les degrés d'occupation positifs sont indiqués en gris tandis que les degrés négatifs sont matérialisés en vert. Plus un degré est élevé (en valeur absolue), plus la couleur correspondante est foncée au sein d'une même carte.

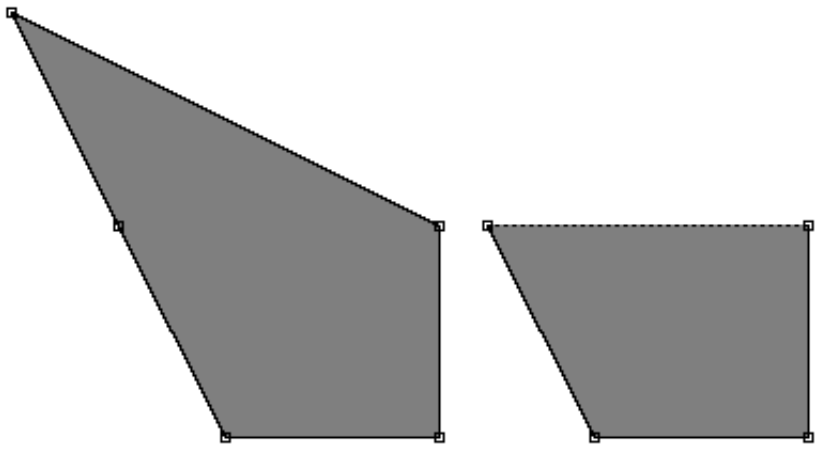


FIG. *7.29: Polygones d'espace libre.*

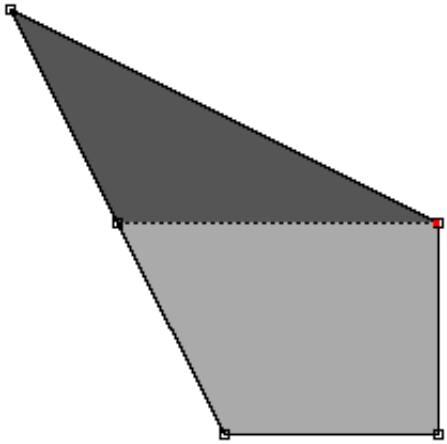


FIG. *7.30: Carte combinatoire résultant de la fusion des deux polygones d'espace libre de la figure 7.29, en tenant compte de l'information d'appariement des segments superposés (ces segments sont donc réellement fusionnés et non simplement superposés).*

Les figures 7.31 et 7.32 détaillent le processus de déplacement d'un sommet dans la carte de la figure 7.30.

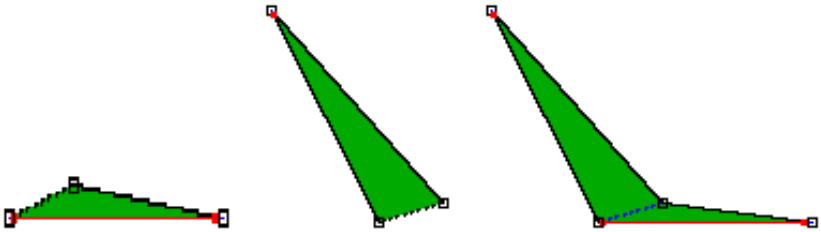
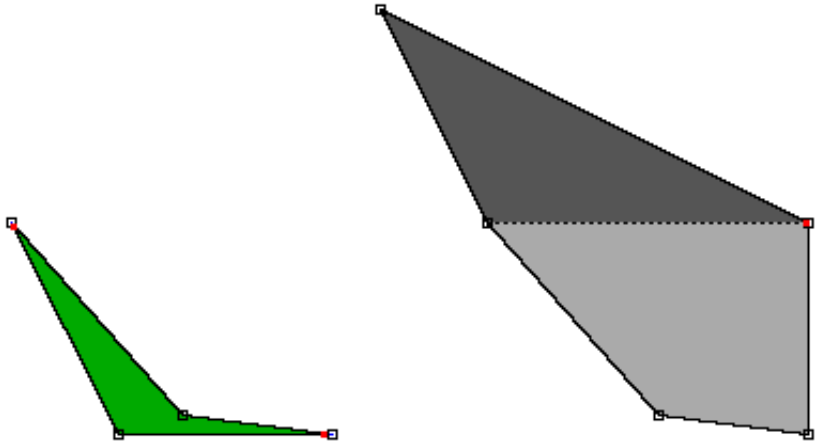


FIG. *7.31: Déplacement du premier sommet en bas à gauche de la figure 7.30. (a) Création du triangle élémentaire correspondant à la première arête adjacente à ce sommet. (b) Création du triangle élémentaire correspondant à la seconde arête. (c) Raffinement de la superposition de ces deux triangles afin de créer la mini-carte combinatoire.*

La figure 7.33 détaille le déplacement séquentiel des quatre autres sommets avec les mini-cartes combinatoires associées. On note que le déplacement du troisième sommet implique la création de 3 triangles élémentaires, d'où une mini-carte un peu plus complexe que dans les autres cas (cf. Fig. 7.34). Par ailleurs, le déplacement du dernier sommet est artificiel puisque sa nouvelle position est identique à sa position de départ. En pratique, ce déplacement peut être économisé mais nous l'avons tout de même effectué pour vérifier la robustesse de l'approche dans un cas très dégénéré.

La figure 7.35 illustre deux déplacements supplémentaires de sommets, avec un cas très dégénéré impliquant le déplacement du sommet de droite sur l'arête gauche. L'image suivante montre que la superposition d'arêtes qui en résulte est réversible : en particulier, les informations concernant les degrés d'occupation sont maintenues.

La figure 7.36 illustre le déplacement des sommets en une seule étape selon les positions précédentes.



(a) (b)

FIG. *7.32: (a) Suppression des arêtes provisoires (en pointillés bleus sur la Fig. 7.31 (c) car l'arête visible est « non obstacle » par définition et constituée de brins bleus). (b) Carte globale résultant de cette opération.*

Enfin, la figure 7.38 illustre l'ajout d'un sommet sur l'arête $[BC]$ et la figure 7.39 illustre l'ajout du même sommet sur l'arête $[CD]$ de la carte initiale (avec les notations de la figure 7.37).

7.9.1 Illustration sur des données réelles

Les figures 7.40 et 7.41 illustrent un déplacement de sommets séquentiel permettant de construire une carte combinatoire colorée (cf. Fig. 7.42) à partir du polygone de la figure 7.43 (extrait de l'un des scans utilisé au chapitre 5), suite à une projection sur un cercle garantissant une bonne initialisation de la structure de carte combinatoire colorée (comme cela a été expliqué dans les sections précédentes). Cette première carte tient lieu de carte globale pour notre algorithme. Pour donner un ordre de grandeur, elle est constituée de 106 grands brins et de 2484 petits brins. Le pas de discrétisation (correspondant à un pixel) est de 5 cm. L'ensemble de l'opération de déplacement séquentiel de sommets menant du cercle à la carte finale a été réalisé en 5 secondes environ sur un ordinateur équipé d'un processeur Pentium IV.

La figure 7.44 montre la carte locale résultant du polygone extrait du second scan (en vert) de la figure 5.17 du chapitre 5. Cette carte a également été construite par déplacement de sommets après projection sur un cercle, et suite à l'ajout de sommets résultant de « cassures » sur les segments locaux (liées à des appariements multiples avec des segments globaux).

La figure 7.45 présente la carte globale mise à jour suite à l'appariement avec le polygone local d'espace libre. En particulier, certains segments ont été découpés en raison des « cassures » liées aux appariements multiples : cette opération est réalisée via la fonction d'ajout de sommet. En outre, les sommets ont été légèrement décalés suite à la mise à jour du vecteur d'état par application du filtrage de Kalman.

Enfin, la figure 7.46 illustre le résultat de l'algorithme complet de cartographie appliqué aux cartes globale et locale précédentes, après appariement (Fig. 5.36), extraction des observations, mise à jour du vecteur d'état par filtrage de Kalman, ajout de sommets de cassure sur la carte globale, déplacement des sommets selon les nouvelles positions indiquées dans ce vecteur d'état et fusion des cartes globale et locale

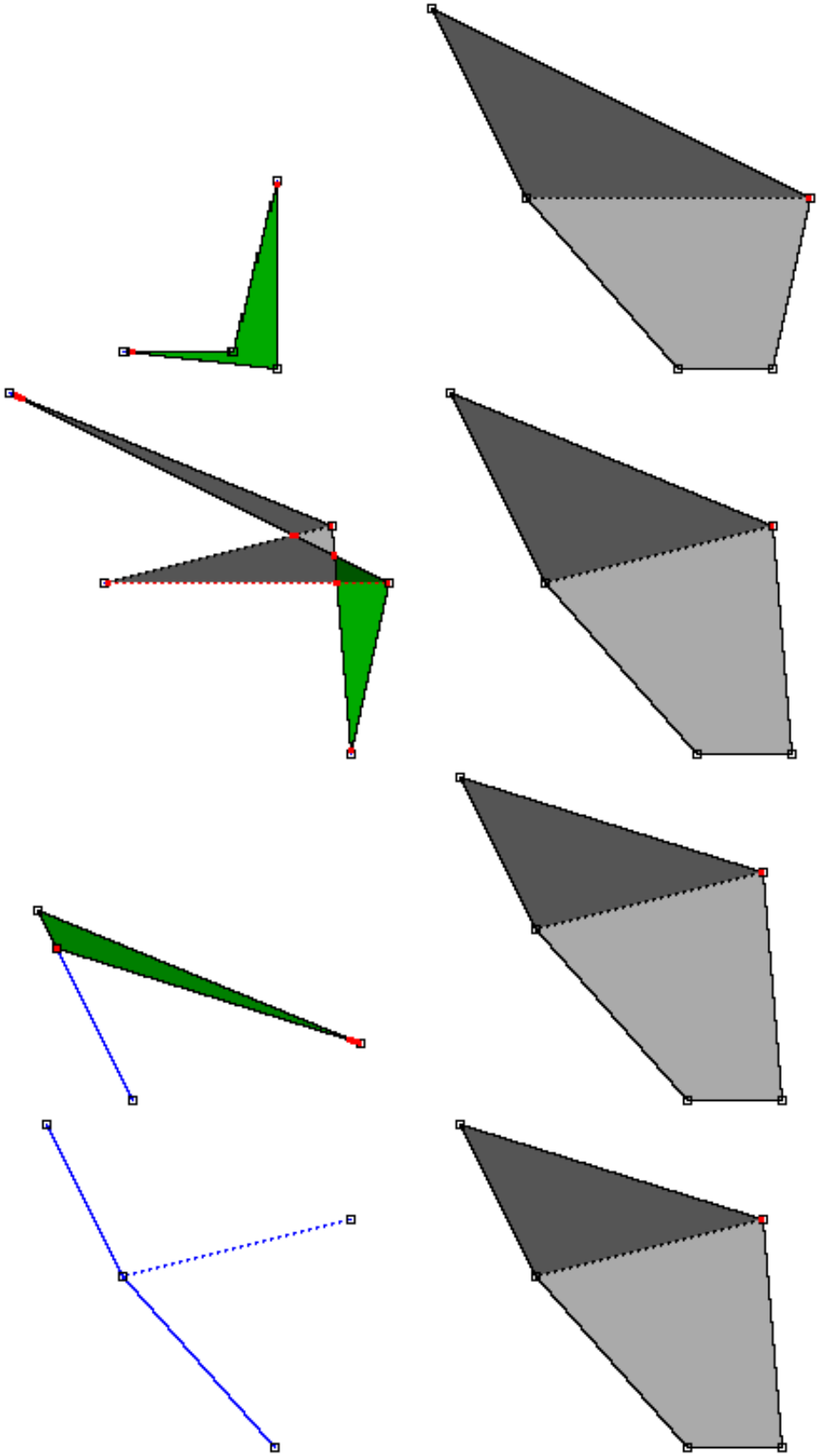


FIG. *7.33: Déplacement séquentiel des quatre autres sommets de la carte globale, avec à gauche les mini-cartes combinatoires correspondantes.*

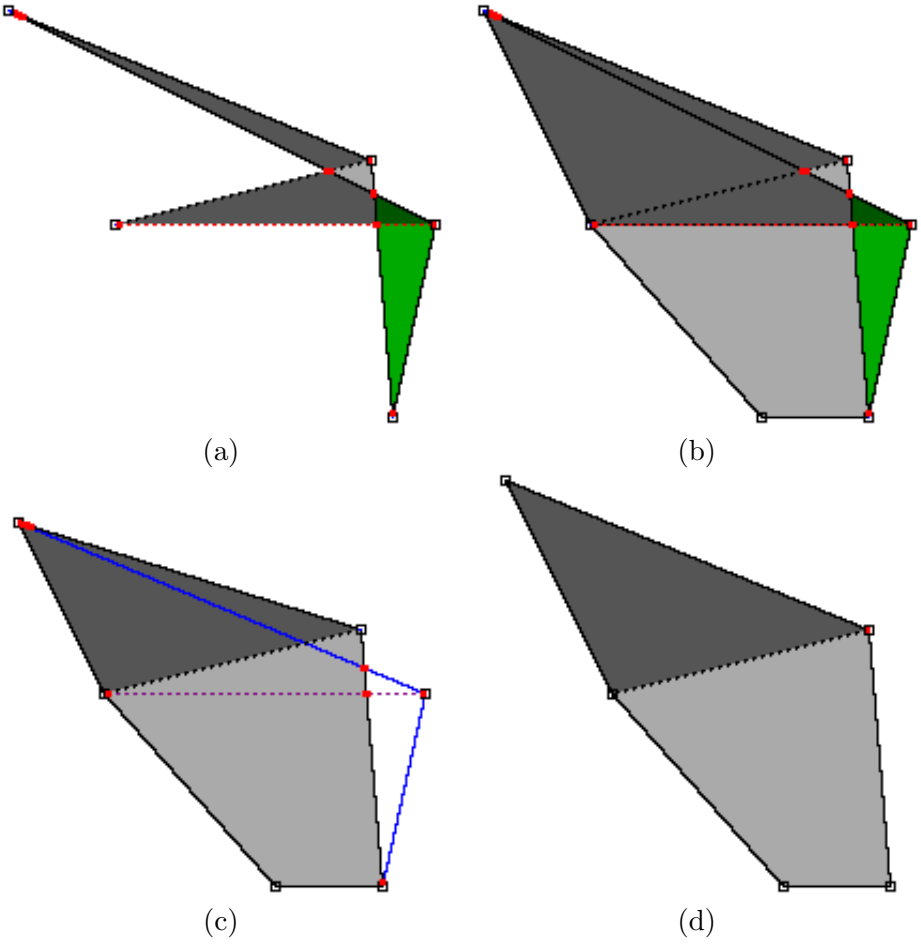


FIG. *7.34: Détail du déplacement du troisième sommet de la carte globale. (a) Mini-carte combinatoire définie pour le déplacement du troisième sommet (cette mini-carte est constituée de trois triangles élémentaires puisque le sommet correspondant est à l'intersection de trois arêtes). (b) Superposition de la carte globale et de la mini-carte avant raffinement. (c) Résultat du raffinement des deux cartes et de la complétion de labels avant suppression des arêtes marquées « à supprimer ». (d) Résultat de la suppression des arêtes « à supprimer ».*

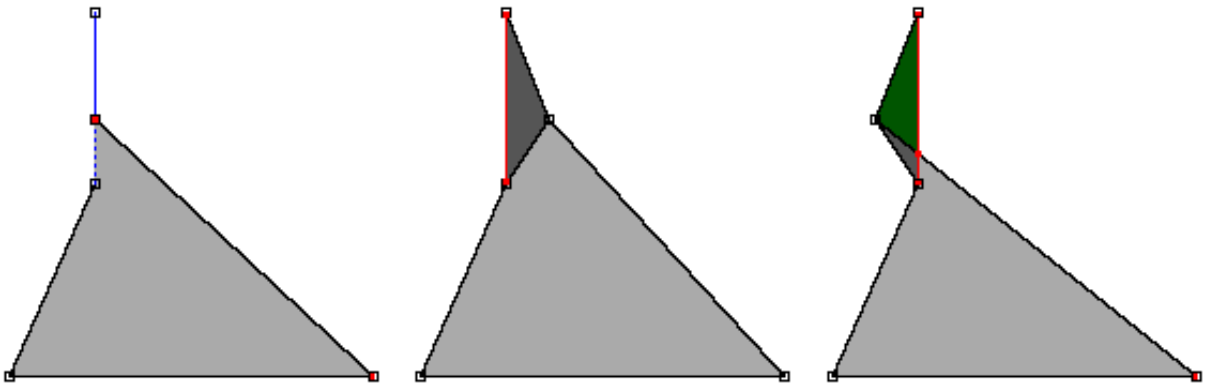


FIG. *7.35: Déplacements supplémentaires de sommets dans le polygone précédent.*

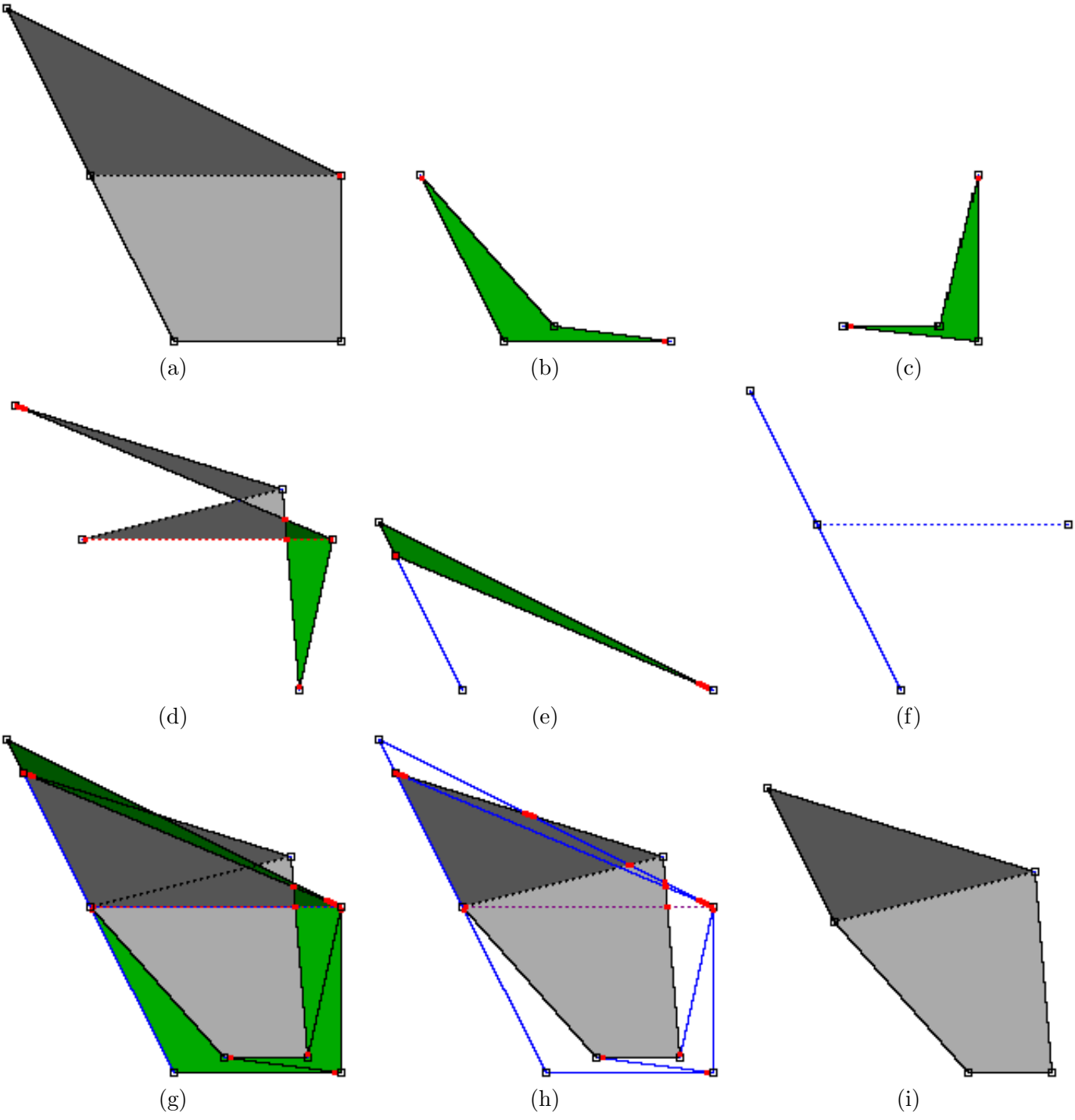


FIG. *7.36: Déplacement simultané de tous les sommets de la carte précédente. Dans ce cas, les sommets sont ordonnés de gauche à droite. (a) Configuration initiale. (b)-(f) Mini-cartes combinatoires correspondant à chacun des sommets. (g) Superposition de la carte initiale et des cinq mini-cartes. (h) Résultat du raffinement et de la complétion de labels avant suppression des arêtes marquées « à supprimer ». (i) Résultat de la suppression de ces arêtes.*

