Localisation d'un robot humanoïde en milieu intérieur non-contraint

Mathieu Nowakowski

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à MINES ParisTech

Localisation d'un robot humanoïde en milieu intérieur non-contraint

Soutenue par
Mathieu NOWAKOWSKI
Le 3 Avril 2019

École doctorale n°621
Ingénierie des Systèmes, Matériaux, Mécanique, Energétique

Spécialité
Informatique temps réel, robotique et automatique

Composition du jury :

David FILLIAT Professeur, ENSTA ParisTech	*Président du jury*
Paul CHECCHIN Professeur, Institut Pascal (UCA)	*Rapporteur*
Olivier STASSE Directeur de recherche, LAAS - CNRS	*Rapporteur*
Samia BOUCHAFA-BRUNEAU Professeur, Univ. Evry Val d'Essonne	*Examinateur*
Fabien MOUTARDE Professeur, Mines ParisTech	*Directeur de thèse*
Cyril JOLY Chargé de recherche, Mines ParisTech	*Maître de thèse*
Sébastien DALIBARD Ingénieur, SoftBank Robotics Europe	*Encadrant industriel*

MINES ParisTech | PSL

Remerciements

Lors de ces trois années de recherche, j'ai eu la chance de rencontrer, de débattre et d'être aidé par de nombreuses personnes à qui je souhaite exprimer ma gratitude ici, en commençant par les membres de mon jury que je remercie pour leurs différents retours et pour l'attention qu'ils ont portée à mes travaux.

Si ce doctorat a constitué une expérience aussi enrichissante, je le dois notamment à mes encadrants. Je tiens ainsi à remercier Fabien Moutarde, mon directeur de thèse, qui m'a fait confiance et sans qui cette thèse n'aurait pas existé. Je remercie également Cyril Joly, mon maître de thèse, pour son aide et ses nombreux conseils. Enfin, je remercie Sébastien Dalibard, mon encadrant industriel, auprès de qui j'ai énormément appris, et qui s'est toujours montré encourageant et compréhensif vis-à-vis de mes responsabilités de doctorant.

Je remercie chaleureusement mes collègues d'Agility à SoftBank Robotics Europe. J'y ai été très bien accueilli et constamment soutenu : venir travailler dans cette équipe a toujours été un plaisir. Chacun m'a aidé à sa manière et sans eux, le contenu de ce manuscrit aurait été très différent. Merci à vous tous : Nicolas, Scarlett, Lucas, Justine, Sébastien B., Kenzo, Kyo, Anne et Aldenis.

Bien que moins souvent présent au laboratoire, je m'y suis tout autant senti à ma place grâce aux membres du CAOR. J'ai eu la chance d'y rencontrer d'autres doctorants brillants et passionnés. Xavier, Yannick, Edgar, Philip, Guillaume, Marion, Paul, Martin, Aubrey, Arthur, Florent, Daniele… je garderai d'agréables souvenirs des moments partagés avec vous.

Pour finir, je tiens à remercier mes proches. Marien, qui s'est intéressé à des sujets qui lui étaient étrangers et m'a encouragé. Adrien, qui m'a parfois bousculé dans ma routine de thésard, me permettant ainsi de me changer les idées. Et bien évidemment Victoria, qui a été là pour moi, partageant mon enthousiasme les jours de réussite et me remotivant lors des inévitables jours de bugs et d'échecs. Je remercie ma famille, en particulier mes parents qui m'ont transmis leur goût des sciences et du partage. Enfin, un grand merci à mon grand-père : tu t'es intéressé à mon travail et ne manquais jamais de prendre des nouvelles de "pé-père".

Table des matières

Liste des figures

Liste des tableaux

Chapitre 1

Introduction et contexte

Sommaire

Les travaux présentés dans ce manuscrit s'intéressent à la localisation autonome de robot humanoïde en milieu intérieur non-contraint. Le problème étant vaste, ce chapitre a pour but de mettre en lumière le contexte dans lequel a été effectuée cette thèse, afin de permettre au lecteur d'identifier le cadre de nos recherches.

Nos travaux sont particulièrement guidés par les conditions d'utilisation de notre robot. Le contexte de cette thèse est donc indissociable de l'aspect pratique. Pour présenter ce contexte particulier, ce chapitre se découpera en trois parties.

Dans la partie 1.1, le domaine de la robotique sociale sera introduit. Nous profiterons de cette partie pour présenter l'entreprise SoftBank Robotics Europe au sein de laquelle nos travaux ont été réalisés.

La partie 1.2 présentera les diverses contraintes liées aux conditions d'utilisation particulières de notre robot.

Pour finir, la partie 1.3 résumera l'organisation de ce manuscrit.

1.1 Les robots sociaux

Après avoir été cantonnés au secteur industriel, les robots investissent de plus en plus notre quotidien. Une nouvelle branche de la robotique s'est ainsi spécialisée dans la conception de robots dont le rôle principal est la création d'interactions fortes avec leurs utilisateurs. Ces robots sont qualifiés de sociaux.

Le secteur de la robotique sociale est aujourd'hui en pleine expansion. Dans de nombreux contextes, la capacité d'établir une interaction forte avec l'utilisateur constitue un atout. Par exemple, chez les particuliers, certains robots sociaux remplissent les fonctions d'assistant personnel. Parmi eux, le robot Jibo [1] (Figure 1.1-a), ajoute une dimension émotionnelle aux interactions des assistants vocaux plus standards comme l'Amazon Echo ou le Google Home. Jibo retransmet des émotions lorsqu'il interagit avec ses utilisateurs grâce à des mouvements, des rire, ou encore à l'aide de modulations de son ton.

Les robots sociaux investissent également le secteur de la santé. Le robot phoque Paro [2] (Figure 1.1-b) est par exemple utilisé à des fins thérapeutiques dans des établissements où sont traitées des personnes atteintes de la maladie d'Alzheimer. Reposant sur le principe de la thérapie animalière, Paro aide à améliorer le bien-être de ces personnes.

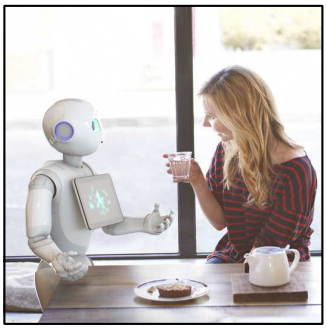
Enfin, d'autres secteurs comme la vente, la finance ou le tourisme s'intéressent aux robots sociaux. Placés dans des magasins, ces robots suscitent la curiosité de la clientèle et permettent d'y augmenter la fréquentation. Dans cette optique, le robot Pepper [3] (Figure 1.1-c) est quotidiennement utilisé dans différentes boutiques au Japon.

Ce dernier exemple nous permet de présenter une branche de la robotique sociale qui se spécialise dans la conception de robots humanoïdes. La forme humanoïde possède divers atouts, notamment pour établir des interactions fortes. En effet, un utilisateur interagit de manière plus naturelle avec des robots à forme humaine, ce qui rend

1. https://www.jibo.com/
2. http://www.phoque-paro.fr/
3. https://www.softbankrobotics.com/emea/index.php/fr/pepper

(a) Robot Jibo

(b) Robot Paro



(c) Robot Pepper

FIGURE 1.1: Exemples de robots sociaux, utilisés respectivement chez les particuliers (a), dans les hôpitaux (b) et dans différents types d'entreprise (c) (vente, tourisme, finance, etc.).

l'interaction plus simple et ergonomique.

Les travaux de cette thèse s'ancrent dans le domaine des robots sociaux à forme humanoïde. Plus particulièrement, elle s'intéresse à la localisation autonome du robot Pepper, conçu par l'entreprise SoftBank Robotics Europe.

1.1.1 SoftBank Robotics Europe

Fondée en 2005 par Bruno Maisonnier, Aldebaran Robotics est la première société française qui a investi le marché du robot humanoïde. Rachetée depuis par le groupe japonais SoftBank, et renommée SoftBank Robotics Europe, la société a gardé sa volonté de développer des robots bienveillants destinés au grand public.

Trois plates-formes sont développées par ses ingénieurs (Figure 1.2). NAO, petit robot humanoïde d'une soixantaine de centimètres, est particulièrement présent dans les domaines de la recherche et de l'éducation. Romeo, son cadet, mesurant 1 m 40, a fait l'objet de partenariats avec six laboratoires de robotique dans l'objectif de créer un robot capable de venir en aide aux personnes âgées, ou en perte d'autonomie. Pepper, du haut de ses 1 m 20 est le plus jeune du groupe. Il a été conçu pour servir de vendeur, conseiller et animateur dans les magasins japonais du groupe SoftBank. Aujourd'hui, 25 000 robots de SoftBank Robotics Europe sont utilisés dans plus de 70 pays.

L'objectif de SoftBank Robotics Europe est d'amener ses robots au contact des gens. Au Japon, cette aspiration est visible par l'utilisation quotidienne de robots Pepper dans de nombreux magasins. En France, des projets en partenariat avec des grands groupes comme la SNCF, Renault, Costa Croisière ou certains magasins Carrefour, ont permis à Pepper de rencontrer le grand public. Dans ces exemples, Pepper avait pour principales missions d'enquêter sur la satisfaction des clients, de les divertir et de les renseigner.

Cependant, les utilisations classiques de Pepper ne profitent pas pleinement de la mobilité du robot. Bien que le robot soit équipé d'une base holonome, il est dans la plupart des cas utilisé à une pose [4] fixe, comme cela est visible sur la Figure 1.3.

4. Dans ce manuscrit, la pose d'un robot correspond à sa position et son orientation.

FIGURE 1.2: Robots développés par SoftBank Robotics Europe. De gauche à droite : NAO (2006), Romeo (2011) et Pepper (2014).

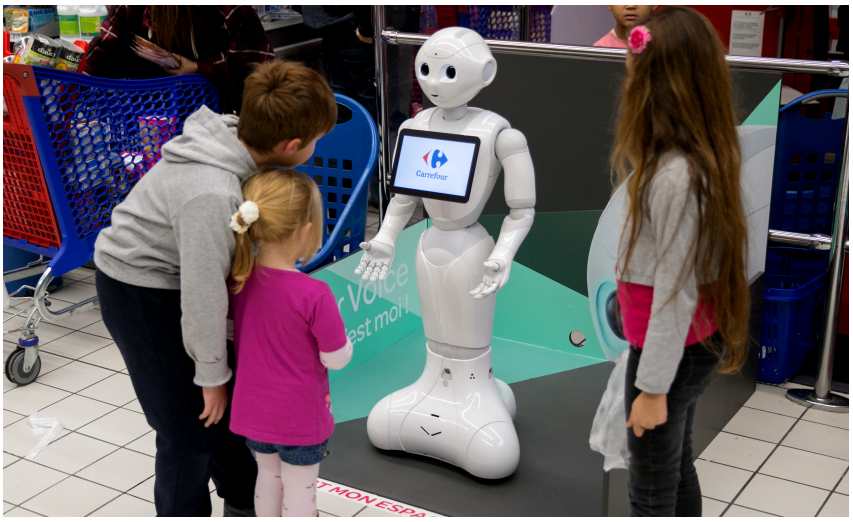


FIGURE 1.3: Pepper divertissant des enfants dans un magasin Carrefour. Bien que le robot soit équipé d'une base holonome, ce dernier est utilisé de manière statique. Cette utilisation vient du fait que le robot ne sait pas estimer correctement sa pose.

Cette utilisation statique est due au fait que lorsque Pepper se déplace, il se perd et n'est pas capable de revenir tout seul à des poses clés de fonctionnement, comme sa base de rechargement. Ce problème est inévitable en robotique mobile mais peut être résolu en donnant au robot le moyen d'estimer sa pose dans son environnement. Cette capacité correspond à la localisation autonome du robot.

1.1.2 Intérêts de la localisation autonome

La localisation autonome d'un robot correspond à la capacité pour ce dernier d'estimer sa pose dans son environnement. D'un point de vue pratique, donner à Pepper

la capacité de se localiser revient à le rendre capable de se déplacer dans l'espace. Sans capacité de localisation, si le robot se déplace, il risque de se perdre et de ne pas savoir revenir à des poses stratégiques de fonctionnement.

Le déplacement dans l'espace constitue un des atouts majeurs de la robotique mobile. Il ajoute une dimension physique au système qu'on ne peut retrouver avec des agents virtuels. De nombreuses applications peuvent être envisagées concernant Pepper. Par exemple, celle de guide accompagnant des personnes à des endroits précis (un rayon dans un magasin, une voie dans une gare, ou bien une salle dans un musée). De ce fait, l'intérêt pour SoftBank Robotics Europe de rendre leur robot capable de se localiser devient évident.

Des travaux ont déjà été réalisés à cet effet, notamment dans le cadre d'une précédente thèse s'intéressant à la localisation des robots NAO et Pepper (WIRBEL, 2014). Cependant, la méthode proposée a rencontré peu de succès en pratique, en raison des contraintes lors de la mise en application.

1.2 Localisation et limites applicatives

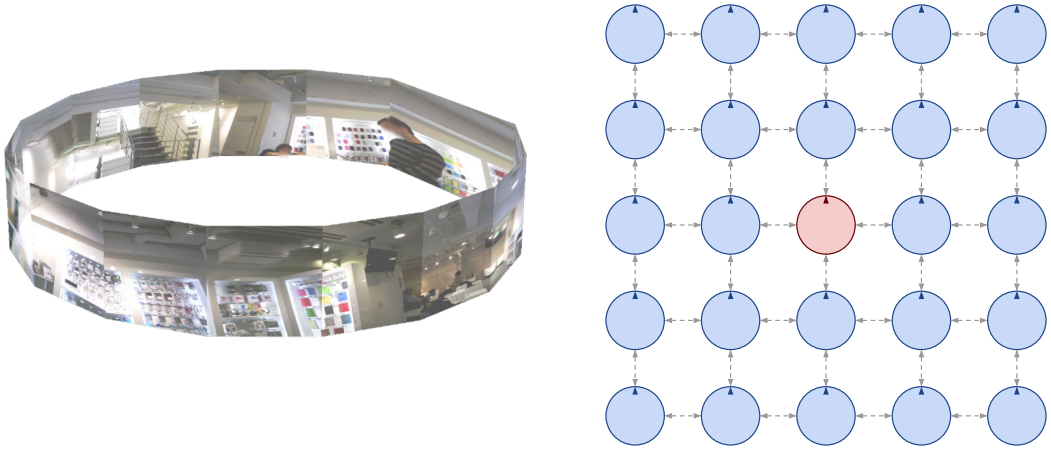
L'objectif de cette partie est de présenter les problèmes rencontrés par la solution de localisation proposée dans (WIRBEL, 2014), qui viennent des conditions d'utilisation particulières du robot Pepper. Pour cela, nous devons y introduire le concept de carte qui sera plus détaillé dans le Chapitre 2, où nous dresserons un portrait des différentes méthodes de localisation autonome de la littérature.

Le terme de localisation renvoie fortement au concept de carte, puisque l'estimation de pose est réalisée par rapport à un référentiel. Dans le cadre de la localisation autonome de robots, la carte peut être connue ou doit être construite en même temps que le robot découvre son environnement. Comme Pepper est utilisé dans de nombreuses boutiques, il est très souvent manipulé par des personnes non-expertes en robotique. Le robot doit être le plus autonome possible et ne peut compter sur la spécification d'une carte par un utilisateur. Le problème de la localisation de Pepper comprend donc une phase de cartographie (automatique ou supervisée), dans laquelle le robot crée une représentation de son environnement.

1.2.1 Approche par panoramas visuels

Dans (WIRBEL, 2014), la méthode de localisation et de cartographie présentée se veut compatible avec les robots NAO et Pepper de SoftBank Robotics Europe (Figure 1.2). Pour cela, l'approche se base sur les caméras monoculaires placées dans la tête de ces robots (WIRBEL et al., 2013). L'environnement est représenté par un graphe de panoramas visuels connectés entre eux, comme l'illustre la Figure 1.4. En fonctionnement, lorsque le robot acquiert une image, il se localise par rapport à un panorama de référence et en déduit sa pose dans l'environnement.

En pratique, l'utilisation de cette approche sur Pepper donne au robot la capacité d'estimer correctement son orientation par rapport à un panorama. Dans une boutique,



(a) Panorama visuel (b) Carte utilisée

FIGURE 1.4: Utilisation de panoramas visuels. Dans (WIRBEL, 2014), des expériences menées dans des boutiques japonaises utilisent une carte qui s'organise sous la forme d'une grille de panoramas, espacés de 50 cm. Chaque panorama est orienté dans la même direction et sa position est connue par rapport au panorama d'origine, en rouge sur (b). Se localiser dans cette carte revient à estimer sa pose par rapport à un panorama de référence, puis d'en déduire sa pose par rapport au panorama d'origine.

cette capacité est essentielle. Elle permet par exemple d'éviter des situations où le robot est tourné vers l'étalage d'un rayon au lieu d'être tourné vers les clients qu'il doit renseigner.

Cependant, cette solution possède des limitations. D'abord, bien qu'elle estime correctement l'orientation du robot, elle ne fournit pas d'information précise sur sa position dans le plan. Ensuite, la construction de la carte est une étape complexe puisqu'elle nécessite l'acquisition de plusieurs panoramas visuels.

Ce second défaut est particulièrement problématique dans les utilisations classiques de Pepper. Du fait de son aspect humanoïde[5], mais aussi puisque Pepper est vendu comme un robot social, les contraintes comportementales de Pepper sont très importantes. Pour ne pas paraître étrange à ses utilisateurs non-experts en robotique, le robot doit adopter un comportement proche de celui des humains. Ainsi, la localisation doit s'effectuer en arrière plan et être imperceptible par les utilisateurs. Or, dans cette méthode, l'étape de cartographie, correspondant à la création des panoramas visuels, active différentes articulations du robot. Pour acquérir un panorama visuel à partir de la caméra placée sur son front, Pepper doit tourner la tête et tourner sur lui-même. Ce comportement n'est pas intuitif pour un utilisateur non-expert en robotique et nuit à l'interaction et l'acceptation du robot. De plus, ce comportement rend la phase de cartographie trop longue et fastidieuse pour certains utilisateurs.

Pour ces raisons, cette solution est peu utilisée en pratique. Afin d'aboutir dans nos

5. Bien que Pepper ne soit pas un robot bipède, le terme humanoïde lui est associé et est lié à son expressivité. Pepper possède une tête, deux bras, un torse, et une jambe, qui lui permettent de reproduire des attitudes humaines.

travaux à une solution exploitable sur Pepper, ces derniers sont fortement guidés par le contexte d'utilisation du robot.

1.2.2 Contraintes applicatives

Dans la majorité des cas, Pepper est manipulé par des personnes non-expertes en robotique. Le robot ne doit pas être une source de travail supplémentaire conséquente. Si on peut envisager que le gérant d'une boutique fasse faire à Pepper un « tour du propriétaire », la phase de cartographie du robot ne doit pas résulter en une acquisition longue ou complexe, comme avec l'utilisation des panoramas visuels.

Plusieurs contraintes applicatives peuvent être identifiées et sont liées à : (i) l'environnement de fonctionnement du robot, (ii) la plate-forme Pepper, et (iii) les conditions d'utilisation du robot.

Environnement de fonctionnement : Le robot Pepper est utilisé dans des environnements variés comme des magasins, des banques ou des gares. La solution de localisation doit donc se montrer suffisamment générique pour fonctionner dans des environnements différents.

De plus, Pepper est autonome et ne doit pas dépendre de dispositifs tiers pour se localiser. Cela rend impossible deux types de stratégies. D'abord, celles où l'environnement est équipé de marqueurs d'aide à la localisation, comme par exemple des QR-Codes[6] (GEORGE et MAZEL, 2013) ou des Beacons (FARAGHER et HARLE, 2015). Ensuite, Pepper ne peut compter sur l'utilisation d'un robot spécialement équipé pour cartographier l'environnement à sa place (JIRKU, KUBELKA et REINSTEIN, 2016).

Le robot Pepper : La solution de localisation doit être directement fonctionnelle sur Pepper. Ainsi, aucun capteur additionnel ne peut être ajouté à la plate-forme. De même, les calculs de localisation doivent être réalisés sur le robot. En effet, il n'est pas possible d'assurer la présence d'une connexion internet de bonne qualité dans les divers environnements de fonctionnement de Pepper, ce qui empêche de déporter les calculs sur une plate-forme plus puissante.

Plus de détails sur la plate-forme Pepper seront donnés dans la partie 2.3 de ce manuscrit.

Les conditions d'utilisation : Pour finir, plusieurs contraintes découlent de la nature sociale de Pepper. Rappelons ici que le rôle principal de Pepper est de créer des interactions fortes avec ses utilisateurs. Les tâches qui ne participent pas à l'interaction ou à des problématiques de sécurité sont alors moins prioritaires et ne doivent pas entraver les comportements du robot. Ainsi les stratégies de localisation ne doivent pas générer de mouvements parasites (comme des arrêtes impromptus, des rotations complètes, ...), et doivent se satisfaire des ressources computationnelles restantes.

6. https://www.youtube.com/watch?v=p8m4wH2a068

Une autre difficulté provient des nombreuses interactions que mène Pepper, et est visible au niveau des acquisitions de ses différents capteurs. Par exemple, les images issues des caméras de Pepper contiennent souvent des occultations causées par un utilisateur se tenant dans le champ de vision du capteur.

1.3 Organisation du manuscrit et contributions

Cette thèse se penche donc sur un problème de localisation fortement ancré dans un contexte applicatif. Notre objectif final, qui cherche à rendre Pepper capable d'estimer sa pose dans un environnement, doit donc prendre en compte les contraintes d'utilisation du robot.

Quatre chapitres suivront cette introduction. Le Chapitre 2, constituera un état-de-l'art des principales méthodes de localisation et de cartographie. Une analyse des approches de la littérature nous permettra de montrer les limitations de la solution logicielle de localisation utilisée sur Pepper, et de mettre en lumière les objectifs de cette thèse. La présentation de nos travaux s'organisera ensuite selon deux chapitres.

Dans le Chapitre 3, le problème de la relocalisation dans une carte sera étudié. Lorsque le robot est allumé dans un environnement connu, il doit être en mesure d'estimer sa position sans information *a priori* sur sa pose. Si les méthodes de l'état-de-l'art se basent souvent sur l'utilisation de l'apparence visuelle pour résoudre ce problème, cette dernière est tenue en échec dans des environnements visuellement redondants. Nos travaux cherchent alors à combiner les données issues des capteurs visuels et Wi-Fi de Pepper pour permettre au robot de se relocaliser dans une carte.

Le chapitre 4 aborde ensuite la question de la création d'une carte métrique à partir de la caméra stéréo de Pepper. Les approches classiques de SLAM visuel métrique basent leur fonctionnement sur le suivi de primitives visuelles entre plusieurs observations. Cependant, l'utilisation de Pepper et les contraintes matérielles de la plate-forme conduisent à l'acquisition d'images floues ou dont les champs de vue se recouvrent peu, ce qui empêche le suivi de telles primitives. Pour compenser ces pertes de suivi visuel, nous proposons d'inclure dans notre système des contraintes issues des mesures odométriques du robot. Ces contraintes sont prises en compte dans une optimisation par ajustement de faisceaux et permettent la création et maintenance d'une carte cohérente dans une approche de SLAM basse fréquence.

Chapitre 2

Localisation d'un robot en milieu intérieur

Sommaire

Ce chapitre constitue un état-de-l'art des méthodes de localisation en milieu intérieur, et se divise en trois parties. Dans un premier temps, le problème de la localisation et cartographie simultanées est introduit en 2.1. Ensuite, un portrait de la littérature des méthodes de localisation et de cartographie est dressé en 2.2. Pour finir, la partie 2.3 détaille la plate-forme robotique utilisée dans cette thèse, ainsi que les limitations de sa solution de localisation actuelle.

2.1 Introduction à la localisation et cartographie simultanées

2.1.1 Se localiser dans l'espace

L'action de localisation revient à déterminer sa pose par rapport à un repère de référence. Cette problématique n'est pas apparue avec l'émergence des nouvelles technologies mais est profondément ancrée dans le vivant. De nombreux exemples de localisation pourraient être listés, allant de l'oiseau messager parcourant plusieurs centaines de kilomètres, à l'étudiant se rendant à la machine à café la plus proche. Si l'un de ces exemples paraît moins impressionnant, il correspond pourtant bien à une tâche de localisation brillamment exécutée. La localisation est une tâche réalisée en permanence par notre cerveau. Cependant, cette dernière est réalisée de manière quasi-inconsciente, puisqu'elle ne nous apparaît que lorsqu'elle échoue ou demande un effort particulier.

Il est possible d'identifier deux stratégies de localisation lorsqu'on se déplace dans son environnement. La première repose sur l'apport d'information proprioceptive. Dans le noir complet, il est par exemple possible de monter l'escalier vers sa chambre en comptant ses pas. Cependant, se rendre jusqu'à la station de métro la plus proche les yeux fermés est une tâche beaucoup plus complexe. Pour ce type de trajet, une information extéroceptive s'ajoute à notre proprioception et se base sur l'emploi de repères fixes dans l'environnement. Pour les humains, ces repères sont majoritairement visuels. C'est d'ailleurs pour cette raison que les environnements visuellement redondants mettent souvent notre localisation en échec (SOUMAN et al., 2009).

Les produits technologiques les plus connus du grand public répondant à des problématiques de localisation sont sans aucun doute les systèmes de navigation. Utilisés quotidiennement par de nombreux automobilistes, leur fonctionnement repose sur l'emploi des signaux satellites du programme GPS (*Global Positionning System*) et le principe de la trilatération. Sur des produits industriels destinés à un large public comme nos smartphones, la précision du GPS varie entre 5 et 10 m. Dans notre contexte de localisation d'un robot mobile, une telle précision est insuffisante puisqu'elle ne permet pas de distinguer de quel côté d'un mur se situe un robot. Cette incertitude sur la pièce dans laquelle se trouve le robot peut alors générer des erreurs comme le lancement de comportements inadéquats. De plus, dans les environnements urbains, les signaux satellites utilisés souffrent de l'effet canyon (CUI et GE, 2003) qui dégrade la précision du système. Pour finir, les signaux du programme GPS deviennent tout bonnement inutilisables à l'intérieur de la plupart des bâtiments, car ils sont absorbés par les parois.

À la fin des années 1980, les chercheurs en robotique ont commencé à se poser la

question de la localisation autonome de systèmes robotiques en milieu intérieur (CHATILA et LAUMOND, 1985 ; SMITH et CHEESEMAN, 1986 ; DURRANT-WHYTE, 1988). De nombreux travaux ont proposé de considérer le problème de la localisation dans un environnement inconnu comme allant de pair avec le problème de la cartographie. Ces méthodes ont alors été regroupées sous l'acronyme SLAM pour *Simultaneous Localization And Mapping*, soit le problème de la localisation et cartographie simultanées.

2.1.2 Principes du SLAM

Le problème du SLAM se formule comme :

> *« la démarche par laquelle un robot mobile peut construire une carte de son environnement et l'utiliser simultanément pour en déduire sa position »*
> traduit de (DURRANT-WHYTE et BAILEY, 2006).

Il s'applique donc aux plates-formes mobiles qui possèdent au minimum un capteur extéroceptif qui leur permet de percevoir leur environnement. De nombreux capteurs sont couramment utilisés à cet effet : télémètres laser, sonars, caméras, etc. À ces derniers peuvent s'ajouter des capteurs proprioceptifs qui renseignent sur les déplacements effectués (accéléromètres, gyroscopes, roues codeuses, etc.). Le problème du SLAM se matérialise alors par la boucle de fonctionnement suivante, illustrée sur la figure 2.1 :

— À un instant t, le robot perçoit son environnement à travers les observations z_t issues de ses capteurs.

— Le robot passe de l'état s_t à s_{t+1} via le déplacement contrôlé par le vecteur u_t, appliqué à l'instant t.

— Le système estime son nouvel état et le corrige grâce aux nouvelles observations z_{t+1} des amers déjà perçus, en supposant ces derniers immobiles.

À chaque itération, le robot cherche ainsi à estimer sa pose par rapport à l'ensemble des amers déjà observés qui constituent la carte.

Bien que ce problème soit correctement posé et étudié depuis des années, il ne serait pas honnête d'affirmer qu'aujourd'hui le SLAM est un problème résolu. En effet, le problème est vaste puisqu'il dépend aussi bien de la plate-forme considérée (qualité de ses capteurs, mouvements autorisés, etc.) que de l'environnement d'utilisation (dynamique, sans repère évident, etc.). Ainsi, les recherches dans ce domaine proposent des solutions spécifiquement adaptées à un type de fonctionnement, une plate-forme et un environnement particuliers. Dans la section qui suit, nous présentons les méthodes de SLAM et de localisation les plus communes dans la littérature.

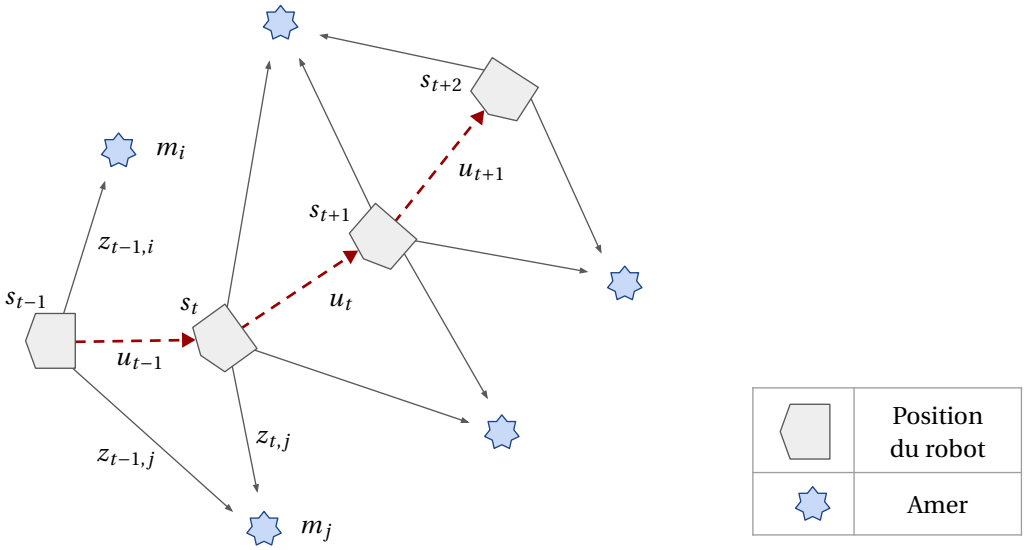


FIGURE 2.1: Problème du SLAM tel que présenté dans (DURRANT-WHYTE et BAILEY, 2006 ; MEI, 2007).