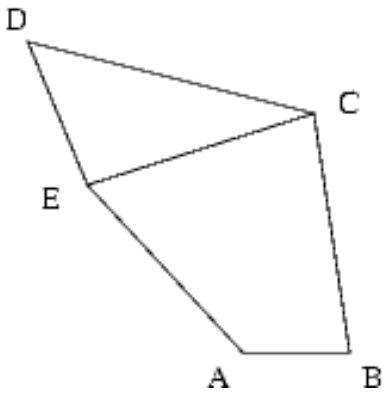


FIG. *7.37: Notations pour l'ajout de sommets.*

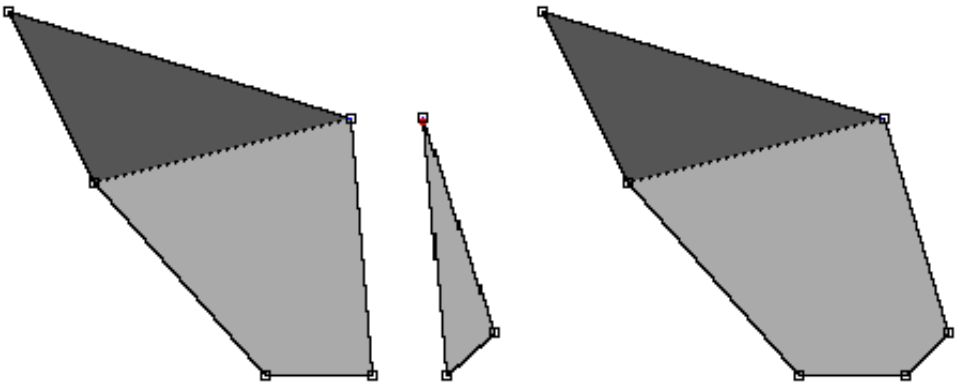


FIG. *7.38: Ajout de sommet sur l'arête [BC] de la carte précédente. (a) Configuration initiale. (b) Triangle défini pour l'ajout sur [BC]. (c) Résultat de l'ajout.*

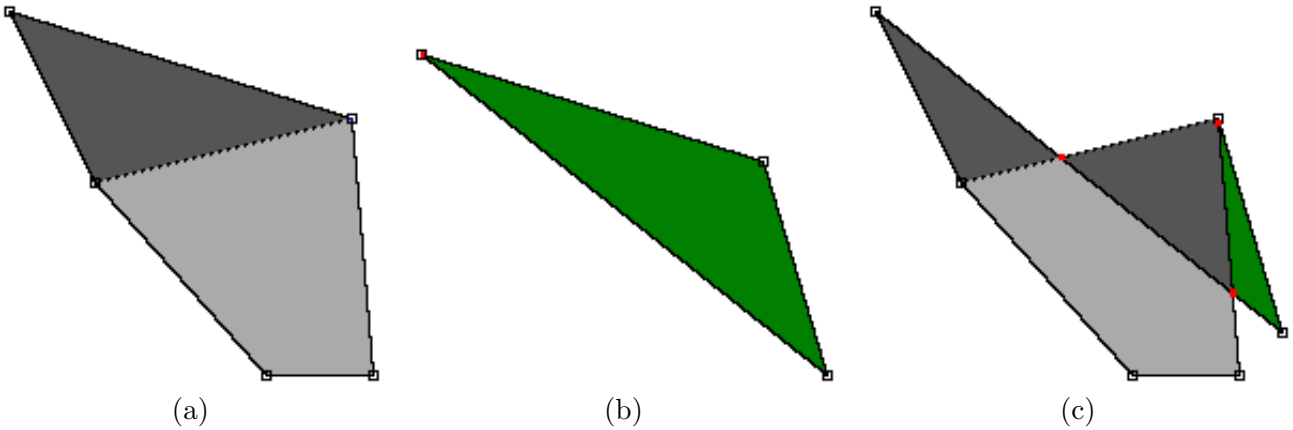


FIG. *7.39: Ajout du même sommet que pour la figure 7.38 mais cette fois sur l'arête [CD] de la carte précédente. (a) Configuration initiale. (b) Triangle défini pour l'ajout sur [CD]. (c) Résultat de l'ajout.*

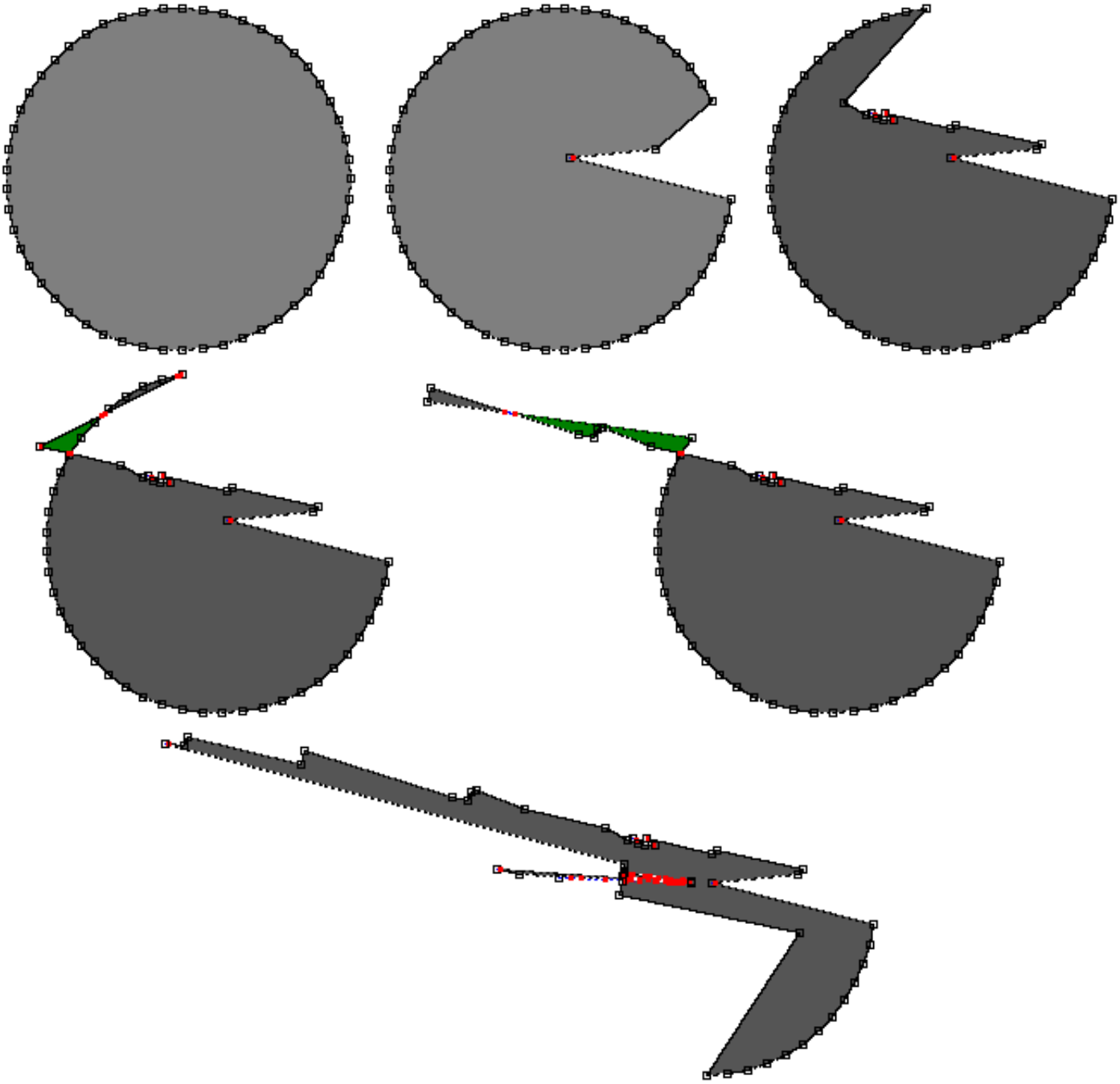


FIG. *7.40: Quelques étapes intermédiaires du déplacement séquentiel de sommets permettant de créer la carte combinatoire colorée discrète correspondant au polygone de la figure 7.43, suite à une projection sur un cercle.*

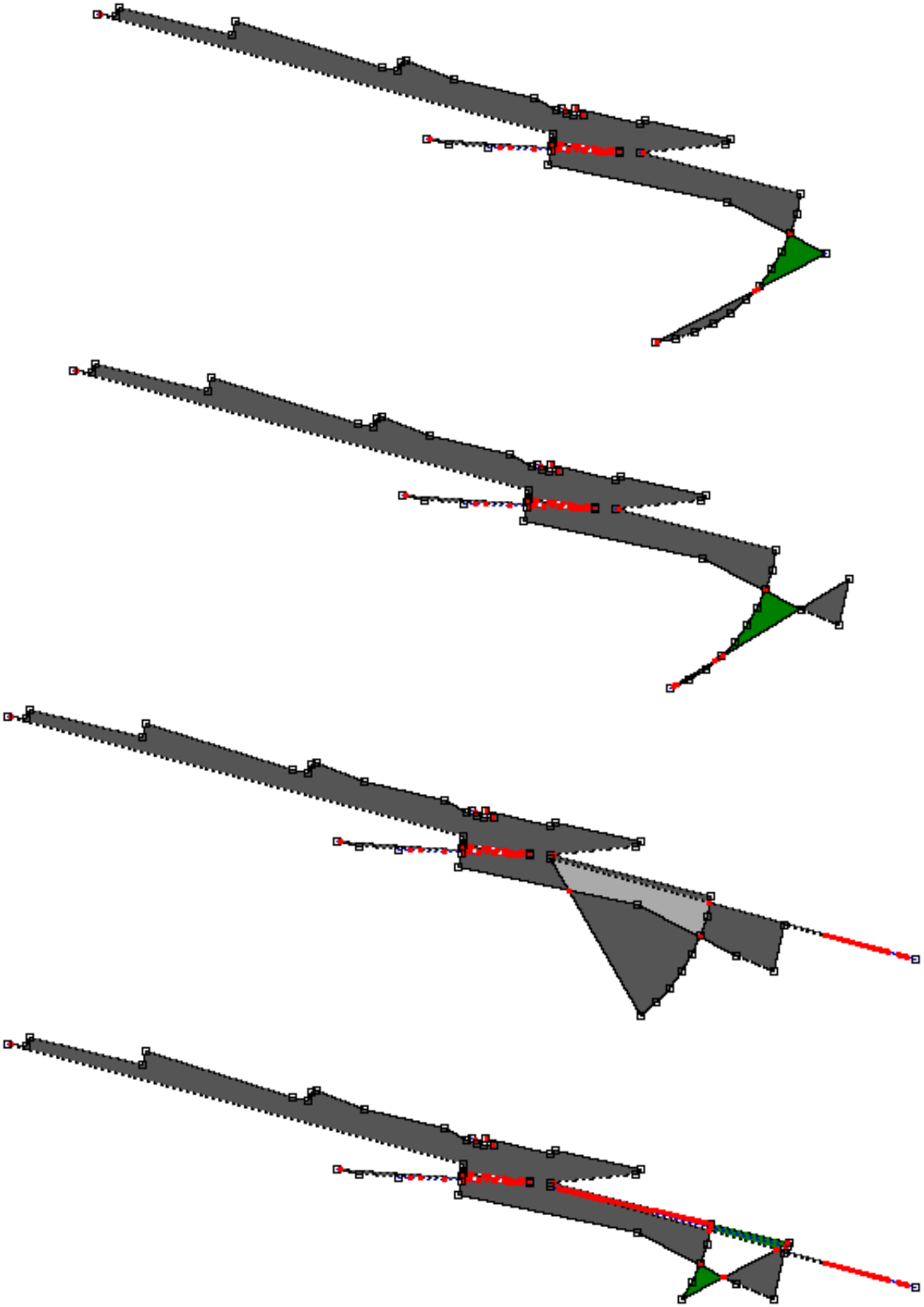


FIG. *7.41: Suite des étapes intermédiaires du déplacement séquentiel de sommets permettant de créer la carte combinatoire colorée discrète correspondant au polygone de la figure 7.43, après une projection sur un cercle (suite de la figure 7.40).*

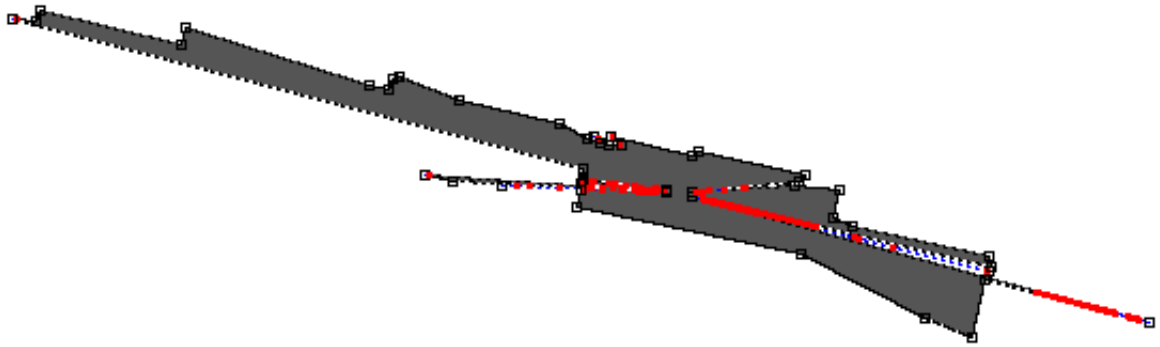


FIG. *7.42: Carte combinatoire colorée correspondant au polygone de la figure 7.43.*

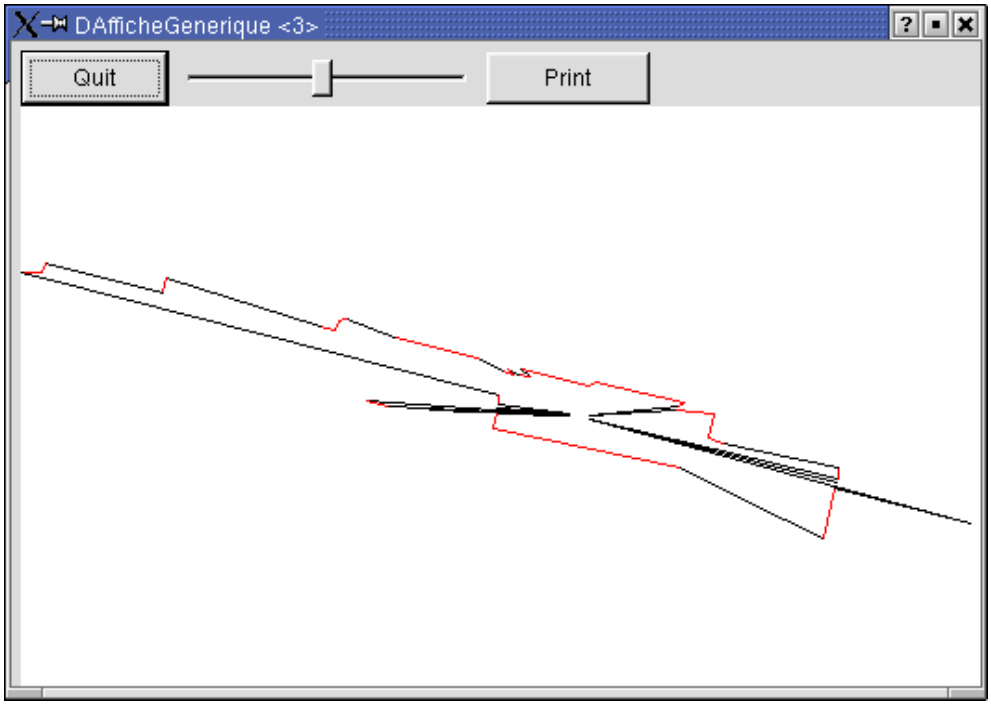


FIG. *7.43: Polygone extrait de l'un des scans utilisé au chapitre 5 : les segments « obstacles » apparaissent en rouge tandis que les segments « non obstacles » sont tracés en noir.*

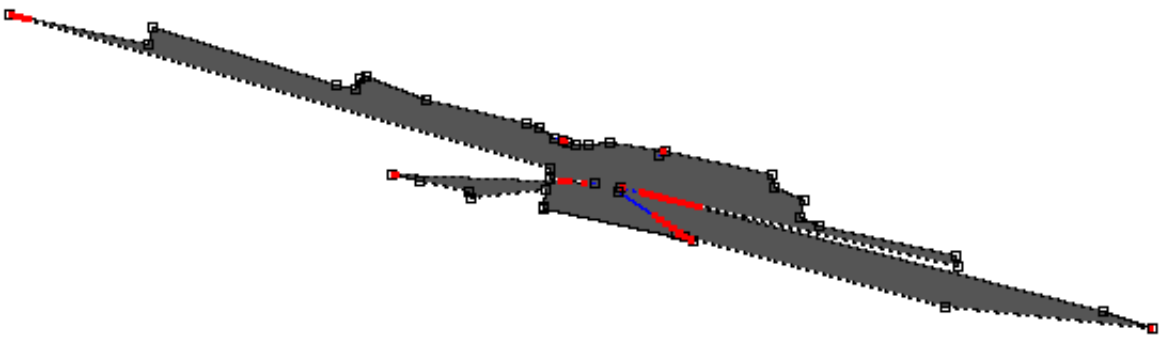


FIG. *7.44: Carte combinatoire colorée locale résultant également d'un déplacement de sommets, après ajout des points de « cassure ».*

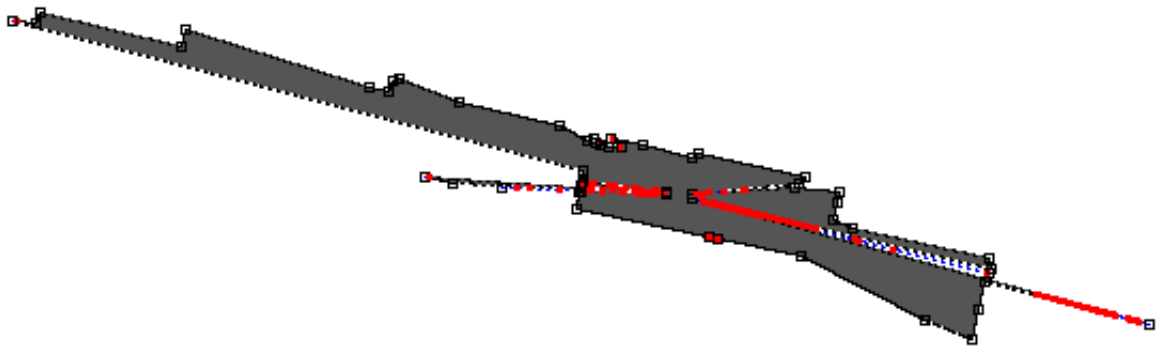


FIG. *7.45: Carte combinatoire colorée globale résultant du découpage des arêtes le long des « cassures » et du déplacement de sommets selon les nouvelles positions calculées par filtrage de Kalman.*

avec calcul des nouveaux degrés d'occupation. Pour mettre en œuvre la chaîne algorithmique complète liée à l'acquisition du nouveau scan laser, il a fallu moins de 40 secondes sur un ordinateur équipé d'un processeur Pentium IV. A l'heure actuelle, nous n'avons cependant pas encore cherché à optimiser notre programme en terme de temps de calcul : des accélérations nous semblent donc tout-à-fait possibles (en particulier via un traitement accéléré de certains cas dégénérés). Certaines pistes d'accélération liées à l'utilisation d'opérations plus locales sont d'ailleurs évoquées dans la section suivante.

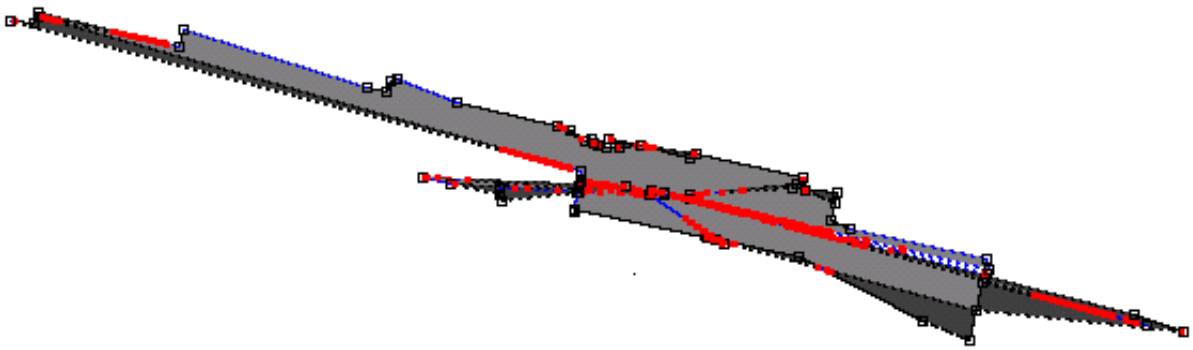


FIG. *7.46: Carte combinatoire colorée discrète finale, réalisée à partir des deux scans de la figure 5.17. du chapitre 5.*

La figure 7.47 illustre le même processus dans un cas où les cartes sont directement traitées dans leur version réelle au lieu d'être discrétisées [51]. Dans ce cas particulier, l'algorithme a pu être simplifié de manière à utiliser uniquement des cartes monochromes. On peut constater que les bords de la carte sont particulièrement nets par rapport aux cartes classiques issues des algorithmes de SLAM. On peut également remarquer que l'un des bords sur la partie haute de la carte n'a pas été apparié en raison de paramètres de mise en correspondances trop « prudents » : ce type d'incohérence est toutefois facile à repérer dans la structure de carte combinatoire comme une face de degré d'occupation non nul entourée de segments de type « obstacle ». Elle pourra ainsi être traitée comme nous le précisons dans la section suivante.

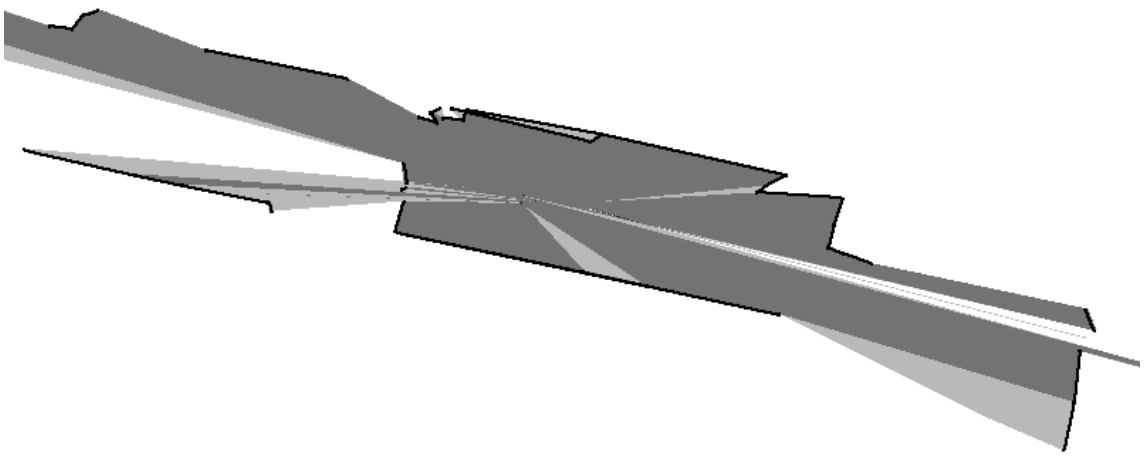


FIG. *7.47: La même carte que dans la figure 7.45 (avec un léger zoom), dans une version en nombres réels (non discrétisée).*

7.10 Conclusion

Dans ce chapitre, nous avons défini un nouvel outil, intitulé « carte combinatoire colorée », qui nous permet de gérer des configurations incohérentes provisoires. Associé à une nouvelle opération dite de « déplacement sûr » des sommets, cet outil permet de maintenir correctement les degrés d'occupations (même dans des situations pathologiques de croisement ou de retournement de polygones) et de rétablir une situation normale lorsque les incohérences transitoires disparaissent au gré des mises à jour géométriques successives. Cette représentation permet également de détecter certaines incohérences dans le processus de cartographie, en vue de les corriger.

Nous avons en outre défini deux stratégies pour la mise à jour géométrique et topologique de la carte globale suite à l'observation du polygone local d'espace libre. Pour chacune de ces stratégies, nous avons discuté de leurs avantages et de leurs inconvénients et précisé la complexité des algorithmes qui interviennent lors de cette étape.

La complexité des opérations utilisées est très dépendante de la configuration de la carte. Toutefois, dans le cas réel, avec la stratégie la plus efficace (la mise à jour incrémentale), on peut dire qu'elle est proche de $O(n \log n)$ en général (sauf dans des configurations très défavorables où elle passe en $O(n^2 \log n)$), n étant le nombre de brins (ou d'arêtes) de la carte. Dans le cas discret, il faut remplacer n par $n \times l$ (l étant la longueur discrète de la carte).

Ainsi, nous avons introduit une nouvelle représentation de carte colorée et des opérations de manipulation qui pourraient peut-être servir dans d'autres domaines que le SLAM, en particulier si l'on parvient à les étendre aux objets en trois dimensions (ce qui paraît tout-à-fait possible à première vue, à partir des cartes combinatoires classiques 3D et de l'opération de raffinement associée). La reconstruction d'objets 3D, l'animation (des objets flexibles notamment) et la simulation pourraient donc sans doute exploiter de la même manière des opérations de déformation de cartes combinatoires par déplacement ou ajout de sommets.

Parmi les perspectives envisageables, outre le passage au 3D, nous pouvons citer les pistes suivantes :
– une plus grande formalisation des définitions et des opérations décrites dans ce chapitre en vue de

vérifier leurs propriétés, à la manière de la thèse de Cazier [23] ;
- une accélération des opérations ;
- une mise en forme de la carte, notamment en vue de la simplifier et de remédier aux éventuelles incohérences.

Les deux derniers points sont détaillés dans les sections suivantes.

7.10.1 Accélération des opérations

Les opérations décrites ci-dessus pourraient sans doute être accélérées en travaillant sur des parties réduites de la carte. En particulier, lors du déplacement de sommets (déplacement un par un), les mini-cartes combinatoires utilisées sont généralement d'étendue limitée et le raffinement est a priori très localisé. Des opérations locales semblent toutefois plus faciles à gérer avec la représentation discrète qui évite les effets de bord. Plus généralement, si la carte globale considérée est vaste et contient de nombreux sommets, il pourrait être judicieux de la découper en sous-cartes de taille réduite, de manière à éviter de vérifier d'éventuelles interactions entre brins trop éloignés lors du balayage du plan mis en œuvre pour le raffinement.

Par ailleurs, il est envisageable d'éviter une restructuration topologique complète à chaque étape. En effet, l'appariement et le filtrage de Kalman ne nécessitent qu'une mise à jour de la partie noire de la carte. Le déplacement effectif des sommets, l'ajout des nouveaux polygones d'espace libre, le raffinement et le calcul des degrés d'occupation rouges pourraient être retardés et calculés en toile de fond jusqu'à ce que le système robotisé ait besoin d'une carte correcte.

7.10.2 Mise en forme de la carte

Cette mise en forme revêt plusieurs aspects. Il s'agit en premier lieu d'obtenir une carte exploitable par le robot : pour cela, nous devons détecter les éventuelles incohérences et chercher à les corriger. De plus, comme nous l'avons vu précédemment, la représentation ne cesse de s'agrandir au fil du temps. Ainsi, pour limiter la complexité de la construction de la carte, il serait utile de pouvoir la simplifier.

Repérer et corriger les incohérences :

Avec la première stratégie de mise à jour, il est possible de remonter aux polygones individuels. Nous sommes donc en mesure de détecter les incohérences existant au niveau de chacun de ces polygones, en particulier les croisements ou le retournement complet. Une solution drastique pour remédier à ces incohérences consisterait par exemple à retirer le polygone de la carte. Toutefois, les appariements réalisés avec ce polygone et leurs conséquences en terme de déplacement de sommets demeurent irréversibles.

Au niveau de la carte complète, les incohérences se matérialisent notamment par des degrés d'occupation négatifs (suite au retournement de polygones) ou par l'existence de segments « obstacles » délimitant deux régions observées au moins une fois comme étant libres. En particulier, le croisement de deux segments obstacles ou l'intersection d'un segment obstacle et d'un segment « non obstacle » conduisent nécessairement à ce type de situation. Concernant les degrés d'occupation rouges négatifs, on peut provisoirement les considérer comme nuls pour des applications de planification : ces zones doivent être considérées comme inconnues. Quant aux segments « obstacles » délimitant deux cellules de degré d'occupation non nul, une solution drastique consisterait par exemple à les transformer en segments « non obstacles » (cf. Fig. 7.48).

On remarque que cette dernière opération peut générer de nouveaux sommets noirs. Par exemple, sur la figure 7.48, le sommet A, initialement rouge (intersection de deux segments « non obstacles »), délimite désormais un segment « obstacle » et un segment « non obstacle ». Il faut donc le matérialiser si cette correction est considérée comme définitive, de manière à ce qu'il puisse évoluer avec les autres éléments de la carte. Par exemple, ce sommet et les covariances associées peuvent être créés à partir de la fonction qui calcule la position de l'intersection en fonction des positions des quatre extrémités de segments auxquels il appartient.

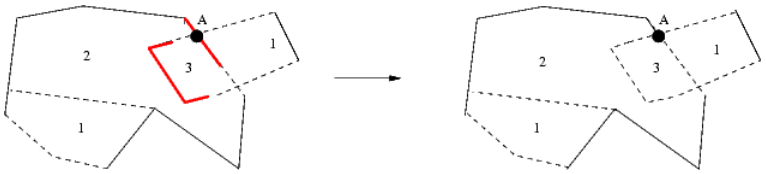


FIG. *7.48: Détection d'incohérences dans la carte globale : des segments « obstacles » (en rouge) délimitent deux cellules qui sont toutes deux de degré d'occupation non nul. Pour y remédier, ces segments sont transformés en segments « non obstacles ».*

Toutefois, les solutions drastiques proposées ci-dessus conduisent à une perte d'information très importante, alors que ces incohérences peuvent n'être que transitoires, entre deux estimations successives de la carte. De plus, elles ne conduisent pas nécessairement à une représentation plus juste de l'environnement et risquent de figer certaines configurations : dans l'exemple de la figure 7.48, le polygone de droite (formé des cellules de degré d'occupation 1 et 3) aurait peut-être dû être décalé vers la droite pour éviter de chevaucher la cellule de gauche (de degré d'occupation 2) et dans ce cas, l'intersection A n'existerait plus. Certaines incohérences sont d'ailleurs plus difficiles à déceler dans la carte complète : en particulier, certains retournements de polygones ne se traduisent pas par des degrés d'occupation négatifs car ils sont superposés à d'autres polygones d'espace libre. Des corrections moins draconiennes pourraient en revanche favoriser un compromis entre maintien de l'information et suppression des incohérences.

Ainsi, on pourrait notamment s'attacher à supprimer en priorité les zones incohérentes de type « slivers », qui correspondent à de petites superpositions erronées de polygones (cf. chapitre 3). Ces zones correspondent à des cellules dont une arête « obstacle » présente de part et d'autre un degré d'occupation rouge non nul. En particulier, si cette cellule est allongée et étroite, elle résulte peut-être d'un appariement manqué (si les degrés d'occupation noirs des lignes polygonales proches sont orientés dans le même sens) ou bien d'un léger dépassement (si les degrés d'occupation de ces lignes polygonales sont orientés dans des sens opposés) : la figure 7.49 en fournit une illustration. Les incertitudes sur les positions des arêtes et les degrés d'occupation rouges peuvent aider à trancher. Par exemple, une petite zone entourée uniquement de segments obstacles et observée une seule fois comme étant libre alors que des cellules voisines ont un degré d'occupation bien supérieur résulte a priori d'un dépassement et peut être supprimée.

Par ailleurs, du fait de la stratégie de fusion des segments appariés, un autre type d'incohérence risque d'apparaître sur les segments obstacles, sous la forme de petites arêtes chaînées en forme « Z ». En effet, l'ordre des sommets le long d'une ligne polygonale est en principe fixé une fois pour toutes. Pourtant, dans certains cas, il peut y avoir un doute sur cet ordre au moment de la mise en correspondance entre carte locale et carte globale (cf. Fig. 7.50).

Nous avons déjà évoqué un problème similaire lors de la définition de l'ordre de chaînage des points supplémentaires ajoutés sur une arête (points de cassure) : dans ce cas, nous avions proposé une solution locale en fonction de l'estimation courante des positions des points. Si le « Z » est aplati, les degrés d'oc-

FIG. *7.49: Détection de petites zones incohérentes dans la carte. Dans ces exemples, les arêtes en bleu (avec des degrés d'occupation noirs indiqués en bleu) correspondent à celles du dernier polygone local inséré dans la carte. Les degrés d'occupation noirs des autres arêtes sont indiqués en noir et les degrés d'occupation rouge modifiés suite à l'insertion de ce dernier polygone sont indiqués en rouge. (a) Les degrés d'occupation noirs des lignes polygonales proches sont orientés dans le même sens : il s'agit sans doute d'un appariement manqué. (b) Les degrés d'occupation de ces lignes polygonales sont orientés dans des sens opposés : il s'agit plutôt d'un dépassement.*

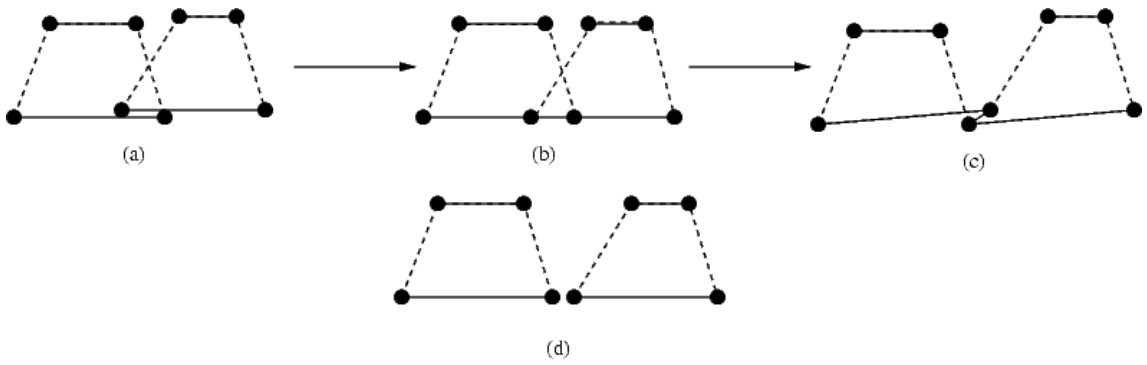


FIG. *7.50: Incohérences en forme de « Z » liées à une erreur de chaînage des sommets le long de la ligne polygonale. (a) Polygones d'espace libre initiaux, à leurs positions estimées. (b) Fusion des segments appariés et mise à jour par filtrage de Kalman. (c) Mise à jour ultérieure des positions des sommets : cette nouvelle estimation introduit une forme de « Z » sur la ligne polygonale obstacle. (d) Carte idéale (qui aurait été créée si la position initiale des polygones avait été correcte).*

cupation restent corrects mais le petit segment retourné (la barre oblique du Z) risque éventuellement de générer de mauvais appariements. Pour y remédier, on pourrait imaginer la création de points « glissants » au niveau des cassures, dont le déplacement pourrait se faire au travers des sommets adjacents, comme si le sommet « glissait » sur la ligne polygonale. On peut également envisager une détection géométrique explicite de ce type de configuration : la correction consisterait par exemple à rétablir le bon ordre de chaînage ou à transformer les arêtes retournées en arêtes non obstacles afin d'éviter les mauvais appariements.

A ce titre, on peut noter que pour la polygonalisation a posteriori d'une carte basée sur la superposition de scans bruts, Veeck et Burgard utilisent un certain nombre d'opérateurs élémentaires qui réalisent notamment les tâches suivantes [190] : suppression de superpositions de segments, fusion et suppression de lignes en zig-zag. Les deux premiers opérateurs pourraient nous servir pour la correction et la mise en forme générale de la carte tandis que le dernier opérateur pourrait peut-être s'adapter à notre problème de zig-zags, du moins pour leur détection.

Enfin, la stratégie globale de correction de la carte peut prendre diverses formes. D'abord, suivant les applications, ces corrections peuvent être manuelles (comme c'est souvent le cas pour les cartes géographiques) ou automatiques. En outre, la stratégie peut varier suivant la fréquence des corrections. On peut envisager une recherche d'incohérences à chaque ajout de polygone : dans ce cas, si ces incohérences sont trop nombreuses, il est possible de revenir en arrière en supprimant purement et simplement cet ajout de polygone. On pourrait également revenir sur le processus d'appariement. Une autre stratégie consisterait à réaliser ces corrections à chaque fois que l'on a besoin d'une carte correcte, pour effectuer une planification par exemple. Dans ce cas, les corrections peuvent être provisoires, de manière à conserver la réversibilité des configurations (par exemple, les degrés d'occupation négatifs doivent le rester si l'on espère corriger le retournement au gré des mises à jour futures). Ces corrections peuvent également être appliquées régulièrement sur la carte, afin de maintenir en permanence une représentation aussi cohérente que possible.

Mise en forme de la carte :

Pour limiter la taille de la représentation, il nous paraît indispensable de mettre en œuvre des opérations de « nettoyage » et de mise en forme régulière pour la simplifier (cf. Fig. 7.51) :

- fusion de deux sommets très proches qui sont soit consécutifs sur une même ligne polygonale, soit à l'extrémité de lignes polygonales « obstacles » distinctes ;
- suppression du sommet commun à deux segments « obstacles » adjacents et alignés ;
- fusion de cellules voisines présentant un degré d'occupation important (suppression des arêtes « non obstacles » communes).

Concernant la fusion de cellules voisines, il faut définir un seuil de fusion pour les degrés histogrammiques : les arêtes « non obstacles » délimitant deux cellules de degré histogrammique supérieur ou égal à ce seuil peuvent être supprimées. Comme précédemment, on remarque que ces suppressions peuvent générer de nouveaux sommets noirs (cf. Fig. 7.51 (e)). Si l'on souhaite les éviter, il faut uniquement supprimer des lignes polygonales candidates qui mènent aux extrémités à deux segments obstacles (cf. Fig. 7.51 (d)). Sinon, on peut les initialiser comme proposé ci-dessus.

Par ailleurs, jusqu'à présent, nous avons supposé implicitement que chaque nouveau polygone d'espace libre intersecte au moins une arête de la carte globale : la carte n'est constituée que d'une seule composante connexe et il n'existe pas de configurations où une composante connexe se trouverait strictement à l'intérieur d'une face de la carte (cf. Fig. 7.52 (a)). Toutefois, dans certains cas extrêmes (en particulier

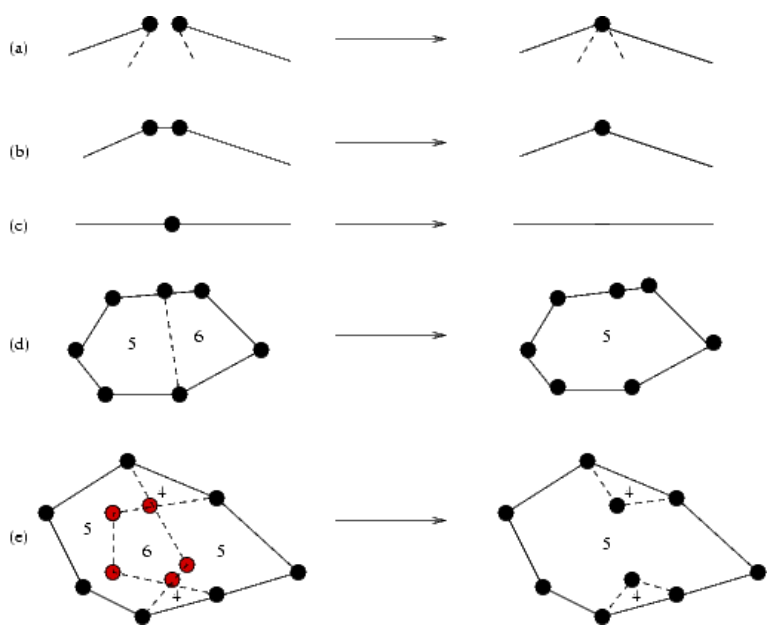


FIG. *7.51: Opérations de mise en forme de la carte. (a) Fusion de sommets proches à l'extrémité de lignes polygonales « obstacles » distinctes. (b) Fusion de sommets proches et consécutifs sur une même ligne polygonale. (c) Suppression d'un sommet séparant deux segments « obstacles » consécutifs alignés. (d) Suppression d'un segment « non obstacle » séparant deux cellules observées souvent comme libres. (e) Suppression d'une ligne polygonale « non obstacle » séparant deux cellules observées souvent comme libres.*

si les observations du robot ne sont pas intégrées régulièrement dans la carte), de telles configurations pourraient survenir. Elles pourraient également résulter de la suppression d'arêtes « non obstacles » dans les zones souvent observées comme libre lors de la mise en forme de la carte. Pour les gérer, il suffirait d'ajouter une arête virtuelle qui relierait cette composante connexe englobée à l'un des sommets de la face englobante (cf. Fig. 7.52 (b)) : de cette manière, on s'assure qu'elles sont bien prises en compte lors du raffinement et de la complétion de labels [23]. La détection de telles configurations pourrait se faire par le biais de méthodes géométriques, via un lancer de rayons par exemple [72].