2.2 État-de-l'art et problématiques du SLAM et de la localisation

Cette partie cherche à présenter comment les problèmes de localisation autonome sont abordés dans la littérature. Deux situations peuvent être identifiées. La première correspond au problème du SLAM. Le robot n'a pas de carte de l'environnement dans lequel il évolue et se localise en même temps qu'il cartographie ce dernier. La deuxième situation est celle où le robot possède une représentation de son environnement. Dans cette situation, estimer la pose du robot dans la carte correspond à un problème de localisation pure. Si cet état-de-l'art se structurera de sorte à bien présenter les enjeux des systèmes de SLAM, des aspects relevant de la localisation pure y seront également introduits, chacune de ces problématiques constituant différents cas d'utilisation de Pepper.

(THRUN et al., 2002) constitue un rapport assez complet des méthodes de SLAM. Dans celui-ci, Les auteurs identifient les grandes problématiques du SLAM, qu'ils regroupent sous cinq grands axes :

1. contrôler la taille du problème,
2. modéliser l'erreur des capteurs,
3. détecter les fermetures de boucles,
4. gérer les changements dans l'environnement,
5. naviguer dans la carte créée.

Cette thèse se concentrant sur les actions de localisation et cartographie, les problématiques de navigation telles que l'évitement d'obstacles ou la planification de trajectoires n'y seront pas abordées. Notre état-de-l'art se divisera donc sous quatre axes reprenant les quatre premiers points de la liste dressée ci-dessus.

2.2.1 Carte et formalismes

Les cartes sont des outils connus de tous. Cependant, la représentation qu'elles font du monde et les informations qu'elles donnent peuvent être variées. En reprenant l'exemple d'un système de navigation GPS, l'utilisateur obtient une position à la surface de la Terre, décrite par sa latitude, sa longitude et son altitude. Les systèmes de SLAM utilisent eux différentes représentations du monde selon les besoins induits par leur utilisation et les capteurs dont ils disposent. D'autres contraintes influent également sur le choix de la représentation comme l'utilisation temps-réel [1] , l'espace mémoire limité sur la plate-forme, la qualité des processeurs CPU et GPU, etc.

2.2.1.1 Représentations métriques

Historiquement, les premières approches de SLAM ont été effectuées à partir de capteurs fournissant une information métrique sur leur environnement, tels que des lasers, des sonars, ou des caméras 3D. Dans ce type de représentation la carte contient directement des informations de distance. La pose du robot estimée correspond alors à une position métrique dans un repère de référence.

Le type de repère choisi est bien souvent fonction du degré de liberté de la plate-forme mobile considérée. Par exemple, pour un robot mobile à roues se déplaçant dans un plan, l'utilisation d'un repère 2D est suffisante, alors que pour un drone, qui se déplace lui dans l'espace, un repère 3D est nécessaire. La localisation du robot s'exprime ensuite comme une pose par rapport au repère de référence qui se traduit par une position et une orientation.

Les approches nécessitant une grande précision ont souvent recours à cette représentation métrique. Une telle carte se construit à partir de capteurs métriques comme des télémètres et la carte créée est d'autant plus précise que la précision des capteurs utilisés pour la construire est grande. Cette technologie a cependant un coût. Aujourd'hui, un capteur comme le Lidar Velodyne VLP-16, fréquemment monté sur les voitures autonomes (figure 2.2), vaut 4 000$.

Tous les capteurs permettant une localisation métrique ne sont pas aussi coûteux. Les systèmes de navigation GPS que nous prenions en exemple précédemment sont des produits destinés à un public large et fournissent une localisation métrique. D'autres systèmes permettent d'obtenir une pose métrique à partir de capteurs peu chers. Mais à l'instar des systèmes de navigation qui utilisent les satellites du programme GPS, ces approches nécessitent d'équiper l'environnement avec des marqueurs artificiels comme des marqueurs visuels (BABINEC et al., 2014), ou des marqueurs émettant des signaux sous forme d'ondes (LEONARD et DURRANT-WHYTE, 1991 ; VERMEIREN et al., 2018). Si ces méthodes présentent de nombreux avantages, elles ne sont pas applicables en toutes circonstances puisqu'elles reposent sur l'utilisation de ces amers artificiels.

En robotique mobile, obtenir une position métrique présente un avantage dans de

1. Dans les problématiques de SLAM, les contraintes temporelles s'accommodent de certains dépassements qui ne conduisent pas à des situations critiques. L'expression temps-réel renverra ainsi au temps-réel *souple*, qui cherche à respecter un temps de cycle mais tolère des dépassements exceptionnels.

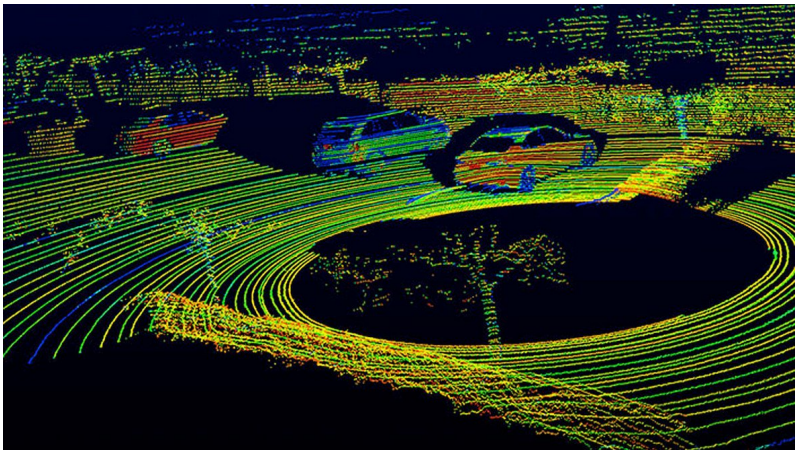


FIGURE 2.2: Exemple de scan lasers obtenu par un capteur Lidar monté sur une voiture autonome (SIMONITE, 2017).

nombreuses applications. Cela permet par exemple de planifier des chemins avec plus de précision et de cohérence. Cependant, dans un contexte de SLAM où l'environnement n'est pas connu, la représentation métrique peut nuire au fonctionnement temps-réel. En effet, les capteurs utilisés fournissent souvent des informations denses, comme des nuages de points, dont la manipulation ralentit le temps de traitement d'une requête. Pour pallier ce problème, un autre type de formalisme est couramment employé en robotique : le formalisme topologique.

2.2.1.2 Représentations topologiques

Dans une représentation topologique, l'environnement est décrit par des nœuds reliés entre eux par des transitions. Ce formalisme représente donc l'environnement comme une succession de lieux, à la manière dont on indiquerait le chemin à une personne dans la rue : « Pour aller à l'épicerie, passez d'abord devant la pharmacie, le feu de signalisation, puis le boucher. ».

Une bonne illustration de ce formalisme est la carte du métro. L'usager qui souhaite se rendre à son travail ne se soucie pas particulièrement du trajet effectué au sens métrique. Il surveille simplement que les stations s'enchaînent normalement jusqu'à la sienne.

En robotique ce type de représentation est assez populaire pour plusieurs raisons. Peu consommatrice en place mémoire ou temps de calcul, la représentation topologique permet également de résoudre facilement des problèmes comme ceux de la localisation globale ou du robot capturé (CUMMINS et NEWMAN, 2008) (plus clairement présentés en partie 2.2.3). Dans un contexte de SLAM, comme le robot effectue des tâches de navigation locale entre nœuds, l'incertitude sur sa pose topologique est moins grande que dans une représentation métrique où celle-ci croît avec les déplacements. Cependant, une telle représentation se révèle insuffisante pour des utilisations qui requièrent une grande précision comme rejoindre une station de recharge. De plus, perdre l'information métrique est dommageable dans les tâches de planification de trajectoires. Certaines ap-

proches préfèrent alors combiner les formalismes métriques et topologiques pour profiter des avantages de chacun.

2.2.1.3 Les approches topo-métriques

Les approches topo-métriques introduisent une information métrique dans une carte topologique. Cette information métrique peut se retrouver dans :

— les transitions entre nœuds, en conservant une information de pose relative entre eux par exemple,

— le nœud lui-même, avec des données comme la position métrique d'amers par rapport au nœud courant.

Cette représentation hybride est choisie dans de nombreux algorithmes de SLAM puisqu'une information métrique sur les transitions entre nœuds peut aisément être extraite grâce à l'odométrie du robot (CUMANI et al., 2004 ; BAZEILLE et FILLIAT, 2011 ; BADINO, HUBER et KANADE, 2012). Par exemple, un robot équipé d'une caméra monoculaire et de roues avec encodeurs magnétiques peut représenter l'environnement comme une succession d'images reliées entre elles par des déplacements métriques estimés à l'odométrie.

Les avantages de ce type de représentation sont multiples. Moins dense que l'approche métrique, ce formalisme génère des cartes plus légères en mémoire et plus facilement compatibles avec un fonctionnement temps-réel. De plus, il est facile de maintenir la carte créée étant donné que toute modification d'amers ou de position sera relative à un nœud. Pour finir, ce formalisme permet aussi d'adopter différentes stratégies de navigation, par exemple, une navigation métrique optimisée lorsque les nœuds sont suffisamment proches, ou une navigation topologique entre deux nœuds éloignés.

2.2.1.4 Cartes sémantiques

Un autre type de représentation de l'environnement a émergé ces dernières années et est qualifié de sémantique (KOSTAVELIS et GASTERATOS, 2015 ; SCHÖNBERGER et al., 2018). En milieu intérieur, par exemple dans un appartement, la localisation humaine définit un lieu par sa fonction. Nous identifions aisément la cuisine, la chambre, le salon, etc... Ces pièces et les façons de passer de l'une à l'autre constituent notre représentation mentale de l'environnement visité.

Si on retrouve des similarités avec le formalisme topologique, le formalisme sémantique identifie clairement les limites ou la fonction d'une pièce. Dans le formalisme topologique, un nœud ne correspond pas forcément à une pièce et aucune donnée ne permet de diviser l'espace topologique en différents espaces clairement identifiés. Les nœuds topologiques sont créés suivant plusieurs critères comme les limitations des capteurs utilisés, les zones d'intérêts, les poses de départ et d'arrivée, etc.

Le choix de l'approche sémantique présente un fort intérêt dans les interactions du robot. En effet, savoir qu'il se trouve dans une cuisine permet de lancer un comportement adapté comme la suggestion de différentes recettes de cuisine. Les interfaces de

navigation se retrouvent également simplifiées car l'utilisateur peut alors donner des consignes humaines : « va dans le couloir » (ZENDER et al., 2008). Comme ce type de représentation dépend fortement des performances de reconnaissance et classification d'un système, son utilisation s'est popularisée ces dernières années avec l'essor des techniques d'apprentissage profond (MCCORMAC et al., 2017).

2.2.2 Méthodes d'estimation

L'estimation de la position des amers et du robot est considérée comme le cœur du problème du SLAM. Dans le monde réel, les défis sont nombreux. Le bruit des capteurs, l'environnement changeant, l'usure de la plate-forme... tous ces problèmes mis bout-à-bout font la complexité du SLAM. Dans cette partie, nous tentons de dresser un portrait des différentes solutions proposées dans la littérature.

2.2.2.1 Approches probabilistes

De nombreux algorithmes ont été développés pour résoudre le problème du SLAM, mais dans un premier temps, la plupart ont abordé ce problème de façon probabiliste.

Les grilles d'occupation : Un des premiers raisonnements a par exemple conduit à échantillonner le monde sous la forme d'une grille. À chaque case de cette grille est associée une probabilité d'occupation (MORAVEC, 1988 ; ELFES, 1989). On parle alors d'approches par grille d'occupation.

Ces approches se basent en général sur l'odométrie du robot et la modélisation du bruit de ses capteurs. Différents types de capteurs peuvent être utilisés comme des capteurs de distance lasers (ADARVE et al., 2012) ou des caméras stéréo (PERROLLAZ et al., 2010). Les grilles d'occupation présentent l'avantage de bien gérer les signaux bruités, mais restent souvent très dépendantes de la qualité de l'odométrie. Cependant, d'autres méthodes ont également cherché à résoudre le problème du SLAM par filtrage.

L'EKF-SLAM : La boucle de fonctionnement présentée en 2.1.2 s'adapte particulièrement bien au filtre de Kalman. Visant à estimer les paramètres d'un système évoluant dans le temps, le filtre de Kalman repose sur trois hypothèses : un modèle d'évolution linéaire du système, une relation linéaire entre l'état et les mesures, et des mesures soumises à un bruit blanc gaussien centré de covariance connue.

Le filtre de Kalman présente l'inconvénient de fonctionner à partir d'un modèle linéaire. Pour cette raison, son extension est préférée dans les solutions de SLAM : le filtre de Kalman étendu (EKF), plus compatible avec les non-linéarités des mesures et modèles de déplacement usuels des robots. Ainsi, les premières approches cherchant à estimer conjointement la pose du robot et des amers dans l'environnement ont été réalisées via un filtre de Kalman étendu (SMITH et CHEESEMAN, 1986 ; SMITH, SELF et CHEESEMAN, 1990). Relativement simple à intégrer et fournissant des performances intéressantes sur des systèmes embarqués, l'EKF-SLAM est communément utilisé dans la littérature de-

puis (CHATILA et MOUTARLIER, 1989 ; CSORBA, 1998 ; CASTELLANOS et al., 1999 ; NEWMAN, 1999 ; DURRANT-WHYTE et al., 2003).

L'emploi du filtre de Kalman étendu montre cependant des limitations (MEI, 2007). Comme ce dernier utilise une approximation linéaire, l'erreur est souvent sous-estimée, ce qui peut rendre le filtre inconsistant (BAILEY et al., 2006). L'EKF-SLAM est également incompatible avec de grandes cartes (DISSANAYAKE et al., 2001 ; JULIER et UHLMANN, 2001). En effet, puisque chaque nouvel amer est créé à partir d'une observation du robot, son erreur de position est directement reliée à l'erreur de pose du robot. Or, comme la pose du robot est déterminée à partir des amers déjà présents dans la carte, son erreur dépend directement de celle des amers utilisés. Ainsi, tous les amers de la carte sont corrélés. La complexité algorithmique de l'EKF-SLAM augmente donc en $\mathcal{O}(M^2)$ où M correspond au nombre d'amers dans la carte. Lorsque M devient trop important, le filtre ne peut plus garantir un fonctionnement en temps-réel.

Plusieurs approches ont cherché à diminuer cette complexité comme le filtre de Kalman étendu compressé (GUIVANT et NEBOT, 2001). Son principe consiste à restreindre ses calculs à un espace local. L'utilisation de filtrage pour résoudre les problèmes du SLAM et de la localisation ne s'est cependant pas limité au filtre de Kalman. Certaines solutions se concentrent sur l'usage d'un filtrage particulaire (DOUCET, GODSILL et ANDRIEU, 2000 ; JOLY, PEYRET et al., 2008).

Le filtrage particulaire : L'emploi d'un filtre à particules classique passant par l'échantillonnage des poses du robot et de celle des amers n'est pas compatible avec la complexité du problème du SLAM. Le nombre important de particules pour obtenir une représentation convenable résulterait en une complexité algorithmique critique pour des usages temps-réel. Pour pallier ce problème, l'algorithme FastSLAM présenté dans (MONTEMERLO et al., 2002), propose l'emploi d'un filtrage particulaire Rao-Blackwellisé, proposé dans (MURPHY, 2000). Comme les positions des amers observés dépendent des observations du robot, l'échantillonnage peut n'être réalisé que sur la pose du robot. De cette manière, FastSLAM améliore sa complexité en $\mathcal{O}(log(M))$.

Le filtrage particulaire a été utilisé dans de nombreux travaux. Dans (HAHNEL et al., 2003), les auteurs combinent FastSLAM avec des associations de scans lasers utilisées pour corriger l'odométrie. Cette démarche permet d'éviter le problème de dégénérescence du filtre tout en diminuant la nécessité de ré-échantillonner les particules. L'algorithme peut alors générer de grandes cartes en temps-réel en maintenant 100 particules. L'approche de (STACHNISS, HAHNEL et BURGARD, 2004) applique l'algorithme FastSLAM aux grilles d'occupation et cherche à fermer les boucles. Le filtre à particules de FastSLAM est également employé dans des algorithmes fonctionnant à partir de capteurs visuels métriques comme une caméra stéréo (BARFOOT, 2005) ou une caméra de profondeur (HARTMANN et al., 2012).

Les travaux de (GRISETTI, STACHNISS et BURGARD, 2005) proposent de coupler l'information de mouvement du robot à l'observation la plus récente pour déterminer une distribution plus précise sur la pose courante. Dans (MONTEMERLO et THRUN, 2007), les auteurs du FastSLAM reprennent cette idée et proposent une nouvelle version de leur

algorithme sous le nom de FastSLAM 2.0. L'algorithme FastSLAM original échantillonne les poses des particules suivant la commande de mouvement appliquée au système, et la mesure n'est incorporée qu'au moment du ré-échantillonnage. Ainsi, lorsque le bruit sur le mouvement du robot est important et celui sur la mesure faible, cette approche se révèle inefficace. Pour éviter cet écueil, FastSLAM 2.0 échantillonne les poses en prenant en compte à la fois le mouvement et l'observation.

* * *

Les roboticiens ont ainsi résolu le problème du SLAM via des outils probabilistes qui reposent souvent sur l'emploi de filtres. Une autre grande famille d'algorithmes cherche à résoudre le problème du SLAM en utilisant des outils géométriques. D'abord perfectionnés dans le domaine de la vision par ordinateur, ces algorithmes sont aujourd'hui de plus en plus utilisés sur des plate-formes robotiques embarquées.

2.2.2.2 Approches visuelles géométriques

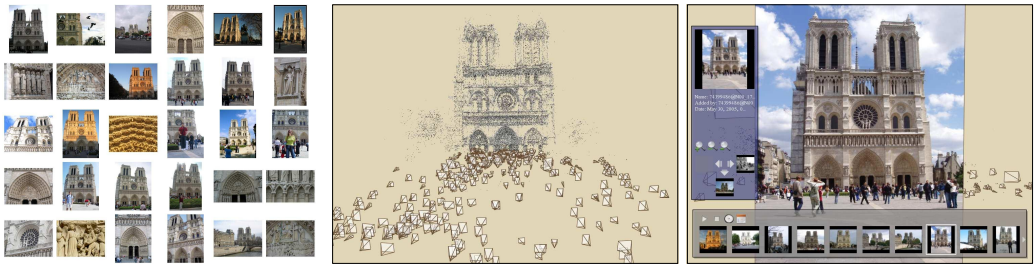
Si le problème du SLAM est très étudié en robotique à partir de capteurs extéroceptifs variés, la communauté de la vision par ordinateur s'est penchée sur cette problématique en utilisant spécifiquement des capteurs visuels. Lorsqu'en robotique les premières méthodes de SLAM se sont basées sur l'utilisation d'outils probabilistes, l'approche géométrique a été plus utilisée en vision par ordinateur.

Structure-from-motion **:** Le terme Structure-from-Motion (SFM) est beaucoup plus utilisé par la communauté de la vision par ordinateur pour désigner le même problème que celui du SLAM. Le problème du SFM revient à résoudre un problème de SLAM à partir d'images. Son principe est de déterminer la pose des caméras ayant acquis ces images. Cette pose s'exprime sous forme métrique dans un repère (coordonnées 3D du centre optique et orientation de la caméra).

Un exemple issu du domaine du SFM est l'application *photo tourism* (SNAVELY, SEITZ et SZELISKI, 2006). Ce logiciel propose d'explorer une collection d'images, par exemple de photographies touristiques d'un monument, en fonction de l'endroit d'où elles ont été prises (figure 2.3). Pour cela, l'algorithme crée une structure géométrique à partir des observations partagées entre ces images : cette structure prend le rôle de la carte. Les poses des images sont déterminées par rapport à elle.

La principale différence entre les problèmes de SFM et de SLAM réside en la disponibilité des données d'entrée. Dans une approche de SLAM, les données d'entrées sont acquises au fur et à mesure que l'environnement est découvert. Dans *photo tourism*, l'algorithme possède directement en entrée toutes les images de l'environnement considéré.

Les approches classiques de SFM reposent sur des outils de vision par ordinateur. Elles se divisent généralement en plusieurs étapes :



(a) (b) (c)

FIGURE 2.3: Dans l'application *photo tourism* (SNAVELY, SEITZ et SZELISKI, 2006), une collection d'images (a) permet de générer un modèle 3D de l'environnement (b) par rapport auquel chaque image est localisée (c).

1. Une étape de détection de primitives visuelles dans chaque image. Les primitives les plus couramment utilisées sont les points remarquables de l'image, aussi appelés points d'intérêt (HARRIS et STEPHENS, 1988 ; LOWE, 1999 ; BAY, TUYTELAARS et VAN GOOL, 2006 ; ROSTEN et DRUMMOND, 2006 ; RUBLEE et al., 2011).
2. Une étape d'association de ces primitives. Grâce à leurs descripteurs, dont les plus connus pour les points d'intérêt sont les SIFT (LOWE, 1999) et SURF (BAY, TUYTELAARS et VAN GOOL, 2006), ou bien les descripteurs binaires BRIEF (CALONDER et al., 2010) et ORB (RUBLEE et al., 2011).
3. Une étape de calculs géométriques de reconstruction 3D. Ces derniers reposent majoritairement sur les principes de la géométrie épipolaire, dont une très bonne présentation est disponible dans (HARTLEY et ZISSERMAN, 2003).

La troisième étape listée ci-dessus est souvent ramenée à un problème d'optimisation. Aujourd'hui, la méthode la plus populaire pour résoudre ce problème d'optimisation est *l'ajustement de faisceaux*. Considérée comme la méthode de reconstruction la plus précise (TRIGGS et al., 1999 ; DEANS et HEBERT, 2001 ; STRASDAT, MONTIEL et DAVISON, 2012), elle optimise à la fois la pose des images considérées, et celles des points 3D reconstruits. Son fonctionnement repose sur les observations partagées entre les différentes images et les lois de projection reliant un point réel 3D à une coordonnée pixellique sur le plan image. Le problème d'optimisation peut alors se formuler via un graphe où des arêtes directes existent entre amers 3D et images qui les observent, comme présenté en Figure 2.4. Cette optimisation a cependant un coût puisqu'elle induit une grande complexité algorithmique. Ainsi, elle a longtemps été incompatible avec des usages temps-réel.

L'odométrie visuelle : Pour des applications temps-réel, certaines méthodes préfèrent se passer de l'ajustement de faisceaux. Dans (NISTÉR, NARODITSKY et BERGEN, 2004), les auteurs présentent un système d'*odométrie visuelle* qui ne nécessite aucune connaissance *a priori* sur le mouvement de la caméra ou la scène observée. L'objectif de leur démarche est d'estimer la transformation géométrique entre deux images consécutives d'un flux vidéo. Ces estimations successives permettent alors de reconstruire la trajec-

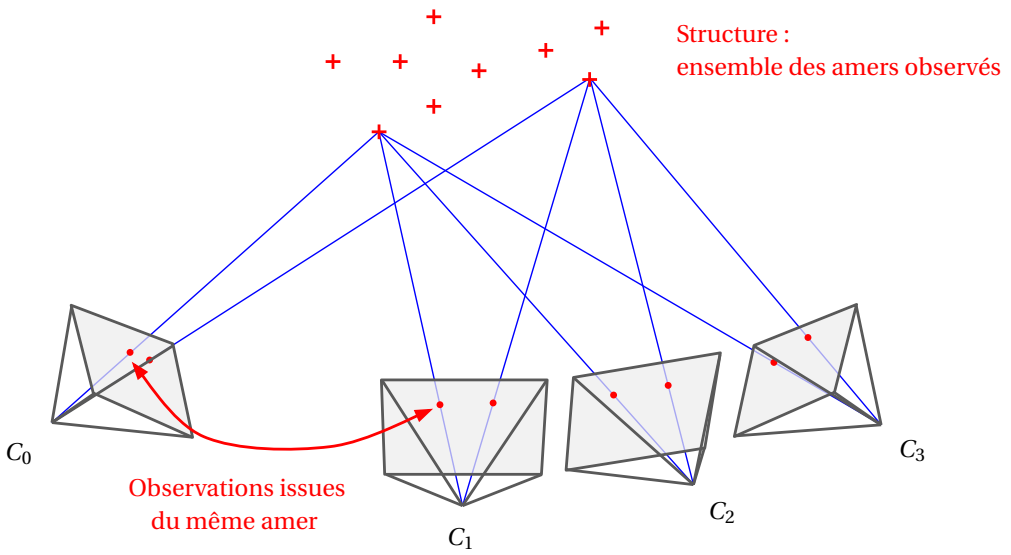


FIGURE 2.4: Approche typique des algorithmes de vision par ordinateur : l'ajustement de faisceaux. La pose des images issues des caméras C_i est optimisée conjointement avec la position des amers visuels.

toire réalisée par la caméra. Cependant, l'odométrie visuelle est basée sur le suivi de points d'intérêt. Elle n'utilise donc pas de carte à proprement parler. Ainsi, les points qui sortent du champ de vision de la caméra ne sont jamais reconnus si ils y reviennent. Un phénomène de marche aléatoire est donc inévitable, ce qui se traduit par une dérive de la pose estimée, causée par l'accumulation d'erreurs.

Les approches directes : D'autres méthodes font le choix de ne pas avoir recours à des primitives visuelles. Ces approches sont couramment qualifiées de *directes* (SILVEIRA, MALIS et RIVES, 2008 ; ENGEL, SCHÖPS et CREMERS, 2014 ; FORSTER, PIZZOLI et SCARAMUZZA, 2014) et reposent directement sur l'intensité des pixels des images. Fonctionnant sans points d'intérêt, ces approches s'affranchissent des étapes de détection et description. Elles évitent ainsi les erreurs issues de mauvaises détection ou description.

Cependant, ces méthodes supportent difficilement les variations photogrammétriques comme les changements d'illumination. De plus, ces dernières sont efficaces sur l'estimation de déplacements locaux, mais sont tenues en échec lors de plus grands déplacements.

Images-clés et optimisation locale : Parallèlement aux approches directes, certains travaux se sont employés à faire fonctionner l'ajustement de faisceaux dans des applications temps-réel. Leur idée est de simplifier l'optimisation par ajustement de faisceaux en considérant qu'il n'est pas nécessaire d'inclure toutes les images d'un flux vidéo dans le problème d'optimisation. Le tri sur les images à utiliser est réalisé de deux façons :

1. Par sélection d'images-clés : toutes les images ne sont pas ajoutées dans la carte

mais seulement certaines, appelées images-clés. L'idée derrière cette sélection est de maximiser l'information contenue dans la carte avec un nombre minimal d'images-clés. Le nombre de variables à optimiser est ainsi réduit. Pour cela, les critères d'ajout sont variés et peuvent se baser sur des informations spatiales, temporelles, ou sur le nombre de nouveaux amers observés par exemple.

2. Par optimisation locale itérative : l'idée derrière cette approche est que l'ajout d'une image dans le graphe ne va perturber les poses que de ses proches voisins. Ainsi, il n'est pas nécessaire de réaliser une optimisation coûteuse sur l'ensemble du graphe, mais plutôt sur le sous-ensemble directement impacté par l'ajout des nouvelles données.

Les premiers algorithmes fonctionnant en temps-réel reprenant ces principes ont été introduits par (MOURAGNON et al., 2006) et (KLEIN et MURRAY, 2007). Dans leurs approches l'optimisation locale est définie par des critères de co-observabilité. Pour satisfaire les contraintes d'une application temps-réel, dans (KLEIN et MURRAY, 2007) l'algorithme se divise en deux routines tournant en parallèle. Les tâches de suivi des amers et de localisation du système sont réalisées dans une première routine. La seconde routine est dédiée aux tâches d'optimisation de la carte.

Avec de telles approches, le nombre d'amers visuels est souvent bien plus élevé que le nombre d'images-clés. Certaines méthodes préfèrent alors, dans le cas d'optimisations globales coûteuses, concentrer leurs efforts sur les transitions entre images-clés. Elles utilisent un *squelette* (KONOLIGE et AGRAWAL, 2008 ; KONOLIGE et al., 2010). Malgré tout, le rôle des amers visuels ne doit pas être sous-estimé. Les travaux présentés dans (STRASDAT, MONTIEL et DAVISON, 2012) expliquent que l'augmentation du nombre d'amers visuels a plus d'incidence sur la précision du système que l'augmentation du nombre d'images-clés. Aujourd'hui, les méthodes de SLAM visuel les plus populaires basées sur des outils géométriques combinent ces méthodes d'optimisation locale et de squelette, comme dans ORB-SLAM (MUR-ARTAL, MONTIEL et TARDOS, 2015).

SLAM contraint : Un défaut bien connu vient cependant des approches de SFM monoculaires. Le principe de création d'une image étant la projection d'un espace 3D sur un plan image 2D, une dimension est perdue. Si on peut associer un pixel d'une image à une droite réelle, aucune information n'est disponible sur la profondeur du point réel projeté dans l'image. Les approches de SLAM monoculaire génèrent ainsi des cartes correctes à un facteur d'échelle près. Pour contourner ce problème, plusieurs approches proposent d'ajouter une contrainte dans la carte. Dans le cas de la vision monoculaire, cette contrainte est souvent implicitement forcée par un modèle de mouvement. D'autres approches utilisent un capteur supplémentaire qui permet d'introduire une information métrique dans le problème, comme une centrale inertielle (KNEIP, CHLI et SIEGWART, 2011 ; FLEPS et al., 2011 ; LOVEGROVE, PATRON-PEREZ et SIBLEY, 2013 ; LEUTENEGGER et al., 2015), l'odométrie du robot (CUMANI et al., 2004 ; KAESS et DELLAERT, 2006 ; BAZEILLE et FILLIAT, 2011 ; EUDES, 2011 ; BADINO, HUBER et KANADE, 2012), ou une deuxième caméra (MEI et al., 2009 ; PIRE et al., 2015 ; ENGEL, STÜCKLER et CREMERS,

2015 ; MUR-ARTAL et TARDÓS, 2017a), transformant alors le système de SLAM monoculaire en SLAM stéréo. Une autre façon de contraindre le système est de connaître au préalable les dimensions d'éléments observés. Ces éléments peuvent être des objets (TAMAAZOUSTI et al., 2011) voire même une partie de la structure déjà connue (LARNAOUT et al., 2013 ; RAMADASAN, CHEVALDONNÉ et CHATEAU, 2014). Ces dernières approches peuvent s'apparenter à ce que nous percevons lorsqu'on regarde une photo d'un appartement inconnu. Sans avoir jamais mis les pieds dans la pièce, on imagine bien ses dimensions grâce à des éléments de référence : une table, une porte, une chaise...