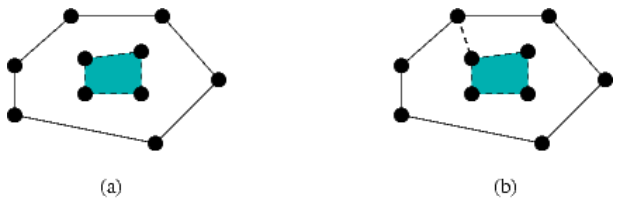


FIG. *7.52: (a) Exemple de composante connexe totalement incluse dans une autre. (b) Solution exploitant une arête virtuelle qui relie la composante connexe englobée à un sommet de la face englobante.*

Enfin, d'autres opérations de mise en forme ou d'analyse de la carte peuvent être envisagées en vue de préparer une description plus sémantique de l'environnement. Par exemple, il serait possible d'extraire les objets isolés dans l'espace libre, tels que des piliers ou des meubles. A priori, si les frontières de ces obstacles ont été complètement observées, ils se présentent sous la forme d'une région jamais observée comme libre (de degré d'occupation nul) et entourée de segments obstacles. On peut également imaginer une recherche de formes particulières dans la représentation : portes (ouvertes ou entrouvertes), couloirs, etc. Ces formes

pourraient ensuite être étiquetées dans la carte combinatoire au moyen de labels de sommets, d'arêtes ou de faces : les raisonnements spatiaux pourraient alors exploiter ce type d'informations. Non seulement l'extraction de tels objets faciliterait l'interaction avec l'homme mais elle pourrait également préparer la cartographie d'environnements dynamiques, exploitant la notion d' « objets mobiles », séparés du fond de carte fixe.

FIG. *7.53: Extraction d'objets particuliers dans la carte. (a) Exemple de porte. (b) Exemple de pilier.*

Chapitre 8

Conclusion

« Un tableau naît-il jamais d'une seule fois ? Non pas ! Il se monte pièce par pièce, point autrement qu'une maison. »
P. Klee (« Théorie de l'art moderne »)

8.1 Bilan

Dans le cadre de cette thèse sur la problématique de localisation et de cartographie simultanées (SLAM), nous avons commencé par comparer différents formats de cartes 2D afin de déterminer leurs avantages et leurs inconvénients pour les applications de robotique, en particulier à l'intérieur des bâtiments. Nous nous sommes d'abord intéressés aux modèles élémentaires (chapitre 2) pour lesquels nous avons défini un certain nombre de critères de comparaison : compacité, généricité par rapport au type de milieu, généricité par rapport aux capteurs, adaptation à la représentation de grands environnements, souplesse des modes de construction existants, exploitation par un robot et exploitation par l'homme. Il ressort de cette étude une complémentarité entre les différents modèles, en particulier entre les paradigmes topologiques et métriques d'une part et entre les approches métriques surfaciques et les approches à base d'amers géométriques d'autre part.

Nous avons ensuite recensé les modèles hybrides combinant différents types de modèles élémentaires (chapitre 3) et en avons proposé une classification originale. Nous avons alors constaté que les modèles de cartes utilisés par les roboticiens dans le domaine du SLAM demeurent souvent relativement peu structurés, à la différence des formats utilisés dans le domaine des systèmes d'information géographiques (SIG) par exemple. Une telle structuration offre cependant de nombreux avantages. En particulier, elle peut contribuer à l'accélération de certains traitements spatiaux via le stockage d'informations précalculées, à la vérification de cohérence globale de la carte via l'adjonction de contraintes structurelles et elle facilite en principe l'introduction d'informations sémantiques.

En conséquence, nous avons défini un modèle de carte très structuré par rapport aux modèles existants, qui combine de manière cohérente différents types de représentations (cf. chapitre 4) : représentation surfacique, représentation à base de primitives géométriques, grille d'occupation. Ce modèle, qui s'appuie sur un outil algébrique appelé « carte combinatoire », fournit également des informations topologiques telles que les liens d'adjacence, correspondant aux couches topologiques utilisées dans les systèmes d'information géographiques. En outre, il gère des incertitudes de différentes natures, à la fois géométriques (via la prise en compte des erreurs gaussiennes de position sur les éléments de la carte et de leurs corrélations) et topologiques (via les degrés d'occupation). De plus, nous avons défini une version discrète des cartes combinatoires qui fournit un lien direct avec les modèles de type « grilles d'occupation », tout en restant

plus compacte et plus structurée.

Nous avons ensuite expliqué comment construire automatiquement des cartes selon ce modèle, en utilisant un robot équipé d'un télémètre laser à balayage. Les techniques de cartographie existantes doivent être adaptées pour profiter des niveaux de représentation multiples, en particulier durant la phase délicate d'appariement entre la carte locale (issue des données laser courantes) et la carte globale en cours de construction. D'autres adaptations s'avèrent nécessaires afin de maintenir la structure de carte combinatoire. En effet, comme chaque arête est intrinsèquement liée aux arêtes adjacentes, ces liens topologiques induisent des contraintes sur l'ensemble du processus de cartographie. Celles-ci doivent être combinées aux contraintes spécifiques du SLAM, qui dérivent des corrélations entre les erreurs d'estimation de position des différents éléments de la carte. Dans notre implémentation, nous avons opté pour une approche basée sur le filtre de Kalman étendu qui maintient l'ensemble des corrélations entre ces erreurs d'estimations. L'algorithme global se décompose en trois étapes :

- Mise en correspondance de la carte locale (issue du télémètre laser) et de la carte globale en cours de construction (chapitre 5). Cette mise en correspondance fait notamment appel à une technique de programmation dynamique qui tire parti des liens d'adjacence entre segments : il s'agit d'apparier des chaînes polygonales plutôt que des segments individuels. Cette approche impose en outre la gestion des appariements multiples (tantôt acceptés, tantôt écartés), qui sont très rarement pris en compte dans les algorithmes de SLAM (où l'on utilise généralement de mises en correspondance individuelles, faisant souvent intervenir des approches gloutonnes). Il a donc fallu définir des critères heuristiques de compatibilité entre segments et des scores d'appariement tenant compte des informations d'incertitudes (qui correspondent *grosso modo* aux longueurs des portions de segments appariées, modulées par le score du test de Mahalanobis). En outre, un prétraitement de type « scan-matching » permet d'améliorer la robustesse de cette opération. La mise en correspondance fournit alors en sortie un « chemin d'appariement » entre les données locales (polygone d'espace libre local) et la carte globale en cours de construction.

- Mise en œuvre du filtre de Kalman étendu afin de calculer les nouvelles estimations de position des éléments de la carte globale en fonction des observations réalisées suivant la carte locale (chapitre 6). Par rapport aux méthodes classiques, ce filtre gère les observations qui correspondent à des « cassures » sur les segments de la carte globale. Nous avons montré que trois types d'observations permettent de traiter l'ensemble des cas (observations « sommet global sur sommet local », « sommet global sur droite locale » et « cassure » c'est-à-dire « sommet virtuel global sur sommet local »). Pour mettre en œuvre cette approche, nous avons défini une méthode qui permet d'extraire les observations selon ces trois catégories à partir du chemin d'appariement issu de l'étape précédente.

- Mises à jour géométriques et topologiques (chapitre 7). La mise à jour géométrique permet de déplacer les éléments de la carte selon la nouvelle estimation de position issue du filtrage de Kalman, tout en maintenant la cohérence du modèle de carte combinatoire. La mise à jour topologique revient à calculer les nouveaux degrés d'occupation des cellules de la carte (nombres de fois où une cellule a été observée comme libre d'obstacle). Pour cela, nous avons défini un nouveau concept de « carte combinatoire colorée », dans sa version réelle et discrète, ainsi que diverses opérations permettant de les manipuler (raffinement coloré et complétion de labels associée, déplacement de sommets, ajout de sommets...). Par rapport aux cartes combinatoires « monochromes » classiques, ce modèle permet notamment de gérer des incohérences transitoires du modèle telles que des retournements de cellules ou des croisements.

Nous décrivons en outre des opérations de mise en forme qui peuvent être activées à intervalles réguliers pour garantir la lisibilité de la carte, limiter sa complexité et vérifier sa cohérence (possibilité de correction explicite des erreurs).

Différentes expérimentations réalisées sur des données simulées et sur des données réelles ont permis d'illustrer ces mécanismes et de montrer la faisabilité de la chaîne complète de construction incrémentale. Le caractère très structuré du modèle de cartes employé induit des traitements plus complexes que dans le cas des modèles habituels (grilles d'occupation, superposition de données télémétriques brutes ou ensemble de primitives de type points ou segments). Toutefois, nous avons montré que la complexité théorique de ces traitements demeure proche de celle des approches classiques fondées sur le filtrage de Kalman et que l'algorithme global ne nécessite de régler qu'un nombre réduit de paramètres (variance des bruits de mesure extraite par calibrage, quatre seuils pour la polygonalisation, un pas de discrétisation des histogrammes pour le recalage global, un seuil d'appariement probabiliste pour la mise en correspondance de chaînes polygonales, un facteur régissant l'incertitude de position des sommets virtuels pour le filtrage de Kalman et un pas de discrétisation des cartes combinatoires). En outre, la structuration et la richesse du modèle permettent en principe d'améliorer la robustesse de la phase d'association de données (lors de l'appariement carte locale / carte globale, voire lors du bouclage de cycles) et fournit des informations utiles en vue de raisonnements spatiaux de haut niveau, pour la planification par exemple. La structure de carte combinatoire, bien formalisée, facilite également la vérification de certaines propriétés (terminaison, convergence...) sur les algorithmes mis en œuvre, ce qui favorise la sûreté de fonctionnement du système robotisé. Enfin, les similitudes avec les modèles géographiques offrent plusieurs avantages : possibilité de transposer les requêtes classiques effectuées sur les SIG aux applications de robotique et possibilité d'employer des opérations de manipulation de SIG pour gérer les cartes construites par les robots.

8.2 Perspectives

Les premières perspectives concernent les tests expérimentaux. L'ensemble de la chaîne algorithmique de construction de cartes représente environ 100 000 lignes de code en langage C++ écrites au CEP Arcueil, dont plus de la moitié développées exclusivement dans le cadre de cette thèse. La vérification et le test extensif de tous ces programmes nécessiterait donc un travail très conséquent qu'il serait intéressant d'effectuer en vue d'obtenir des résultats sur des cartes de plus grandes dimensions.

Par ailleurs, nous proposons ci-dessous quelques pistes d'améliorations et des possibilités d'extension concernant le processus de cartographie et l'exploitation des cartes construites.

8.2.1 Vers une évaluation plus systématique de la qualité des cartes

Pour faire progresser les techniques de SLAM et pour obtenir des systèmes de cartographie opérationnels et performants, qui répondent à des spécifications de besoin précises, il nous paraît indispensable de pouvoir évaluer correctement la qualité des représentations construites par ces systèmes, à divers stades de leur développement. En particulier, il s'agit d'élaborer des méthodologies d 'évaluation qui soient la fois :

- quantitatives (pour s'affranchir des jugements purement visuels et qualitatifs des représentations construites),
- automatiques (pour limiter le travail de l'opérateur d'évaluation, qui peut parfois s'avérer très fastidieux),
- reproductibles (pour pouvoir tester tous les systèmes dans des conditions équivalentes),

- comparatives (pour comparer la qualit de plusieurs représentations similaires ou de nature différente),
- et représentatives de la tâche considérée ainsi que des conditions environnementales prévues pour les systèmes considérés.

C'est pourquoi nous proposons en annexe (« Annexe 2 : Vers une évaluation plus systématique de la qualité des cartes ») un tour d'horizon des méthodes d'évaluation existantes et quelques techniques ou métriques d'évaluation susceptibles de répondre aux exigences mentionnes ci-dessus.

8.2.2 Améliorations de la représentation

Nous avons évoqué au chapitre 4 des extensions possibles de notre modèle qui permettraient de traiter des environnements 2D statiques plus variés. En particulier, dans des milieux intérieurs peu structurés présentant de nombreux points isolés dûs à de petits obstacles tels que des pieds de meubles, il serait intéressant d'envisager un traitement séparé de ces objets de petites dimensions, dans une autre couche de représentation. Cela éviterait notamment de générer de nombreuses lignes de visée parasites. La structure de carte combinatoire unique pourrait toutefois être préservée en raccordant ces petits objets à la carte globale par le biais d'arêtes virtuelles (cf. chapitre 7). Par ailleurs, la structure de carte combinatoire n'est pas réservée aux subdivisions polygonales : le plongement des arêtes peut être courbe, ce qui permettrait de gérer plus explicitement les surfaces non rectilignes dans l'environnement.

Par ailleurs, nous avons vu que notre représentation pourrait bénéficier de diverses opérations de mise en forme et de correction explicite des erreurs : fusion des cellules voisines souvent observées comme libres, fusion des sommets proches et des segments adjacents parallèles, détection des retournements de cellules, gestion de l'inversion de l'ordre de chaînage des sommets (configurations en « Z »), détection et correction des superpositions de segments « obstacles », etc.

En outre, un découpage hiérarchique des cartes combinatoires de grandes dimensions permettrait sans doute d'accélérer les calculs de raffinement local et la recherche des primitives appariées lors de la consruction de la carte. Plus généralement, un tel découpage pourrait contribuer à l'efficacité des opérations exploitant la carte telles que la planification de trajectoires [205] ou la mise en œuvre de raisonnements spatiaux, exploitant notamment des informations sémantiques (partitions multiples [74]...). Il pourrait également favoriser l'adjonction d'index spatiaux, utiles aux opérations de pointage ou de fenêtrage employées dans les SIG [39].

Enfin, il serait intéressant d'enrichir notre représentation afin d'en exploiter pleinement la structure, et notamment l'existence des liens topologiques entre éléments. En particulier, nous avons vu au chapitre 4 que l'adjonction d'informations sémantiques (issues par exemple de la reconnaissance d'objets dans les données visuelles) pouvait faciliter l'interaction homme / robot et la mise en œuvre de raisonnements spatiaux de plus haut niveau.

8.2.3 Un algorithme d'estimation plus élaboré

Notre méthode de construction de cartes est fondée classiquement sur un filtre de Kalman étendu. Nous avons vu au chapitre 2 que ce type de méthodes a fait l'objet ces dernières années de nombreuses améliorations afin d'accélérer les temps de calcul et de limiter les problèmes de linéarisation. En particulier, en vue de limiter la complexité du processus de filtrage, il semble que les méthodes de report temporel [40] [99] [200] [80] soient applicables à notre représentation : il s'agirait alors de travailler momentanément sur une partie bornée de l'état avant de propager les corrections au reste de la carte. En outre, ces méthodes impliquent souvent un découpage de la carte qui permettrait en principe d'accélérer par la même occasion les opérations de mise à jour géométrique et topologique (cf. section précédente).

Les techniques de SLAM hiérarchique (vues au chapitre 3) paraissent également prometteuses [67], puisqu'elles assurent en ligne la cohérence de la carte (suite aux fermetures de cycle) et gèrent à la fois les problèmes de linéarisation et de coût de calcul important. Toutefois, à l'heure actuelle, elles ne proposent pas de mécanisme permettant de réestimer la position de primitives communes à deux sous-cartes : on obtiendrait alors un ensemble de cartes combinatoires locales non fusionnées, ce qui risquerait de nuire à la mise en œuvre de raisonnements spatiaux globaux.

Les approches fondées sur la « Rao-Blackwellisation » (FastSLAM [137]) nous paraissent tout aussi intéressantes et offrent l'avantage d'être directement applicables à notre représentation, qui garde déjà en mémoire les positions successives du robot (cf. chapitre 2). Ces positions successives seraient alors modélisées par des particules tandis que chaque sommet de la carte pourrait être réestimé via un filtre de Kalman individuel (où l'état serait représenté par un vecteur de dimension 2 seulement). En revanche, si l'on souhaite tirer parti des hypothèses d'association de données multiples, il faudrait maintenir une carte combinatoire par particule, ce qui risque de s'avérer très coûteux en temps de calcul si le nombre de particules est important. Cela permettrait toutefois de gérer certaines difficultés de mise en forme, telles que le changement de l'ordre de chaînage des sommets sur les arêtes « obstacles » (configurations en « Z » - cf. chapitre 7).

Quant aux méthodes qui s'appuient spécifiquement sur des représentations flexibles de l'environnement (superposition de scans bruts notamment, susceptibles de se déplacer librement les uns par rapport aux autres pour maximiser l'appariement lors des estimations successives de la carte) [179], elles paraissent peu compatibles avec notre modèle, qui exploite la fusion progressive (et irréversible) des primitives appariées dans la carte. Toutefois, on pourrait imaginer de transformer les informations obtenues localement par ces techniques de « dense SLAM » en cartes combinatoires (en plus de l'extraction de polylignes [190], il faudrait calculer les incertitudes de position de ces polylignes ainsi que les degrés d'occupation) puis de fusionner ces sous-cartes structurées en une seule. Cela impliquerait également de gérer la mise en correspondance de deux cartes combinatoires au lieu d'une carte et d'un simple polygone d'espace libre. On obtiendrait alors une représentation dense et flexible localement (ce qui permettrait de retarder certaines décisions d'appariement) puis, une fois cette représentation locale considérée comme correcte, on pourrait la structurer et la fusionner avec les autres cartes locales dans une représentation globale compacte et structurée.

8.2.4 Extensions à d'autres types d'environnement

Dans le cadre de cette thèse, nous nous sommes limités aux environnements 2D statiques relativement bien structurés. Toutefois, dans certaines applications concrètes, il pourrait être utile de savoir cartographier des environnements moins structurés : on peut songer par exemple à des applications de reconnaissance en milieu urbain, où certains bâtiments ont été partiellement détruits. Dans ce cas, l'hypothèse de sol plan risque d'être mise à mal et il est à craindre qu'une carte 2D n'apporte que peu d'informations. C'est pourquoi le passage au 3D peut s'avérer nécessaire. Plus généralement, le SLAM 3D favoriserait également la navigation autonome de robots bipèdes ou de microdrones. La plupart des techniques actuelles exploitent des superpositions de données laser 3D brutes, éventuellement maillées (dans ce cas, il semble toutefois que ce maillage soit définitif et qu'il ne soit pas possible de le raffiner via de nouvelles observations [179]), ou des modèles volumiques (découpage de l'espace en voxels). Nous pensons toutefois qu'à l'instar du cas 2D, des représentations plus structurées pourraient faciliter l'exploitation de la carte et améliorer la robustesse du processus de cartographie.

Il se trouve que les cartes combinatoires et l'opération de raffinement associée sont généralisables en 3D [23] : a priori, il devrait en aller de même pour les cartes combinatoires discrètes et colorées. Ainsi, l'ensemble de notre représentation peut être définie en 3D (les degrés d'occupations de cellules peuvent être directement exprimées en 3D, de même que les incertitudes en position des sommets). De même, le filtre de Kalman et la définition des observations associées est en principe généralisable au 3D (il suffirait d'introduire des observations de points sur plan et des points virtuels sur les faces pour les cassures). En revanche, notre technique de mise en correspondance entre carte locale et carte globale paraît plus difficile à mettre en œuvre dans le cas 3D : en particulier, la programmation dynamique ne pourrait plus exploiter directement le fait que la carte locale est ordonnée (en 2D, le polygone présentait naturellement une séquence de segments). Il faudrait donc revoir en particulier la méthode d'appariement.