2.2.2.3 Apprentissage et bio-mimétisme

Nous avons vu jusqu'ici que les approches probabilistes et géométriques ont été très étudiées en robotique et vision par ordinateur pour résoudre les problèmes de SLAM. Si ces techniques sont prédominantes, elles ne sont pas les seules. Certaines méthodes ont en effet parié sur l'émergence des outils d'apprentissage profonds, mais aussi sur des représentations plus proches du vivant.

Reproduire le vivant : Tout comme la robotique s'est souvent inspirée du vivant, certains travaux de SLAM se sont basés sur les mécanismes biologiques de la localisation et cartographie mis en place chez différents animaux. Dans (COLLETT, 2010), l'auteur s'intéresse à la navigation des fourmis du désert. Il met en évidence leur utilisation de repères visuels, et s'il ne propose pas directement d'algorithmes de SLAM, il met en évidence l'intérêt de démarches similaires pour des systèmes robotisés.

L'algorithme bio-inspiré de SLAM le plus utilisé a été présenté dans (MILFORD, WYETH et PRASSER, 2004) sous le nom de RatSLAM. Comme son nom l'indique, cet algorithme s'inspire des modèles de cartographie retrouvés dans l'hippocampe des rats. RatSLAM repose sur l'emploi de différentes cellules qui vont être excitées par différents stimuli. La carte créée par l'algorithme peut être considérée comme un ensemble de cellules qui vont être excitées selon la pose et les observations du robot. Cette carte possède donc à la fois des propriétés de représentations métriques, comme les grilles d'occupation, mais aussi topologiques. L'algorithme du RatSLAM cartographie en continu son environnement. Si cette stratégie permet d'obtenir des représentations cohérentes de l'environnement exploré, elle conduit a la croissance inévitable de la carte dans des environnements dynamiques. Cependant, les expériences de (MILFORD et WYETH, 2008) démontrent que son utilisation est envisageable à partir d'une caméra monoculaire sur des trajets de plusieurs dizaines de kilomètres.

En prenant pour exemple les méthodes de localisation du vivant, l'importance de la perception est remarquable. Or, ces dernières années, une révolution des tâches de perception automatique s'est opérée grâce au développement des méthodes d'apprentissage profond par réseaux de neurones convolutifs. Il n'est donc pas surprenant d'avoir vu leur utilisation étudiée dans le domaine de la localisation et cartographie simultanées.

Réseau de neurones et SLAM : Si les réseaux de neurones ont été développés dès les années 80, leur utilisation a explosé ces dernières années. Cet engouement s'explique par les performances de ces systèmes, la disponibilité croissante de bases d'apprentissage annotées et l'amélioration du matériel. L'emploi de GPU dans les systèmes embarqués tend en effet à se démocratiser car il permet de réaliser rapidement des opérations de convolution. Les réseaux de neurones convolutifs ont en effet révolutionné le domaine de la vision par ordinateur dans des tâches comme la reconnaissance d'objets (classification et segmentation). Cette technologie a alors été logiquement employé en robotique, notamment dans les problèmes de localisation.

Les travaux de (KENDALL, GRIMES et CIPOLLA, 2015) introduisent PoseNet, une approche par réseau de neurones convolutif permettant de déterminer la pose métrique d'une caméra en 6 dimensions dans un environnement connu. Leur méthode consiste à entraîner un réseau par des techniques de *transfer learning*, à partir d'images RGB dont les positions sont annotées. Le robot embarque le réseau, à travers duquel les images requêtes vont être traitées. Chaque requête se voit associée en sortie du réseau une pose 6D. L'avantage de ce genre d'approche est de ne nécessiter que peu de place mémoire puisque seuls les poids du réseau sont nécessaires à son fonctionnement. Cependant, elle ne correspond pas à proprement parler à un problème de SLAM puisque la pose des images de l'environnement doit être connue lors de l'apprentissage des poids du réseau.

L'apprentissage profond a également été utilisé dans des problématiques d'odométrie visuelle. L'idée est cette fois de définir la transformation entre deux images successives d'un flux vidéo grâce aux réseaux de neurones (KONDA et MEMISEVIC, 2015 ; COSTANTE et al., 2016 ; MELEKHOV et al., 2017 ; WANG et al., 2017).

2.2.3 Fermeture de boucle

En situation réelle, toutes les méthodes de SLAM présentées jusqu'ici ont tendance à dériver, et ce malgré la précision des capteurs embarqués par le robot. Cela s'explique par l'accumulation d'erreurs au fil du temps correspondant à un phénomène de marche aléatoire. Il arrive que le robot repasse par un endroit déjà visité mais que la dérive de sa trajectoire ne lui permette pas d'estimer correctement sa pose. Idéalement, le robot devrait alors être capable de :

1. détecter qu'il a déjà visité ce lieu,
2. corriger la trajectoire qu'il a estimée jusqu'ici.

Ces deux actions constituent le problème de la fermeture de boucle, et sont détaillées respectivement en 2.2.3.1 et 2.2.3.2.

2.2.3.1 Reconnaissance de lieux

Le problème de la détection de fermeture de boucle peut se retrouver sous différentes formes et appellations dans la littérature. Résoudre ce problème revient à donner au système la capacité de se localiser dans une carte sans *a priori* valable sur sa pose courante. Ce problème général peut donc correspondre à deux problèmes plus particuliers,

celui du « *robot capturé* » et de la « *localisation globale* ». Le problème de la localisation globale arrive à l'allumage du robot : comme le robot « ne sait pas » s'il a été déplacé pendant qu'il était éteint, il n'a aucune information sur sa pose courante. Il doit alors se *relocaliser* dans sa carte. Le problème du robot capturé diffère en un point de celui de localisation globale : le robot possède cette fois une information sur sa pose dans la carte mais cette dernière est fausse. Le problème du robot capturé peut alors apparaître comme plus compliqué puisqu'il demande de prendre conscience de son erreur dans un premier temps. Cependant, les méthodes de localisation globale peuvent facilement être adaptées pour résoudre le problème du robot capturé (ROHRIG et KIINEMUND, 2007).

Plusieurs méthodes existent pour reconnaître un lieu déjà visité. La plupart ont pour point commun de préférer l'emploi d'un formalisme topologique ou topo-métrique. L'avantage de ce type de formalisme est que la recherche s'effectue dans un monde discrétisé. Chaque nœud est testé comme une position potentielle. Par exemple, pour une carte où à chaque nœud correspond une image, la tâche de relocalisation revient à trouver l'image de référence la plus similaire à l'image requête. Dans la littérature, l'approche par *apparence visuelle* est souvent utilisée pour de telles problématiques.

L'apparence visuelle : La relocalisation visuelle fait face à de nombreuses difficultés. La comparaison d'une image requête aux images références de la carte se révèle vite être une opération coûteuse en temps de calculs sans adaptation spécifique. De plus, comparer des images de manière brute n'a pas de sens puisque deux images issues du même lieu possèdent des variations multiples en fonction de l'angle de prise de vue, des variations lumineuses ou du mobilier déplacé. L'apparence visuelle est une solution consistant à décrire une image de façon compacte et en adoptant un formalisme permettant la gestion de ces variations (Figure 2.5). (LOWRY et al., 2016) constitue un très bon état-de-l'art des différentes approches dans ce domaine.

FIGURE 2.5: Reconnaissance de lieux par apparence visuelle. L'apparence visuelle ne prend pas en compte des contraintes géométriques fortes telles que l'angle de prise de vue ou la position de primitives visuelles sur l'image, mais cherche à reconnaître un lieu en prenant en compte l'apparence de la scène observée.

Plus de détails sur ces méthodes sont donnés dans la partie 3.1 du chapitre suivant. Cependant, il convient ici de présenter un problème récurrent que cherchent à éviter les méthodes de détection de fermeture de boucle : *l'ambiguïté perceptuelle.*

L'ambiguïté perceptuelle : Dans les solutions de localisation, l'ambiguïté perceptuelle correspond à la confusion du système entre plusieurs lieux distincts. Ce phénomène apparaît lorsque différents lieux de l'environnement sont identiques du point de vue de l'information extraite des capteurs. La Figure 2.6 illustre ce problème à partir de capteurs visuels.

FIGURE 2.6: Problème de l'ambiguïté perceptuelle. Dans cet exemple, les deux images semblent avoir été acquises dans le même lieu. En réalité, chacune a été prise à deux étages différents d'un bâtiment. Dans des environnements visuellement très redondants comme celui-ci, les approches basées sur l'apparence visuelle souffrent de cette ambiguïté perceptuelle.

L'ambiguïté perceptuelle est un problème bien connu dans le domaine de la localisation et cartographie simultanées. En effet, confondre deux lieux pendant la phase de cartographie conduit à une forte dégradation de la carte, voire à des erreurs irrécupérables. Pour cette raison, l'ambiguïté perceptuelle a souvent conditionné le choix des capteurs utilisés dans les problèmes de fermeture de boucle, ainsi que de l'information extraite de ces derniers. Certains choix, qui peuvent s'avérer pertinents dans un environnement peuvent l'être beaucoup moins dans d'autres. Par exemple, dans un appartement où chaque pièce possède une couleur prédominante, une reconnaissance de lieux basée sur la comparaison d'histogrammes couleurs comme dans (ULRICH et NOURBAKHSH, 2000 ; BLAER et ALLEN, 2002) est une bonne idée. Mais dans un intérieur où chaque pièce possède des murs blancs, cette approche révèle rapidement ses limites.

L'ambiguïté perceptuelle est un phénomène particulièrement problématique dans les environnements redondants. Les bureaux d'entreprises, hôpitaux, hangars ou parties communes de grands immeubles en sont de bons exemples. Ainsi, les méthodes de SLAM cherchent à ne pas tomber dans ce piège. Pour cela, différentes approches existent. Dans le domaine de la vision monoculaire par exemple, des outils probabilistes fréquentistes (SALTON et BUCKLEY, 1988) ou bayésiens (CUMMINS et NEWMAN, 2008) ont été développés. Une autre approche présentée dans (MILFORD et WYETH, 2012) ne cherche plus à associer des images individuellement mais plutôt des séquences d'images. Elle ajoute alors un facteur spatio-temporel dans la détection de fermeture de boucle.

Empreintes Wi-Fi : Certaines méthodes de SLAM préfèrent miser sur des amers dont l'unicité dans l'environnement est assurée, et ainsi éviter les risques d'ambiguïté perceptuelle. La plupart équipent l'environnement avec des amers artificiels reconnaissables et uniques. Nous pouvons citer l'exemple des marqueurs ArUco en vision (BABINEC et al., 2014), ou du système Pozyx basé sur des ondes radio Ultra Large Bande (ULB) capables de pénétrer les murs (VERMEIREN et al., 2018). Équiper l'environnement n'est pas toujours possible et peut constituer une étape complexe pour des utilisateurs non-experts. Pour éviter à l'utilisateur final une phase active d'installation d'amers, des méthodes de localisation se sont basées sur l'utilisation d'amers uniques et « naturellement » présents en milieu urbain : les bornes Wi-Fi.

La présence de nombreuses bornes Wi-Fi dans les milieux urbains a permis l'exploitation de leur signal dans des problématiques de localisation. Depuis une dizaine d'années, plusieurs travaux se sont appliqués à la résolution du problème de relocalisation à partir de données Wi-Fi. L'idée derrière ces travaux est de profiter de l'unicité du signal émis par une borne d'accès Wi-Fi, ainsi que de la propagation de ces signaux dans l'espace. Sur un système embarqué équipé d'une carte Wi-Fi, il est facile de collecter à une position spatiale une *signature Wi-Fi* qui correspond à la liste des signaux Wi-Fi perçus. Dans cette signature, chaque signal Wi-Fi est décrit par son intensité (noté RSSI pour *Received Signal Strength Indication*) et l'adresse MAC de sa borne émettrice. Historiquement, deux familles d'algorithmes de localisation basée sur le Wi-Fi sont identifiables (WEI et BELL, 2011) :

1. les méthodes de trilatération,
2. les méthodes basées empreintes.

Le but des premières est d'arriver à associer à chaque signal une distance entre les antennes réceptrice et émettrice. Ainsi, comme avec le GPS, en connaissant la position de plusieurs bornes d'accès Wi-Fi et le signal de puissance qui leur est associé, il est possible de trilatérer sa position et de se localiser dans l'espace. On retrouve cependant ici le problème de la nécessité d'annoter la position des bornes Wi-Fi manuellement (Figure 2.7). De plus, la propagation des signaux Wi-Fi est perturbée dans les environnements intérieurs. Les murs contenant des métaux peuvent par exemple jouer le rôle d'une cage de Faraday. Pour obtenir une localisation précise, il devient alors nécessaire de modéliser finement l'environnement.

Les secondes méthodes basées sur l'empreinte cherchent à modéliser la répartition des signaux Wi-Fi dans l'environnement à l'aide d'une première phase d'exploration. Cette phase d'exploration correspond à un apprentissage. Elle permet de créer une représentation de la répartition des signaux Wi-Fi dans l'environnement. Dans (BISWAS et VELOSO, 2010), pendant l'exploration d'apprentissage, les intensités des signaux Wi-Fi sont relevées à différentes positions pendant une durée pré-définie puis moyennées. Chaque position est ainsi décrite par une signature de référence. Au moment de la localisation, le système acquiert une signature requête et la compare à toutes ses références. Différentes heuristiques peuvent permettre de chercher la signature la plus proche comme dans les travaux de (ROBERTS et PAHLAVAN, 2009) ou (BOONSRIWAI et

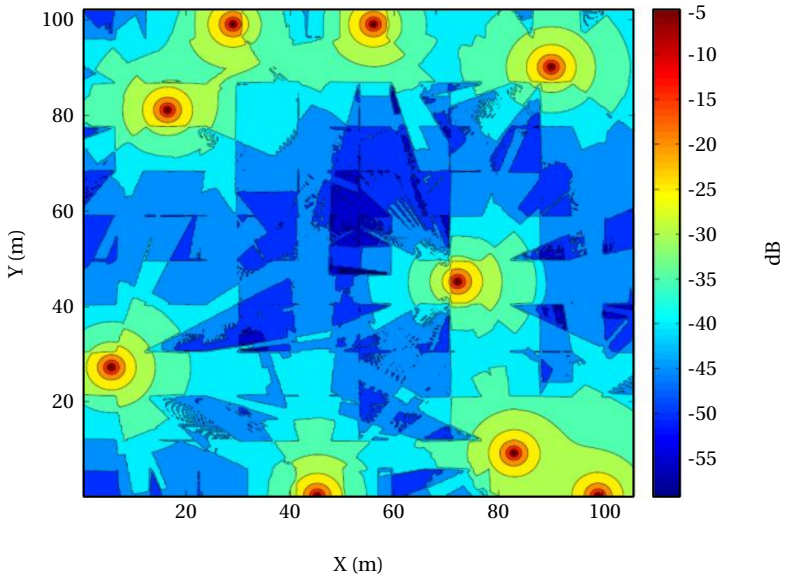


FIGURE 2.7: Modélisation de la propagation des signaux Wi-Fi, valeurs du RSSI simulées dans un environnement (FARKAS, HUSZÁK et GÓDOR, 2013). Les approches de trilatération nécessitent de modéliser finement l'environnement puisque la propagation des signaux Wi-Fi n'est pas uniforme suivant les obstacles qu'ils traversent. De plus, la position des points d'accès doit être connue.

APAVATJRUT, 2013). D'autres approches tentent une modélisation plus fine compatible avec un fonctionnement métrique. Dans (JIRKU, KUBELKA et REINSTEIN, 2016), un robot équipé de capteurs métriques performants mais coûteux crée une carte métrique en relevant les Wi-Fi perçus. La carte Wi-Fi créée est ensuite utilisée par des robots mobiles moins chers. À l'aide d'une modélisation gaussienne de la propagation des signaux Wi-Fi, les auteurs permettent à leur système de corriger sa localisation en 6 dimensions.

Ces méthodes présentent l'avantage de ne pas tomber dans le piège de l'ambiguïté perceptuelle. Cependant ces dernières manquent de précision puisque leurs erreurs sont comprises entre 5 et 10 m selon le matériel utilisé. Ces erreurs peuvent s'expliquer par les variations temporelles fortes des signaux Wi-Fi mais aussi par des perturbations externes comme la présence d'humains. En effet, les humains, étant composés majoritairement d'eau, absorbent une grande partie du signal Wi-Fi. Notons pour finir que dans un environnement avec très peu de bornes Wi-Fi, ces approches sont inenvisageables.

2.2.3.2 Correction de la carte

Une fois qu'une fermeture de boucle a été détectée, la dérive du robot peut être corrigée afin de retrouver une trajectoire plus cohérente. L'approche la plus employée est de considérer ce problème comme un problème d'optimisation. Cette optimisation est par exemple assimilée à une méthode de lissage dans (LU et MILIOS, 1997). Mais les méthodes d'optimisation les plus courantes aujourd'hui considèrent notre carte comme un graphe, dont les poses du robot (ou des capteurs) ainsi que les amers sont les sommets. Dans (THRUN et MONTEMERLO, 2006) et (GRISETTI et al., 2010) les auteurs introduisent le nom de *GraphSLAM* largement repris par la communauté robotique depuis.

Un problème d'optimisation revient à déterminer la solution optimale dans l'ensemble des solutions d'un problème. La plupart des méthodes de SLAM par optimisation font appel à des méthodes de moindres carrés non linéaires, utilisant le plus souvent l'algorithme de Levenberg-Marquart (MARQUARDT, 1963 ; LEVENBERG, 1944). Cet algorithme est généralement employé dans les solutions de SLAM puisqu'il présente l'avantage de bien gérer une mauvaise initialisation ou des erreurs sur les données d'entrée. Pour la majorité des usages SLAM, ce problème d'optimisation prend la forme suivante :

$$F(\mathrm{X}) = \sum_{(i,j)} e(\mathrm{x}_i, \mathrm{x}_j, \mathrm{z}_{ij})^T \, \Omega_{ij} \, e(\mathrm{x}_i, \mathrm{x}_j, \mathrm{z}_{ij}) \tag{2.1a}$$

$$\mathrm{X}^* = \underset{\mathrm{X}}{\operatorname{argmin}} \, F(\mathrm{X}) \tag{2.1b}$$

Les notations des équations (2.1a) et (2.1b) sont reprises de (KÜMMERLE et al., 2011). Elles correspondent à :

— $\mathrm{X} = (\mathrm{x}_1^T, \ldots, \mathrm{x}_n^T)$ le vecteur de paramètres,

— z_{ij} la moyenne de la contrainte reliant x_i et x_j,

— Ω_{ij} la matrice d'information associée à la contrainte reliant x_i et x_j,

— $e(\mathrm{x}_i, \mathrm{x}_j, \mathrm{z}_{ij})$ la fonction d'erreur mesurant à quel point les paramètres x_i et x_j respectent la contrainte z_{ij}.

Dans le cas d'une fermeture de boucle, la contrainte ajoutée dans le graphe est loin d'être respectée avant optimisation. Nous illustrons ce problème en figure 2.8 pour un robot dont les poses sont reliées par des transitions odométriques. L'optimisation va « diluer » cette erreur sur tout le graphe si chaque contrainte est associée à une matrice d'information similaire. Cette matrice d'information peut être vue comme la confiance de notre système sur une contrainte. Par exemple, dans le cas de contraintes odométriques pures, l'information diminue lorsque le déplacement grandit entre deux nœuds topométriques puisque le robot accumule plus d'erreurs.

Dans le domaine du SLAM, l'expression *back-end* fait souvent référence à cette étape d'optimisation de la carte. Puisque le problème du SLAM est un problème épars, certains projets *open-source* se sont employés à fournir des solveurs efficaces pour le résoudre. Les plus connus sont les bibliothèques g^{2}o (KÜMMERLE et al., 2011), GTSAM (DELLAERT, 2012) et CERES (AGARWAL et MIERLE, 2012). Grâce à l'amélioration du matériel embarqué et au développement de ces différents solveurs, la correction de la carte est de plus en plus réalisée en temps-réel dans les solutions de SLAM actuelles.

La capacité d'un système à fermer les boucles permet d'obtenir une carte spatialement cohérente. Cependant, une telle correction entraîne un risque de détérioration de la carte dans le cas où elle est déclenchée par une fausse fermeture de boucle. Ainsi certaines approches de la littérature cherchent à éviter ce phénomène en s'assurant de la

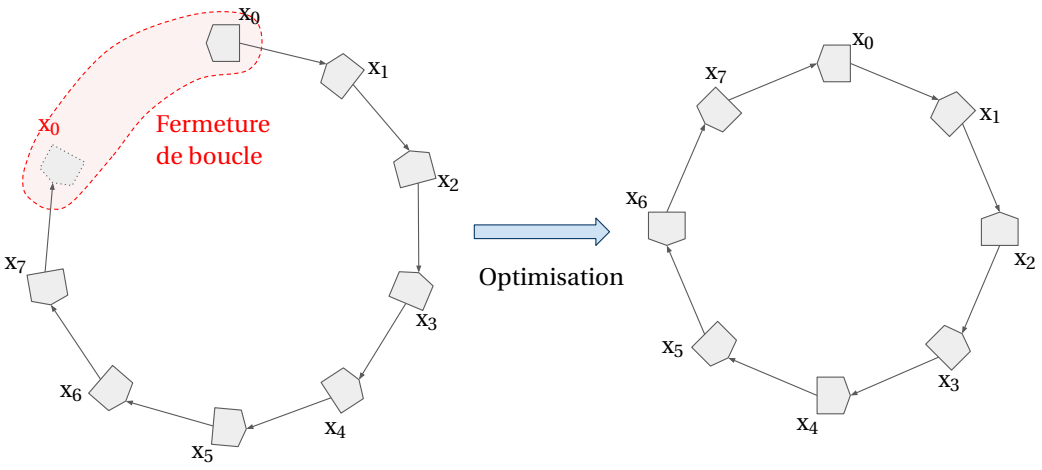


FIGURE 2.8: Correction de trajectoire après une fermeture de boucle pour un robot effectuant une trajectoire circulaire. Les poses successives du robot estimées à l'odométrie constituent les paramètres X du problème d'optimisation.

validité de la détection de fermeture de boucle, comme dans (MILFORD et WYETH, 2012), ou via l'utilisation d'un processus *back-end* capable de gérer de mauvaises contraintes en les commutant comme dans (SÜNDERHAUF et PROTZEL, 2012).

Mais la dérive d'un système de SLAM peut également s'observer d'un point de vue temporel. Ainsi, certains travaux se sont concentrés sur la gestion des variations temporelles pour des applications long-terme.

2.2.4 Localisation et cartographie long-terme

En pratique, la pérennité des cartes générées par les premières méthodes de SLAM n'est pas l'objectif principal. Bien souvent, la carte réalisée n'est utilisée que pendant le fonctionnement du robot. Dans ce genre d'expérience, le système testé est alors soumis à des limites à la fois spatiales et temporelles. Ces dernières années, la question d'une localisation et cartographie long-terme a été plus étudiée. Cette question est loin d'être triviale car elle entraîne nécessairement deux problèmes majeurs sur un système embarqué : (i) la saturation de l'espace mémoire disponible et (ii) l'augmentation du temps de calcul. Ces deux problèmes sont intimement liés puisque plus la quantité d'information stockée est grande, plus le temps de calcul augmente. Si la capacité de fermer des boucles aide le système à diminuer la taille de sa carte (en évitant de dupliquer les observations et lieux ajoutées dans celle-ci), elle ne solutionne pas entièrement le problème. En effet, comme l'illustre la Figure 2.9, les observations vont être soumises à des variations. Ces variations peuvent être chroniques comme la luminosité qui varie en fonction de l'heure, ou causées par des objets dynamiques (personnes en mouvement, mobilier déplacé, etc.). L'un des objectifs du SLAM long-terme est d'être robuste à ce type de variations.

Dans (DYMCZYK et al., 2016), les auteurs classifient les solutions de SLAM long-terme en trois grandes approches qui cherchent à :

FIGURE 2.9: Exemple de variations visuelles dans une problématique de SLAM long-terme.

1. réduire la taille de la carte,
2. gérer les changements dans l'environnement,
3. sélectionner les amers pertinents.

Il est possible de catégoriser ces trois approches respectivement par le choix d'un formalisme adapté, l'entretien *back-end* de la carte, et la sélection *front-end* d'amers pertinents. Nous reprenons ici cette catégorisation pour présenter un état-de-l'art des méthodes de SLAM long-terme.

2.2.4.1 Taille de la carte

L'utilisation d'une grande carte génère plusieurs problèmes. Une complication courante est la recherche dans la carte. En prenant pour exemple une tâche de détection de fermeture de boucle par apparence visuelle, l'image requête doit être comparée à toutes les images de référence contenues dans la carte. Si le nombre de références est élevé, cette solution devient incompatible avec un fonctionnement temps-réel. Des outils comme des index inversés permettent une recherche accélérée parmi les références comme dans (CUMMINS et NEWMAN, 2011).

Certaines approches remarquent que les changements de l'environnement sont souvent chroniques. Elles proposent alors de charger des sous-cartes en fonction des observations courantes. Par exemple, en utilisant un horodatage comme dans (CARLEVARIS-BIANCO et EUSTICE, 2012) ou (KRAJNÍK et al., 2014). La carte chargée est alors fonction de l'heure à laquelle le robot commence à naviguer dans son environnement. Ce type de méthode permet non seulement de diminuer la taille de la carte à charger et à utiliser, mais également de limiter le risque d'associations aberrantes en ne considérant que les références les plus probables.

Néanmoins, cette démarche nécessite de pouvoir relier entre elles des observations qui décrivent le même lieu malgré leurs différences. Si certains travaux mélangent différents algorithmes pour tirer les avantages de chacun (GLOVER et al., 2010), on retrouve principalement deux raisonnements dans la littérature. Le premier consiste à extraire de l'observation les caractéristiques les plus robustes possibles aux variations. Dans le domaine visuel, on peut par exemple citer l'essor des méthodes via réseau de neurones (SÜNDERHAUF et al., 2015 ; ARROYO et al., 2016). Le second revient à modifier l'observation en entrée pour qu'elle corresponde aux données de la carte, comme dans (NEUBERT,

SÜNDERHAUF et PROTZEL, 2015) où les auteurs permettent à leur système de se relocaliser à travers différentes saisons. Pour cela, leur solution prédit l'apparence d'une image de requête acquise en hiver pour qu'elle corresponde aux images de référence cartographiées en été.

2.2.4.2 Maintenance de la carte

La question de la mise-à-jour de la carte se pose lorsque le robot repasse par un endroit déjà visité. Quand une boucle est détectée, deux observations correspondent au même endroit : une issue de la référence, l'autre de la requête. Dans un système basique, l'information portée par la requête peut être ajoutée simplement à la carte. Deux références identifient alors le même lieu. Cependant, dans des applications long-terme, cette stratégie conduit à générer une carte dont la taille grandit infiniment.

Afin de limiter la taille de la carte, les auteurs de (KONOLIGE et BOWMAN, 2009) proposent de conserver un nombre limité d'observations pour décrire chaque lieu de l'environnement. Leur algorithme cherche à conserver la diversité des observations tout en supprimant pour chaque lieu les observations les plus vieilles, et les moins souvent associées lors de détection de fermeture de boucle.

D'autres stratégies tentent d'optimiser l'information contenue dans la carte en supprimant amers ou références selon différents critères. Dans (SOO PARK et al., 2013 ; DYMCZYK et al., 2015), les amers de la carte à conserver sont identifiés via un critère de couverture maximale. L'idée est de favoriser les amers vus dans plusieurs observations. Une autre stratégie adoptée dans (STEINER, HUANG et LEONARD, 2015) cherche à favoriser les lieux particulièrement adaptés pour la localisation. Les auteurs définissent pour cela un critère prenant en compte la probabilité qu'un lieu soit visité, ainsi que sa position par rapport aux lieux voisins cartographiés.

2.2.4.3 Sélection d'amers

À l'inverse de la maintenance de la carte qui s'effectue sur des données déjà incluses dans la carte, la sélection d'amers souhaite n'ajouter que l'information la plus pertinente.

Une stratégie assez intuitive consiste à n'ajouter que les amers issus d'objets statiques. En effet, lors d'une phase de cartographie, conserver des repères sur des objets susceptibles d'être déplacés peut conduire à de futures erreurs de localisation, en particulier dans un formalisme métrique. On retrouve ici l'intérêt des approches sémantiques, où la segmentation et classification d'objets dynamiques permettent un tri des entrées utilisées.

De plus, selon les auteurs de (DYMCZYK et al., 2016), la qualité d'un système de SLAM dépend directement de la re-détectabilité des amers de la carte. L'utilisation de caractéristiques re-détectables et invariantes à diverses variations est alors encouragée par ce constat.

* * *

La localisation et cartographie simultanées constitue donc un problème essentiel dans le secteur de la robotique mobile. Ainsi, de nombreuses méthodes de SLAM sont proposées dans la littérature. Cependant, comme nous l'expliquions au début de ce chapitre, le SLAM ne peut pas être considéré comme résolu. Différents facteurs entrent en jeu comme la plate-forme utilisée et la fonction du robot. Pour donner un cadre plus précis à nos travaux nous présentons dans la partie suivante le robot utilisé dans cette thèse ainsi que les lacunes de la solution de localisation employée en pratique sur ce dernier.