Quant aux environnements dynamiques, ils ont déjà fait l'objet de travaux spécifiques, avec des représentations à base de scans bruts [193] ou de grilles d'occupation [1] notamment. Le fait que notre représentation permette d'exprimer assez naturellement la notion d'objet (agrégation de cellules connexes ou extraction aisée d'un objet isolé dans un espace libre - cf. chapitre 7) faciliterait peut-être la mise en place d'algorithmes de détection de mouvement, de reconnaissance et de suivi d'objets dynamiques [1]. Dans une certaine mesure, l'identification de ces objets (homme, robot...) permettrait en outre de prédire leurs mouvements, comme le suggèrent Pradalier et al. par exemple [158].

8.2.5 Exploitation de la carte

Nous avons vu au chapitre 4 que dans un cadre robotique, la structuration et la richesse de notre représentation (et en particulier la couche topologique) pouvait contribuer à diverses applications : localisation (combinaison de plusieurs couches du modèle), planification de trajectoires (utilisation des informations topologiques pour réaliser une planification grossière et des informations métriques pour la définition plus fine des trajectoires...), exploration (extraction efficace des frontières de l'espace libre restant à explorer notamment), raisonnements spatiaux plus élaborés exploitant les opérations booléennes voire des informations sémantiques (opérations typiques des SIG telles que le fenêtrage ou la superposition de couches thématiques, requêtes plus complexes dans le cadre d'interactions homme / machine, etc.). Plus généralement, la richesse de la carte permet de combiner des algorithmes destinés à différents types de représentations, tandis que la structuration facilite a priori la mise en œuvre des raisonnements spatiaux de haut niveau. En outre, la définition formelle des cartes combinatoires favorise la vérification de certaines propriétés concernant les opérations de manipulation de ces modèles, ce qui peut contribuer à la robustesse des traitements correspondants.

Enfin, on peut remarquer que si à l'avenir, les capteurs proprioceptifs deviennent plus précis (avec le progrès et la miniaturisation des capteurs inertiels) et si les techniques de localisation par satellites s'améliorent (réception du signal à l'intérieur des bâtiments et précision accrue), le problème de localisation tendra à s'effacer. Le problème de cartographie ne disparaîtra pas pour autant si les capteurs extéroceptifs restent imprécis et entachés d'erreurs. En particulier, la question de la structuration et de la fusion successive des données locales devrait demeurer, ce qui plaide en faveur de travaux sur la mise en place de modèles compacts et structurés (dans l'esprit des post-traitements proposés récemment par Veeck et Burgard [190] ou des opérations de géométrie discrète décrits il y a quelques mois par Lakaemper et al. [110]), dans la lignée de ce travail de thèse.

Annexe 1 : Pour aller plus loin...

Dans le cadre de cette thèse, nous avons proposé une chaîne algorithmique complète permettant de construire de manière incrémentale des modèles d'environnement riches et structurés basés sur la notion de cartes combinatoires. En particulier, nous avons introduit une représentation originale qui combine des éléments issus à la fois du domaine du SLAM (représentation de l'incertitude et maintien des informations de corrélation entre primitives géométriques) et du domaine de la géométrie algorithmique (cartes combinatoires). Pour les besoins de l'algorithme de construction de cartes, nous avons dû en outre enrichir ce modèle au moyen de nouveaux éléments (« sommets virtuels » pour modéliser les cassures) et de nouvelles structures de données (cartes combinatoires discrètes, cartes combinatoires colorées...).

Nous avons également introduit diverses opérations qui exploitent la structuration du modèle, et notamment la couche topologique maintenant les liens d'adjacence entre les divers éléments de la carte, dont certaines pourraient probablement servir dans d'autres domaines que le SLAM (tels que la synthèse et le traitement d'images, la reconnaissance des formes, la géométrie algorithmique, etc.) :

- un algorithme de mise en correspondance entre chaînes polygonales qui exploite les informations d'incertitude sur la position des arêtes à apparier ;
- une méthode d'extraction des observations (destinées au filtre de Kalman) à partir du chemin d'appariement fourni par le précédent algorithme (cette méthode exploitant la notion de sommets virtuels de cassure) ;
- des opérations de manipulation de cartes combinatoires discrètes et colorées (raffinement et complétion de labels, déplacement « sûr » et réversible des sommets de manière à conserver des informations liées au degré d'occupation des cellules).

Cependant, c'est la mise au point de l'algorithme global qui constitue le cœur de notre travail, avec l'exploitation conjointe des informations de corrélation et d'adjacence entre primitives géométriques, ainsi que le chaînage cohérent de chaque phase de mise à jour de la carte combinatoire globale à partir des observations locales. En outre, comme nous l'avons indiqué dans les chapitres précédents, nous nous sommes attachés à réduire le nombre de paramètres (une douzaine au total, dont certains, comme les bruits de mesure sont extraits par calibrage) ainsi que la complexité algorithmique, qui demeure proche des algorithmes classiques de SLAM opérant sur des modèles moins sophistiqués.

L'implémentation de l'ensemble de cette chaîne algorithmique représente un travail conséquent puisqu'elle correspond à plusieurs dizaines de milliers de lignes de code, qui nécessiteraient d'être testées de manière plus intensive (ce qu'il n'était pas possible de réaliser dans le cadre d'une seule thèse). En outre, nous avons dégagé plusieurs pistes d'amélioration concernant certaines parties de la chaîne algorithmique et nous avons proposé des opérations de mise en forme de la carte (en vue de détecter puis de corriger les éventuelles incohérences et de limiter l'encombrement de l'espace mémoire) qui n'ont pas été testées. C'est pourquoi nous pensons qu'il serait intéressant de poursuivre ce travail afin d'obtenir des résultats expérimentaux plus globaux et de raffiner les algorithmes.

Ainsi, l'objet de cette annexe est de préciser les travaux à réaliser afin de compléter les expérimentations effectuées dans le cadre de cette thèse et à plus long terme en vue d'améliorer les performances des algorithmes proposés. En particulier, elle synthétise et apporte quelques compléments aux orientations indiquées à la fin des chapitres 5 , 6 et 7 ainsi qu'en conclusion. Le lecteur pourra donc se reporter à ces différents chapitres pour obtenir certains détails ainsi que des éléments de contexte.

8.3 Améliorations envisageables à court et moyen terme

8.3.1 Améliorer la mise en correspondance

Les pistes d'amélioration indiquées ci-dessous correspondent aux algorithmes décrits au chapitre 5.

*** Polygonalisation :**

L'algorithme de segmentation de scans que nous avons utilisé fournit en général des résultats corrects, même s'il présente parfois quelques imperfections (coins rognés notamment). Il pourrait sans doute être amélioré en s'inspirant par exemple des méthodes proposées par Charbonnier [25] (pour la détection des points anguleux ou de points de décrochement) et par Castellanos et Tardos [21] (pour la distinction entre portions obstacles et non obstacles et l'extraction des segments successifs). Il serait également intéressant d'étudier si des méthodes récentes de segmentation de scans superposés, telles que celle de Veeck et Burgard [190] basée sur l'algorithme E-M, pourraient être exploitées, afin de garantir le chaînage des segments adjacents : en particulier, il faudrait vérifier s'il est possible, avec une telle méthode, de calculer les incertitudes concernant la position des segments extraits.

Une autre piste consisterait à combiner des données issues d'un télémètre laser et les images acquises simultanément par un capteur omnidirectionnel catadioptrique, ce qui permettrait par exemple d'utiliser la détection d'arêtes verticales dans les images (des radiales si le capteur est placé verticalement) pour évaluer la position des points anguleux et des lignes de visée (ruptures verticales entre cloisons). Enfin, il semble intéressant d'extraire et de stocker les petits objets (tels que les pieds de table) dans une couche à part de manière à éviter les longues lignes de visée qui « parasitent » l'espace libre : il faudrait alors étudier les conséquences de cette dissociation sur l'ensemble de la chaîne algorithmique de construction de carte.

*** Recalage global :**

Une garantie supplémentaire de robustesse de la phase de recalage consisterait à exploiter la couche de représentation surfacique de notre modèle (en particulier la grille d'occupation que l'on peut extraire facilement dans le cas des cartes combinatoires discrètes). Par exemple, il s'agirait de combiner la technique actuelle avec un appariement entre grilles d'occupation locale et globale [168] [81] pour vérifier la cohérence du recalage obtenu via la corrélation d'histogrammes (suivie éventuellement d'une phase d'ICP).

Le calcul de l'incertitude de recalage pourrait également être revu : il s'agirait par exemple de comparer la pertinence d'une extraction de cette incertitude par analyse de la largeur du pic de corrélation d'histogrammes, par propagation d'incertitude à travers l'optimisation par moindres carrés (pour l'ICP) et par corrélation entre grilles d'occupation [168] [129]. Une autre possibilité consisterait à recourir à un système de vote pondéré en prenant différentes hypothèses de départ pour le calcul d'appariement

(échantillonnées à l'aide des informations odométriques), à la manière de Wang et Thorpe [194] : chaque résultat final serait évalué (de manière à estimer la pondération à appliquer) et la distribution représentée par l'ensemble des échantillons finaux pondérés permettrait d'évaluer l'incertitude du recalage.

*** Extraction des hypothèses d'appariements individuels :**

Pour accélérer la recherche de segments globaux candidats à l'appariement avec un segment local donné, il serait utile de restreindre cette recherche à une zone d'intérêt positionnée autour du segment local et de largeur proportionnelle à son incertitude en position [139].

Par ailleurs, il semble intéressant d'introduire les incertitudes de position des sommets lorsque l'on teste si deux segments se situent bien « l'un en face de l'autre » (via un calcul des projections orthogonales), pour éviter de rejeter des hypothèses acceptables lorsque le recalage global n'a pas fourni un résultat de bonne qualité ou lorsqu'il existe un décalage important entre polygone local et carte globale suite à un bouclage de cycle.

*** Mise en correspondance de chaînes polygonales :**

Ici encore, l'introduction des incertitudes de position des sommets permettrait sans doute d'améliorer les tests de compatibilité entre couples candidats à l'appariement, notamment dans le cas où l'incompatibilité résulte d'une superposition très réduite des projections orthogonales.

En outre, le passsage à une modélisation plus locale des incertitudes, de type SP-Maps [21], rendrait a priori les tests de Mahalanobis plus pertinents : en effet, à l'heure actuelle, les estimations d'incertitude de position absolue des segments ne sont pas indépendantes du repère global considéré (par exemple, en coordonnées polaires, si un mur est très loin de l'origine, une faible incertitude en orientation d'un mur donne lieu à une variance très importante pour la distance du segment à l'origine), ce qui peut fausser l'appariement. L'exploitation de repères locaux comme dans les SP-Maps éviterait cet inconvénient.

Par ailleurs, réaliser un test de compatibilité globale [145] sur les couples de segments appariés dans le chemin fourni par programmation dynamique offrirait un garde-fou supplémentaire pour éviter les appariements erronés.

Enfin, il est possible que la stratégie globale d'appariement, basée aujourd'hui sur une technique de programmation dynamique, puisse être améliorée en posant le problème différemment, par exemple comme un problème d'ordonnancement, ce qui conduirait à de nouvelles méthodes de résolution.

8.3.2 Améliorer la mise en œuvre du filtrage

Comme nous l'avons indiqué au chapitre 6, l'introduction des positions successives du robot dans le vecteur d'état permettrait généralement de limiter la taille de ce vecteur : en effet, de cette manière, il n'est plus nécessaire de maintenir l'estimation de position des sommets placés aux intersections de segments « non obstacles » (qui ne seront plus observés) car seule la position relative du sommet observé par rapport à la position correspondante du robot (lors de leur observation) suffit.

Par ailleurs, la gestion des points virtuels pourrait être améliorée car la grande incertitude associée à ces points risque de conduire à des problèmes de linéarisation ou de divergence des calculs numériques (lors de l'inversion de matrices notamment) : une solution intéressante consisterait à introduire plu-

sieurs points de cassure sur les grands segments obstacles, de manière à borner l'incertitude de chacun des points dans la direction parallèle au segment. Des techniques de « décision retardée »[118] ou des méthodes dérivées de l'initialisation des amers en SLAM visuels [109] pourraient également constituer une source d'inspiration.

Enfin, les améliorations au cadre classique du filtrage de Kalman seraient les bienvenues afin de limiter les coûts de calcul : en particulier le FastSLAM 2.0 [136] apparaît comme une piste particulièrement prometteuse sur plusieurs points, notamment grâce au découplage de l'estimation de chaque balise : a priori, il permettrait de réduire les coûts de calcul et de limiter certains problèmes de linéarisation (liés par exemple aux grandes incertitudes sur les points virtuels). En théorie, le FastSLAM offre également l'avantage d'autoriser le maintien de différentes hypothèses d'appariement (une pour chaque particule) : cela nécessiterait toutefois de maintenir plusieurs hypothèses de cartes combinatoires (une pour chaque particule) et risquerait de s'avérer très coûteux en espace mémoire et en temps de calcul. Une alternative consisterait à n'utiliser qu'une seule carte et d'exploiter la distribution de particules modélisant la localisation du robot comme s'il s'agissait d'une gaussienne (en extrayant moyenne et variance).

8.3.3 Améliorer la phase de mise à jour géométrique et topologique

Une amélioration simple à court terme des cartes combinatoires (colorées) discrètes décrites au chapitre 7 consisterait à exploiter le calcul des directions principales du premier scan (effectué lors du recalage par histogrammes du scan local par rapport au précédent, dans la phase de mise en correspondance) afin de définir l'orientation de la grille de discrétisation des cartes. En effet, si cette grille est alignée avec les directions principales, les grands segments seront en général peu découpés lors de leur transformation en petits segments (verticaux ou horizontaux dans la grille), ce qui permettra de réduire le nombre total de petits brins dans la carte et donc d'accélérer les opérations de raffinement (et les autres opérations associées comme la complétion de labels ou le déplacement de sommets).

A moyen terme, il s'agirait par ailleurs d'étudier la possibilité de réaliser les opérations de manipulation de cartes de manière plus locale, en vue d'accélérer les calculs. En particulier, les mini-cartes combinatoires présentent généralement une emprise spatiale réduite, ce qui permettrait sans doute de réaliser un raffinement plus localisé avec la carte globale (d'autant que dans le cas des cartes discrètes, les intersections sont calculées de manière exacte et ne génèrent pas de petits déplacements de segments qui pourraient se répercuter sur l'ensemble de la carte, comme nous l'avions illustré à la section 4.4). Cette amélioration nécessiterait peut-être cependant un découpage hiérarchique de la carte globale, avec une structure de données spécifique qu'il faudrait insérer convenablement dans la représentation et dans l'algorithme de construction de cartes actuels.

8.3.4 Implémenter les opérations de mise en forme de la carte

Au chapitre 7, nous avons proposé certaines opérations de mise en forme de la carte globale visant à réduire la taille mémoire de la représentation, à détecter d'éventuelles incohérences et à les corriger. Par manque de temps, ces opérations n'ont pas pu être implémentées dans le cadre de la thèse mais elles peuvent s'avérer utiles dans le cadre de la construction de modèles d'environnement de grande dimension.

8.3.5 Expérimentations

Il nous paraît important de tester de manière plus intensive (en simulation et en environnement réel) les algorithmes proposés dans le cadre de cette thèse, en particulier dans le cas de représentations de taille

relativement importante avec bouclage de cycles. Pour mettre en œuvre de telles expérimentations, il faudra auparavant s'assurer précisément de la validité du code programmé et vérifier module par module les quelques dizaines de milliers de lignes de code C++ programmées. En outre, il serait intéressant de comparer les représentations obtenues à des représentations moins structurées obtenues par des algorithmes classiques : à ce titre, l'annexe 2 fournit des pistes pour des méthodologies et métriques d'évaluation quantitative et comparative d'algorithmes de construction de cartes.

Enfin, l'état de l'art évolue très vite : il serait donc utile de vérifier dans la littérature (en SLAM mais aussi plus généralement en robotique, en reconnaissance des formes ou en traitement d'images par exemple) s'il existe des travaux permettant de remplacer directement certains modules de la chaîne algorithmique par des modules existants plus performants (par exemple pour la polygonalisation ou le recalage global par appariement de scans).

8.4 Perspectives à plus long terme

A plus long terme, les pistes d'améliration concernent essentiellement :

- **le découpage de la carte selon une structure hiérarchique** : nous avons déjà mentionné l'éventualité d'une hiérarchisation dans le cadre de l'accélération des opérations de manipulation de cartes combinatoires. Plus généralement, le découpage de la carte globale en sous-cartes offrirait a priori divers avantages sur l'ensemble de la chaîne algorithmique : il permettrait sans doute la construction de cartes cohérentes à plus grande échelle (cf. travaux sur le SLAM hiérarchique par exemple [67]), il faciliterait la définition de zones d'intérêt pour la recherche de segments globaux candidats à l'appariement avec des segments locaux, il faciliterait la mise en œuvre des techniques de report temporel qui consistent à travailler sur une zone locale avant de propager les mises à jour à l'ensemble de la carte [200] [80] [176], et il pourrait accélérer la mise à jour topologique et géométrique de la carte combinatoire (raffinement et déplacement de sommets traités plus localement). Dans ce cas, il faudra toutefois porter une attention toute particulière à la gestion des éléments à cheval sur deux sous-parties de la carte (les segments dans les opérations de manipulation de cartes combinatoires notamment).

  On pourrait également envisager de **structurer des ensembles locaux de scans en sous-cartes combinatoires avant de les fusionner à la carte globale**, afin de limiter les risques d'erreur d'association. Il faudrait alors revoir l'étape de mise en correspondance entre chaînes polygonales locales et globales. En effet, contrairement au cas où les segments locaux sont organisés en séquence (dans le polygone local), les segments d'une sous-carte locale ne présentent pas un ordre évident : en particulier, il arrive que, suite à des difficultés d'appariement, certains segments obstacles soient adjacents à plusieurs autres (cf. section 7.10.2). Dans ce cas, il sera difficile de calculer les scores de Mahalanobis corrects prévus au chapitre 5 entre couples local et global de segments candidats lorsque les segments locaux sont adjacents : l'algorithme de programmation dynamique décrit au chapitre 5 ne fonctionnera plus de manière optimale. Une solution consisterait alors à résoudre les incohérences d'appariement sur chaque carte locale avant mise en correspondance : on se ramènerait ainsi au cas de véritables lignes polygonales (sans embranchement), ce qui permettrait d'appliquer l'algorithme du chapitre 5 en les chaînant (comme si elles étaient reliées par des segments non obstacles) dans un ordre quelconque.

- **l'exploitation de la carte** : nous avons vu au chapitre 4 que la richesse et la structuration de la représentation choisie (notamment la couche topologique) offraient des perspectives intéressantes en

termes de planification de trajectoires (utilisation des informations topologiques pour réaliser une planification grossière et des informations métriques pour la définition plus fine des trajectoires...), de localisation (combinaison de plusieurs couches du modèle), d'exploration (extraction efficace des frontières de l'espace libre restant à explorer notamment) et de requêtes de plus haut niveau (opérations typiques des SIG telles que le fenêtrage ou la superposition de couches thématiques, requêtes plus complexes dans le cadre d'interactions homme / machine, etc.). En particulier, l'extraction et l'introduction d'informations sémantiques dans la carte permettrait de tirer pleinement parti de la structure de couche topologique, comme c'est le cas dans les modèles géographiques de type vecteur.

- **l'extension à d'autres types d'environnements** : il s'agit en particulier d'étendre la représentation existante et l'algorithme de construction de carte aux environnements 3D et aux milieux dynamiques. L'extension aux environnements 3D ne semble pas présenter de point dur en termes de filtrage (ajout d'une coordonnée z pour la position des sommets) ou de manipulation des cartes combinatoires (existence d'un algorithme de raffinement des cartes combinatoires 3D [23] notamment). En revanche, la mise en correspondance par programmation dynamique entre carte locale et carte globale peut poser problème en raison de l'absence de chaînage linéaire des segments ou des faces 3D (comme dans le cas de la mise en correspondance entre sous-cartes mentionné ci-dessus). Quant à l'application à des environnements dynamiques, elle peut être facilitée par la notion d'objet, qu'il est facile d'isoler et d'étiqueter dans la représentation que nous utilisons.

- **l'évaluation quantitative et comparative plus systématique des cartes construites** par rapport à d'autres types de représentations et de méthodes de construction de cartes, dans la lignée des propositions développées dans l'annexe 2 de ce mémoire, en vue d'estimer plus précisément les performances respectives et les conditions de fonctionnement des divers algorithmes de SLAM et de faciliter la mise en œuvre de systèmes réellement opérationnels.