2.3 SLAM embarqué sur le robot Pepper

Les travaux développés dans cette thèse se concentrent sur la localisation des robots Pepper en milieu intérieur. Créé par Aldebaran Robotics pour le groupe japonais SoftBank, le robot Pepper (visible en Figure 2.10) a été dévoilé au public en juin 2014. Pepper est un robot social. Son objectif est de créer des interactions fortes avec des personnes non-expertes en robotique. Aujourd'hui, il est possible de trouver ces robots dans des boutiques, où ils exercent des fonctions d'accueil, d'animation ou de conseil aux clients.

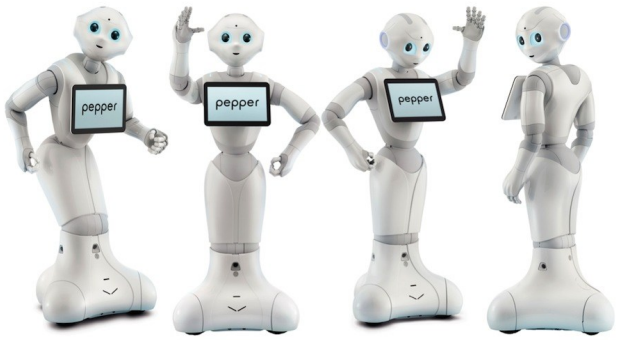


FIGURE 2.10: Le robot Pepper : présenté pour la première fois dans (LAFAYE, GOUAILLIER et WIEBER, 2014). L'aspect humanoïde du robot est pensé pour rendre l'interaction avec des humains aussi intuitive et naturelle que possible.

Une précédente thèse (WIRBEL, 2014) a cherché à développer une solution de localisation pour les robots humanoïdes d'Aldebaran Robotics. La solution créée devait être compatible avec les robots Pepper et NAO. Depuis, la solution de SLAM sur Pepper s'est spécifiquement adaptée pour pouvoir profiter des capteurs métriques de sa base. Les deux sous-parties qui suivent détaillent respectivement le robot Pepper, plate-forme utilisée au cours de cette thèse, et la solution logicielle de localisation intégrée sur ces robots, ses avantages et ses lacunes.

2.3.1 Plate-forme et capteurs disponibles

La conception de Pepper est pensée de sorte à favoriser l'interaction avec ses utilisateurs. De forme humanoïde, le robot est doté d'une tête, de deux bras, d'un torse et d'une jambe. Il possède 20 degrés de liberté. Sa base triangulaire est équipée de trois roues motrices omnidirectionnelles, disposées au niveau des trois sommets, qui rendent

la plate-forme holonome. Pepper mesure 120 cm de haut et pèse 28 kg. Sa batterie lui confère une autonomie d'au moins 12 h en fonctionnement normal.

Pepper est équipé d'un processeur quatre cœurs Atom E3845 fonctionnant à une fréquence de 1,91 GHz. Dans un contexte applicatif standard, trois de ces cœurs sont utilisés pour le calcul de tâches fondamentales comme le contrôle et la planification de mouvements, la détection de personnes et la reconnaissance de la parole.

En 2018, Pepper est vendu 20 000 €, dans 80% des cas à des partenaires qui développent des solutions logicielles pour leurs clients. 12 000 unités de Pepper sont ainsi en circulation dans le monde, ce qui en fait le robot le plus populaire de sa catégorie.

Pour percevoir le monde et créer de l'interaction, Pepper embarque de nombreux capteurs extéroceptifs [2]. Dans le contexte de localisation et navigation, les plus pertinents sont listés ci-dessous et positionnés sur la Figure 2.11 :

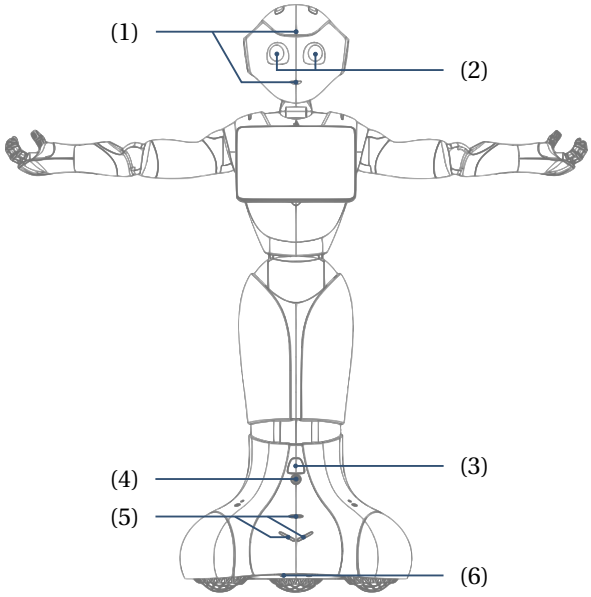


FIGURE 2.11: Capteurs extéroceptifs de Pepper potentiellement utilisables pour la localisation et cartographie : deux caméras monoculaires (1), une caméra stéréo (2), trois caméras infrarouge (3), deux sonars (4) et des émetteurs lasers (5) (6).

(1) Deux caméras monoculaires, placées respectivement sur le front et dans la bouche. La caméra du front est principalement utilisée pour de l'interaction, quand celle du bas permet de voir le sol dans une zone proche du robot [3]. Les champs de vue de ces caméras ne se recouvrent cependant pas suffisamment pour les utiliser comme un dispositif stéréo.

(2) Une paire de caméras calibrées fonctionnant comme un dispositif stéréo. L'information de profondeur extraite de cette caméra aide pendant l'interaction mais également pour l'évitement d'obstacle.

2. Un aperçu technique de Pepper est disponible en ligne à l'adresse http://doc.aldebaran.com/2-5/family/pepper_technical/index_pep.html. Cette thèse s'applique à la version hardware 1.8 du robot.

3. En général utilisée pour la détection d'obstacles.

(3) Trois caméras infrarouge, disposées sur chaque côté de la base du robot, observant les points d'impact des lasers émis par les émetteurs (5) et (6).

(4) Deux sonars, un à l'avant et un à l'arrière. Leur rôle est similaire à celui des capteurs lasers. Malgré leur moins bonne précision, l'utilisation de ces sonars permet de détecter des obstacles imperceptibles autrement, comme des baies vitrées.

(5) Des émetteurs lasers orientés vers le bas ou en biais. Leur rôle consiste à détecter des obstacles mais aussi des ruptures de pente, par exemple des rampes ou des marches.

(6) Des émetteurs lasers sur chaque côté de la base du robot, émettant un plan laser parallèle au sol, à une hauteur d'environ 3 cm, dont les points d'impact sont observés par les caméras infrarouge (3).

D'autres capteurs extéroceptifs tels que quatre microphones placés sur la tête du robot, trois capteurs de pression au niveau des roues et des capteurs tactiles sur la tête et les mains sont présents sur le robot. Leur rôle est principalement dédié à l'interaction (microphones et capteurs tactiles) ou à des problématiques de sécurité (capteurs de pression).

2.3.1.1 Capteurs métriques lasers

Conçus à l'origine pour de l'évitement d'obstacle, les lasers horizontaux de Pepper [4] sont utilisés dans une solution de localisation déjà intégrée sur le robot. Ces lasers permettent d'extraire une information métrique sur la position d'obstacles aux alentours du robot grâce au couplage émetteur (6) - récepteur (4) visible en Figure 2.11. Trois couples émetteur-récepteur sont disposés sur chaque côté de la base triangulaire de Pepper, comme le montre la Figure 2.12.

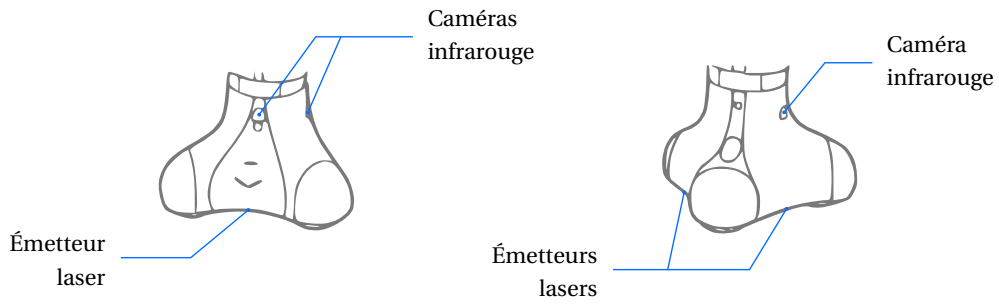


FIGURE 2.12: Capteurs lasers utilisés pour la localisation de Pepper.

Le fonctionnement de ce capteur est le suivant. L'émetteur émet un plan laser parallèle au sol, à environ 3 cm de haut. La caméra scrute ensuite les points d'impact du plan émis avec les éventuels obstacles présents autour du robot. Grâce à une calibration industrielle et une méthode géométrique simple, une distance est ensuite associée aux points d'impact observés (Figure 2.13).

4. Documentation disponible : http://doc.aldebaran.com/2-5/family/pepper_technical/laser_pep.html

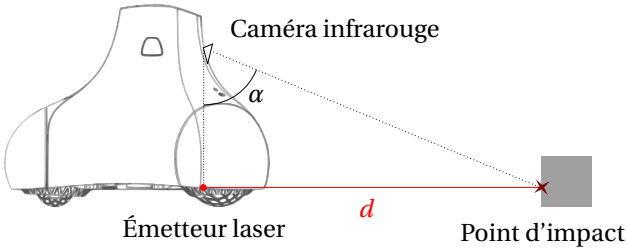


FIGURE 2.13: Mesure de distance à partir des capteurs lasers de Pepper. Grâce à la calibration industrielle de la caméra, un point d'impact est associé à un angle α puis à une distance d.

Cependant, l'objectif originel de ce dispositif est de résoudre des problématiques d'évitement de collision. Ainsi, ces lasers fournissent des données fiables pour l'évitement d'obstacles mais pas forcément compatibles avec des tâches de localisation.

Champs de vue des caméras infrarouge : Les champs de vue des caméras infrarouge sont limités. Si un obstacle est placé trop près du robot, le système ne le perçoit pas. Il y a donc une incertitude sur la présence d'un obstacle en dessous la distance minimale détectable (20 cm) et au-dessus de la distance maximale détectable (5 m).

De plus, les champs de vue des trois caméras ne se recouvrent pas et ne permettent pas de générer un scan laser avec un champ de vue à 360°autour du robot. Il existe donc des zones aveugles, comme le montre la Figure 2.14.

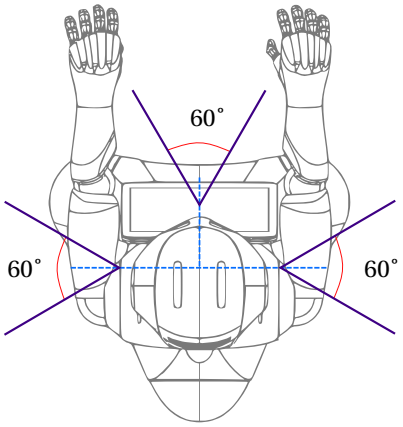


FIGURE 2.14: Capteurs lasers et angles morts : chacune des trois caméras infrarouge a un champ de vue limité de 60°. Le cumul des trois dispositifs laisse donc des angles morts.

Une stratégie logicielle pour obtenir un scan laser à 360°consiste à utiliser une mémoire temporelle des obstacles [5]. Cela revient à exploiter les mouvements du robot pour combler les zones aveugles.

Discrétisation et précision : Ce dispositif étant initialement destiné à l'évitement de collision, la précision du capteur est loin d'être équivalente à celle de télémètres la-

5. http://doc.aldebaran.com/2-5/naoqi/motion/alnavigation.html?highlight=safety%20map

sers couramment utilisés dans les méthodes de SLAM. Avec ce dispositif, la précision de la distance estimée diminue lorsque le point d'impact observé s'éloigne. Typiquement, pour un point d'impact à 1 m du robot, l'incertitude sur la distance estimée est de 2 cm. À 2 m, cette incertitude passe à une vingtaine de centimètres.

De plus, pour des raisons de bande passante, les données issues de ce dispositif sont discrétisées. L'image de chaque caméra infrarouge est divisée en 15 bandes, comme le montre la Figure 2.15. Sur chaque bande, seul le point d'impact le plus proche du robot est conservé. Ce choix, pertinent pour de l'évitement de collision, ajoute encore une difficulté dans l'utilisation de ces capteurs pour des tâches de localisation et cartographie simultanées, puisqu'il génère du bruit de quantification.

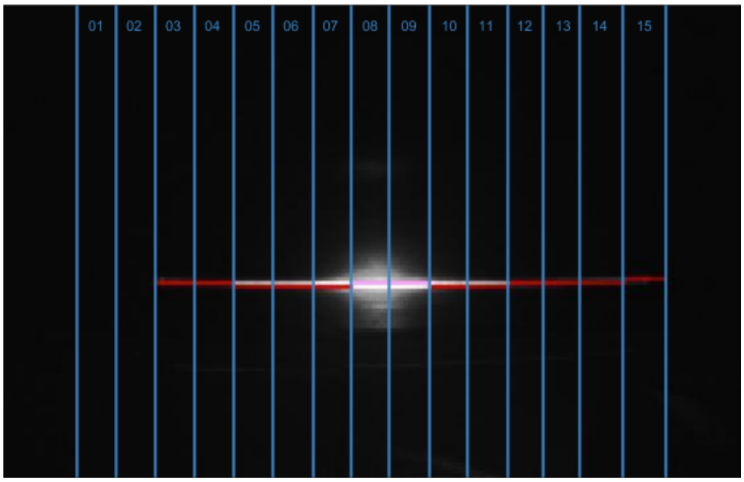


FIGURE 2.15: Observation d'une caméra infrarouge et bandes de discrétisation. Sur chacune des 15 bandes, seul le point d'impact le plus proche de la base est conservé.

Malgré les limitations d'un tel dispositif, une solution de localisation et cartographie simultanées basée sur ces capteurs est présente dans la version logicielle des robots Pepper. Nous présentons brièvement cette solution dans la sous-partie qui suit.

2.3.2 Solution de SLAM basée lasers

Bien que la majorité des robots Pepper soient utilisés de façon statique, une solution logicielle de localisation et de cartographie simultanées a été développée par les ingénieurs d'Aldebaran Robotics pour ce robot. Pour pallier les lacunes des dispositifs lasers présentés plus haut, la solution de SLAM se base sur l'utilisation d'une méthode de filtrage particulaire classique pour générer une carte topo-métrique de l'environnement.

La solution utilise une approche similaire à celle introduite par FastSLAM (MONTEMERLO et al., 2002). Elle utilise les mesures de l'odométrie pour propager des particules et les ré-échantillonne en fonction de l'observation lasers courante. Cette méthode permet à Pepper de générer une carte de son environnement. La qualité de la carte créée dépend fortement de l'odométrie du robot, mais grâce au filtrage particulaire, le robot peut reparcourir des zones déjà visitées en étant constamment localisé. De cette manière, Pepper peut réaliser des tâches de navigation sans dérive dans sa carte.

2.3.2.1 Performances

Cette solution a montré de bonnes performances en pratique. Contrairement à une solution basée uniquement sur l'odométrie, l'utilisation des lasers apporte une information extéroceptive qui permet au système d'estimer sa pose sans dérive dans la carte utilisée.

Lors de Pepper World 2017 à Tokyo, un robot Pepper a pu se localiser via cette solution logicielle pendant toute une journée de fonctionnement [6] sans dérive. Dans cette démonstration, Pepper se déplaçait le long d'une estrade de 7 m, et délivrait un discours en fonction du panneau devant lequel il se trouvait (Figure 2.16).

La carte était construite par un utilisateur non-expert qui poussait simplement le robot et lui indiquait les poses correspondant à un discours. Cette carte est visible sur la Figure 2.16. La solution logicielle de localisation était ensuite utilisée en localisation pure pour permettre à Pepper de ne pas se perdre pendant sa journée de fonctionnement.

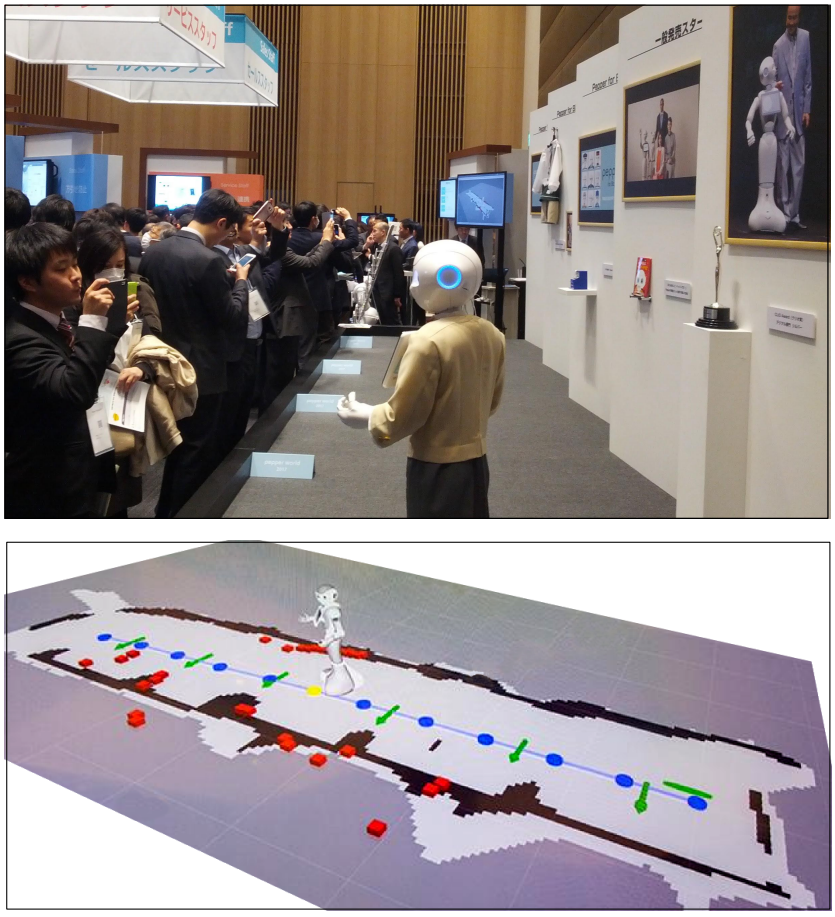


FIGURE 2.16: Exemple d'application utilisant la localisation lasers : Pepper World 2017, à Tokyo. Pepper se déplace sur une estrade et délivre un discours face à certains panneaux. La carte laser créée est affichée avec : les nœuds topologiques mis en évidence en bleu ; les obstacles perçus pendant la cartographie en noir ; les zones libres en gris clair ; les poses de discours en vert ; les impacts lasers observés en rouge.

6. 10 heures d'affilée, sur deux jours consécutifs.

Une telle application aurait été impossible en se basant sur la seule information de l'odométrie. Très vite cette dernière aurait dérivé et le robot aurait été incapable de savoir devant quel panneau il se trouvait.

Un autre atout fondamental de cette solution est qu'elle requiert très peu de ressource computationnelle. Sur le robot Pepper, elle n'utilise que 1 à 2 % d'un seul des quatre cœurs du robot. De plus, la solution est passive et ne nécessite aucun comportement particulier. Cette caractéristique est très importante puisque la fonction sociale de Pepper fait que tout système de localisation doit être caché à l'utilisateur. En effet, une localisation reprenant la main sur les articulations du robot, ou perceptible par des temps de calcul trop longs, nuirait fortement à l'interaction, ainsi qu'à l'aspect humain du robot.

Néanmoins cette solution présente différentes lacunes, directement liées à l'information extraite de ce type de dispositif laser.

2.3.2.2 Limitations

L'information issue des capteurs lasers est de très faible qualité pour des tâches de localisation. Ce dispositif n'observe en effet que les points d'impact proches du sol. Dans l'exemple présenté en Figure 2.16, Pepper ne perçoit à travers ses lasers que les murs, et les petits chevalets posés au sol (bien visibles sur la carte construite).

De plus, si Pepper peut se déplacer dans la carte sans dérive, la construction de cette dernière repose fortement sur l'odométrie. Les observations lasers ne contiennent pas suffisamment d'information pour corriger significativement la trajectoire. Cette déformation de la carte est illustrée en Figure 2.17.

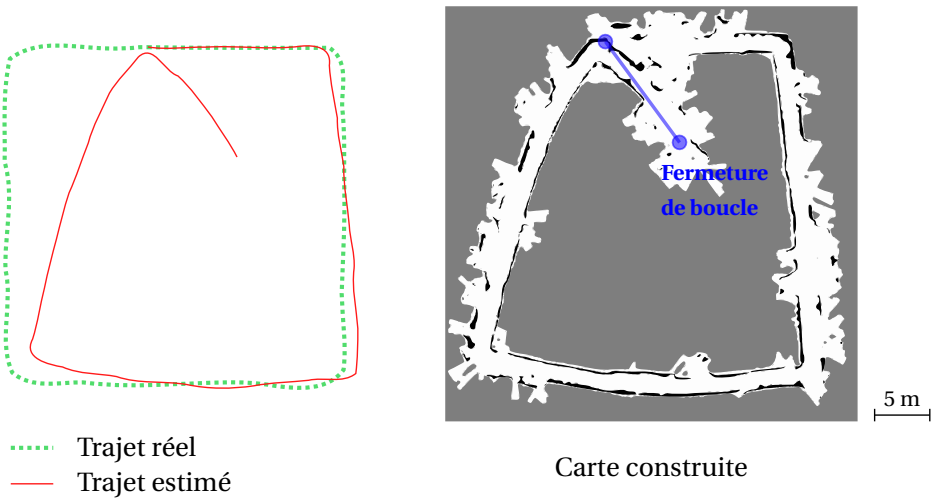


FIGURE 2.17: Dérive odométrique et carte laser (avec en noir les obstacles, en blanc les zones libres, et en gris les zones incertaines). Le trajet estimé par la solution de localisation lasers diverge du vrai chemin parcouru. La carte construite perd ainsi sa cohérence métrique globale. Une fermeture de boucle pourrait être utilisée pour corriger la carte, mais sa détection est trop incertaine à partir de l'information apportée par les lasers.

Pour compenser cette dérive, une optimisation de graphe pourrait être réalisée lorsqu'une boucle est détectée. Mais l'information des lasers reste trop pauvre pour détecter avec certitude une fermeture de boucle. En effet, il est par exemple compliqué pour le robot de différencier deux couloirs différents à l'aide de ses seuls lasers. La détection de fermeture de boucle par le dispositif lasers est donc peu fiable, et optimiser la carte créée en utilisant de fausses fermetures de boucle détériorerait forcément cette dernière.

De la même manière, les lasers permettent difficilement au robot d'estimer directement sa pose dans la carte sans aucune connaissance *a priori* sur celle-ci. Typiquement, à l'allumage du robot, ou après le chargement d'une carte, l'utilisateur doit spécifier manuellement la position actuelle de Pepper pour initialiser son filtre à particules. Ce défaut est particulièrement gênant pour des utilisateurs non-experts.

Pour finir, cette solution basée sur les lasers est mal adaptée à certains environnements (Figure 2.18) ; par exemple, dans des environnements contenant beaucoup d'objets avec une faible empreinte au sol, comme des tables ou des chaises. Sur ce type d'objets, seuls les pieds sont observables à travers les lasers de Pepper. Or la discrétisation du capteur présentée en 2.3.1.1 génère dans ces situations de fortes discontinuités qui dégradent le système de localisation. De plus, dans de grandes salles avec peu d'obstacles au sol, la précision combinée aux limites perceptuelles du capteur font que la localisation basée sur les lasers y est impossible.

En résumé, les principales lacunes de cette solution de localisation basée sur les lasers, du point de vue utilisateur sont les suivantes :

1. l'impossibilité de se relocaliser dans une carte sans un bon *a priori* sur sa position courante ;
2. la nécessité pour le robot d'être placé dans un environnement contenant des obstacles proches et dont l'empreinte au sol est suffisamment importante.

2.4 Conclusion et objectifs de la thèse

Ce chapitre a présenté la littérature associée à la localisation autonome des robots en milieu intérieur non-contraint et a introduit le problème du SLAM. Une présentation de la solution de localisation actuelle sur le robot Pepper a mis en évidence ses lacunes.

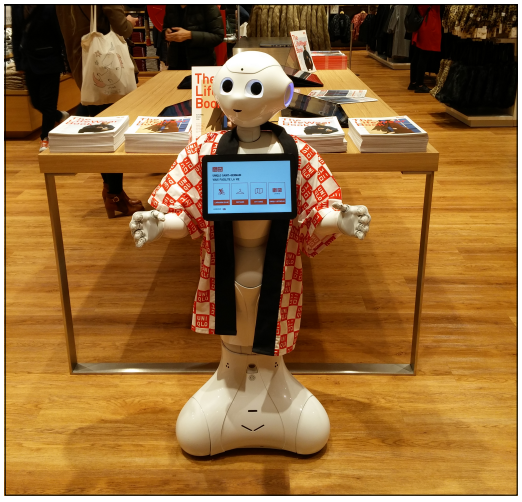
Les objectifs de cette thèse sont donc la mise en place d'une solution de localisation industrialisable sur Pepper qui pallie les problèmes de la solution actuelle, basée sur les capteurs lasers du robot. La solution créée doit prendre en compte les différentes contraintes liées aux fonctions de Pepper (s'exécuter de manière passive avec peu de ressource computationnelle) et ne nécessiter l'ajout d'aucun capteur supplémentaire sur le robot, ni équipement spécifique dans l'environnement.

Les travaux présentés dans cette thèse sont donc fortement influencés par le contexte applicatif lié au robot Pepper et peuvent se diviser en deux parties, chacune répondant à un des problèmes mis en évidence dans la partie précédente.

Dans le chapitre 3, nous répondons au problème de la relocalisation d'un robot dans une carte. Notre principale contribution est la mise en place d'un processus de fusion précoce, mêlant données visuelles et Wi-Fi dans une approche probabiliste.

Dans le chapitre 4, nous proposons une solution de SLAM métrique compatible avec les contraintes et utilisations classiques de Pepper. Notre contribution est ici essentiellement d'intégration, et propose d'ajouter l'information apportée par l'odométrie du robot pour compenser les lacunes du SLAM visuel sur Pepper.

(a) Costa Croisière



(b) Magasin Uniqlo

(c) Banque Mizuho

FIGURE 2.18: Exemples d'environnements compliqués : dans le hall d'un bateau de croisière (a), dans un magasin de vêtements (b) et dans l'entrée d'une banque (c). Dans l'exemple (a), les obstacles sont éloignés du robot et la précision sur les distances estimées est insuffisante pour des tâches de localisation. L'exemple (b) met en lumière le problème des obstacles possédant une faible empreinte au sol (ici la table). Ces derniers sont difficilement utilisables à cause de la discrétisation de l'image infrarouge en 15 bandes verticales. Enfin, (c) illustre un environnement trop uniforme du point de vue des dispositifs lasers, le seul obstacle perceptible étant le mur placé derrière le robot.

Chapitre 3

Relocalisation à partir de données Wi-Fi et visuelles

Sommaire

Ce chapitre se concentre sur le problème de localisation dans une carte sans *a priori* valable sur la pose courante du robot. En pratique on retrouve ce problème dans diverses situations. Par exemple, lorsque Pepper est allumé, il ne sait pas si il a été déplacé pendant qu'il était éteint et ne possède donc pas de connaissance *a priori* valable sur sa pose. Dans notre contexte, le robot doit être capable d'identifier de manière autonome sa pose dans un environnement connu à l'allumage afin de lancer des comportements adéquats.

Pour résoudre le problème de la localisation globale, certaines solutions de l'état-de-l'art se basent sur l'apparence visuelle. La partie 3.1 présente ces solutions, et parmi ces dernières, la méthode FAB-MAP, que nous utilisons dans nos travaux, est détaillée en partie 3.2.

Pour combler les lacunes de la localisation visuelle, nous cherchons à y ajouter l'information issue du Wi-Fi. La partie 3.3 introduit diverses stratégies de relocalisation fusionnant des données Wi-Fi et visuelles, afin de les comparer au processus que nous introduisons dans ce chapitre et que nous qualifions de *fusion précoce*. L'intérêt de combiner Wi-Fi et vision pour des tâches de relocalisation est ensuite démontré expérimentalement à travers les résultats présentés en partie 3.4.

3.1 L'apparence visuelle

Depuis les années 2000, l'emploi des caméras dans les problèmes de SLAM constitue un large champs de recherche. Cet engouement s'explique par le faible coût de ces capteurs et par leur présence sur la plupart des systèmes d'électronique embarquée (smartphones, tablettes, ordinateurs portables, etc.).

Un sous-problème du SLAM est d'ailleurs très souvent résolu à partir de capteurs visuels : la détection de fermeture de boucle. La détection de fermeture de boucle est à prendre ici au sens large du terme et regroupe aussi bien le problème du robot capturé que celui de la localisation globale. Pour cela, la plupart des approches font appel à des méthodes de localisation basées sur l'apparence visuelle.

Les méthodes de reconnaissance de lieu par apparence visuelle consistent à associer deux images provenant du même lieu en utilisant leur apparence (LOWRY et al., 2016). Le terme apparence est choisi pour illustrer un degré d'abstraction par rapport à l'image brute. Ces méthodes souhaitent en effet fonctionner malgré des variations qui peuvent être temporelles (variations de luminosité, objets déplacés, etc.), ou encore un léger changement de pose lors de la prise de vue. L'apparence visuelle est ainsi très sollicitée pour résoudre les tâches de détection de fermeture de boucle.

En général, le traitement d'une image requête consiste à comparer cette dernière à toutes les images références qui constituent la carte. Pour associer deux images, les techniques d'apparence visuelle peuvent faire appel à différentes métriques comme des distances ou des valeurs normalisées correspondant à la probabilité que les deux images aient été acquises dans le même lieu. Il est cependant difficilement concevable de comparer de manière brute deux images, avec par exemple une distance sur les valeurs RGB de chaque pixel. Cela conduirait à des calculs trop coûteux mais aussi donnerait trop d'importance à des variations anecdotiques. Les algorithmes employant l'apparence vi-

suelle passent donc par une description compacte de l'image, souvent sous forme de vecteur, appelé *vecteur d'apparence*. Cette partie réalise un état-de-l'art des méthodes couramment utilisées dans les tâches de détection de fermeture de boucle par apparence visuelle.

3.1.1 Les descripteurs globaux

Parmi les premières approches de relocalisation par apparence visuelle, certaines se basent sur l'utilisation de l'information de couleur contenue dans les images. Dans (ULRICH et NOURBAKHSH, 2000 ; BLAER et ALLEN, 2002), les auteurs emploient des vecteurs d'apparence correspondant à des histogrammes de couleurs extraits sur l'image entière.