Annexe 2 : Vers une évaluation plus systématique de la qualité des cartes

Les recherches menées durant les dernières années sur le SLAM ont permis le développement de multiples algorithmes mais il existe encore peu de systèmes opérationnels dotés de capacités de cartographie autonome en ligne. Ainsi, on dispose rarement de données chiffrées sur la qualité des cartes obtenues et l'évaluation de ces systèmes reste souvent très subjective, avec des expérimentations réalisées sur des environnements bien particuliers (en général le laboratoire qui a développé l'algorithme) et des comparaisons souvent très qualitatives avec une vérité terrain : la généralisation des résultats n'est pas acquise (et nous n'échappons pas à cette règle dans cette thèse !).

L'évaluation de performance se révèlera toutefois cruciale lorsque l'on cherchera à déployer des robots autonomes ou semi-autonomes dont les missions reposeront largement sur leur capacité à modéliser leur environnement et à se localiser : surveillance de zones, reconnaissance en zone urbaine ou recherche de personnes sous les décombres, etc. En effet, pour spécifier de tels systèmes, il importera de connaître plus précisément certaines caractéristiques des approches existantes et en particulier leur domaine de fonctionnement (environnements très structurés à sol plan, environnements urbains déstructurés, milieux naturels...), les risques d'échec, ainsi que la précision à espérer pour les cartes générées. C'est une question qui nous intéresse donc particulièrement à la DGA et comme nous l'avons indiqué précédemment, notre plate-forme robotisée est avant tout destinée à mener de telles évaluations de performance. C'est pourquoi nous indiquons ici quelques pistes pour une évaluation plus rigoureuse des algorithmes de cartographie, en lien avec les travaux que nous avons menés dans le domaine du traitement d'images (une carte 2D pouvant d'ailleurs être considérée comme une image) [52] [53] [54].

Approches existantes

Parmi les approches existantes, nous distinguons celles qui sont consacrées à l'évaluation « absolue » de la qualité de la carte et celles qui procèdent de manière « relative » par rapport à une tâche donnée (planification de trajectoires notamment).

a) Evaluation absolue

L'évaluation absolue de la qualité des cartes reste souvent visuelle et qualitative : nombreux sont les articles qui se contentent de montrer un ou deux exemples de modèles d'environnement construits par le robot, parfois superposés à un plan d'architecte ou à une vérité terrain construite manuellement. Certaines publications proposent néanmoins des évaluations analytiques ou expérimentales de la stratégie de cartographie employée, avec parfois la définition explicite de métriques quantitatives.

*** Evaluation analytique**

L'évaluation analytique d'un algorithme de cartographie peut se définir comme une évaluation théorique a priori de cet algorithme. Par exemple, de nombreux articles précisent la complexité de la technique utilisée et la comparent aux approches existantes : c'est l'un des enjeux des découpages de cartes ou du report temporel notamment. Nous avons également vu dans les chapitres 2 et 3 que certains chercheurs ont examiné les propriétés de convergence du SLAM basé sur le filtre de Kalman [46] tandis que d'autres se sont intéressés aux problèmes de linéarisation, proposant ainsi des améliorations aux approches fondées sur le filtre de Kalman étendu [22].

Dans notre état de l'art (cf. chapitre 2), nous avons également tenté de décrire a priori les points forts et les points faibles des représentations existantes, selon différents critères (compacité, généricité par rapport à l'environnement, aux capteurs, etc.). Nous nous sommes certes inspirés des résultats expérimentaux fournis dans les différents articles (en particulier la taille des environnements cartographiés et le temps de calcul) mais la plupart des indications concernent les approches théoriques, indépendamment des résultats exhibés. Ces indications sont donc avant tout destinées à guider la phase de développement et le choix d'une approche lors d'une phase de conception des algorithmes.

*** Evaluation expérimentale**

Parmi les évaluations expérimentales menées en traitement d'images, on peut distinguer les approches suivantes [20] [52] :

- les méthodes qui jouent sur les entrées de l'algorithme (variation des paramètres ou ajout de perturbations connues telles qu'un bruit ou une saturation) et examinent leur influence sur les résultats. L'objectif est souvent de définir le « domaine de fonctionnement » de cet algorithme et sa sensibilité aux modifications de paramètres et aux perturbations (ce qui peut s'avérer crucial lorsque l'on enchaîne différents algorithmes). Pour cela, on peut également chercher à caractériser les entrées en cherchant des caractéristiques corrélées avec les résulats.
- les méthodes qui s'intéressent essentiellement aux sorties de l'algorithmes : certaines comparent ces sorties à des « vérités terrain » (résultat « idéal » servant de référence) au moyen de métriques adéquates tandis que d'autres évaluent les résulats dans l'absolu, selon des critères censés représenter un résultat « correct » (par exemple la régularité des contours...), sans recourir aux vérités terrain.

Pour la cartographie, nous n'avons pas retrouvé toute cette palette de méthodes (du moins, celles-ci ne sont pas décrites explicitement). En particulier, peu d'approches ont cherché à faire varier les paramètres d'entrée des algorithmes (si ce n'est éventuellement les niveaux de bruit en simulation).

Les évaluations les plus systématiques concernent en général des **sous-tâches du problème de cartographie**, en particulier l'appariement de scans et la localisation du robot. Concernant l'appariement de scans, on peut citer les travaux de Gutmann et Schlegel qui comparent trois méthodes d'appariement dans quelques types d'environnements (très structurés avec des parois polygonales ou moins structurés) et mesurent explicitement l'écart entre le recalage estimé par l'algorithme et le décalage réel entre les scans [82]. Par ailleurs, de nombreux articles sur le SLAM comparent la position estimée du robot à sa localisation réelle (qu'il s'agisse d'un paradigme métrique ou topologique) : implicitement, ils supposent souvent que si la localisation (c'est-à-dire un sous-produit du SLAM) fonctionne, l'ensemble du processus de cartographie se déroule convenablement. Pourtant, la qualité de la localisation du robot ne garantit pas la lisibilité de la carte et l'absence d'incohérences : par exemple, l'omission de certains appariements peut notamment conduire à des segments redondants dans une carte métrique, alors qu'en principe, cette omission n'influe pas sur la stabilité du filtre de Kalman qui peut être utilisé pour la construire.

Quelques travaux proposent quant à eux des **critères d'évaluation de la qualité de la carte, qui peuvent être vérifiés en ligne** (en particulier, ils ne nécessitent pas de vérité terrain). Par exemple, certaines méthodes de cartographie sont fondées sur une estimation par maximum de vraisemblance, qui peut passer par la minimisation d'une fonction d'énergie, dont le calcul explicite est possible (cf. par exemple les travaux de Folkesson et Christensen [73]). Ainsi, cette énergie constitue une mesure de la qualité de la représentation (compte tenu des observations) et il est possible de l'utiliser pour comparer deux hypothèses de cartes notamment. Par ailleurs, dans les méthodes à base de filtre de Kalman, on peut examiner l'évolution de la précision des estimations de position, à partir de la matrice de covariance de la carte.

Finalement, certaines publications proposent une **évaluation a posteriori par rapport à une vérité terrain**. Si cette évaluation est souvent laissée au lecteur, qui peut apprécier visuellement la superposition de la carte reconstruite par le robot et de la carte de référence (ou la juxtaposition de représentations de même nature - par exemple des grilles d'occupation - construites à partir des mêmes données selon des méthodes différentes [164]), quelques présentations fournissent la précision chiffrée du modèle reconstruit, en indiquant éventuellement s'il s'agit d'une précision minimale, maximale ou moyenne. L'évaluation d'une précision est toutefois très dépendante de la technique d'association de données utilisée pour mettre en correspondance la carte reconstruite et la vérité terrain [52] : la diversité des méthodes d'appariement exposées au chapitre 5 en témoignent. Or la plupart du temps, cette technique de mise en correspondance n'est pas précisée. D'autres critères quantitatifs concernent la taille des environnements qui ont été cartographiés « correctement » (et notamment la longueur des cycles parcourus par le robot) ou le temps mis pour générer la carte : ces critères ne sont toutefois pas directement liés à la qualité de la carte obtenue. Par ailleurs, dans le cadre de la construction de grilles d'occupation, certains auteurs ont proposé des métriques d'appariement explicites entre une carte reconstruite et une carte idéale (dont toutes les probabilités sont à 0, à 1 ou à $0,5$) [129] : il s'agit bien de métriques quantitatives, mais réservées à un type de représentation bien particulier.

Enfin, il est possible de vérifier expérimentalement la convergence et la cohérence de certaines techniques de cartographie : plus que la qualité de la carte, c'est le processus de cartographie qui est évalué dans ce cas. Castellanos et al. rappellent ainsi la définition d'un estimateur « consistant » [22] : cet estimateur doit être non biaisé et son erreur quadratique moyenne doit correspondre à la covariance calculée par le filtre. Lorsque l'on dispose d'une vérité de terrain pour ces variables d'état (dans une simulation par exemple), on peut appliquer un test statistique pour s'assurer de la cohérence du filtre : il s'agit de vérifier que l'erreur d'estimation quadratique normalisée (ou NEES pour « Normalized Estimation Error Squared ») est inférieure à un certain seuil (test du χ^2). Ainsi, une covariance légèrement pessimiste est acceptable tandis qu'une covariance trop optimiste conduit à une incohérence et à une probable divergence du filtre (ce qui risque de se produire en présence d'erreurs de linéarisation).

b) Evaluation relative (par rapport à une tâche)

Dans le cadre de la recherche de victimes sous les décombres (suite à un tremblement de terre par exemple), le NIST (National Institute of Standards and Technology aux Etats-Unis) a mis en place des arènes de test standardisées qui ont été copiées à l'identique dans divers pays du monde, à l'occasion de compétitions annuelles [88]. Ces arènes comportent trois zones correspondant à des niveaux de difficultés variables pour l'environnement : structuré avec un sol plan pour l'arène jaune, relativement structuré mais non planaire pour l'arène orange (présence de trous dans le sol, plans inclinés, d'où la nécessité d'une cartographie et d'une localisation en trois dimensions a priori) et destructuré pour l'arène rouge (présence de gravats, structures fragiles et flexibles, etc.). De plus, chaque zone présente des éléments

variés, suceptibles de mettre en difficulté les différents capteurs du robot (revêtements de murs et angles d'incidence peu favorables à la télémétrie laser, motifs perturbants pour les capteurs stéréoscopiques, etc.).

L'ensemble de la tâche de recherche de personnes est évaluée suivant un barême précis qui prend en compte la qualité de la carte générée, la précision de localisation des victimes, ainsi que l'estimation de leur état et de leur situation [89]. En outre, l'équipe candidate est sanctionnée si le robot a malencontreusement heurté l'environnement ou la victime et si le nombre de téléopérateurs requis est important. Les scores sont en général plus élevés lorsque les différentes tâches sont réalisées de manière autonome.

Concernant la cartographie, le modèle d'environnement est destiné à servir de support pour le sauvetage des victimes détectées : il est donc censé représenter la position de ces victimes, les éléments discernables de l'environnement, les dangers potentiels et les outils nécessaires à l'extraction des victimes. La note attribuée est maximale lorsque la carte est générée de manière automatique par le système robotisé et que l'opérateur se contente de l'annoter. Cette note est divisée par deux lorsque l'opérateur intervient dans le processus de cartographie et elle devient très faible lorsque cette carte est entièrement construite par l'opérateur ou lorsqu'il ne s'agit que d'une carte topologique (donc difficile à exploiter par les hommes), du fait de la surcharge de travail associée pour le téléopérateur. Une telle évaluation reste donc très qualitative : les critères d'évaluation du contenu de la carte, de sa précision, de sa validité et de son exhaustivité ne sont pas détaillés et ne semblent pas réellement entrer en ligne de compte. Une telle approche a toutefois le mérite de permettre des évaluations reproductibles (les conditions de test peuvent être répétées à l'identique dans différents pays grâce à la standardisation des arènes) et de proposer une métrique globale quantitative de la tâche, basée sur un barême prédéfini.

Lee propose pour sa part une évaluation de la carte par rapport à une tâche de planification de trajectoire (pour une application de distribution de courrier) [114]. Pour cela, le système spécifie un ensemble de trajets définis par leur point de départ et d'arrivée (dans des zones libres d'obstacles) sur la carte idéale discrétisée (de type grille d'occupation). Pour chacun de ces trajets, la carte du robot est utilisée afin de générer la meilleure trajectoire possible. Cette trajectoire est ensuite superposée à la carte idéale pour vérifier sa validité. Ainsi, si le point de départ ou d'arrivée se situe dans une zone occupée dans la carte du robot, le trajet est étiqueté « impossible ». Si la trajectoire calculée intersecte des obstacles de la carte idéale, le trajet est marqué « en collision ». Sinon, le trajet est considéré comme « sûr ». Ensuite, pour estimer la qualité du modèle d'environnement reconstruit, on peut compter le pourcentage de trajets qui tombent dans chaque catégorie et comparer la longueur des trajets sûrs tels qu'ils sont planifiés sur la carte idéale et sur la carte reconstruite. Suivant l'application précise qui sera faite du système robotisé, le concepteur peut ensuite opter pour la métrique la plus adéquate. Par exemple, si le robot est amené à travailler seul, il faudrait qu'il puisse planifier un nombre maximal de trajectoires sûres. En revanche, si le robot transporte des colis fragiles, il est a priori préférable qu'il évite les arrêts d'urgence (quitte à ne pas tenter certains trajets hasardeux) : il faudrait donc limiter le ratio de trajets marqués « en collision ».

Enfin, on peut mentionner une évaluation réalisée par Thrun dans le cadre de l'extraction de cartes topologiques à partir de grilles d'occupation [178]. Les cartes topologiques extraites sont comparées aux grilles d'occupation initiales (qui constituent une référence, même s'il ne s'agit pas à proprement parler de vérités terrain) pour une tâche de planification de trajectoire. Les critères d'évaluation sont la cohérence (chaque solution - trajectoire planifiée - dans l'une des cartes doit correspondre à une solution dans l'autre carte), la perte (en terme de performance sur la longueur du chemin trouvé) et l'efficacité (en terme de complexité du calcul de trajectoire). Ces critères peuvent être quantifiés et fournissent donc un moyen de comparer des représentations de nature différente.

Quelques propositions pour une évaluation plus systématique

Pour être véritablement exploitable, nous pensons qu'une évaluation doit vérifier certaines propriétés :
- Pour limiter le caractère subjectif des jugements humains, il est préférable qu'elle repose sur des métriques **quantitatives** plutôt que sur des appréciations purement qualitatives.
- Elle doit être **reproductible** au sens où l'on doit obtenir les mêmes résultats en se plaçant dans des conditions expérimentales identiques. On peut également privilégier les systèmes qui permettent effectivement de reproduire les mêmes conditions expérimentales, de manière à vérifier certains résulats ou à comparer différents systèmes par exemple.
- De par le caractère fastidieux et répétitif de certaines évaluations, il vaut mieux disposer de systèmes de test **automatiques** ou semi-automatiques, qui limitent la charge de l'opérateur (en particulier lorsque les données expérimentales sont volumineuses) ;
- Autant que possible, cette évaluation doit être **comparative**, au sens où elle doit permettre de comparer des stratégies variées, basées éventuellement sur des représentations différentes.
- Enfin, elle doit être **représentative** au sens où l'on doit s'attacher à vérifier le comportement des algorithmes testés sur des environnements variés, représentatifs des milieux dans lesquels le robot est amené à évoluer.

Si l'on s'intéresse aux évaluations absolues (qui autorisent une certaine généricité puisqu'elles ne se limitent pas à une tâche donnée), on ne peut donc pas se contenter de tester les stratégies de cartographie sur quelques environnements prédéfinis. Une véritable campagne d'évaluation devrait prendre en compte des environnements variés. Par exemple, en milieu intérieur, il s'agit de faire varier les revêtements des murs et du sol, les densités d'obstacles, la forme des obstacles (notamment leur caractère polygonal ou moins structuré), etc. A ce titre, les arènes du NIST se révèlent particulièrement intéressantes et elles permettent en outre de reproduire les conditions expérimentales à l'identique, tout en introduisant une certaine modularité (les parois peuvent être déplacées par exemple). Dans le cadre de l'évaluation d'algorithmes de traitement d'images, nous avions introduit différents scénarios généraux de difficulté variable, qui pourraient correspondre par exemple aux différentes zones (jaune, orange, rouge) de ces arènes. Quant aux scénarios spéciaux qui isolent des difficultés particulières, ils pourraient inclure par exemple la présence de trous ou de légères irrégularités dans le sol ou d'obstacles semi-transparents (grilles, verre fumé...).

L'utilisation de vérités terrain pose des problèmes pratiques puisque leur construction peut s'avérer particulièrement fastidieuse (même si l'utilisation d'arènes standardisées limiterait cette difficulté). Pour l'évaluation des algorithmes de suivi de routes, nous avons conçu une interface qui permet de tracer facilement des bords de routes dans l'image et qui gère l'organisation des fichiers de vérité terrain [52]. Dans le cadre de la cartographie, on peut envisager des aide au recalage manuel de scans par exemple, puis une segmentation automatique à partir des points de mesure [190], de manière à générer des cartes de type plans d'architecte par exemple. L'utilisation d'algorithmes de cartographie de référence pourrait également réduire le travail de l'opérateur qui se contenterait alors d'intervenir lors des échec de cet algorithme pour corriger la vérité terrain [54]. En matière d'automatisation, il peut également être utile de disposer d'environnements logiciels dédiés à l'évaluation, qui permette notamment de gérer de grandes bases de données de test et de faire varier automatiquement les paramètres de l'algorithme sur certaines plages définies : c'est par exemple le cas de l'outil SENA conçu au CEP Arcueil [52].

Concernant l'aspect quantitatif de d'évaluation, nous pensons qu'il est important de disposer de métriques variées, capables de juger l'algorithme selon différents critères. Ensuite, les utilisateurs peuvent sélectionner celles qui sont le plus adaptées à la tâche (ou les combiner [54]) comme nous le proposons en

traitement d'images pour les algorithmes de suivi de routes [54] et comme nous l'avons vu pour Lee [114] dans le cadre du SLAM. Parmi les propriétés d'une carte que l'on souhaite examiner, on peut notamment citer la validité (par exemple l'interprétation correcte de la présence voire de la nature d'un obstacle), la précision (en terme d'erreur de localisation des éléments de la carte) ou l'exhaustivité (obtention d'une carte aussi complète que possible d'une zone donnée).

Suivant le type de représentation employée, il existe des métriques relativement naturelles pour comparer des cartes de même nature. Par exemple, les cartes basées sur des primitives géométriques peuvent être évaluées selon le nombre de primitives retrouvées (avec éventuellement une estimation des « sur-segmentations » ou des « sous-segmentations » dans le cas des représentations polygonales à base de segments) ou sur la précision de localisation de ces primitives. Pour les représentations basées sur l'apparence, il s'agit de comparer des ensembles de points de mesure laser par exemple : on peut donc songer à des distances de type Hausdorff. Pour les représentations surfaciques, il s'agit par exemple de calculer des scores d'appariement moyens entre les valeurs des cellules individuelles correspondant à la même zone géographique [129]. Quant aux représentations topologiques, elles peuvent être jugées en terme de compacité ou d'exhaustivité, même si une évaluation realtive par rapport à une tâche (planification de trajectoires notamment) paraît plus aisée [178].

Pour améliorer la généricité de la méthodologie d'évaluation, il paraît toutefois plus intéressant de rechercher des métriques permettant de comparer des représentations de nature différente. En particulier, comme la représentation que nous avons choisie dans le cadre de cette thèse combine différents types de modèles élémentaires, il peut être utile de vérifier si cette hybridation améliore la qualité de ces modèles pris individuellement. C'est l'optique que nous avons choisie dans le cadre du traitement d'images pour comparer des algorithmes basés sur la segmentation en régions ou l'extraction de contours par exemple.