Un autre type de description global se base sur le *Gist* d'une scène. Le *Gist* d'une scène correspond aux informations significatives qu'un observateur identifie très rapidement (POTTER, 1975 ; OLIVA, 2005).

L'extraction du *Gist* présentée dans (OLIVA et TORRALBA, 2006) repose ainsi sur l'application de filtres de Gabor sur l'image, à diverses orientations et fréquences. De tels descripteurs Gabor-Gist ont été utilisés dans des problématiques de reconnaissance de lieu par apparence visuelle (MURILLO et KOSECKA, 2009 ; SIAGIAN et ITTI, 2009 ; LIU et ZHANG, 2012).

Plus récemment, les travaux présentés dans (SÜNDERHAUF et PROTZEL, 2011) suggèrent d'adapter des descripteurs de points d'intérêt locaux à l'image globale. Pour cela, les auteurs appliquent les descripteurs binaires BRIEF [1] (CALONDER et al., 2010) non plus au voisinage d'un pixel mais à l'échelle d'une image. Ils obtiennent de cette façon un vecteur d'apparence BRIEF-Gist composé de valeurs binaires. Grâce à la nature binaire de leurs éléments, ces vecteurs d'apparence se comparent facilement deux à deux par distance de Hamming.

Ces diverses méthodes sont qualifiées de globales puisqu'elles s'appliquent directement à l'image entière. À l'inverse, d'autres techniques passent par l'utilisation de primitives visuelles locales comme les points d'intérêt. Cependant, l'association brute de points d'intérêt entre deux images ne suffit pas à résoudre les problématiques de détection de fermeture de boucle par apparence visuelle. Décrire une image par tous ses points d'intérêt se révèle vite être une approche très chronophage. Pour pallier ce problème, de nombreuses méthodes locales choisissent alors d'avoir recours à l'outil des sacs-de-mots.

3.1.2 Les sacs-de-mots

L'outil des sacs-de-mots est directement inspiré des algorithmes de recherche de texte (SIVIC et ZISSERMAN, 2003). Pour classifier des documents texte, l'idée des sacs-de-mots est de résumer le contenu d'un document en comptant les occurrences des mots qu'il contient, peu importe leur position dans le texte ou leur enchaînement.

1. Pour *Binary Robust Independent Elementary Features*.

Pour fonctionner, un dictionnaire contenant les mots de référence est donc nécessaire. Pour un dictionnaire contenant k mots, un document est résumé par un vecteur $V = (v_1, v_2, ..., v_k)$. Chaque valeur v_i indique alors la fréquence d'apparition du mot i du vocabulaire dans le document.

Le principe reste le même appliqué à l'image. Par analogie avec un document texte contenant des mots, l'image est vue comme un ensemble de points d'intérêt décrits par leurs descripteurs. Les sacs-de-mots visuels utilisent alors un vocabulaire dont les mots sont des descripteurs moyens. En général, ces descripteurs moyens sont construits à l'aide de méthodes de regroupement, comme les k-moyennes par exemple, sur un grand nombre de descripteurs de points d'intérêts extraits à partir d'images d'apprentissage. Chaque point de l'image est alors relié à un mot visuel du vocabulaire, soit le descripteur moyen le plus proche, pour construire V, le vecteur d'apparence de l'image. La comparaison de deux images s'effectue ensuite à l'aide de leur vecteur d'apparence.

Remarquons que l'utilisation des sacs-de-mots visuels repose en général sur deux phases. Dans une phase réalisée de manière hors-ligne, le vocabulaire est appris. Cette construction du vocabulaire est réalisée à partir d'un grand nombre d'images d'apprentissage sur lesquelles ont été extraits des descripteurs de points d'intérêt. Le résultat de cet apprentissage est ensuite utilisé dans le fonctionnement en-ligne. Ce dernier consiste à acquérir un image requête, construire son vecteur d'apparence et le comparer aux vecteurs d'apparence des références de la carte pour tenter de localiser la requête.

Diverses stratégies existent pour rendre pertinente la comparaison entre les vecteurs d'apparence de plusieurs images. En effet, pour éviter les problèmes d'ambiguïté perceptuelle, les mots du vocabulaire ne possèdent pas tous le même pouvoir discriminant pour la localisation. La méthode du TF-IDF, pour *term frequency - inverse document frequency* (SALTON et BUCKLEY, 1988), va accorder plus d'importance aux mots rares du vocabulaire. L'intérêt d'une telle approche est mis en lumière sur la Figure 3.1, où des points redondants de l'environnement apportent finalement très peu d'information.

Le TF-IDF : La méthode du TF-IDF cherche à éviter cet écueil dans la construction de son vecteur d'apparence V à l'aide de deux termes. Le premier, réduit l'impact des mots apparaissant de nombreuses fois dans l'image requête, et le deuxième accorde plus d'importance aux mots présents dans peu d'images de la base d'apprentissage comme dans l'équation (3.1) :

$$v_i = \frac{n_{ir}}{n_r} \log\left(\frac{N}{N_i}\right) \tag{3.1}$$

où n_{ir} correspond au nombre d'apparitions du mot i dans l'image requête r, et n_r est le nombre total de mots détectés dans l'image. N_i et N sont déterminés durant l'apprentissage et correspondent respectivement au nombre d'images d'apprentissage contenant le mot i, et au nombre total d'images d'apprentissage. Il est alors possible de calculer une distance D entre deux vecteurs d'apparence V_r et V_d avec l'équation (3.2) :

FIGURE 3.1: Fréquence d'apparition des mots visuels et pouvoir de distinction. Dans cet exemple, les points d'intérêt associés au mot visuel le plus vu dans l'image sont mis en évidence en vert. On comprend que ce mot visuel a peu d'impact pour distinguer des lieux différents dans un environnement où les murs sont faits de briques. Les mots visuels associés à des points extraits sur le banc seraient préférables pour décrire ce lieu.

$$D = \frac{V_r^\top V_d}{\|V_r\|_2 . \|V_d\|_2} \tag{3.2}$$

Vocabulaire hiérarchique : Dans (NISTER et STEWENIUS, 2006), les auteurs définissent une méthode d'association d'images mettant en œuvre un *vocabulaire hiérarchique*. Pour créer ce vocabulaire, leur base d'apprentissage[2] est d'abord divisée via une méthode de regroupement en k groupes. Une fois cette première étape réalisée, chaque groupe obtenu est à nouveau divisé en k groupes ; l'algorithme itère ainsi L fois. Un vocabulaire hiérarchique est ainsi créé et se présente sous la forme d'un *arbre*. Les descripteurs de points d'intérêt extraits sur une image sont propagés à travers l'arbre. Le vecteur d'apparence de l'image contient alors des valeurs v_i définies par $v_i = n_i w_i$, où n_i correspond au nombre de descripteurs passant par le nœud i de l'arbre, et w_i est le poids associé au nœud i. La mesure de distance D, entre deux vecteurs V_r et V_d se calcule ensuite par :

$$D = \left\| \frac{Vr}{\|Vr\|_1} - \frac{V_d}{\|V_d\|_1} \right\|_1 \tag{3.3}$$

En pratique, les auteurs de (NISTER et STEWENIUS, 2006) mettent en évidence que les nœuds n_f des feuilles de l'arbre sont les plus puissants pour associer correctement des images entre elles. Souvent, seuls les poids w_f, associés aux nœuds des feuilles de l'arbre, sont donc différents de 0.

2. Composée des descripteurs de points d'intérêt, extraits sur les images d'apprentissage.

Grâce à cette organisation hiérarchique, la propagation d'un descripteur de point d'intérêt est plus rapide. Les auteurs peuvent ainsi utiliser un vocabulaire plus grand, ce qui permet à leur algorithme d'être plus discriminant.

(GÁLVEZ-LÓPEZ et TARDOS, 2012) décrit un algorithme de localisation qui repose également sur un vocabulaire hiérarchique. Leur vocabulaire correspond au dernier niveau de l'arbre, et les mots sont pondérés à l'aide de la TF-IDF. Remarquons que les auteurs font le choix d'employer des descripteurs binaires BRIEF, pour diminuer leur temps d'extraction et de propagation dans l'arbre. Dans (MUR-ARTAL et TARDÓS, 2014), les auteurs reprennent la même approche avec des descripteurs ORB.

Cooccurrence des mots : Reposant aussi sur l'outil des sacs-de-mots, l'algorithme FAB-MAP (CUMMINS et NEWMAN, 2008), pour *Fast Appearance-Based Mapping*, ne calcule plus une mesure de similarité entre deux images, mais la probabilité que deux observations proviennent du même endroit. Cette probabilité prend ainsi en compte qu'une image requête puisse avoir été acquise dans un endroit inconnu.

Le principal atout du FAB-MAP est l'utilisation des cooccurrences entre mots visuels du vocabulaire appris. En effet, certains mots visuels du vocabulaire ont tendance à apparaître ensemble, car ils apparaissent sur un même objet. Pour capturer au mieux ces corrélations dans un vocabulaire pouvant contenir plusieurs milliers de variables[3] FAB-MAP emploie les arbres de Chow-Liu.

Construction en-ligne du dictionnaire : Dans les méthodes faisant appel à l'outil des sacs-de-mots, le choix du vocabulaire à utiliser est très important. Il est par exemple plus judicieux de construire son dictionnaire à partir d'images d'apprentissage acquises dans un environnement visuellement proche de celui de fonctionnement. Le vocabulaire peut donc être spécifique et mieux fonctionner dans certains milieux. Pour obtenir un vocabulaire adapté, certaines approches font le choix de construire leur vocabulaire en même temps que le robot découvre son environnement de fonctionnement. On parle alors d'*apprentissage en-ligne* du vocabulaire, en opposition à l'*apprentissage hors-ligne* plus communément utilisé.

Dans (FILLIAT, 2007), l'auteur met en place une méthode de relocalisation par vote à l'intérieur de différentes pièces. Les lieux sont ajoutés à la carte sous la supervision d'un utilisateur, et l'apprentissage du dictionnaire est réalisé de manière incrémentale. Pour une nouvelle image ajoutée à la carte, chacun des descripteurs de ses points d'intérêt est associé au mot du vocabulaire courant le plus proche. Si leur distance est supérieure à un seuil, le descripteur initialise un nouveau mot dans le vocabulaire. Dans (ANGELI et al., 2008), la démarche présentée reprend cette construction en-ligne du dictionnaire, mais ajoute des contraintes temporelles ainsi qu'une vérification géométrique lors de l'étape de localisation. Les auteurs de (NICOSEVICI et GARCIA, 2009) créent leur vocabulaire en-ligne par agglomération de *clusters*, lors d'un apprentissage non-supervisé.

3. 11 000 mots dans (CUMMINS et NEWMAN, 2008).

Au-delà de la spécialisation du vocabulaire, appris directement sur des images correspondant aux conditions d'utilisation du robot, ces méthodes permettent de supprimer certains paramètres arbitrairement fixés comme la taille du dictionnaire.

Représentations continues : Plus récemment, certaines approches de relocalisation visuelle, basées sur l'apparence, ont introduit le concept de la localisation « *sans lieu* ». L'idée introduite par ces techniques est de conserver une information spatiale continue, sur la configuration de l'environnement. Par exemple, dans (MEI, SIBLEY et NEWMAN, 2010), les auteurs s'aident de graphes de *co-visibilité*. Le but de cette méthode est de compenser le fait qu'une image requête, acquise entre deux images de référence, puisse être considérée comme un nouveau lieu si elle ne partage pas suffisamment d'information avec chacune des deux images de référence. Repris dans les travaux de (STUMM, MEI et LACROIX, 2013), cette approche génère une carte dont les nœuds sont des repères visuels, et les liaisons entre nœuds indiquent que deux repères ont été observés sur la même image (Figure 3.2). Un suivi des amers visuels est réalisé pour augmenter cette carte avec de nouvelles images, et conserver ainsi une information spatiale sur la répartition des mots du vocabulaire dans l'environnement. Au moment de la localisation, des *lieux virtuels* sont générés à partir de ces informations, et des mots présents dans l'observation de requête. La localisation s'effectue alors sur ces lieux virtuels, qui ne correspondent pas strictement à des observations réelles, mais à des observations possibles.

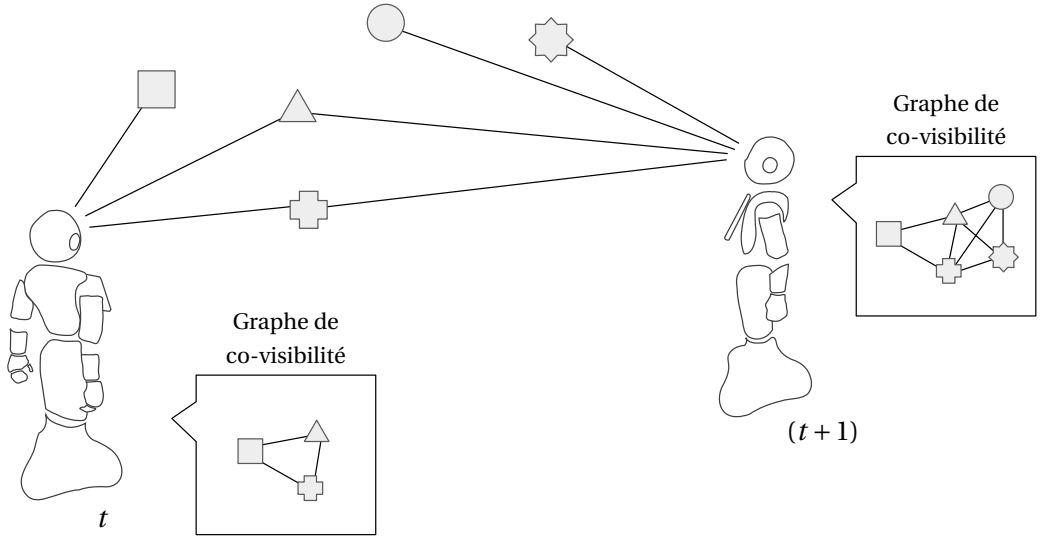


FIGURE 3.2: Représentation du monde par graphe de co-visibilité (STUMM, MEI et LACROIX, 2013).

Actuellement, l'outil des sacs-de-mots est très utilisé pour détecter des fermetures de boucle en se basant sur l'apparence visuelle. Son principal atout est de réduire fortement les dimensions du problème. Néanmoins, des alternatives cherchent à réduire les artefacts provenant de la quantification des descripteurs, tout en gardant le maximum d'information disponible. Par exemple, dans (LYNEN et al., 2014), les auteurs se passent de l'outil des sacs-de-mots. Pour garder une application temps-réel, les auteurs

mettent en place une réduction des dimensions des descripteurs de points d'intérêt à l'aide d'une transformation linéaire. Les fermetures de boucle sont détectées par vote, dans un espace 2D, et sont associées aux régions de forte densité.

* * *

De nombreuses solutions au problème de la relocalisation par apparence visuelle sont proposées dans la littérature. Cependant, l'emploi de la vision rencontre des difficultés dans certains environnements particulièrement redondants. Nos travaux cherchent à profiter du fait que Pepper possède d'autres sources d'informations extéroceptives que ses caméras. Ainsi, nous nous sommes intéressés à l'algorithme du FAB-MAP qui présente l'avantage de pouvoir être adapté pour fonctionner à partir de données issues d'autres types de capteurs. Pour présenter nos travaux sur la relocalisation de Pepper, il convient donc d'introduire dans un premier temps le fonctionnement de l'algorithme FAB-MAP.

3.2 FAB-MAP

3.2.1 Fonctionnement

L'algorithme de localisation FAB-MAP, pour *Fast Appearance-Based Mapping*, a été présenté dans (CUMMINS et NEWMAN, 2008). Reposant sur l'apparence visuelle, FAB-MAP discrétise l'environnement en une succession de nœuds topologiques. Chaque nœud représente un lieu L_i et est associé à une observation. En pratique, une observation Z_i associée à un lieu L_i est le vecteur d'apparence de l'image acquise en ce lieu cartographié.

L'objectif de l'algorithme FAB-MAP est de calculer, pour une image requête, la probabilité que celle-ci provienne d'un lieu de la carte. À chaque requête de localisation, la valeur suivante est calculée pour chaque nœud topologique L_i de la carte :

$$p(L_i|\mathcal{Z}^k) = \frac{p(Z_k|L_i,\mathcal{Z}^{k-1})p(L_i|\mathcal{Z}^{k-1})}{p(Z_k|\mathcal{Z}^{k-1})} \tag{3.4}$$

où Z_k est la $k^{ième}$ observation et $\mathcal{Z}^k$ est l'ensemble des observations, jusqu'à k. Trois termes peuvent être identifiés dans l'équation (3.4) :

— le terme de vraisemblance $p(Z_k|L_i,\mathcal{Z}^{k-1})$,

— le terme de normalisation $p(Z_k|\mathcal{Z}^{k-1})$,

— et $p(L_i|\mathcal{Z}^{k-1})$, correspondant à une connaissance *a priori* sur la pose du robot.

Remarquons que nous traitons ici du problème de la relocalisation. Dans notre utilisation du FAB-MAP, ce dernier terme nous importe donc peu car nous cherchons à identifier les fermetures de boucle lors de grosse dérives du système ou à l'allumage du robot.

Les principales étapes de l'algorithme FAB-MAP sont brièvement présentées dans les sous-parties suivantes.

3.2.1.1 Construction du vecteur d'apparence

La première étape de l'algorithme FAB-MAP consiste à transformer une image requête en une représentation compacte, adaptée au contexte de la localisation. Cette représentation est généralement qualifiée de vecteur d'apparence. Ici, nous parlerons aussi bien de vecteur d'apparence que d'observation et y ferons référence via la notation Z, comme dans l'équation (3.4).

L'algorithme FAB-MAP emploie l'outil des sacs-de-mots pour créer son vecteur d'apparence. Le fonctionnement est exactement le même que celui présenté en 3.1.2, à ceci près que le vecteur n'encode pas les occurrences des mots visuels dans l'image mais simplement leur présence. Pour un vocabulaire de N mots, le vecteur d'apparence Z contient alors N valeurs binaires renseignant sur la présence ou l'absence du mot visuel correspondant dans l'image requête.

Pour fonctionner, cette étape a donc besoin en entrée d'un vocabulaire. Ce vocabulaire est un ensemble de N descripteurs moyens, correspondant chacun à un mot visuel. Il est appris à partir de nombreux descripteurs extraits sur des images d'apprentissage. La méthode la plus populaire pour cela est celle des k-moyennes.

La transformation d'une image en une observation Z est résumée sur la Figure 3.3. L'étape d'association entre caractéristiques et mots visuels revient à réaliser une recherche du plus proche voisin dans les descripteurs moyens qui constituent le dictionnaire.

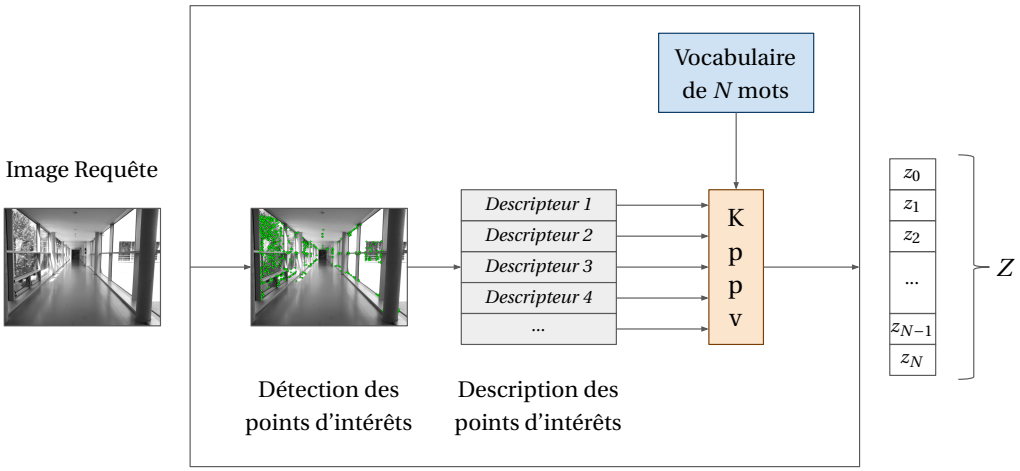


FIGURE 3.3: Création d'une observation FAB-MAP à partir d'une image, appelé aussi vecteur d'apparence. La recherche des K-plus-proches-voisins correspond à la recherche du mot le plus proche dans le vocabulaire pour chaque descripteur ($K = 1$).

Remarquons que ce type de représentation fait perdre toute l'information géométrique sur la position des points d'intérêt dans l'image. Si cela peut constituer un avantage dans le cas où des objets sont déplacés, cette information peut se révéler utile pour rejeter des associations aberrantes entre deux images (PHILBIN et al., 2007 ; CUMMINS et NEWMAN, 2011).

3.2.1.2 Vraisemblance

La deuxième étape de l'algorithme consiste à calculer la valeur du terme de vraisemblance $p(Z_k|L_i, \mathcal{Z}^{k-1})$, conditionné sur le lieu. Dans (CUMMINS et NEWMAN, 2008), ce terme est simplifié en $p(Z_k|L_i)$, en présumant l'indépendance entre la position courante et les observations passées.

La principale contribution du FAB-MAP est l'apprentissage de corrélations entre les différents mots du vocabulaire et leur utilisation lors du calcul de la vraisemblance. Dans (CUMMINS et NEWMAN, 2008), les auteurs démontrent l'intérêt de leur approche par rapport à une approche naïve bayésienne. L'apprentissage de ces corrélations aide ainsi à éviter les fausses associations dues à l'ambiguïté perceptuelle. De plus, l'apprentissage de ces corrélations permet d'associer correctement des images provenant du même endroit, même si ces dernières ont peu de mots visuels en commun.

Cependant, l'apprentissage et l'encodage de ces corrélations peuvent constituer des étapes coûteuses puisqu'un vocabulaire peut facilement contenir plusieurs milliers de mots visuels. FAB-MAP choisit d'utiliser un arbre de Chow-Liu (CHOW et LIU, 1968) pour capturer les corrélations entre les différents mots du dictionnaire. Les arbres de Chow-Liu constituent un sous-ensemble des réseaux bayésiens et permettent d'approximer de grandes distributions discrètes. Leur structure non-dirigée est acyclique et chaque nœud du graphe possède au maximum un parent. Dans (CUMMINS et NEWMAN, 2008), à chaque nœud de l'arbre correspond ainsi un mot du vocabulaire.

La démarche présentée dans (CUMMINS et NEWMAN, 2008) construit cet arbre de Chow-Liu lors d'un apprentissage hors-ligne, à partir d'images d'exemples, pour un vocabulaire de 11 000 mots. Au moment de cet apprentissage, un graphe d'information mutuelle est nécessaire. Pour le fonctionnement en-ligne du FAB-MAP, l'arbre de Chow-Liu construit est suffisant et ce graphe d'information mutuelle peut être jeté.

3.2.1.3 Lieu connu ou inconnu ?

Si l'environnement était entièrement connu, la valeur de vraisemblance suffirait à déterminer la position du robot dans la carte. Il suffirait alors de chercher le lieu dont la valeur de vraisemblance est la plus forte. Or, ici le robot découvre son environnement en même temps qu'il l'explore. L'étape de normalisation permet de détecter si un lieu est déjà dans la carte ou non. En pratique, cette étape permet également de redonner une valeur cohérente à la probabilité $p(L_i|\mathcal{Z}^k)$ calculée, et de rejeter ainsi certains scores trop bas.

La méthode de normalisation présentée dans (CUMMINS et NEWMAN, 2008) divise le terme de normalisation $p(Z_k|\mathcal{Z}^{k-1})$ en deux sommes. La première représente les lieux visités M et la seconde le monde inconnu $\bar{M}$:

$$p(Z_k|\mathcal{Z}^{k-1}) = \sum_{m \in M} p(Z_k|L_m)\, p(L_m|\mathcal{Z}^{k-1}) + \sum_{u \in \bar{M}} p(Z_k|L_u)\, p(L_u|\mathcal{Z}^{k-1}) \tag{3.5}$$

La deuxième somme ne peut être évaluée directement. Les auteurs du FAB-MAP proposent alors d'approximer l'équation (3.5) par :

$$p(Z_k|\mathcal{Z}^{k-1}) \approx \sum_{m\in M} p(Z_k|L_m)p(L_m|\mathcal{Z}^{k-1}) + \; p(L_{new}|\mathcal{Z}^{k-1}) \sum_{u=1}^{n_s} \frac{p(Z_k|L_u)}{n_s} \tag{3.6}$$

où $p(L_{new}|\mathcal{Z}^{k-1})$ correspond à la probabilité d'être dans un nouveau lieu. Cette probabilité est spécifiée en entrée par l'utilisateur (et est réglée à 0.9 dans (CUMMINS et NEWMAN, 2008)). La seconde somme de l'équation (3.6) consiste à échantillonner une observation Z pour créer un modèle de lieu inconnu. Cet échantillonnage de Z est réalisé à partir de n_s images d'entraînement, que nous qualifierons par la suite d'échantillons.

3.2.1.4 Entrées et pré-requis du FAB-MAP

L'algorithme FAB-MAP peut ainsi se diviser en trois étapes correspondant à :

1. la génération d'une observation Z,
2. le calcul de la vraisemblance associée à chaque lieu de la carte,
3. la normalisation de ces valeurs pour les exprimer sous la forme de probabilités.

Chacune de ces étapes nécessite le résultat d'un apprentissage hors-ligne. L'enchaînement de ces étapes et leurs entrées sont rappelés sur la Figure 3.4.

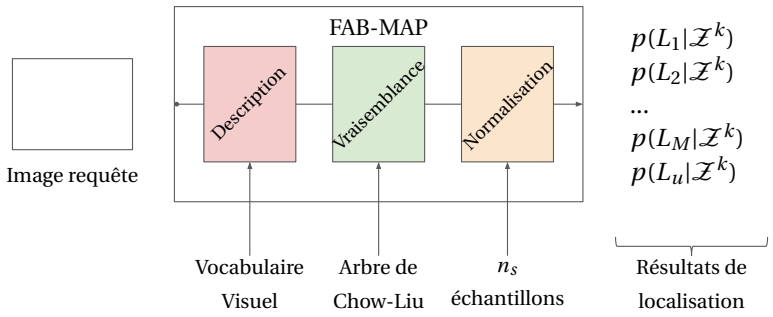


FIGURE 3.4: Fonctionnement global de l'algorithme FAB-MAP : entrées de chaque étape présentées en 3.2.1.

Le vocabulaire, l'arbre de Chow-Liu et les échantillons, peuvent être générés une seule fois, de façon hors-ligne. L'algorithme FAB-MAP requiert cependant quatre autres entrées, qui sont spécifiées par l'utilisateur. Deux scalaires correspondent à une modélisation de l'erreur du détecteur de caractéristiques visuelles utilisé. Un paramètre de lissage σ, et un terme correspondant à la probabilité qu'un lien topologique inconnu mène à un lieu non-visité ($p(L_{new}|\mathcal{Z}^{k-1})$, dans l'équation (3.6)). Dans leur publication originale (CUMMINS et NEWMAN, 2008), les auteurs règlent ces deux derniers paramètres avec :

— $\sigma = 0,99$

— $p(L_{new}|\mathcal{Z}^{k-1}) = 0,9$

Les auteurs mettent également en évidence que rien n'empêche leur algorithme de fonctionner avec des données issues d'autres types de capteurs, du moment qu'un formalisme compatible est défini.

Comme Pepper est équipé de plusieurs caméras, FAB-MAP doit pouvoir fonctionner sur le robot. En pratique, l'algorithme présenté dans (CUMMINS et NEWMAN, 2008) ne peut pas être utilisé tel quel dans notre contexte. La sous-partie qui suit explique pourquoi et présente nos adaptations.

3.2.2 Utilisation sur Pepper

Comme FAB-MAP constitue aujourd'hui un algorithme couramment employé dans le domaine de la localisation par apparence, des projets *open-source* en proposent différentes intégrations. Dans nos travaux, nous reprenons la librairie OpenFABMAP présentée dans (GLOVER et al., 2012) dont l'unique dépendance est OpenCV. Cette intégration du FAB-MAP présente plusieurs avantages comme la possibilité de créer et de réutiliser son propre dictionnaire.

L'algorithme originel FAB-MAP utilise les caractéristiques visuelles SURF, pour *Speeded Up Robust Features* (BAY, TUYTELAARS et VAN GOOL, 2006). Les descripteurs SURF sont des vecteurs de 128 dimensions très utilisés en vision par ordinateur. Cependant, l'utilisation des caractéristiques SURF pose problème dans notre contexte industriel. En effet, les caractéristiques visuelles SURF sont brevetées. Leur utilisation n'est donc pas compatible avec nos usages commerciaux.

L'algorithme FAB-MAP reste malgré tout envisageable puisque d'autres caractéristiques visuelles peuvent être utilisées, sous réserve de bien définir ses entrées, présentées sur la Figure 3.4.

3.2.2.1 Le choix des caractéristiques visuelles

L'utilisation de points d'intérêt dans des problématiques de vision par ordinateur est courante, notamment dans les tâches d'identification ou de suivi. Elle se divise en deux phases. Une première consiste à détecter les points d'intérêt sur une image, ce qui revient à déterminer leur position pixellique. La seconde cherche à caractériser ces points en associant à chacun un descripteur. Ces deux problèmes sont respectivement résolus à l'aide d'un détecteur et d'un descripteur.

Détection : Divers algorithmes de détection de points d'intérêt existent dans la littérature. Un objectif de ces détecteurs est de pouvoir extraire des points remarquables de manière suffisamment robuste pour pouvoir les suivre dans des images pouvant comporter des variations comme des rotations, du flou, des changements d'échelle ou d'intensité lumineuse, etc... Tous les algorithmes de détection ne fournissent pas la même

robustesse à ces variations. Les plus robustes et les plus souvent utilisés sont les détecteurs SURF (BAY, TUYTELAARS et VAN GOOL, 2006) et SIFT[4] (LOWE, 1999). Cependant, ces deux détecteurs sont brevetés ce qui les rend incompatibles avec notre contexte industriel. D'autres détecteurs peuvent être envisagés, comme par exemple :

— le détecteur de coins de Harris (HARRIS et STEPHENS, 1988), qui constitue l'un des premiers détecteurs historiques de la littérature. Peu coûteux à extraire, les points restent néanmoins sensibles aux diverses variations mises en lumière plus haut ;

— le détecteur MSER, *Maximally Stable Extremal Regions* (MATAS et al., 2004), qui se base sur un système de seuillage progressif des images. Les caractéristiques extraites supportent bien les transformations affines et changements d'intensité lumineuse ;

— le détecteur FAST, *Feature Accelerated Segment Test* (ROSTEN et DRUMMOND, 2006), dont la détection est rapide et relativement robuste au flou.

Description : L'objectif de l'étape de description est d'associer à chaque point d'intérêt un descripteur qui le caractérise au mieux, et de manière unique. De la même façon que pour l'étape de détection, les algorithmes de description cherchent à décrire les points d'intérêt de manière robuste pour pouvoir être correctement associés sur différentes images. Les plus couramment utilisés sont ici aussi les descriptions SIFT (LOWE, 1999) et SURF (BAY, TUYTELAARS et VAN GOOL, 2006). Cependant, un autre type de description a été popularisé plus récemment : les descripteurs binaires.

Ces descripteurs trouvent un écho particulier dans les applications temps-réel sur des systèmes avec une puissance de calcul limitée. En effet, ces descripteurs présentent divers avantages. D'abord, ils sont rapides à extraire et reposent souvent sur la comparaison de paires de pixels au voisinage d'un point d'intérêt (CALONDER et al., 2010). Ensuite, la comparaison de deux descripteurs se fait par distance de Hamming, qui consiste à compter le nombre d'éléments différents pour une même position dans ces deux descripteurs. Numériquement cette comparaison est très facile à réaliser avec des descripteurs composés d'éléments binaires puisqu'elle revient à compter les résultats d'une opération *ou-exclusif* (XOR) entre chaque bit des descripteurs, qui peut s'implémenter comme une opération processeur bas-niveau. Enfin, ce type de descripteur nécessite moins de place mémoire que des descripteurs basés sur des valeurs à virgule flottante (MIKSIK et MIKOLAJCZYK, 2012).

Les algorithmes générant des descripteurs binaires les plus employés sont :

— les descripteurs BRIEF, *Binary Robust Independent Elementary Feature* (CALONDER et al., 2010), constitués de 256 éléments binaires. Chaque élément est associé à une paire de pixels définie autour du point d'intérêt décrit et indique le résultat de la comparaison entre les intensités de ces deux pixels par une valeur binaire.

— les descripteurs ORB, *Oriented BRIEF* (RUBLEE et al., 2011), qui rendent les descripteurs BRIEF robustes à la rotation.

4. Pour *Scale Invariant Feature Transform.*

* * *

Dans nos travaux, nous avons suivi les choix d'une précédente thèse s'intéressant à la localisation des robots de SoftBank Robotics Europe (WIRBEL, 2014). Le choix de l'auteure s'est porté sur une détection FAST (ROSTEN et DRUMMOND, 2006) couplée à une description ORB (RUBLEE et al., 2011). Dans notre contexte, ce couple détecteur-descripteur est intéressant puisque les caractéristiques FAST sont rapides à extraire et peuvent l'être sur des images floues (ce qui peut arriver lors de certains mouvement du robot), et les caractéristiques ORB présentent les avantages d'une description binaire. Ce choix est notamment conforté par les limites computationnelles de Pepper et la contrainte d'une localisation fonctionnant en temps-réel.

Les caractéristiques FAST s'extraient en effet beaucoup plus rapidement que les caractéristiques SURF du FAB-MAP classique. Mesurée à partir des librairies visuelles OpenCv, l'extraction de caractéristiques FAST est en moyenne 23 fois plus rapide que celle des caractéristiques SURF. De même, le calcul d'un descripteur associé à un point d'intérêt est accéléré par 22 avec une description ORB, en comparaison avec une description SURF.

Ces performances se répercutent au niveau du traitement d'une image, où plusieurs centaines de caractéristiques peuvent être détectées. Cependant, l'utilisation de descripteurs binaires demande une adaptation de l'algorithme de création du vocabulaire.

3.2.2.2 Création d'un vocabulaire ORB

Il n'est pas possible d'utiliser une méthode de regroupement comme les k-moyennes avec des descripteurs binaires. En effet, lors des k-moyennes, un descripteur moyen évolue en fonction des descripteurs d'apprentissage qui lui sont associés. Pour cela, les éléments du descripteur moyen sont actualisés par la moyenne des éléments de tous les descripteurs d'apprentissage dont il est le plus proche. Or, les éléments d'un descripteur binaire ne pouvant prendre pour valeur que 1 ou 0, en réaliser une moyenne n'a pas de sens.

Pour pallier ce problème, l'apprentissage de notre dictionnaire se fait à l'aide d'une méthode de regroupement par vote. Notre approche reprend celle présentée dans (GRANA et al., 2013).

À chaque itération de cette méthode, tous les descripteurs d'apprentissage sont associés au descripteur moyen, ou centroïde, le plus proche par la distance de Hamming. Au lieu d'une moyenne, les éléments des descripteurs associés à un même centroïde vote chacun pour 1 ou 0. Le descripteur moyen évolue ensuite en actualisant la valeur de chacun de ses éléments par le résultat majoritaire du vote.

La Figure 3.5 montre la stabilisation des centroïdes ORB au cours de l'apprentissage. Cet apprentissage est réalisé à partir de caractéristiques FAST-ORB extraites sur 3 000 images de la base de données présentée dans (QUATTONI et TORRALBA, 2009). Cette base de données se révèle pertinente pour nos travaux puisque ses images sont acquises dans divers milieux intérieurs (salle de classes, couloirs, hôpitaux, boutiques, etc.) qui correspondent à des environnements de fonctionnement possibles pour Pepper.

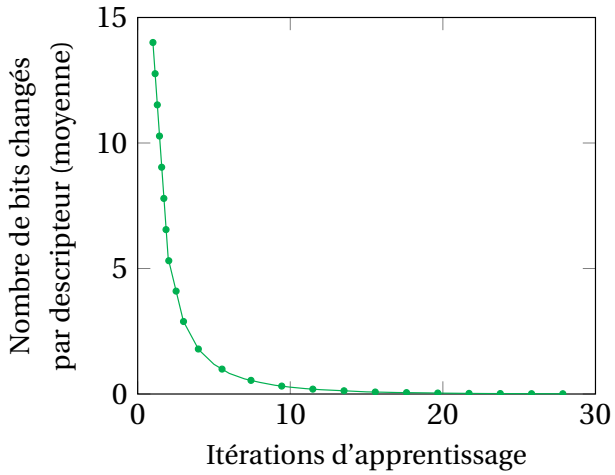


FIGURE 3.5: Stabilisation des centroïdes ORB (256 bits) via méthode de regroupement par vote : 6 000 mots visuels sont appris à partir de 1 892 301 descripteurs ORB d'apprentissage extraits sur 3 000 images de la base de données présentée dans (QUATTONI et TORRALBA, 2009).

Un vocabulaire de 6 000 mots est ainsi créé. Chaque centroïde est décrit par un vecteur de 256 valeurs binaires. La distance de Hamming moyenne entre chaque mot du vocabulaire est de 127,58 bits, avec un écart-type de 30,81 bits.

Malgré les bonnes performances du FAB-MAP dans les tâches de relocalisation visuelle, certains environnements génèrent de nombreuses situations d'ambiguïté perceptuelle, comme l'illustre la Figure 3.6. Ces situations sont inévitables dans ces environnements fortement redondants, et pervertissent la localisation. Une solution pour éviter ces erreurs est d'ajouter une information issue d'un autre capteur : le Wi-Fi.

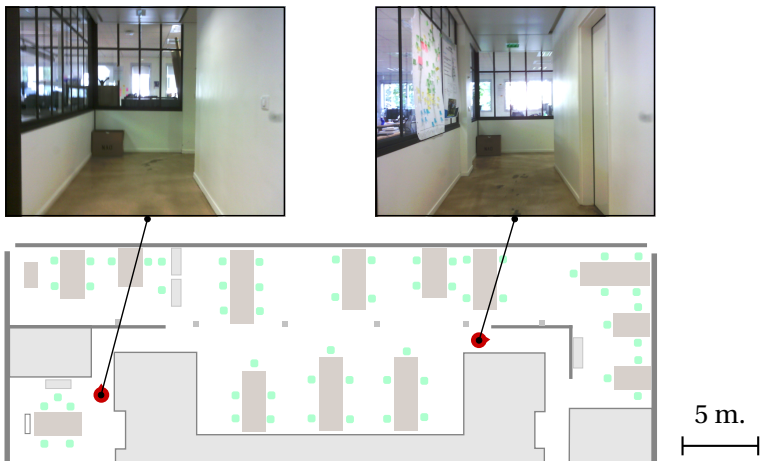


FIGURE 3.6: Erreur de relocalisation visuelle causée par une ambiguïté perceptuelle. Les deux images ont été associées par l'algorithme visuel classique du FAB-MAP bien qu'elles proviennent de lieux distants de 25 m en réalité.

3.3 Fusion Wi-Fi et vision

Certains environnements sont particulièrement redondants visuellement. Par exemple, les couloirs dans un même bâtiment se ressemblent souvent au point d'induire en erreur des humains dans leur localisation. Tout comme les différents étages d'un immeuble qui ne sont souvent identifiables que grâce à des panneaux. Cependant, si ces différents lieux se ressemblent visuellement, ils ne sont pas parcourus par les mêmes signaux Wi-Fi. L'utilisation de signaux Wi-Fi peut alors aider à désambiguïser certains cas de localisation visuelle complexes.

Comme nous l'avons présenté en 2.2.3.1, les méthodes de relocalisation basées sur les signaux Wi-Fi se sont développées ces dernières années. Puisque le Wi-Fi n'est pas soumis au problème de l'ambiguïté perceptuelle, son erreur est bornée (OLIVERA, PLAZA et SERRANO, 2006). La précision de ces méthodes étant plutôt faible (LIU et al., 2012), le Wi-Fi est souvent utilisé dans des approches de localisation multi-capteurs (OCAÑA et al., 2005 ; APARICIO et al., 2008 ; MIROWSKI et al., 2013). La localisation basée sur la vision étant précise mais soumise à l'ambiguïté perceptuelle, elle est donc particulièrement complémentaire d'une localisation basée sur le Wi-Fi.

3.3.1 État-de-l'art : combinaison de données Wi-Fi et visuelles

La localisation basée sur les signaux Wi-Fi possède une moins bonne précision que celle basée sur des signaux visuels. Malgré cela, elle ne tombe jamais dans le piège de l'ambiguïté perceptuelle et sa marge d'erreur est délimitée. Plusieurs stratégies de localisation tirent profit de cette synergie et utilisent des capteurs visuel et Wi-Fi pour mettre en place un système de localisation robuste à moindre coût.

La plupart des travaux se concentrant sur le problème de localisation à partir d'images et d'ondes magnétiques font appel à un filtre à particules pour fusionner ces données comme dans (QUIGLEY et al., 2010 ; SCHWIEGELSHOHN, NICK et GÖTZE, 2013 ; LIU et al., 2017). Cependant, dans ce genre d'approche, les hypothèses ne convergent qu'après suffisamment de déplacements du système.

D'autres techniques emploient les signaux Wi-Fi comme des guides. Elles définissent un ensemble de positions possibles dans la carte à partir des Wi-Fi perçus puis raffinent leur estimation finale grâce à une localisation visuelle dans cet ensemble (RUIZ-RUIZ et al., 2011 ; WERNER, KESSEL et MAROUANE, 2011 ; NOWICKI, 2014 ; JIANG et YIN, 2015). Pour finir, une autre méthode consiste à choisir quel signal est à même de fournir la meilleure estimation pour déterminer la pose courante (BISWAS et VELOSO, 2013).

Cependant, dans ces deux dernières approches, un capteur induit en erreur peut conduire à une estimation finale fausse. Nous proposons ici une nouvelle approche pour fusionner le Wi-Fi et la vision à des fins de relocalisation dans une carte topologique. Ce processus a été présenté dans (NOWAKOWSKI et al., 2017) sous le nom de *fusion précoce*. Basée sur l'algorithme FAB-MAP, cette méthode ne requiert pas de réglage particulier. En comparaison avec les approches de fusion de la littérature, notre méthode cherche un compromis sur l'estimation courante en prenant conjointement en compte les données issues des deux capteurs. Les sous-parties suivantes présentent respectivement com-

ment les données Wi-Fi sont rendues compatibles avec le formalisme du FAB-MAP, puis diverses stratégies de fusion dont notre méthode de fusion précoce.

3.3.2 Inclure le Wi-Fi dans FAB-MAP

Dans la littérature (BISWAS et VELOSO, 2010 ; LIU et al., 2012 ; HE et CHAN, 2016), une signature Wi-Fi - parfois appelée empreinte Wi-Fi - correspond à une liste des points d'accès visibles, chacun caractérisé par son adresse MAC et la force de son signal (RSSI, pour *Received Signal Strength Indication*). La plupart des algorithmes de localisation par Wi-Fi collectent des signatures Wi-Fi lors d'une phase d'exploration, puis génèrent une carte modélisant la distribution des signaux dans l'environnement. De telles approches présentent l'avantage de ne pas avoir à connaître la position des bornes Wi-Fi dans l'environnement pour fonctionner. Ces stratégies sont particulièrement appropriées à la représentation topologique employée par FAB-MAP. Nous présentons dans cette partie notre approche pour adapter les signaux Wi-Fi à l'algorithme FAB-MAP. Cette présentation se divisera en trois étapes reprenant celles de la chaîne de traitement du FAB-MAP visuel classique, comme sur la Figure 3.4 (p. 56).

3.3.2.1 Observation

La première étape du FAB-MAP visuel consiste à transformer une image requête en un vecteur d'éléments binaires : l'apparence visuelle Z_v. Ce vecteur est construit grâce à un vocabulaire de mots visuels, et chacun de ses éléments indique si le mot du vocabulaire qui lui est associé est présent ou absent dans l'image. Pour faire cela à partir de signatures Wi-Fi, une définition de ce qu'est un mot Wi-Fi est nécessaire.

Un choix assez basique consiste à considérer chaque adresse MAC comme un mot Wi-Fi (NOWAKOWSKI et al., 2017). Cette définition du mot Wi-Fi entraîne une particularité par rapport à l'utilisation d'un vocabulaire visuel : le vocabulaire Wi-Fi ne peut être construit que pendant, ou après, une phase d'exploration. En effet, il est inutile de définir un vocabulaire Wi-Fi de manière globale et complète car il n'est pas possible de connaître les points d'accès qu'un robot va rencontrer dans son environnement sans l'explorer. Après une phase d'exploration, les adresses MAC des signaux perçus constituent donc le vocabulaire. Cependant, cette définition ne tire pas d'avantages de l'information portée par l'intensité du signal (RSSI).

Comme le vecteur d'apparence employé par l'algorithme du FAB-MAP est composé d'éléments binaires, une solution pour décrire le RSSI associé à un signal Wi-Fi consiste à discrétiser cette puissance sur plusieurs bits. De cette manière, le vocabulaire Wi-Fi correspond à un répertoire d'adresses MAC rencontrées au cours d'une phase d'exploration, chacune étant associée à plusieurs « mots Wi-Fi » ayant pour rôle d'encoder les différentes puissances perçues.

Plusieurs stratégies de discrétisation peuvent alors être envisagées. Par exemple, dans (WIETRZYKOWSKI, NOWICKI et SKRZYPCZYŃSKI, 2016), les auteurs discrétisent la puissance du signal Wi-Fi sur 10 bits entre $]-110\,\text{dB}, -10\,\text{dB}]$. Chaque bit est associé

à un seuil et est mis à 1 si l'intensité du signal perçu dépasse ce seuil. De cette manière, dans une signature Wi-Fi, chaque signal identifié par une adresse MAC est décrit par un vecteur de 10 éléments binaires. Cette méthode peut être considérée comme une représentation *incrémentale*.

La construction du vecteur d'apparence Z_w associé à cette signature Wi-Fi consiste ensuite à concaténer de manière ordonnée ces vecteurs selon les adresses MAC répertoriées dans le vocabulaire comme illustré en Figure 3.7. À chaque adresse MAC du vocabulaire absente de la signature requête est associé un vecteur composé uniquement de 0. Toutes ces valeurs binaires composant Z_w peuvent, par analogie avec le vecteur d'apparence visuel, être considérées comme nos mots Wi-Fi. Or, rappelons ici que l'algorithme du FAB-MAP repose sur l'apprentissage de corrélations entre les différents mots d'un vocabulaire. Ainsi, bien que cette représentation *incrémentale* doit pouvoir s'accommoder des petites variations temporelles du RSSI, il est possible que les corrélations les plus fortes entre mots du vocabulaire proviennent des mots Wi-Fi associés à la même adresse MAC. En effet, sur la Figure 3.7, il est visible que les mots Wi-Fi associés à des RSSI forts ne s'activent que si les mots précédents ont été activés.

Pour éviter cela, nous introduisons une autre représentation que nous qualifions d'*exclusive*. Un signal associé à une adresse MAC est toujours décrit par plusieurs mots, mais un seul mot peut prendre 1 pour valeur. On parle alors d'encodage *one-hot*. Pour cela, une plage de RSSI utilisable est définie et divisée en b intervalles. Chaque intervalle est associé à une valeur binaire. Lorsque l'intensité du signal perçu se situe entre les limites inférieure et supérieure d'un intervalle, la valeur qui lui est associée passe à 1. Sinon, elle est mise à 0. De cette façon, il est possible de générer un vecteur de b valeurs binaires pour chaque signal Wi-Fi perçu dans une signature (Figure 3.7).

De la même façon que pour la représentation *incrémentale*, le vecteur d'apparence Wi-Fi final Z_w correspond à la concaténation ordonnée de tous les vecteurs binaires décrivant le signal des points d'accès connus dans une signature requête. Avec un vocabulaire de K_w adresses MAC connues, une signature Wi-Fi est transformée en un vecteur d'apparence Z_w de $b \times K_w$ valeurs binaires. Les différences entre représentations *incrémentale* et *exclusive* sont mises en évidence sur la Figure 3.7.

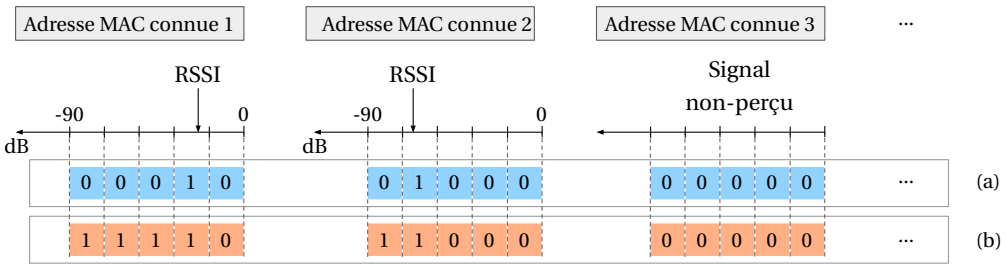


FIGURE 3.7: Utilisation de l'intensité Wi-Fi : différences entre les approches *exclusive* (a), et *incrémentale* (b) présentée dans (WIETRZYKOWSKI, NOWICKI et SKRZYPCZYŃSKI, 2016).

À une position donnée, l'intensité du signal Wi-Fi peut varier significativement sur des échelles de temps très courtes. Pour rendre leurs données d'entrée plus fiables, la

plupart des systèmes de localisation basés sur le Wi-Fi forcent le robot à rester immobile pour calculer la moyenne et l'écart-type du RSSI perçu depuis chaque point d'accès (BISWAS et VELOSO, 2010 ; JIRKU, KUBELKA et REINSTEIN, 2016). Notre cas d'usage ne permet pas d'avoir recours à une telle stratégie, car les comportements de Pepper ne doivent pas être entravés par la tâche de localisation. Pour prendre en compte cette contrainte, nous avons étudié l'impact du nombre d'éléments binaires à utiliser pour décrire chaque signal Wi-Fi. En effet, si l'emploi d'un grand nombre d'éléments binaires servant à encoder le RSSI de chaque signal est supposé accroître la précision du système, une discrétisation trop fine pourrait induire notre système en erreur. Cette question est expérimentalement traitée en partie 3.4.6.

3.3.2.2 Corrélations

Nous venons d'expliquer qu'un vocabulaire Wi-Fi peut être construit en-ligne ou complété après une phase d'exploration. Il est alors possible de construire un arbre de Chow-Liu qui capture les corrélations entre les mots Wi-Fi du vocabulaire, à partir des signatures Wi-Fi collectées. Cependant, pour éviter l'apprentissage de corrélations redondantes, il est nécessaire de s'assurer que les vecteurs d'observation utilisés ne proviennent pas d'un même lieu topologique. Dans notre cas, cela est garanti grâce à l'information odométrique et à l'horodatage des signatures collectées.

3.3.2.3 Normalisation

Dans le monde visuel, échantillonner une observation pour l'étape de normalisation est relativement facile. Pour cela, des images d'entraînement provenant d'environnements d'apprentissage peuvent être utilisées. Dans le monde du Wi-Fi, cette stratégie n'est pas si simple à mettre en œuvre.

Un vocabulaire Wi-Fi est spécifique à un environnement. Pour cette raison, l'utilisation de signatures Wi-Fi collectées dans des environnements d'apprentissage n'a pas beaucoup de sens puisqu'avec le vocabulaire Wi-Fi courant, les vecteurs d'observation créés seraient uniquement composés de zéros. Une solution est de simuler des signatures Wi-Fi virtuelles à partir des données collectées dans l'environnement courant.

En considérant la propagation des signaux Wi-Fi, plusieurs variations dans les signatures collectées peuvent être identifiées. Ainsi, dans un environnement, il est possible qu'un lieu inconnu :

— partage la même signature Wi-Fi qu'un endroit connu,

— ait dans sa signature des adresses MAC jamais vues, et ne faisant donc pas partie du vocabulaire courant,

— mette en évidence de nouvelles combinaisons d'adresses MAC connues.

Nous cherchons donc à simuler des signatures virtuelles en considérant ces variations.

Dans nos expériences, nous constatons que le nombre de points d'accès visibles dans une signature Wi-Fi suit une distribution normale pour un environnement. La moyenne

μ et l'écart-type σ sur le nombre de points d'accès perçus dans les signatures Wi-Fi collectées durant la phase d'exploration sont identifiés. Pour générer aléatoirement une signature Wi-Fi virtuelle, notre méthode consiste premièrement à sélectionner un nombre de points d'accès suivant la distribution normale $\mathcal{N}(\mu,\sigma)$. Chacun de ces points d'accès est ensuite aléatoirement associé à une adresse MAC connue ou inconnue, et un RSSI dans la gamme d'intensité utilisable (dans notre cas $]-90$ dB, 0 dB$]$).

Avec ce formalisme, l'algorithme de localisation FAB-MAP peut fonctionner à partir de signaux Wi-Fi. Cependant, nous montrons dans nos expériences que l'utilisation du Wi-Fi seul conduit à des résultats de localisation moins précis que par apparence visuelle. La prochaine sous-partie introduit donc comment tirer avantage de la synergie entre vision et Wi-Fi grâce à des stratégies de fusion.

3.3.3 Les stratégies de fusion

Nous présentons ici diverses stratégies de fusion pour combiner Wi-Fi et vision à partir d'une localisation basée sur l'algorithme FAB-MAP.

3.3.3.1 Fusions séquentielles

Dans les systèmes multi-capteurs, les fusions séquentielles essaient de tirer parti des différents niveaux de précision des capteurs. Avec l'emploi du FAB-MAP, deux méthodologies peuvent être identifiées :

1. l'approche par guidage Wi-Fi, dans laquelle le Wi-Fi détermine un sous-ensemble de lieux probables sur lesquels est réalisée une localisation visuelle ;
2. l'approche par confirmation Wi-Fi, où le résultat d'une localisation visuelle doit être confirmé par les données Wi-Fi disponibles.

Ces approches utilisent le fait que la localisation Wi-Fi est moins précise que la localisation visuelle, mais ne résulte jamais en des estimations aberrantes. Cependant, le style de normalisation du FAB-MAP ne permet de détecter des fermetures de boucle que sous l'hypothèse où aucune n'a été manquée. En pratique, FAB-MAP détecte donc une unique fermeture de boucle pour chaque requête. Pour que les fusions séquentielles présentées ici fonctionnent, la localisation Wi-Fi doit fournir un ensemble de lieux probables.

Les travaux présentés dans (STUMM, MEI et LACROIX, 2013) proposent de résoudre ce problème à travers une autre approche de normalisation. Avec la normalisation classique de FAB-MAP, une requête q se voit associer une probabilité $p(L_i|\mathcal{Z}^q)$ d'être issu du lieu L_i de la carte, ainsi que $p(L_u|\mathcal{Z}^q)$ indiquant la probabilité que la requête vienne d'un lieu non-cartographié, pour qu'au final :

$$p(L_1|\mathcal{Z}^q) + p(L_2|\mathcal{Z}^q) + \ldots + p(L_u|\mathcal{Z}^q) = 1 \tag{3.7}$$

Dans (STUMM, MEI et LACROIX, 2013), les auteurs considèrent ce problème séparément pour chaque lieu L_i :

$$p(L_i|\mathcal{Z}^q) + p(L_u|\mathcal{Z}^q) = 1 \tag{3.8}$$

Nous employons cette méthode de normalisation dans nos stratégies de fusions séquentielles pour calculer les résultats de localisation basés sur le Wi-Fi. Chaque lieu L_i de la carte est ainsi considéré comme correct (pour construire un sous-ensemble de localisations probables dans le cas de l'approche par guidage Wi-Fi, ou pour valider un résultat visuel dans notre approche par confirmation Wi-Fi), si $p(L_i|\mathcal{Z}^q) > p(L_u|\mathcal{Z}^q)$.

3.3.3.2 Fusions précoce et tardive

Un moyen assez intuitif pour fusionner des résultats de localisation provenant de plusieurs capteurs peut être qualifié de *fusion tardive*. Chaque capteur s est associé à une probabilité $p_s(L_i|\mathcal{Z}_s^k)$ d'être dans un lieu L_i connaissant ses observations $\mathcal{Z}_s^k$. Pour un système composé de deux capteurs s_1 et s_2, le résultat p_{lf} correspondant à un fusion tardive s'écrit comme :

$$p_{lf}(L_i|\mathcal{Z}_{s_1,s_2}^k) = \alpha \times \frac{p_{s_1}(L_i|\mathcal{Z}_{s_1}^k)\, p_{s_2}(L_i|\mathcal{Z}_{s_2}^k)}{p(L_i)} \tag{3.9}$$

où α garantit que $\sum_i p_{lf}(L_i|\mathcal{Z}_{s_1,s_2}^k) = 1$. Rappelons que notre objectif est de résoudre le problème de la localisation globale dans une carte, soit les situations où aucune connaissance *a priori* n'est disponible sur la pose du robot. En considérant ainsi que n_M lieux ont été cartographiés, plus un lieu correspondant au monde inconnu, nous remarquons que pour tout i, $p(L_i) = \frac{1}{n_M+1}$.

Avec la fusion tardive, chaque algorithme de localisation fournit une estimation réalisée à partir des capacités perceptuelles de ses capteurs. Ainsi, un capteur induit en erreur par l'ambiguïté perceptuelle peut clairement pervertir la qualité de l'estimation finale. Notre approche par fusion précoce cherche à éviter ce problème. Nous proposons de fusionner les données avant les étapes de calcul de vraisemblance et de normalisation. Nous notons la probabilité obtenue en sortie de l'algorithme par cette démarche $p_{ef}(L_i|\mathcal{Z}^k)$. L'idée est simple : concaténer les vecteurs d'observation issus des données visuelles Z_v et Wi-Fi Z_w en un seul vecteur Z (Figure 3.8). Le vecteur d'observation obtenu devient alors l'entrée d'un unique processus de localisation. L'avantage de notre fusion précoce est de fournir la meilleure estimation finale, basée sur un compromis entre les différentes sources de données.

3.3.3.3 Quelles corrélations apprendre pour la fusion précoce ?

En mêlant des données Wi-Fi et visuelles dans notre processus de fusion précoce, la question de quelles corrélations apprendre apparaît. En effet, FAB-MAP repose sur

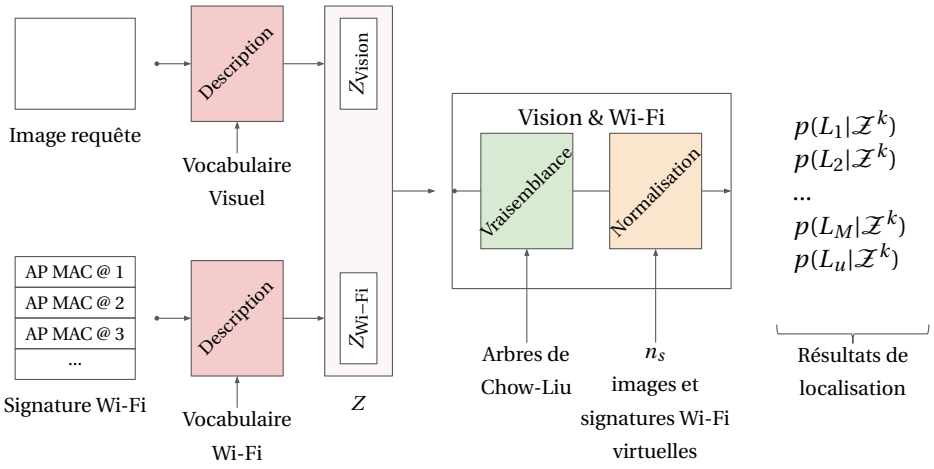


FIGURE 3.8: Principe de la fusion précoce, combinant données visuelles et Wi-Fi.

l'apprentissage des corrélations entre les mots du vocabulaire utilisé. Maintenant que notre dictionnaire contient des mots issus des mondes visuel et Wi-Fi, cette étape n'est plus si triviale.