Ainsi, dans le cadre du SLAM, nous avons vu que Thrun a comparé une carte topologique avec une grille d'occupation [178]. Dans le cadre d'une cartographie métrique, si l'on considère une vérité terrain constituée d'une grille d'occupation, il paraît aisé de discrétiser des représentations de différentes natures pour les comparer au moyen de métriques d'appariement de grilles d'occupation [129] [194] : cette discrétisation a déjà été réalisée sur des représentations à base de scans laser bruts [81] ou sur des représentations à base de primitives [168]. On peut également compter plus précisément le nombre de fausses alarmes ou de détections d'obstacles manquées d'un point de vue surfacique. Si l'on considère plutôt une vérité terrain constituée de segments, le problème devient indissociable de l'association de données. Pour cela, on peut recourir à des techniques génériques utilisées pour l'évalution d'algorithmes d'extraction de réseaux routiers par exemple (les frontières d'obstacles peuvent s'apparenter aux routes), qui utilise une zone de tolérance autour des segments de la vérité terrain et considère comme appariés tous les objets situés dans cette zone de tolérance [197] [54] (cf. Fig. 8.1). Une telle approche permettrait de prendre en compte des mesures ponctuelles (superpositions de scans bruts, primitives ponctuelles, cellules individuelles de la grille d'occupation) mais aussi des primitives géométriques (segments, arcs) et de comparer les cartes sur la base de l'extraction des frontières d'obstacles.

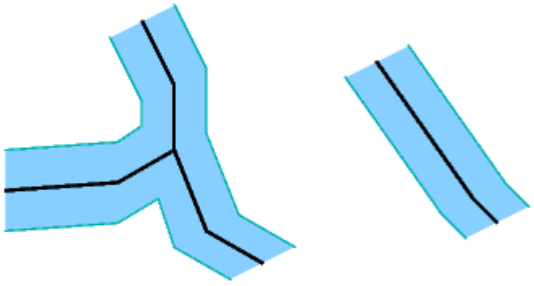


FIG. *8.1: Zone de tolérance (en bleu) définie autour des segments de la vérité terrain (en noir).*

Exploitation des données d'évaluation

Les données d'évaluations peuvent en principe servir à différents stades de conception de l'algorithme. Nous avons vu que les évaluations théoriques peuvent être utilisées en phase de conception pour sélectionner les algorithmes les plus efficaces a priori. Si des évaluations expérimentales sont disponibles sur les divers algorithmes envisagés, le plus simple consiste à choisir l'approche la plus prometteuse, par exemple celle qui donne les meilleurs résulats en moyenne (sur une métrique globale ou adaptée à la mission du robot si celui-ci est relativement spécialisé). On peut également chercher les complémentarités entre techniques de manière à tester des combinaisons d'approches [54]. Ensuite, durant le développement du système de cartographie, les différentes variantes d'un même algorithme peuvent être comparées afin de sélectionner la plus performante. Enfin, l'efficacité du système peut être validée a posteriori. La distinction de ces différentes étapes peut donc conduire à diviser les bases de données d'évaluation en différents ensembles (comme c'est couramment le cas dans le domaine du traitement de la parole, où des évaluations quantitatives sont réalisées périodiquement à l'échelle mondiale) : une base de développement destinée aux concepteurs (dont il est possible d'extraire une base d'apprentissage si les algorithmes en ont besoin) et une base de test destinée plutôt à un organisme indépendant chargé de comparer les approches [54].

En outre, au lieu de figer a priori la stratégie de cartographie, il serait intéressant d'envisager des systèmes robotisés capables de s'adapter en ligne à l'environnement, afin de mettre en œuvre la technique de SLAM la mieux adaptée. Si les évaluations ont permis de déterminer le domaine de fonctionnement des algorithmes, le système pourrait ainsi analyser l'environnement (par exemple le niveau de structuration, la densité d'obstacles, etc.) et sélectionner ainsi en ligne la stratégie de cartographie adéquate (ce type d'approche a déjà été suggéré pour la localisation à travers le système AMOS de Gutmann et Schlegel par exemple [82]). Enfin, si l'on dispose de métriques d'évaluation en ligne (pouvant s'affranchir d'une vérité terrain), celles-ci peuvent également contribuer à la sélection des algorithmes, favorisant ainsi des transitions transparentes entre environnements de nature différente. Cette approche reviendrait notamment à poursuivre les travaux de Dalgalarrondo sur l'architecture Harpic implantée sur notre robot Pioneer, qui avait été testée dans le cadre de comportements de suivi de références linéaires (murs, trottoirs...) avec des transitions entre différents types d'environnements [36].

Bibliographie

[1] D. Anguelov, R. Biswas, D. Koller, B. Limketkai, S. Sanner, and S. Thrun. Learning hierarchical object maps of non-stationary environments with mobile robots. In *Proceedings of the 17th Annual Conference on Uncertainty in A.I. (UAI)*, 2002.

[2] G. C. Anousaki and K. J. Kyriakopoulos. Simultaneous localization and map building for mobile robot navigation. In *IEEE Robotics and Automation Magazine*, septembre 1999.

[3] R. C. Arkin. Integrating behavioral, perceptual and world knowledge in reactive navigation. In *Robotics and Autonomous Systems*, volume 6, pages 105–122, 1990.

[4] A. Arleo, J. del R. Millán, and D. Floreano. Efficient learning of variable-resolution cognitive maps for autonomous indoor navigation. In *IEEE Transactions on Robotics and Automation*, décembre 1999.

[5] D. Arquès. Enumération et codage des cartes et hypercartes planaires pointées - calcul formel sur les fractions continues. Technical report, 1985.

[6] D. Arquès. Une relation fonctionnelle nouvelle sur les cartes pointées. In *Journal of Combinatorial Theory (B)*, volume 39(1), pages 27–42, 1985.

[7] N. Ayache. *Construction et fusion de représentations visuelles 3D. Applications la robotique mobile.* PhD thesis, Université d'Orsay, mai 1988.

[8] T. Bailey. *Mobile robot localisation and mapping in extensive outdoor environments.* PhD thesis, Australian Center for Field Robotics, University of Sydney, Sydney (Australie), août 2002.

[9] B. G. Baumgart. A polyhedron representation for computer vision. In AFIPS Press, editor, *Proc. Nat. Comp. Conf.*, pages 589–596, 1970.

[10] J. L. Bentley and T. Ottmann. Algorithms for reporting and counting geometric intersections. In *IEEE Trans. Comput.*, volume 28, pages 643–647, 1979.

[11] J. R. Beveridge. Local search algorithms for geometric object recognition : optimal correspondence and pose. In University of Massachusetts, editor, *Technical Report CS-93*, 1993.

[12] J. Borenstein, H.R. Everett, and L. Feng. *Where am I ? Sensors and methods for mobile robot positionning.* The University of Michigan, avril 1996.

[13] J. Borenstein and Y. Koren. Histogramic in-motion mapping for mobile robot obstacle avoidance. In *IEEE Transactions on Robotics and Automation*, août 1991.

[14] G. A. Borges and M.-J. Aldon. A split-and-merge segmentation algorithm for line extraction in 2-D range images. In *International Conference on Pattern Recognition (ICPR'00)*, Barcelone (Espagne), 2000.

[15] G. A. Borges and M.-J. Aldon. Optimal pose estimation using geometrical maps. In *IEEE Transactions on Robotics and Automation*, février 2002.

[16] M. Bosse, P. Newmann, J. Leonard, M. Soika, W. Feiten, and S. Teller. An Atlas framework for scalable mapping. In *IEEE International Conference on Robotics and Automation (ICRA'03)*, pages 1899–1906, Taipei (Taiwan), 2003.

[17] R. A. Brooks. Intelligence without representation. In *Artificial intelligence*, volume 47, pages 139–159, 1991.

[18] H. Bulata and M. Devy. Incremental construction of a landmark-based and topological model of indoor environments by a mobile robot. In *IEEE International Conference on Robotics and Automation (ICRA'96)*, 1996.

[19] W. Burgard, D. Fox, D. Henning, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Menlo Park (CA, Etats-Unis), août 1996.

[20] R. Capolunghi and V. Ropert. L'évaluation empirique en traitement d'images. Technical report, DGA/Centre Technique d'Arcueil, mars 2000.

[21] J. A. Castellanos, J. M. M. Montiel, J. Neira, and D. Tardos. The SPmap : a probabilistic framework for simultaneous localization and map building. In *IEEE Transactions on Robotics and Automation*, octobre 1999.

[22] J. A. Castellanos, J. Neira, and D. Tardos. Limits to the consistency of EKF-based SLAM. In *5th IFAC Symposium on Intelligent Autonomous Vehicles*, Lisbonne (Portugal), juillet 2004.

[23] D. Cazier. *Construction des systèmes de réécriture pour les opérations booléennes en modélisation géométrique*. PhD thesis, Université Louis Pasteur, Strasbourg, France, novembre 1997.

[24] D. Cazier and J.-F. Dufourd. A formal specification of the geometric refinement. In Springer-Verlag, editor, *The Visual Computer*, volume 15, pages 279–301, 1999.

[25] L. Charbonnier. *Localisation d'un robot mobile par mise en correspondance de cartes télémtriques : utilisation du concept de ressemblance*. PhD thesis, Université de Montpellier II, juillet 1996.

[26] R. Chatila and J.-P. Laumond. Position referencing and consistent world-modeling for mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 138–145, mars 1985.

[27] C.-H. Choi, J.-B. Song, W. Chung, and M. Kim. Topological map building based on thinning and its application to localisation. In *Proceedings of the 2002 Int. Conference on Intelligent Robots and Systems*, Lausanne (Suisse), octobre 2002.

[28] K. S. Chong and L. Kleeman. Feature-based mapping in real, large scale environments using an ultrasonic array. In *The International Journal of Robotics Research*, volume 18(1), pages 3–19, 1999.

[29] J.-P. Cocquerez and S. Philipp, editors. *Analyse d'images : filtrage et segmentation*. Masson, 1995.

[30] B. Collin. *Contribution à l'analyse automatisée de la déformation*. PhD thesis, Université d'Orsay, novembre 1994.

[31] CGAL Community. The CGAL user manual, version 3.1. In *http ://www.cgal.org*, décembre 2004.

[32] J. H. Connel. *Minimalist mobile robotics*. Academic Press, 1990.

[33] R. Cori. Un code pour les graphes planaires et ses applications. In Société mathématique de France, editor, *Asterisque*, volume 27, Paris, 1975.

[34] I. Cox. Blanche - an experiment in guidance and navigation of an autonomous robot vehicle. In *IEEE Transactions on Robotics and Automation*, volume 7(2), pages 193–204, avril 1991.

[35] M. Csorba. *Simultaneous localization and map-building*. PhD thesis, University of Oxford, 1997.

[36] A. Dalgalarrondo. *Intégration de la fonction perception dans une architecture de contrôle de robot mobile autonome*. PhD thesis, Université de Paris Sud - Centre d'Orsay, 2001.

[37] A. Dalgalarrondo, D. Dufourd, and D. Filliat. Controlling the autonomy of a reconnaissance robot. In *Proceedings of SPIE's 15th International Symposium, Unmanned Ground Vehicle Technology VI*, Orlando (Floride, Etats-Unis), avril 2004.

[38] J. Daux. Evaluation d'algorithmes d'appariement de scans laser 2d. Technical report, DGA/Centre Technique d'Arcueil - Stage ENSTA, juillet 2002.

[39] B. David. *Modélisation, représentation et gestion de l'information géographique*. PhD thesis, Université Paris 6, Paris, juillet 1991.

[40] A. Davison. *Mobile robot navigation using active vision*. PhD thesis, Robotics Research Group, Oxford University Department of Engineering Science (Royaume-Uni), février 1998.

[41] G. Dedeoglu, M. J. Matarić, and G. S. Sukhatme. Incremental, on-line topological map building with a mobile robot. In *Proceedings of Mobile Robots XIV - SPIE 99*, pages 129–139, Boston, Massachusetts (Etats-Unis), 1999.

[42] M. Devy and H. Bulata. Multi-sensory perception and heterogeneous representations for the navigation of a mobile robot in a structured environment. In *Proceedings of SIRS'96*, 1996.

[43] J. Diard. *La carte bayésienne - un modèle probabiliste hiérarchique pour la navigation en robotique mobile*. PhD thesis, Institut National Polytechnique de Grenoble, janvier 2003.

[44] J. Diard, P. Bessière, and E. Mazer. Hierarchies of probabilistic models for navigation : the Bayesian Map and the Abstraction operator. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3837–3842, avril 2004.

[45] M. W. M. G. Dissanayake, H. F. Durrant-Whyte, and T. Bailey. Map management for efficient simultaneous localization and mapping (SLAM). In *Autonomous Robots*, volume 12, pages 267–286, mai 2002.

[46] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. In *IEEE Transactions on Robotics and Automation*, volume 17(3), pages 229–241, juin 2001.

[47] J. G. Donnett. *Analysis and synthesis in the design of locomotor and spatial competences for a multisensory mobile robot*. PhD thesis, University of Edinburgh (Royaume-Uni), 1992.

[48] A. Doucet, J. de Freitas, K. Murphy, and S. Russell. Rao-blackwellized particle filtering for dynamic Bayesian networks. In *Proc. of the Conf. on Uncertainty in Computer Science*, 2000.

[49] T. Duckett, S. MArsland, and J. Shapiro. Learning globally consistent maps by relaxation. In *Proceedings of the International Conference on Robotics and Automation (ICRA'00)*, pages 3841–3846, 2000.

[50] D. Dufourd. Autonomous construction of indoor maps with a mobile robot. In *Proceedings of SPIE's 15th International Symposium, Unmanned Ground Vehicle Technology III*, Orlando (Floride, Etats-Unis), avril 2001.

[51] D. Dufourd, R. Chatila, and D. Luzeaux. Combinatorial maps for Simultaneous Localization and Map-building (SLAM). In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai (Japon), septembre-octobre 2004.

[52] D. Dufourd and A. Dalgalarrondo. Performance evaluation of road detection and tracking algorithms. In *Performance Metrics for Intelligent Systems Workshop (PerMIS'02)*, Gaithersburg (Maryland, Etats-Unis), août 2002.

[53] D. Dufourd and A. Dalgalarrondo. Quantitative evaluation of image processing algorithms for ill-structured road detection and tracking. In *Proceedings of SPIE's 17th International Symposium, Unmanned Ground Vehicle Technology V*, Orlando (Floride, Etats-Unis), avril 2003.

[54] D. Dufourd and A. Dalgalarrondo. Results and lessons learned from the quantitative evaluation of road detection and tracking algorithms. In *Performance Metrics for Intelligent Systems Workshop (PerMIS'03)*, Gaithersburg (Maryland, Etats-Unis), septembre 2003.

[55] J.-F. Dufourd. Algebras and formal specification in geometric modelling. In *The Visual Computer Journal*, volume 15, pages 279–301. Springer, 1997.

[56] J.-F. Dufourd and F. Puitg. Functional specification and prototyping with oriented combinatorial maps. In *Computational geometry, theory and applications*. Elsevier, 2000.

[57] H. F. Durrant-Whyte, M. W. M. G. Dissanayake, and P. W. Gibbens. Toward deployment of large scale simultaneous localisation and map building (SLAM) problem. In *Technical Report*, Australian Center for Field Robotics, University of Sydney (Australie), 2000.

[58] H. F. Durrant-Whyte, S. Majumder, M. de Battista, and S. Scheding. A Bayesian algorithm for simultaneous localisation and map building. In *Proceedings of the International Symposium of Robotics Research (ISRR'01)*, Lorne Virginia (Australie), 2001.

[59] J. Edmonds. A combinatorial representation for polyhedral surfaces. In *Notices of AMS*, volume 7, 1960.

[60] T. Einsele. Real-time self localization in unknown environments using a panorama laser range finder. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 191–202, New-York (Etats-Unis), 1997.

[61] A. Elfes. Sonar based real world mapping and navigation. In *IEEE Journal of Robotics and Automation*, volume 3(3), pages 233–247, 1987.

[62] A. Elfes. Multi-source spatial data fusion using Bayesian reasoning. In M. A. Abidi and R.C. Gonzales, editors, *Data Fusion in Robotics and Machine Intelligence (Chapter 3)*, New-York (Etats-Unis), 1992. Academic Press.

[63] A. Eliazar and R. Parr. DP-SLAM : fast, robust simultaneous localization and mapping without predetermined landmarks. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, 2003.

[64] A. Eliazar and R. Parr. DP-SLAM 2.0. In *International Conference on Robotics and Automation*, 2004.

[65] S. Engelson. Continuous map learning for mobile robots. In *The 3rd French-Israeli Symposium on Robotics*, mai 1995.

[66] S. Engelson and D. McDermott. Error correction in mobile robot map learning. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2555–2560, Nice (France), mai 1992.

[67] C. Estrada, J. Neira, and J. D. Tardos. Hierarchical SLAM : real-time accurate mapping of large environments. In *IEEE International Transactions on Robotics and Automation*, A paraître.

[68] E. Fabrizi and A. Saffioty. Augmenting topology-based maps with geometric information. In *Robotics and Autonomous Systems*, volume 40 (2-3), pages 91–97, août 2002.

[69] D. Filliat. *Cartographie et estimation globale de la position pour un robot mobile autonome*. PhD thesis, Université Paris 6, décembre 2001.

[70] D. Filliat and J.-A. Meyer. Map-based navigation in mobile robots. I. A review of localization strategies. In *Cognitive Systems Research*, volume 4(1-4), pages 243–282, décembre 2003.

[71] E. Flato, D. Halperin, I. Hanniel, O. Nechushtan, and E. Ezra. The design and implementation of planar maps in CGAL. In *Journal of Experimental Algorithms (JEA)*, volume 5, 2000.

[72] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer graphics : principles and practice.* Addison-Wesley, 1995.

[73] J. Folkesson and H. Christensen. Graphical SLAM - A Self-Correcting map. In *Proc. of the International Conference on Robotics and Automation (ICRA'04)*, 2004.

[74] D. Fradin, D. Meneveaux, and P. Lienhardt. Partitionnement de l'espace et hiérarchie de cartes généralisées : application aux complexes architecturaux. In *Actes des 15èmes journées de l'Association Française d'Informatique Graphique (AFIG'2002)*, 2002.

[75] J. Gasós and A. Martín. Mobile robot localization using fuzzy maps. In *Proceedings of the IJCAI'95 workshop - Lecture Notes in Artificial Intelligence*, pages 207–224. Springer-Verlag, 1997.

[76] P. Gaussier, C. Joulain, S. Lepretre, and A. Revel. The visual homing problem : an exemple of robotics/biology cross-fertilisation. In *Robotics and Autonomous Systems*, volume 30(1-2), pages 155–180, 2000.

[77] F. Gechter and F. Charpillet. Vision based localization for a mobile robot. In *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI'00)*, Vancouver (Canada), novembre 2000.

[78] M. Golfarelli, D. Maio, and S. Rizzi. Elastic correction of dead-reckoning errors in map building. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98)*, Victoria (Canada), 905-911 1998.

[79] J. Guivant, E. Nebot, J. Nieto, and F. Masson. Navigation and mapping in large unstructured environments. In *The International Journal of Robotics Research*, volume 23 (4-5), pages 449–472, avril-mai 2004.

[80] J. E. Guivant and E. M. Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. In *IEEE Transactions on Robotics and Automation*, volume 17(3), pages 242–257, juin 2001.

[81] J.-S. Gutmann and K. Konolidge. Incremental mapping of large cyclic environments. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA'99)*, pages 318–325, 1999.

[82] J.-S. Gutmann and C. Schlegel. AMOS : comparison of scan matching approaches for self-localization in indoor environments. In *1st Euromicro Workshop on Advanced Mobile Robots (Eurobot'96)*, 1996.

[83] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)*, 2003.

[84] D. Hähnel, D. Schulz, and W. Burgard. Mapping with mobile robots in populated environment. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02)*, 2002.

[85] R. M. Haralick. Propagating covariances in computer vision. In *International Conference on Pattern Recognition*, 1994.

[86] J. Hermosillo, C. Pradalier, S. Sekhavat, and C. Laugier. Autonomous navigation of a bi-steerable car : experimental issues. In *Machine Intelligence and Robotic Control Journal (MIROC)*, volume 5(3), pages 95–102, 2004.

[87] A. Howard. Multi-robot mapping using manifold representations. In *IEEE International Conference on Robotics and Automation*, pages 4198–4203, New Orleans (Louisianne, Etats-Unis), avril 2004.

[88] A. Jacoff, E. Messina, and J. Evans. Experiences deploying test arenas for autonomous mobile robots. In *Proc. of the Performance Metrics for Intelligent Systems (PerMIS'01) Workshop*, septembre 2001.

[89] A. Jacoff, B. Weiss, and E. Messina. Evolution of a performance metrics for urban search and rescue robots. In *Proc. of the Performance Metrics for Intelligent Systems (PerMIS'03) Workshop*, août 2003.

[90] A. Jacques. Constellations et graphes topologiques. In *Combinatorial Theory and Applications*, pages 657–673, 1970.