Nous choisissons de diviser l'apprentissage des corrélations. Au lieu d'apprendre un unique arbre de Chow-Liu résumant les corrélations entre tous les mots (visuels et Wi-Fi), deux arbres capturent respectivement les cooccurrences entre mots visuels d'une part, et mots Wi-Fi de l'autre. Les deux arbres sont passés ensuite comme une unique structure de données à l'algorithme de localisation. Cette division est principalement motivée par deux raisons :

1. Premièrement, apprendre des corrélations entre tous les mots, Wi-Fi et visuels, n'est pas évident. En effet, comme le vocabulaire Wi-Fi est appris à partir de données collectées dans l'environnement, pour apprendre de nouvelles corrélations visuelles depuis les nœuds collectés, tous les mots du vocabulaire visuel doivent avoir été vus au moins une fois pendant la phase d'exploration. En pratique, il n'est pas possible de garantir que cette hypothèse soit vérifiée.
2. Ensuite, l'étape de normalisation encourage notre choix d'apprendre séparément les corrélations entre mots des vocabulaires Wi-Fi et visuel. En effet, simuler le monde inconnu est plus compliqué lorsqu'on considère des observations mixtes. Avec cet apprentissage séparé, l'échantillonnage nécessaire pour la représentation du monde inconnu revient à deux échantillonnages séparés d'observations visuelles et Wi-Fi, puis à une simple concaténation.

* * *

Dans la partie suivante, nos résultats expérimentaux viennent montrer l'avantage de notre processus de fusion précoce dans les tâches de relocalisation. Notre fusion précoce est comparée aux différentes stratégies de fusion présentées précédemment. Pour

garder à l'esprit les différences entre ces stratégies combinant données visuelles et Wi-Fi, la Figure 3.9 rappelle leur fonctionnement.

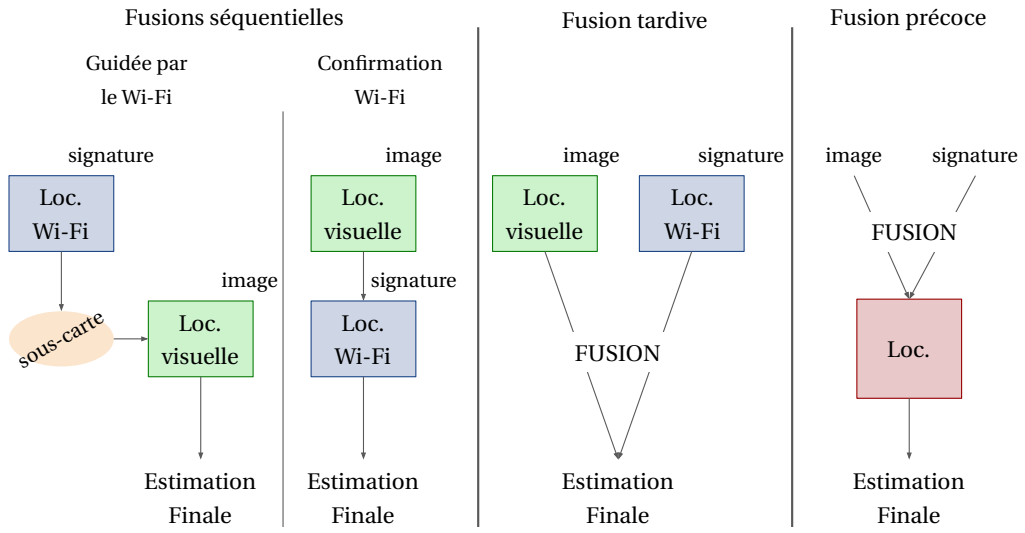


FIGURE 3.9: Différences entre les stratégies de fusion des données visuelles et Wi-Fi à partir d'un algorithme de localisation topologique.

3.4 Expériences et résultats

Notre algorithme a été évalué dans plusieurs environnements depuis des données acquises par différents robots Pepper. Pour présenter nos résultats, cette partie se divisera en trois temps. Dans un premier temps, les conditions expérimentales seront détaillées en 3.4.1. Ensuite, les performances des différents types de relocalisation seront présentées pour chaque environnement de test. Pour finir, nous étudierons l'impact du RSSI et de sa discrétisation dans nos stratégies en 3.4.6.

3.4.1 Détails des acquisitions

3.4.1.1 Conditions expérimentales

Toutes les données utilisées dans nos expériences ont été acquises à partir de robots Pepper. Les chemins parcourus par le robot ont été contrôlés et pilotés par nos soin via un système de téléguidage. Pendant les acquisitions, le robot acquiert de façon autonome une image toutes les 2 secondes et une signature Wi-Fi toutes les 10 secondes. Si la fréquence d'acquisition des images est suffisante dans notre contexte de carte visuelle topologique, celle du Wi-Fi vient d'une contrainte logicielle de sécurité. Pour des raisons de communication Wi-Fi, il n'est pas possible d'augmenter la fréquence d'acquisition des signatures Wi-Fi.

Rappelons que nous ne souhaitons pas que les tâches de localisation soient perceptibles par l'utilisateur et s'exécutent de manière passive. Les contraintes comportementales de Pepper sont ainsi conservées durant nos acquisitions. Certaines images collectées contiennent donc du flou causé par les mouvements du robot car il n'est pas sou-

haitable que ce dernier s'arrête pour des prises de vues. De plus, pour simuler une navigation humaine, le robot regarde toujours dans la direction vers laquelle il se dirige. Lors de lignes droites, les images ne sont donc pas prises dans le sens perpendiculaire de la direction, pourtant plus discriminant, mais dans celui de la direction suivie (Figure 3.10).

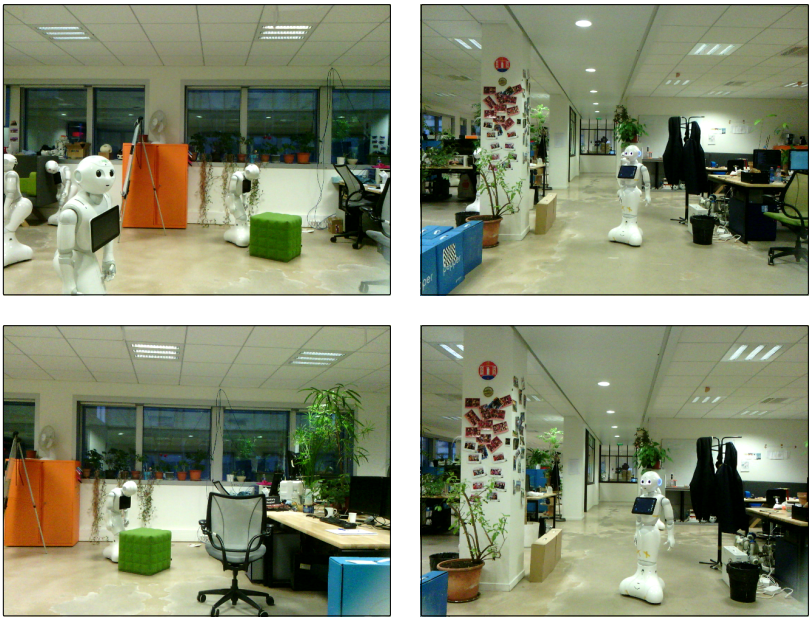


FIGURE 3.10: Direction des prises de vue durant les acquisitions. En général, les images utilisées dans des problématiques d'apparence visuelle sont collectées dans le sens perpendiculaire de la direction, comme ici, les deux images à gauche. De cette manière, les champs de vue se recouvrent moins et des informations plus discriminantes permettent de différencier les deux images. Les contraintes d'utilisation de Pepper ne permettent pas ce genre de comportement puisque, pour paraître plus naturel, le robot doit regarder où il se dirige lorsqu'il se déplace (images de droite).

Afin de vérifier la généricité des différents types de fusions, divers environnements ont été utilisés. Notons que chacun de ces environnements a été utilisé sans aucun équipement spécifique, comme le positionnement de marqueurs visuels ou l'ajout de bornes Wi-Fi. Nos différents jeux de données ont été collectés dans :

— les *open-spaces* d'un immeuble de bureaux sur deux étages (3.4.2),

— les couloirs et les classes d'un collège, à trois différents étages (3.4.3),

— les différentes pièces d'un appartement privé (3.4.4).

Chacun de ces environnements possède des particularités. Cependant, comme Pepper est amené à être manipulé par des personnes non-expertes, le même schéma d'acquisition a été conservé dans ces différentes situations.

Nos acquisitions constituent une collection de 10 120 images et 6 110 signatures Wi-Fi. La distance totale parcourue pendant ces expériences fait 6,4 km de long. Une attention particulière a été portée à l'aspect réaliste des scénarios d'acquisition. Se retrouvent ainsi dans nos données des occultations, des interactions avec des utilisateurs, des variations de luminosité brusques ou évoluant en fonction de l'heure d'acquisition, etc...

3.4.1.2 Annotations : exploration et localisation

Notre formalisme définit un nœud topologique comme l'association d'une image et d'une signature Wi-Fi. En pratique, les images sont collectées plus rapidement que les signatures Wi-Fi. Pour associer les images acquises à une signature Wi-Fi, deux scénarios sont choisis selon que le système soit en phase de cartographie (exploration d'apprentissage) ou de localisation (phase de test) :

1. Pendant la phase initiale d'exploration, les images sont associées avec la signature Wi-Fi la plus proche temporellement, grâce à l'horodatage. Cette approche est possible puisque les résultats de l'exploration peuvent être traités une fois l'acquisition terminée.
2. Pendant la phase test de localisation, les images sont liées à la dernière signature Wi-Fi acquise. En effet, en fonctionnement les requêtes sont traitées de façon incrémentale.

La position de toutes les images est manuellement annotée. Ces annotations constituent notre vérité-terrain sur la pose des nœuds topologiques collectés. Pour chaque environnement, les différentes acquisitions sont divisées en deux groupes : 40% sont utilisées pour la création de la carte et 60% constituent nos requêtes de relocalisation. Notons que le vocabulaire Wi-Fi ainsi que les corrélations entre ses mots, sont appris sur l'ensemble des acquisitions utilisées pour la création de carte.

Chaque requête correspond à un endroit visité. Ainsi la tâche de localisation globale est réalisée sur un environnement entièrement cartographié.

3.4.1.3 Métriques d'évaluation

Chaque requête test est associée au lieu dans la carte dont le score renvoyé par l'algorithme de localisation est le plus haut :

$$L_{max} = \underset{L_i \in M \cup \tilde{M}}{\operatorname{argmax}} \; p(L_i | \mathcal{Z}^k) \tag{3.10}$$

Afin d'évaluer la précision des différentes approches, la distance euclidienne est calculée entre la position annotée (x, y) de la requête, et celle du lieu associé L_{max} dans la carte. Dans certains cas d'erreur aberrante comme la confusion entre deux étages ou deux environnements différents, cette distance est mise à l'infini. Ce choix s'explique par le fait que de telles erreurs nuisent beaucoup plus à notre système, et sont en pratique plus compliquées à récupérer. Finalement, lorsque $L_{max} = L_u$, correspondant à un lieu non-visité, notre évaluation considère la requête comme rejetée.

Les résultats obtenus sont détaillés dans les parties suivantes. Pour les obtenir, nous utilisons l'algorithme FAB-MAP 2.0 (CUMMINS et NEWMAN, 2011), et adaptons l'intégration *open-source* de l'algorithme présentée dans (GLOVER et al., 2012) à notre usage. L'ensemble de nos requêtes est localisé à partir des différentes méthodes présentées :

— employant seulement des données visuelles,

— employant seulement des données Wi-Fi,

— fusionnant ces données suivant les processus de fusion précoce et tardive (3.3.3.2),

— fusionnant ces données suivant les processus de fusion séquentiels (3.3.3.1).

L'appréciation des performances de relocalisation fournies par chacune de ces approches se fera grâce à deux types de courbes, représentant respectivement :

— le *taux de localisations correctes* : défini comme le pourcentage de requêtes dont la distance entre la position estimée et la vraie position est inférieure à une distance donnée : $T_{correctes}(d) = T(erreur < d)$.

— le *taux de localisations fausses* : défini comme le pourcentage de requêtes ayant conduit à des estimations dont la distance à la vérité-terrain est supérieure à une distance donnée : $T_{mauvaises}(d) = T(erreur > d)$.

Ainsi, chacune de ces courbes met en évidence une information différente. Si la première fournit une information sur la précision des algorithmes, la seconde insiste sur ses erreurs. Les figures montrant les taux de localisations fausses permettent également de déduire les taux de rejet de chaque approche, à partir de l'ordonnée à l'origine. Le taux de rejet correspond alors à :

$$T_{rejet} = 1 - T_{mauvaises}(d = 0) \tag{3.11}$$

Nous présentons ci-dessous les performances des diverses stratégies de localisation dans nos environnements de test. Dans un premier temps, remarquons que la localisation Wi-Fi n'utilisera pas l'information associée aux puissances des signaux perçus. Seule la présence ou l'absence des points d'accès connus dans les requête sera prise en compte. L'utilisation du RSSI sera étudiée en 3.4.6.

3.4.2 L'apport du Wi-Fi dans un environnement visuellement redondant

La plus grande partie de nos acquisitions est réalisée dans les bureaux de SoftBank Robotics Europe. Une particularité de cet environnement de test est que ces bureaux sont en grande partie composés d'*open-spaces*. Cette facette est importante, car en considérant la propagation des signaux Wi-Fi, les signatures collectées dans cet environnement sont plus compliquées à distinguer. L'absence de cloisons dans l'environnement limite en effet les variations dans les signatures Wi-Fi. Ces conditions de tests correspondent néanmoins à certains usages possibles de Pepper dans de grands environnements intérieurs comme des centres commerciaux, des gares ou des aéroports.

Les acquisitions sont réalisées à partir de différents robots Pepper, s'étalent sur plusieurs mois et sont faites à deux étages du bâtiment. Des exemples de chemins parcourus par Pepper durant ces acquisitions sont présentés sur la Figure 3.11.

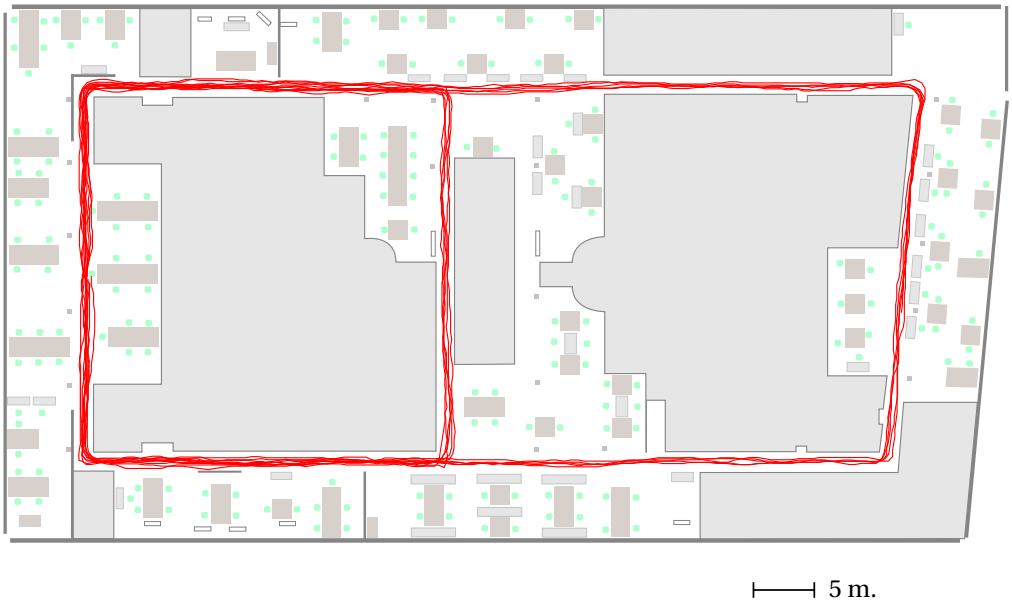


FIGURE 3.11: Exemples de chemins parcourus sur un étage de SoftBank Robotics Europe (tracés en rouge).

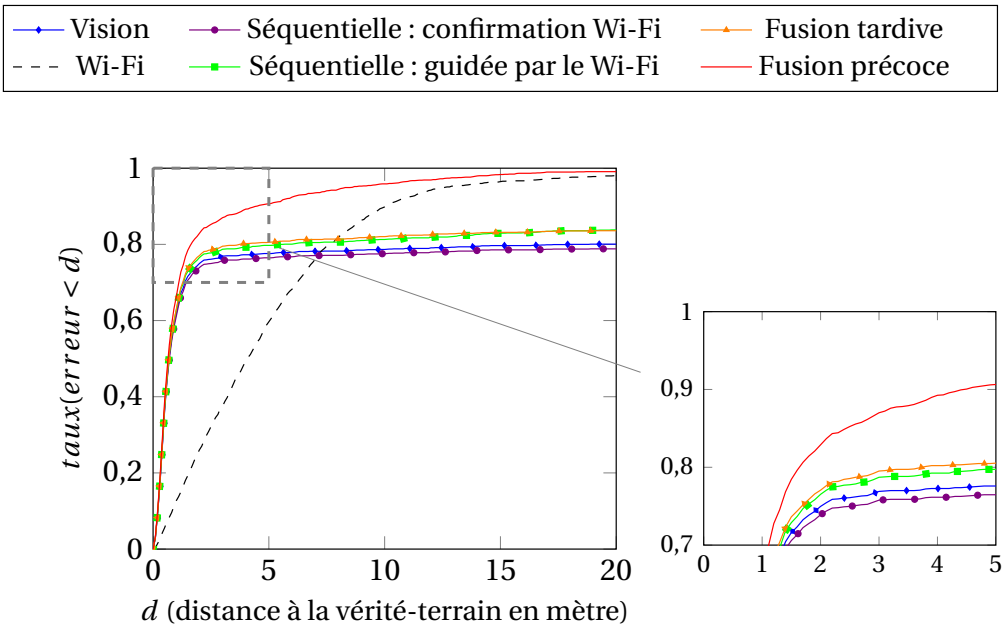
3.4.2.1 Évaluation des différentes méthodes

Dans de telles conditions de tests, il est attendu que l'utilisation des données Wi-Fi permette de diminuer les erreurs de localisation aberrantes. En effet, l'aspect particulièrement dynamique et répétitif de l'*open-space* contribue à faire des points d'accès des repères plus fiables que les points d'intérêt visuels. Ces attentes se vérifient dans les résultats affichés sur la Figure 3.12.

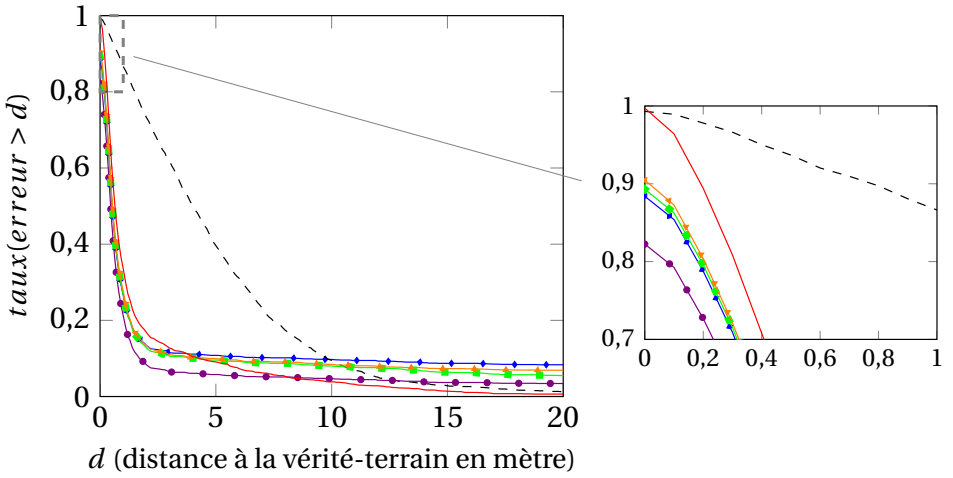
La localisation basée sur le Wi-Fi ne produit que 1,3% d'estimations aberrantes, positionnées à plus de 20 m de leur vraie position, alors que l'utilisation du FAB-MAP visuel classique génère 8,3% d'estimations aberrantes. Cependant, la localisation visuelle résulte en des résultats plus précis avec 75,0% des requêtes localisées à moins de 2 m de la vérité-terrain.

L'intérêt de combiner Wi-Fi et vision pour les tâches de localisation est confirmé par les résultats présentés sur la Figure 3.12. Toutes les approches de fusion présentent des taux de localisations fausses inférieurs à ceux du FAB-MAP visuel. Concernant les taux de localisations correctes, notre fusion précoce surpasse clairement les autres approches avec 90,6% des estimations localisées à moins de 5 m de leur vraie position, contre 77,6% pour le processus de localisation n'utilisant que des données visuelles. Sur la Figure 3.12b, le taux de rejet de l'approche séquentielle par confirmation Wi-Fi permet de réduire le taux de localisations fausses pour ce type de fusion. Cependant, le prix à payer est élevé puisque le robot échoue à estimer sa position dans presque un cinquième des requêtes (17,8%, *cf.* Figure 3.12b pour $d = 0$). Les performances de chaque approche sont extraites et reportées dans le Tableau 3.1.

Des exemples d'images requêtes compliquées, prises dans l'environnement, sont vi-



(a) Taux de localisations correctes.



(b) Taux de localisations fausses.

FIGURE 3.12: Taux de bonnes et mauvaises relocalisations dans les *open-spaces* de SoftBank Robotics Europe, calculés à partir des différentes stratégies de localisation.

sibles sur la Figure 3.13. Les images choisies mettent en lumière certaines situations typiques et complexes. Toutes nos acquisitions étant réalisées pendant les heures d'ouverture des bureaux, l'environnement revêt un aspect dynamique. Entre deux acquisitions, des objets peuvent être déplacés (chaises, porte-manteaux, robots, etc.), les gens peuvent avoir changé de place et les conditions d'éclairage peuvent avoir significativement évolué. De plus, aucune consigne n'ayant été donnée aux employés, ces derniers pouvaient interagir avec le robot ou l'embêter à leur guise durant nos acquisitions. La plupart de nos acquisitions contiennent ainsi des occultations causées par des interac-

	Taux de localisations correctes (%), distance à la vérité-terrain inférieure à				Taux de localisations fausses (%), distance à la vérité-terrain supérieure à			
	2m	5m	10m	20m	2m	5m	10m	20m
Vision	75,0	77,6	78,7	80,1	13,4	10,8	9,7	8,3
Wi-Fi	26,0	59,7	89,8	98,0	73,3	39,6	9,5	1,3
Guidage Wi-Fi	76,5	79,7	81,3	83,8	12,8	9,6	8,0	5,5
Confirmation Wi-Fi	73,9	76,5	77,5	78,8	**8,3**	**5,8**	4,7	3,4
Fusion tardive	77,1	80,5	82,0	83,6	13,4	9,9	8,4	6,9
Fusion précoce	**82,9**	**90,6**	**95,9**	**99,1**	16,8	9,1	**3,9**	**0,6**

TABLEAU 3.1: Comparaison du FAB-MAP visuel classique avec les processus de localisation utilisant des données Wi-Fi seules, et mélangeant données Wi-Fi et visuelles, dans les *open-spaces* de SoftBank Robotics Europe.

tions ou des personnes passant devant le robot (sur la Figure 3.13, les requêtes (a) et (b)). Dans tous les cas d'occultations visuelles, la fusion précoce a soit produit le même résultat que la localisation visuelle, soit fourni une meilleure estimation de position. Un autre problème fréquent dans cet environnement est illustré par les requêtes (c) et (d) de la Figure 3.13 : l'ambiguïté perceptuelle. Dans le cas de la requête (c), la présence d'une plante d'intérieur a généré de nombreux mots visuels qui induisent en erreur la localisation visuelle. Une plante similaire placée à un étage différent dupe ainsi l'algorithme. L'algorithme de fusion précoce ne tombe pas dans ce piège grâce à l'utilisation des données Wi-Fi. Un cas particulièrement dur d'ambiguïté perceptuelle est illustré par la requête (d) sur la Figure 3.13. Dans cette situation, la fusion précoce fournit également une estimation acceptable (à 0,70 m de sa vraie position) contrairement à la vision (à 37,87 m de sa vraie position).

Le processus de fusion précoce se montre ainsi capable de localiser 99,1% des requêtes à moins de 20 m de leur vraie position et avec la meilleure précision. De plus, il est intéressant de remarquer que malgré les taux de rejets plus élevés de toutes les autres approches, ces dernières n'arrivent pas à rattraper certaines erreurs aberrantes. La fusion précoce est ainsi l'approche qui produit le moins d'erreurs aberrantes.

3.4.2.2 Confusion d'étages

La Figure 3.12 ne permet pas de visualiser un type d'erreur très problématique : la confusion entre deux étages. Cette confusion peut se révéler très handicapante dans des stratégies où le résultat de la relocalisation est utilisé pour charger une sous-carte de l'environnement. Néanmoins, ce type d'erreur est commun dans des environnements où tous les étages sont visuellement similaires, comme dans des hôtels ou des centres de conférence. Les bureaux de SoftBank Robotics Europe présentent également des similarités entre chaque étage. La Figure 3.14 indique le pourcentage de requêtes localisées au mauvais étage pour chaque approche de localisation testée.

Une fois encore, les résultats de la Figure 3.14 démontrent l'intérêt d'utiliser les données Wi-Fi. L'algorithme de localisation basé uniquement sur des données visuelles se trompe le plus souvent d'étage. En comparaison avec la localisation basée uniquement

FIGURE 3.13: Exemples de requêtes visuelles difficiles collectées dans les locaux de SoftBank Robotics Europe. Chaque ligne montre une requête (à gauche) et la référence qui lui a été associée dans la carte d'après la localisation visuelle pure (au centre) et la fusion précoce (à droite). Les distances entre l'estimation et la vraie position de la requête sont indiquées sous chaque image référence. Une image grise indique que la localisation visuelle a rejeté la requête (b) et l'a considérée comme provenant d'un lieu inconnu.

sur le Wi-Fi, la vision a conduit à 10 fois plus de confusion entre étages. Utiliser le Wi-Fi en complément de la vision permet donc de diminuer ce type d'erreur. Cependant, seule l'approche par fusion précoce conduit à un taux d'erreur plus bas que celui de la localisation basée uniquement sur les données Wi-Fi. Ces résultats sont particulièrement intéressants puisqu'ils semblent indiquer que le processus de fusion précoce résulte en le meilleur compromis pour l'estimation de position que les autres techniques de fusion.

3.4.2.3 Performances long-terme

Les résultats précédents démontrent l'intérêt de combiner le Wi-Fi et la vision pour des tâches de relocalisation dans cet environnement. Ces bonnes performances encouragent à mener des tests dans des conditions plus difficiles. Nous avons ainsi testé la

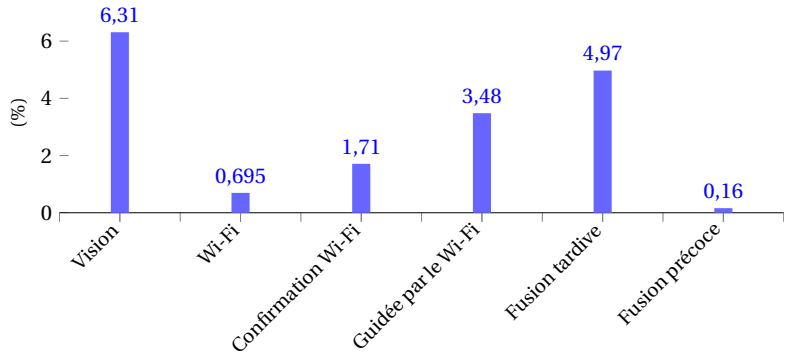


FIGURE 3.14: Pourcentages de requêtes localisées au mauvais étage pour chacune des approches de localisation présentées.

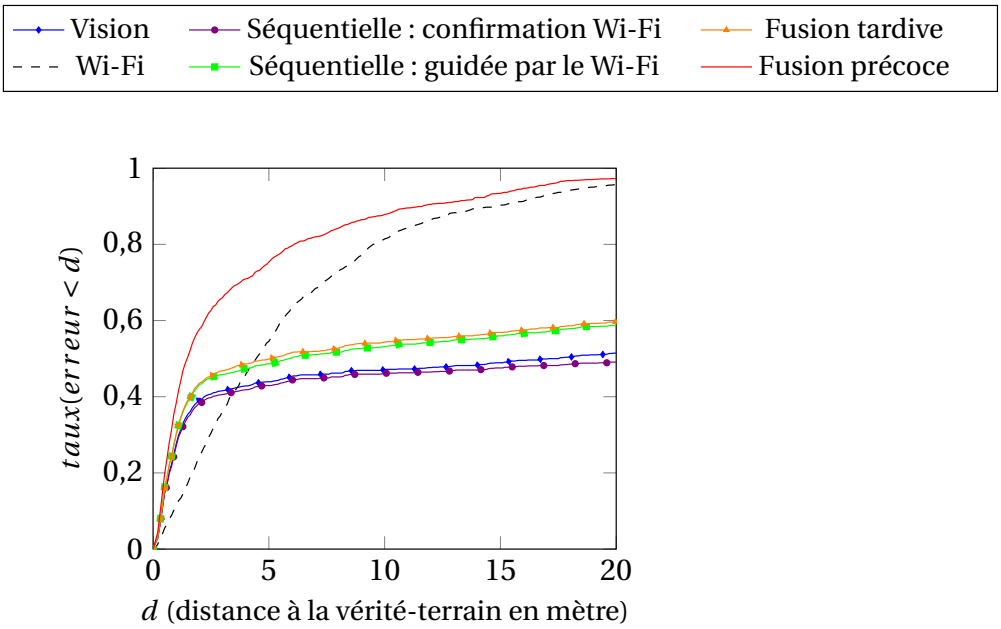
robustesse de ces algorithmes dans le temps. Pour cela, la phase d'exploration initiale et celle de test sont espacées de sept mois. Les résultats obtenus sont présentés sur la Figure 3.15. Ils révèlent que la localisation visuelle est plus détériorée que la localisation basée sur le Wi-Fi. Cette détérioration s'explique facilement par tous les changements visuels ayant eu lieu en sept mois. Ces modifications impactent fortement les résultats de la localisation visuelle et de la localisation par fusion séquentielle. À l'inverse, les changements apparaissant dans les signatures Wi-Fi, tels que des points d'accès mobiles ayant disparu ou apparu ne sont pas suffisamment significatifs pour dégrader les performances de la localisation Wi-Fi.