[91] S. Julier and J. K. Uhlmann. Building a million beacon map. In *Proceedings of SPIE Int. Soc. Opt. Eng.*, volume 4571, pages 10–21, Washington (DC, Etats-Unis), 2001.

[92] S. J. Julier. A sparse weight Kalman filter approach to simultaneous localisation and map building. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1251–1256, Maui (Hawaii), novembre 2001.

[93] S. J. Julier and J. K. Uhlmann. A new extension to the Kalman filter to non linear systems. In *Proceedings of SPIE AeroSense : 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, Orlando (FL, Etats-Unis), 1997.

[94] S. J. Julier and J. K. Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *Proceedings of the American Control Conference*, volume 4, pages 2369–2373, Abulquerque (Nouveau-Mexique, Etats-Unis), juin 1997.

[95] S. J. Julier and J. K. Uhlmann. Simultaneous localisation and map building using split covariance intersection. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1257–1262, Maui (Hawaii), novembre 2001.

[96] I. K. Jung and S. Lacroix. High resolution terrain mapping using low altitude aerial stereo imagery. In *Proceedings of the 11th International Symposium on Robotics Research (ISRR'03)*, Sienne (Italie), octobre 2003.

[97] M. Kieffer, L. Jaulin, E. Walter, and D. Meizel. Robust autonomous robot localization using interval analysis. In *Reliable computing*, pages 337–362. Kluwer Academic Publishers, 2000.

[98] J. Knight. Computationally tractable SLAM. In *Report N OUEL 2232/2001*, Robotics Research Group, University of Oxford (UK), 2001.

[99] J. Knight, A. Davison, and I. Reid. Towards constant time SLAM using postponement. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 406–412, Maui (Hawaii), 2001.

[100] T. Kohonen. The self-organizing map. In *Proceedings of the IEEE*, volume 78(9), pages 1464–1480, 1990.

[101] D. Kortenkamp. *Cognitive maps for mobile robots : a representation for mapping and navigation - A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.* PhD thesis, The University of Michigan, Etats-Unis, 1993.

[102] D. Kortenkamp, R. P. Bonasso, and R. Murphy. *Artificial intelligence and mobile robots : case studies of successful robot systems.* American Association for Artificial Intelligence, 1998.

[103] D. Kortenkamp and T. Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 979–984, Seattle (WA, Etats-Unis), 1994.

[104] J. Kosecka. Visually guided navigation. In *Journal of Robotics and Autonomous Systems*, volume 21, pages 37–51, 1997.

[105] B. J. Kuipers. Modeling spatial knowledge. In *Cognitive science*, volume 2, pages 129–153, 1978.

[106] B. J. Kuipers and Y. T. Byun. A robot exploration mapping strategy based on a semantic hierarchy of spatial representations. In *Robotics and Autonomous Systems*, 1991.

[107] C. Kunz, T. Willeke, and I. Nourbakhsh. Automatic mapping of dynamic office environment. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-97)*, volume 2, pages 1681–1687, 1997.

[108] A. Kurz. Alef : an autonomous vehicle which learns basic skills and constructs maps for navigation. In *Robotics and Autonomous Systems*, volume 14, pages 172–183, 1995.

[109] N. M. Kwok. An efficient multiple hypothesis filter for bearing-only SLAM. In *Proceedings of IROS'04*, septembre 2004.

[110] R. Lakaemper, L. J. Latecki, X. Sun, and D. Wolter. Geometric robot mapping. In *Discrete Geometry for Computer Imagery (DGCI'05)*, Poitiers, avril 2005.

[111] T. Larue. *Un serveur de données géographiques pour robot mobile*. PhD thesis, Université Paris 6, Paris, décembre 1994.

[112] J.-C. Latombe. *Robot motion planning*. Kluwer Academic Publisher, Norwell (Massachusetts), Etats-Unis, 1991.

[113] J.-P. Laumond, editor. *La Robotique Mobile*. Hermès Sciences Publications, Paris, 2001.

[114] D. Lee. *The map-building and exploration strategies of a simple sonar-equipped mobile robot*. Press Syndicate of the University of Cambridge, Cambridge, UK, 1996.

[115] J. J. Leonard and H. F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publisher, Cambridge, MA, 1992.

[116] J. J. Leonard and J. S. Feder. A computationally efficient method for large scale concurrent mapping and localization. In J. Hollerbach and D. Koditschek, editors, *Robotics Research : Proceedings of the 9th International Symposium*, 2000.

[117] J. J. Leonard, P. M. Newman, R. J. Rikoski, J. Neira, and J. D. Tardós. Towards robust data association and feature modeling for concurrent mapping and localization. In *10th International Symposium of Robotics Research (ISRR'01)*, 2001.

[118] J. J. Leonard and R. J. Rikoski. Incorporation of delayed decision making into stochastic mapping. In D. Rus et S. Singh, editor, *Experimental Robotics VII, Lecture Notes in Control and Information Science*. Springer-Verlag, 2000.

[119] S. Leroy. *Outils géométriques pour la planification de chemins de robots mobiles non holonomes*. PhD thesis, Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS, Toulouse, France, décembre 1998.

[120] Levitt and Lawton. Qualitative navigation for mobile robots. In *Artificial Intelligence*, volume 44(3), pages 305–360, 1990.

[121] P. Lienhardt. Topological models for boundary representation : a comparison with n-dimensional generalized maps. In *Computer-Aided Design*, volume 23(1), pages 59–82, 1991.

[122] J. H. Lim and D. W. Cho. Physically based sensor modelling for a sonar map in a specular environment. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1714–1719, mai 1992.

[123] G. S. Lionis and K. J. Kyriapopoulos. A laser scanner based mobile robot SLAM algorithm with improved convergence properties. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02)*, pages 582–587, Lausanne (Suisse), octobre 2002.

[124] Y. Liu and S. Thrun. Results for outdoor-SLAM using sparse extended information filters. In *IEEE International Conference on Robotics and Automation (ICRA'03)*, pages 1227–1233, Taipei (Taiwan), 2003.

[125] C. Loader. Local search algorithms for 2D geometric object recognition. In *Technical report*, The University of Western Australia, 1995.

[126] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. In *Autonomous Robots*, volume 4, pages 333–349, 1997.

[127] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2D range scans. In *Journal of Intelligent and Robotic Systems*, volume 18(3), pages 249–275, 1997.

[128] D. Luzeaux, A. Dalgalarrondo, and D. Dufourd. Autonomous small robots for military applications. In *Proceedings of UVS Tech 2001*, Bruxelles (Belgique), décembre 2001.

[129] M. C. Martin and H. Moravec. Robot evidence grids. Technical report, Robotics Institute, Carnegie Mellon University, juin 1996.

[130] A. Martinelli, A. Tapus, K. O. Arras, and R. Siegwart. Multi-resolution SLAM for real world navigation. In *International Syposium on Robotics Research (ISRR'03)*, Sienne (Italie), octobre 2003.

[131] A. Martinelli, N. Tomatis, and R. Siegwart. Open challenges in SLAM : an optimal solution based on shifts and rotation invariants. In *International Conference on Robotics and Automation (ICRA'04)*, Nouvelle-Orleans (Etats-Unis), avril 2004.

[132] M. J. Matarić. Navigating with a rat brain : A neurobiologically-inspired model for robot spacial representations. In *From Animals to Animats 1*, 1990.

[133] M. J. Matarić. Integration of representation into goal-driven behavior-based robots. In *IEEE Transactions on Robotics and Automation*, volume 8 (3), pages 304–312, 1992.

[134] A. Meystel and A. Bathija. Tesselating and searching uncertain state spaces. In *Proceedings of the Performance Metrics for Intelligent Systems Workshop*, Gaithersburg (Maryland, Etats-Unis), 2002.

[135] D. Michelucci and M. Gangnet. Saisie de plans à partir de tracés à main levée. In *Proceedings of MICAD*, pages 96–110, Paris, 1984.

[136] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0 : an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *International Conference on Artificial Intelligence (AAAI'03)*, 2003.

[137] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM : a factored solution to simultaneous mapping and localization. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2003.

[138] H. P. Moravec and M. C. Martin. Robot navigation by 3D spatial evidence grids. In *Mobile Robot Laboratory, Robotics Institute, Carnegie Mellon University*, 1994.

[139] P. Moutarlier. *Modélisation autonome de l'environnement par un robot mobile*. PhD thesis, Laboratoire d'Automatique et d'Analyse des Systèmes du CNRS, Toulouse, France, octobre 1991.

[140] P. Moutarlier and R. Chatila. Stochastic multisensory data fusion for mobile robot location and environment modelling. In *5th International Symposium on Robotics Research (ISRR'89)*, 1989.

[141] P. Moutarlier and R. Chatila. Incremental free-space modelling from unertain data by an autonomous mobile robot. In *International Conference on Robotics and Intelligent Systems (IROS'91)*, novembre 1991.

[142] K. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neuronal Information Processing Systems 11*. The MIT Press, 1999.

[143] U. Nehmzow, D. Gelder, and T. Duckett. Automatic selection of landmarks for mobile robot navigation. In *Technical report [UMCS-00-7-1]*, University of Manchester (Royaume-Uni), juillet 2000.

[144] U. Nehmzow and T. Smithers. Using motor actions for location recognition. In F. Varela and P. Bourgine, editors, *Towards a Practice of Autonomous Systems : Proceedings of the first Conference on Artificial Life*. MIT Press, décembre 1991.

[145] J. Neira and J. D. Tardos. Data association in stochastic mapping using the joint compatibility test. In *IEEE Transactions on Robotics and Automation*, volume 17(6), décembre 2001.

[146] J. Neira, J. D. Tardos, and J. A. Castellanos. Linear time vehicle relocation in SLAM. In *IEEE International Conference on Robotics and Automation*, Taipei (Taiwan), mai 2003.

[147] J. Neira, J. D. Tardos, J. Horn, and G. Schmidt. Fusing range and intensity images for mobile robot localisation. In *IEEE Transactions on Robotics and Automation*, février 1999.

[148] P. Newman. *On the structure and solution of the simultaneous localisation and map building problem*. PhD thesis, The University of Sydney, mars 1999.

[149] P. M. Newman and J. J. Leonard. Pure range-only sub-sea SLAM. In *IEEE International Conference on Robotics and Automation (ICRA'03)*, Taipei (Taiwan), septembre 2003.

[150] J. Nievergelt and F. P. Preparata. Plane-sweep algorithms for intersecting geometric figures. In *Commun ACM*, volume 25, pages 739–747, 1982.

[151] Department of Defense (Etats-Unis). Interface standard for Vector Product Format. In *Rapport MIL-STD-2407*, juin 1996.

[152] G. Oriolo, G. Ulivi, and M. Vendittelly. Fuzzy maps : a new tool for mobile robot perception and planning. In *Journal of Robotic Systems*, volume 14(3), pages 179–197, 1997.

[153] H. Gonzáles-Ba os, E. Mao, J. C. Latombe, T. M. Murali, and A. Efrat. Planning robot motion strategies for efficient model construction. In *Robotics Research - the 9th International Symposium*, 1999.

[154] D. Pagac, E. M. Nebot, and H. Durrant-Whyte. An evidential approach to map-building for autonomous vehicles. In *IEEE Transactions on Robotics and Automation*, volume 14(4), pages 623–629, 1998.

[155] J. Piaget and B. Inhelder. *The child's conception of space*. Norton, New-York (Etats-Unis), 1967.

[156] A. Poncella, E. J. Perez, A. Bandera, C. Urdiales, and F. Sandoval. Efficient integration of metric and topologic maps for directed exploration of unknown environments. In *Robotics and Autonomous Systems*, volume 41(1), pages 21–40, octobre 2002.

[157] C. Pradalier, P. Bessière, and C. Laugier. Driving on a known sensori-motor trajectory with a car-like robot. In *International Symposium on Experimental Robotics (ISER'04)*, Singapour, juin 2004.

[158] C. Pradalier, J. Hermosillo, C. Koike, C. Braillon, P. Bessière, and C. Laugier. The CyCab : a car-like robot navigating autonomously and safely among pedestrians. In *Robotics and Autonomous Systems*, volume 50(1), janvier 2005.

[159] C. Pradalier and S. Sekhavat. Concurrent matching, localization and map building using invariant features. In *International Conference on Intelligent Robots and Systems (IROS'02)*, 2002.

[160] F. P. Preparata and M. I. Shamos. *Computational Geometry : An Introduction*. Springer Verlag, 1985.

[161] A. J. Prescott. *Explorations in reinforcement and model-based learning (chapter 7 - Representations for wayfinding : topological models).* PhD thesis, University of Sheffield (Royaume-Uni), 1993.

[162] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipees in Pascal : The Art of Scientific Computing.* Cambrige University Press, New-York (Etats-Unis), 1989.

[163] F. Puitg and J.-F. Dufourd. Formalizing mathematics in higher-logic : A case study in geometric modelling. In *Theoretical Computer Science*, volume 234, pages 1–57. Elsevier Science, 2000.

[164] M. Ribo and A. Pinz. A comparison of three uncertainty calculi for building sonar-based occupancy grids. In *Robotics and Autonomous Systems*, volume 35(3-4), pages 201–209, 2001.

[165] P. Rigeaux, M. Scholl, and A. Voisard. *Spatial database with application to GIS.* Academic Press, 2002.

[166] T. Röfer. Using histogram correletation to create consistent laser scan maps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 625–630, octobre 2002.

[167] S. Rolfes and M.-J. Rendas. Statistical environment representation for navigation in natural environments. In *Robotics and Autonomous Systems*, volume 41, pages 129–136, 2002.

[168] B. Schiele and J. L. Crowley. Comparison of position estimation techniques using occupancy grids. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1628–1634, San Diego (Etats-Unis), mai 1994.

[169] A. C. Schultz, W. Adams, B. Yamauchi, and M. Jones. Unifying exploration, localization, navigation and planning through a common representation. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 2651–2658, Detroit (Etats-Unis), 1999.

[170] R. Siegwart and I. R. Nourbakhsh. *Introduction to autonomous mobile robots.* The MIT Press, Cambridge (MA, Etats-Unis), 2004.

[171] R. Sim and G. Dudek. Learning environmental features for pose estimation. In *ICCV'99*, pages 1217–1222, 1999.

[172] S. S. Simhon. Islands of reliability for hybrid topological-metric mapping. In *Technical report TR-CIM-99-01*, Mobile Robotics Laboratory, McGill University (Canada), janvier 1999.

[173] R. Simmons and S. Koenig. Probabilistic navigation in partially observable environments. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1080–1087, San Francisco, Etats-Unis, 1995.

[174] R. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In O. Faugeras and G. Giralt, editors, *Robotics Research, the Fourth International Symposium*, pages 467–474. The MIT Press, 1988.

[175] A. Tapus, N. Tomatis, and R. Siegwart. Topological global localization and mapping with fingerprints and uncertainty. In *International Symposium on Experimental Robotics*, juin 2004.

[176] J. D. Tardos, J. Neira, P. M. Newman, and J. J. Leonard. Robust mapping and localization in indoor environment using sonar data. In *The International Journal of Robotic Research*, volume 21(4), pages 311–330, avril 2002.

[177] C. J. Taylor and D. J. Kriegman. Vision-based motion planning and exploration algorithms for mobile robots. In *IEEE Transactions on Robotics and Automation*, 1999.

[178] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. In *Artificial Intelligence*, volume 99(1), pages 21–71, 1998.

[179] S. Thrun. A probabilistic on-line mapping algorithm for teams of mobile robots. In *The International Journal of Robotics Research*, volume 20(5), pages 335–363, mai 2001.

[180] S. Thrun. Robotic mapping : a survey. In *Technical Report CMU-CS-02-111*, février 2002.

[181] S. Thrun. Learning occupancy grid maps with forward sensor models. In *Autonomous Robots*, volume 15, pages 111–127, 2003.

[182] S. Thrun, J.-S. Gutmann, D. Fox, W. Burgard, and B. Kuipers. Integrating topological and metric maps for a mobile robot navigation : a statistical approach. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98)*, juillet 1998.

[183] S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and A. Y. Ng. Simultaneous mapping and localization with sparse extended information filters. In *Technical Report CMU-CS-02-112*, 2002.

[184] N. Tomatis, I. Nourbakhsh, and R. Siegwart. Hybrid simultaneous localization and map building : a natural integration of topological and metric. In *Robotics and Autonomous Systems*, volume 44, pages 3–14, 2003.

[185] M. A. Trick. A tutorial on dynamic programming. In *Support de cours de Carnegie Mellon University sur http ://mat.gsia.cmu.edu/classes/dynamic/dynamic.html*, 1997.

[186] W. T. Tutte. Graph theory. In Addison-Wesley, editor, *Encyclopedia of Mathematics and its applications*, 1984.

[187] I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In *IEEE International Conference on Robotics and Automation*, pages 1023–1029, San Francisco (Etats-Unis), avril 2000.

[188] R. Unnikrishnan and A. Kelly. A constrained optimization approach to globally consistent mapping. In *International Conference on Intelligent Robots and Systems, (IROS'02)*, pages 564–569, Lausanne (Suisse), 2002.

[189] D. van Zwynsvoorde. *Construction incrémentale de modèles topologiques pour la navigation d'un robot mobile*. PhD thesis, Laboratoire d'Automatique et d'Architecture des Systèmes du CNRS - Université Paul Sabatier, Toulouse, France, décembre 2000.

[190] M. Veeck and W. Burgard. Learning polyline maps from range scan data acquired with mobile robots. In *International Conference on Intelligent Robots and Systems, (IROS'04)*, Sendai (Japon), septembre - octobre 2004.

[191] A. Victorino, P. Rives, and J.-J. Borrelly. Localisation d'un robot mobile et cartographie par télémétrie laser. In *Journées des Jeunes Chercheurs en Robotique (JJCR'00)*, Bourges, février 2000.

[192] A. C. Victorino. *La commande référencée capteur : une approche robuste au problème de navigation, localisation et cartographie simultanées pour un robot d'intérieur*. PhD thesis, Université de Nice-Sophia Antipolis, septembre 2002.

[193] C.-C. Wang and C. Thorpe. Simultaneous localization and mapping with detection and tracking of moving objects. In *International Conference on Robotics and Automation (ICRA'02)*, 2002.

[194] C.-C. Wang and C. Thorpe. A hierarchical object-based representation for simultaneous localization and mapping. In *International Conference on Intelligent Robots and Systems, (IROS'04)*, Sendai (Japon), septembre - octobre 2004.

[195] Z. Wang, S. Huang, and G. Dissanayake. D-SLAM : Decoupled Localization and Mapping for autonomous robots. In *International Symposium on Robotics Research (ISRR'05)*, San Francisco (Etats-Unis), octobre 2005.

[196] G. Weiss, C. Wetzler, and E. Puttkamer. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *International Conference on Intelligent Robots and Systems, (IROS'94)*, pages 595–601, Munich (Allemagne), 1994.

[197] C. Wiedmann, C. Heipke, H. Mayer, and O. Jamet. Empirical evaluation of automatically extracted road axes. In *Empirical Evaluation Techniques in Computer Vision*, pages 172–187. IEEE Computer Society Press, 1998.

[198] O. Wijk and H. I. Christensen. Triangulation-based fusion of sonar data with application in robot pose tracking. In *IEEE Transactions on Robotics and Automation*, volume 16(6), pages 740–741, 2000.

[199] S. B. Williams. *Efficient solutions to autonomous mapping and navigation problems.* PhD thesis, Australian Center for Fields Robotics, University of Sydney, septembre 2001.

[200] S. B. Williams, G. Dissanayake, and H. Durrant-Whyte. An efficient approach to the simultaneous localization and map-building problem. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'02)*, volume 1, pages 406–411, Washington (DC, Etats-Unis), 2002.

[201] B. Yamauchi. A frontier-based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 146–151, 1997.

[202] B. Yamauchi and R. Beer. Spatial learning for navigation in dynamic environment. In *IEEE Transactions on Systems, Man and Cybernetics-Part B*, volume 26(3), pages 496–505, 1996.

[203] W. Yeap. Towards a computational theory of cognitive maps. In *Artificial Intelligence*, volume 32, pages 297–360, 1988.

[204] E. Yeh and D. J. Kriegman. Toward selecting and recognizing natural landmarks. In *IEEE/RSJ International Conference on Robots and Systems*, 1995.

[205] A. Zelinski. A mobile robot exploration algorithm. In *IEEE Transactions on Robotics and Automation*, pages 707–717, décembre 1992.

[206] L. Zhang and B. K. Ghosh. Geometric feature based 2 1/2 map building and planning with laser, sonar and tactile sensors. In *International Conference on Intelligent Robots and Systems (IROS'00)*, 2000.