Les valeurs du Tableau 3.2 montrent que la fusion précoce via FAB-MAP renforcée par Wi-Fi donne le meilleur compromis entre données Wi-Fi et visuelles.

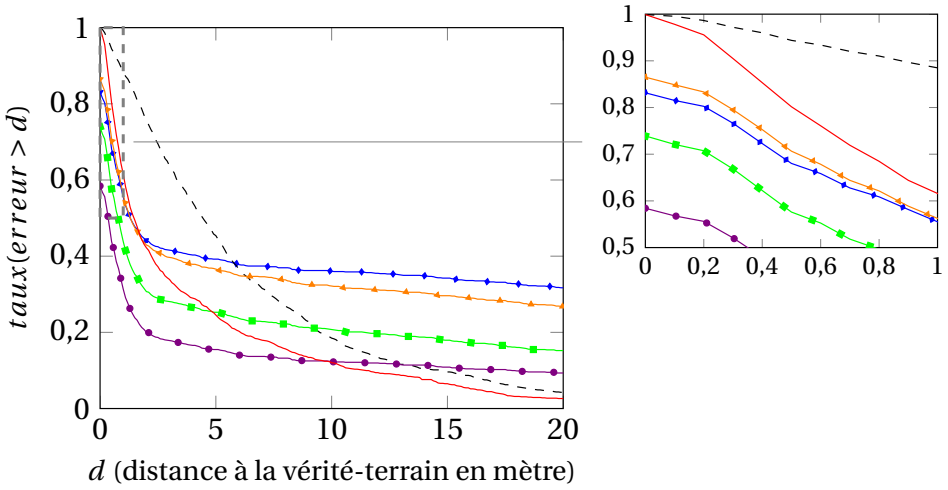
	Taux de localisations correctes (%), distance à la vérité-terrain inférieure à				Taux de localisations fausses (%), distance à la vérité-terrain supérieure à			
	2m	5m	10m	20m	2m	5m	10m	20m
Vision	39,2	43,9	47,1	51,4	43,9	39,2	36,0	31,7
Wi-Fi	24,2	54,5	81,4	95,7	75,7	45,4	18,5	4,2
Guidage Wi-Fi	43,2	48,6	53,2	58,7	30,7	25,3	20,7	15,2
Confirmation Wi-Fi	38,4	42,9	46,1	49,1	**20,0**	**15,5**	12,3	9,3
Fusion tardive	43,6	49,9	54,3	59,7	42,9	36,6	32,2	26,8
Fusion précoce	**57,7**	**75,4**	**87,9**	**97,3**	42,2	24,5	**12,0**	**2,6**

TABLEAU 3.2: Comparaison du FAB-MAP visuel classique avec les processus de localisation utilisant des données Wi-Fi seules, et mélangeant données Wi-Fi et visuelles, dans une utilisation long-terme : les phases d'exploration et de tests sont espacées de sept mois.

Jusqu'ici, l'emploi de données Wi-Fi a permis d'améliorer l'estimation de position. Cependant, cela peut n'être vrai que dans cet environnement de test avec ses particularités : sa redondance visuelle, son aspect dynamique, ses grandes dimensions ainsi que la présence de suffisamment de points d'accès assurant une bonne couverture Wi-Fi. Pour confirmer ces performances dans des situations différentes, deux autres environnements de test ont été utilisés.



(a) Taux de localisations correctes.



(b) Taux de localisations fausses.

FIGURE 3.15: Taux de bonnes et mauvaises relocalisations dans les *open-spaces* de SoftBank Robotics Europe calculés à partir des différentes stratégies de localisation dans une utilisation long-terme.

3.4.3 Localisation avec une mauvaise couverture Wi-Fi

Une partie de nos tests est réalisée dans un collège. Cet environnement a été choisi pour plusieurs raisons. Pour commencer, la localisation visuelle y produit de bonnes estimations. Ensuite, la couverture Wi-Fi n'y est pas optimale. Seulement 25 points d'accès y sont visibles contre 544 dans les locaux de SoftBank Robotics Europe, pour des environnements de tailles comparables. 876 m sont parcourus sur trois étages différents par un robot Pepper. Les acquisitions réalisées s'étalent sur une journée entière et passent à

travers des couloirs et des salles de classe. Quelques exemples d'images collectées dans cet environnement sont donnés en Figure 3.16.

FIGURE 3.16: Exemples d'images acquises dans le collège test.

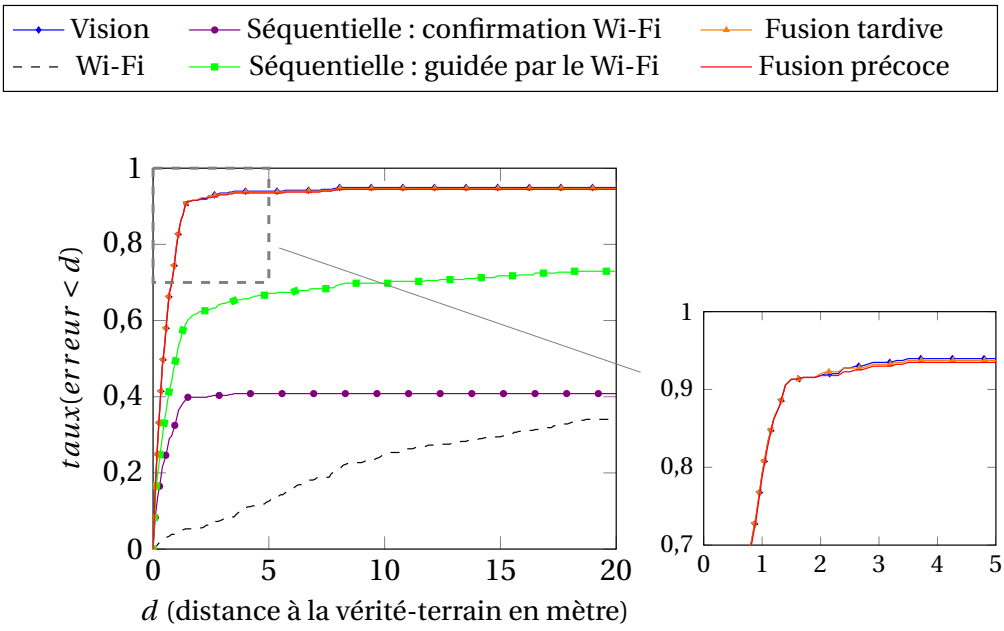
Les courbes présentées en Figure 3.17 montrent que les performances de la localisation Wi-Fi sont bien en deçà de celles de la localisation visuelle. Dans cet environnement, les bonnes performances du FAB-MAP visuel ne sont pas si surprenantes puisque la base de données sur laquelle est appris notre vocabulaire visuel (QUATTONI et TORRALBA, 2009) contient beaucoup d'images acquises dans des écoles.

Cette expérience a pour intérêt de montrer que malgré la faible précision de la localisation Wi-Fi, les processus de fusion précoce et tardive n'ont pas dégradé les bonnes performances de la localisation visuelle de manière significative. En effet, les taux de localisations correctes et fausses présentés en Figure 3.17a et Figure 3.17b sont très similaires.

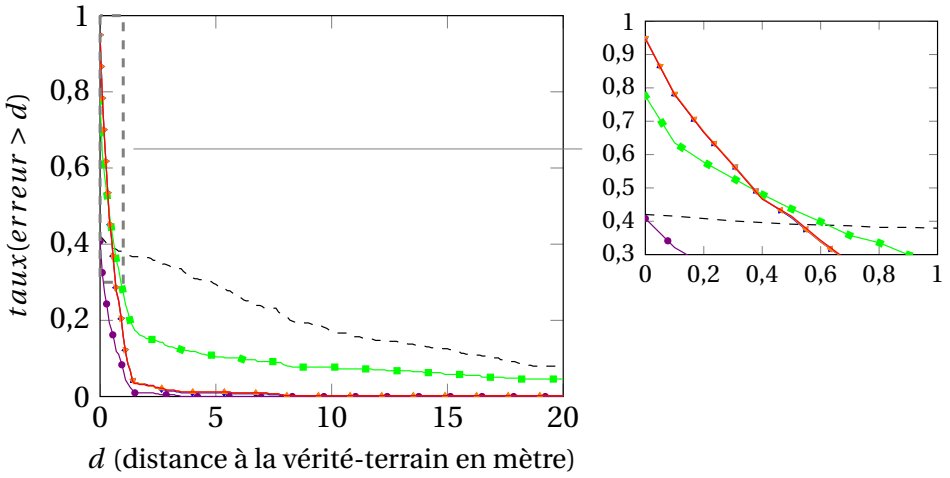
	Taux de localisations correctes (%), distance à la vérité-terrain inférieure à				Taux de localisations fausses (%), distance à la vérité-terrain supérieure à			
	2m	5m	10m	20m	2m	5m	10m	20m
Vision	91,8	**94,0**	**94,9**	**94,9**	3,0	1,0	**0**	**0**
Wi-Fi	5,6	12,6	24,9	34,1	36,5	29,5	17,1	7,8
Guidage Wi-Fi	62,3	67,1	69,8	72,4	15,2	10,4	7,7	4,6
Confirmation Wi-Fi	39,9	40,8	40,8	40,8	**1,0**	**0**	**0**	**0**
Fusion tardive	**92,0**	93,7	94,7	94,7	2,9	1,2	0,2	0,2
Fusion précoce	91,8	93,5	94,4	94,4	2,9	1,2	0,2	0,2

TABLEAU 3.3: Comparaison du FAB-MAP visuel classique avec les processus de localisation utilisant des données Wi-Fi seules, et mélangeant données Wi-Fi et visuelles, dans les couloirs et classes d'un collège avec peu de points d'accès Wi-Fi visibles.

Contrairement aux fusions précoce et tardive, les approches séquentielles souffrent plus des mauvaises performances du Wi-Fi. L'approche séquentielle par confirmation Wi-Fi devient ainsi très restrictive et rejette 59,2% des requêtes, ce qui dégrade fortement



(a) Taux de localisations correctes.



(b) Taux de localisations fausses.

FIGURE 3.17: Taux de bonnes et mauvaises relocalisations dans les couloirs et classes d'un collège avec peu de points d'accès Wi-Fi visibles, calculés à partir des différentes stratégies de localisation.

ses taux de localisations correctes (Tableau 3.3). Par exemple, si la vision localise 94,0% des requêtes à moins de 5 m de leur vraie position, l'approche par confirmation Wi-Fi n'atteint qu'un score de 40,8% pour la même distance.

Dans un environnement où la couverture Wi-Fi n'est pas optimale et où la vision produit de bonnes relocalisations, la fusion précoce permet ainsi de ne pas dégrader significativement les performances du système.

3.4.4 Localisation dans un appartement privé

Le dernier environnement testé correspond à un appartement privé. Tester notre approche de fusion précoce dans cet environnement permet de vérifier que l'utilisation du Wi-Fi dans un plus petit espace ne dégrade pas les performances de localisation visuelle.

Les acquisitions sont réalisées à un seul étage et la distance maximale entre toutes les poses enregistrées du robot est de 16 m. A titre de comparaison, dans les locaux de SoftBank Robotics Europe la distance maximale entre deux poses étaient de 76 m, et de 65 m dans le collège. Pour ces dimensions, la couverture Wi-Fi de l'appartement est acceptable, avec 76 points d'accès visibles. Plusieurs acquisitions et chemins sont explorés, pour une distance totale parcourue de 440 m. Quelques exemples des chemins parcourus par le robot sont affichés sur la Figure 3.18.

FIGURE 3.18: Chemins parcourus dans un appartement privé lors de nos acquisitions (tracés en rouge).

Bien que ce type d'environnement soit moins affecté par le problème de l'ambiguïté perceptuelle, les tâches de relocalisation visuelle y restent difficiles. L'environnement contient des sources d'erreurs potentielles telles que de fortes variations d'intensité lumineuse ou un champ de vue réduit, dû aux dimensions plus réduites des pièces. Des images acquises dans l'appartement exploré sont données en Figure 3.19 pour illustrer ces problèmes.

Dans cet environnement peu propice à l'ambiguïté perceptuelle, la localisation basée sur la vision renvoie de bons résultats, comme le montre la Figure 3.20. Grâce aux murs des différentes pièces, les signatures Wi-Fi sont plus différentes que dans les environnements de tests précédents. Ainsi, en comparaison avec les performances de la localisation Wi-Fi dans les autres environnements testés, la localisation Wi-Fi est plus précise ici. Nous extrayons dans le Tableau 3.4 quelques valeurs des Figures 3.20a et 3.20b pour

FIGURE 3.19: Exemples d'images acquises dans l'appartement test.

une comparaison plus détaillée.

	Taux de localisations correctes (%), distance à la vérité-terrain inférieure à				Taux de localisations fausses (%), distance à la vérité-terrain supérieure à			
	2m	4m	6m	8m	2m	4m	6m	8m
Vision	91,8	92,9	93,6	94,4	4,6	3,5	2,8	2,0
Wi-Fi	35,3	70,6	88,5	97,4	64,7	29,4	11,5	2,6
Guidage Wi-Fi	81,2	82,6	84,8	86,6	6,4	5,0	2,9	1,1
Confirmation Wi-Fi	73,0	73,7	74,4	74,8	**2,0**	**1,3**	**0,6**	**0,2**
Fusion tardive	92,9	94,3	94,9	95,6	4,0	2,6	2,0	1,3
Fusion précoce	**94,7**	**96,7**	**97,8**	**98,2**	4,2	2,2	1,1	0,7

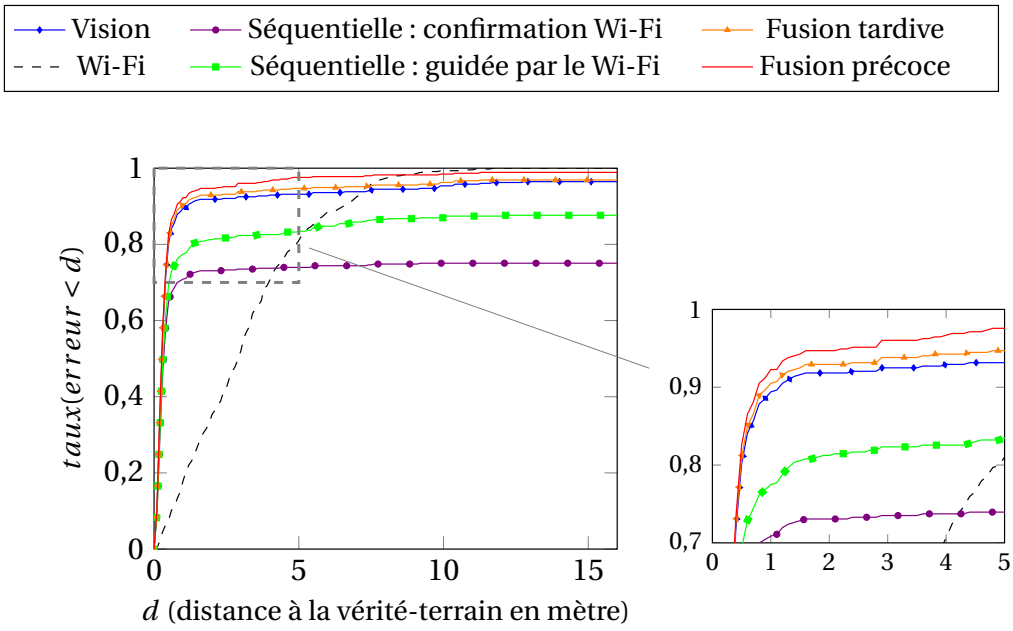
TABLEAU 3.4: Comparaison du FAB-MAP visuel classique avec les processus de localisation utilisant des données Wi-Fi seules, et combinant données Wi-Fi et visuelles dans un appartement privé.

La fusion précoce fournit de nouveau les taux de localisations correctes les plus élevés. Le processus de fusion précoce a également aidé à réduire le nombre d'erreurs. Même si l'approche séquentielle par confirmation Wi-Fi possède des taux de localisations fausses plus bas (2,0% des requêtes placées à plus de 2 m de leur vraie position contre 4,2% pour la fusion précoce), elle rejette 25,0% des requêtes ce qui n'est pas acceptable.

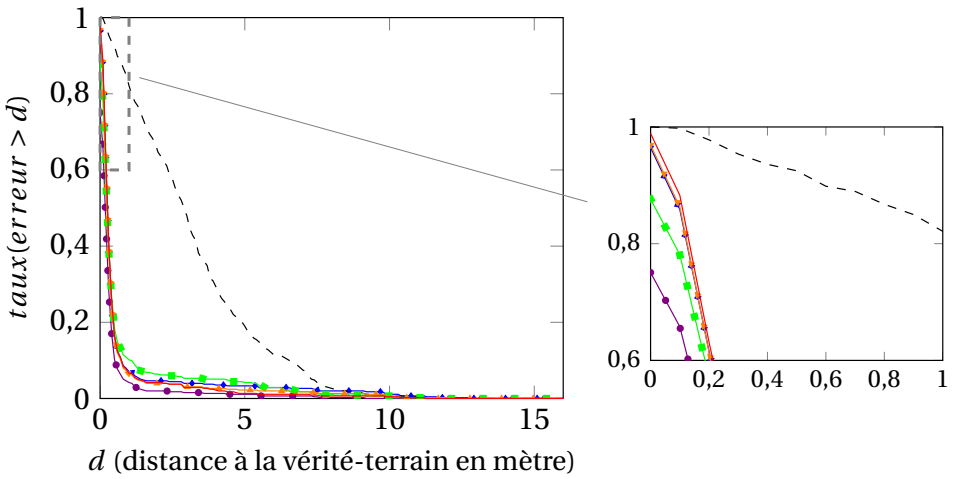
3.4.5 Temps de calcul

Pour finir, les temps de calcul des différentes approches ont été relevés sur le robot. Les robots Pepper sont équipés de processeur quatre cœurs Atom E3845 avec une fréquence d'horloge CPU de 1,91 GHz.

Les temps de traitement d'une requête sont assez similaires et sont de 117 ms ± 59 ms pour tous les algorithmes étudiés employant des données visuelles.



(a) Taux de localisations correctes.



(b) Taux de localisations fausses.

FIGURE 3.20: Taux de bonnes et mauvaises localisation dans un appartement privé calculés à partir des différentes stratégies de localisation.

L'approche de localisation basée uniquement sur le Wi-Fi traite une requête en 0,31 ms ± 0,095 ms. Cette différence s'explique par le temps nécessaire à la création du vecteur d'apparence visuelle. Générer ce vecteur d'apparence nécessite en effet 99% du temps de traitement d'une requête, dans toutes les techniques employant la vision.

3.4.6 L'utilisation du RSSI

Jusqu'ici nous nous sommes intéressés aux performances des différentes stratégies de localisation, sans prendre en compte l'intensité des signaux Wi-Fi perçus. Comme mentionné en partie 3.3.2.1, l'information de puissance portée par le RSSI peut être uti-

lisée pour améliorer la précision de la localisation basée sur le Wi-Fi. Cela est fait grâce à une sorte de discrétisation de l'intensité du signal, compatible avec le formalisme employé par FAB-MAP. Pour chaque point d'accès connu, le RSSI est décrit par plusieurs mots binaires. Nous traitons ici du nombre optimal de mots à utiliser pour décrire l'intensité d'un signal perçu.

3.4.6.1 Représentations exclusive et incrémentale

Les performances de la localisation Wi-Fi sont étudiées dans l'environnement de test correspondant aux bureaux de SoftBank Robotics Europe. Ce choix s'explique par la couverture Wi-Fi particulièrement bonne de cet environnement, avec beaucoup de points d'accès visibles.

Les deux types de représentations, *incrémentale* et *exclusive*, mis en évidence en partie 3.3.2.1, sont étudiés. Remarquons au passage que la plage d'intensité utilisable correspondant à $]-90\text{ dB}, 0\text{ dB}]$, est due à des limitations matérielle et logicielle de Pepper. Le Tableau 3.5 présente les résultats de la localisation basée uniquement sur le Wi-Fi. Différents nombre b de mots binaires encodant le RSSI sont employés. La première ligne du Tableau 3.5, correspondant à $b = 1$, est utilisée comme référence, et reflète le cas où l'information utilisée se résume à la présence ou l'absence des points d'accès connus dans les signatures traitées.

Nombre de mots Wi-Fi utilisés pour encoder le RSSI	Taux de localisations correctes (%), distance à la vérité-terrain inférieure à				Taux de rejet (%)
	2m	5m	10m	20m	
1	26,0	59,7	89,8	98,0	0,7
4	31,3	68,1	92,6	99,1	0,2
	26 ,1	61,5	90,4	98,9	0,4
8	**32,0**	**70,9**	92,9	99,1	0.6
	27 ,4	67,5	92,9	99,6	0,4
12	29,8	68,9	90,9	97,8	1,0
	28 ,0	68,8	**93 ,3**	99,5	0,4
16	31,2	70,1	91,4	97,5	0,6
	26,6	66,5	91,2	**99,7**	0,1
20	29,7	65,8	88,0	96,4	1,2
	28,0	65,7	91,5	**99,7**	0,1

TABLEAU 3.5: Importance du nombre de mots Wi-Fi choisi pour décrire le RSSI associé au signal de chaque point d'accès. Les résultats calculés à partir de la représentation exclusive sont affichés sur un fond gris et comparés aux résultats obtenus par la représentation incrémentale proposée dans (WIETRZYKOWSKI, NOWICKI et SKRZYPCZYŃSKI, 2016).

Les valeurs présentées dans le Tableau 3.5 montrent que prendre en compte l'infor-

mation portée par le RSSI permet d'améliorer la précision de la localisation Wi-Fi. Les deux approches présentées en 3.3.2.1 sont testées. Les meilleures performances de localisations sont obtenues avec un encodage du RSSI sur 8 mots. Avec la représentation exclusive, 32,0% des requêtes sont localisées à moins de 5 m de leur vraie position lorsque $b = 8$, contre 26,0% sans l'utilisation du RSSI ($b = 1$). Les approches exclusive et incrémentale fournissent néanmoins des résultats comparables, avec un petit avantage pour la représentation exclusive pour des distances à la vérité-terrain inférieures à 5 m.

3.4.6.2 Performances dans les environnements de tests

L'utilisation du RSSI est testée dans chaque environnements de test, pour les stratégies de localisation employant les signaux Wi-Fi. Nous présentons les résultats obtenus sous la forme de tableaux de gains de performances, en considérant une distance à la vérité-terrain $d = 5$ m. Ces gains sont calculés par rapport à la démarche qui ne prend pas en compte le RSSI ; soit celle qui considère seulement la présence ou l'absence des points d'accès dans la signature. Les valeurs de ces gains sont reportées :

— dans le Tableau 3.6, pour les performances de localisation dans les bureaux de SoftBank Robotics Europe ;

— dans le Tableau 3.7, pour les performances de localisation dans le collège exploré ;

— dans le Tableau 3.8, pour les performances de localisation dans l'appartement privé de test.

	Gains sur les taux de localisations correctes à 5 m (%), pour différents b					Gains sur les taux de localisations fausses à 5 m (%), pour différents b				
	$b = 4$	$b = 8$	$b = 12$	$b = 16$	$b = 20$	$b = 4$	$b = 8$	$b = 12$	$b = 16$	$b = 20$
Wi-Fi	+8,6	+11,2	+9,3	+10,4	+6,1	-8,6	-11,2	-9,3	-10,4	-6,1
Guidage Wi-Fi	-0,3	+0,3	+0,1	-2,3	-3,6	+0,3	-2,5	-3,2	-1,3	-1,3
Confirmation Wi-Fi	+0,4	-3,2	-5,7	-7,6	-11,8	-0,2	-3,0	-4,1	-3,6	-4,3
Fusion tardive	-0,1	+0,1	-0,7	-0,4	-0,1	+0,2	-0,3	+0,4	+0,3	-0,1
Fusion précoce	+0,9	+2,9	+1,1	+1,0	+1,2	-0,9	-3,0	-1,2	-1,2	-1,6

TABLEAU 3.6: Impact de la discrétisation Wi-Fi *exclusive* dans les *open-spaces* de SoftBank Robotics Europe. Les gains de performance sont calculés pour différents nombres b de mots Wi-Fi encodant le RSSI des signaux perçus.

Pour les expériences réalisées dans les bureaux de SoftBank Robotics Europe (Tableau 3.6) et dans le collège (Tableau 3.7), l'utilisation du RSSI a amélioré les performances de localisation. Les meilleures performances sont atteintes pour une discrétisation de la puissance Wi-Fi sur $b = 8$ mots. Cependant, cette amélioration est surtout significative pour la localisation basée uniquement sur les signaux Wi-Fi. Les différentes approches de fusion sont moins impactées par l'utilisation du RSSI, et certaines sont même dégradées. Les fusions séquentielles deviennent plus restrictives et les gains de performances associés au taux de localisations correctes sont négatifs.

	Gains sur les taux de localisations correctes à 5 m (%), pour différents b					Gains sur les taux de localisations fausses à 5 m (%), pour différents b				
	$b=4$	$b=8$	$b=12$	$b=16$	$b=20$	$b=4$	$b=8$	$b=12$	$b=16$	$b=20$
Wi-Fi	+8,9	+15,7	+9,4	+13,0	+11,8	-8,9	-15,7	-9,4	-13,0	-11,8
Guidage Wi-Fi	+4,1	-6,3	-9,2	-7,2	-13,5	-4,1	+1,4	+3,6	+1,4	+3,6
Confirmation Wi-Fi	+8,2	-4,3	-10,0	-4,1	-10,0	+0,5	-0,2	0	0	-0,2
Fusion tardive	+0,5	+0,2	0	+0,2	+0,7	-0,2	-0,5	0	-0,2	-0,7
Fusion précoce	-0,2	+0,2	-0,5	-0,2	0	0	-0,7	0	-0,2	-0,2

TABLEAU 3.7: Impact de la discrétisation Wi-Fi *exclusive* dans un collège. Les gains de performance sont calculés pour différents nombres b de mots Wi-Fi encodant le RSSI des signaux perçus.

	Gains sur les taux de localisations correctes à 5 m (%), pour différents b					Gains sur les taux de localisations fausses à 5 m (%), pour différents b				
	$b=4$	$b=8$	$b=12$	$b=16$	$b=20$	$b=4$	$b=8$	$b=12$	$b=16$	$b=20$
Wi-Fi	+3,1	-2,9	+4,0	-0,2	-1,1	-3,1	+2,9	-4,0	+0,2	-1,1
Guidage Wi-Fi	-6,2	-5,7	+0,2	-8,4	-9,7	-0,4	0	+0,7	+2,9	+3,5
Confirmation Wi-Fi	-5,5	-7,7	-2,0	-13,0	-16,3	+0,7	+0,2	+0,4	+0,7	0
Fusion tardive	+0,7	+0,2	-0,2	+0,4	+0,4	-0,2	0	+0,2	-0,4	0
Fusion précoce	+0,7	+0,2	-0,4	+1,1	-0,9	-0,2	0	+0,2	-0,7	+0,7

TABLEAU 3.8: Impact de la discrétisation Wi-Fi *exclusive* dans un appartement privé. Les gains de performance sont calculés pour différents nombres b de mots Wi-Fi encodant le RSSI des signaux perçus.

Concernant l'expérience réalisée en appartement, l'impact du RSSI est moins favorable (Tableau 3.8). Dans cet environnement, les pièces fermées permettent de créer suffisamment de variations des signaux Wi-Fi pour que la localisation Wi-Fi puisse reposer sur une stratégie ne prenant compte que la présence ou l'absence des points d'accès connus dans une signature. L'utilisation du RSSI ajoute du bruit de quantification à notre système, puisque les signaux Wi-Fi sont perturbés par les nombreux obstacles qu'ils traversent, et les gains mesurés ne soulignent pas d'amélioration franche.

Considérer la puissance des signaux Wi-Fi perçus n'a pas permis d'améliorer significativement les performances des stratégies de fusion présentées. De plus, remarquons que la complexité algorithmique de la solution est augmentée car l'utilisation de plusieurs mots Wi-Fi pour décrire la puissance de chaque signal Wi-Fi connu conduit à la gestion d'un vocabulaire plus grand.

3.5 Conclusion

Ce chapitre a introduit notre processus de fusion précoce dont l'objectif est de combiner des données Wi-Fi et visuelles pour résoudre des tâches de localisation en milieu intérieur.

La méthode présentée a été testée sur différents jeux de données collectés à partir de robots Pepper, selon des schémas d'acquisition qui suivent des cas d'usage réalistes pour ces robots. Au total, 6,4 km ont été parcourus pendant ces acquisitions dans trois envi-

ronnements différents : les *open-spaces* d'un bureau, les salles et couloirs d'un collège et les différentes pièces d'un appartement privé. La fusion précoce a permis d'améliorer les performances de relocalisation visuelle. Par exemple, dans un environnement où l'emploi de la vision était confronté à divers problèmes comme ceux de l'ambiguïté perceptuelle ou du dynamisme des scènes observées, l'amélioration de la relocalisation est significative : 90,6 % des requêtes sont localisées à moins de 5 m de leur vraie position, contre 77,6 % pour une localisation basée uniquement sur la vision.

De plus, comparée à d'autres stratégies de fusion, notre fusion précoce a généré les meilleurs résultats. En effet, dans toutes nos expériences la fusion précoce a amélioré les performances visuelles sans jamais les dégrader de manière significative, et cela même dans les environnements où les signaux Wi-Fi apportaient très peu d'information. Les résultats présentés dans nos tests montrent ainsi que la fusion précoce constitue le meilleur compromis pour fusionner des données Wi-Fi et visuelles dans un contexte de relocalisation topologique en milieu intérieur.

Chapitre 4

SLAM métrique basse fréquence combinant stéréo-vision et odométrie

Sommaire

Après nous être intéressés au problème de la relocalisation dans une carte connue, nous abordons dans ce chapitre le problème de la construction d'une carte métrique cohérente. Pour cela, une méthode de SLAM est développée, et se base sur la caméra stéréo de Pepper et son odométrie.

Dans un premier temps, les approches les plus courantes de SLAM visuel métrique sont présentées en partie 4.1. Leurs limitations sur notre plate-forme sont mises en évidence pour illustrer l'intérêt de notre solution, qui intègre l'odométrie du robot au problème. La partie 4.2 explique comment l'odométrie est prise en compte par notre système. Notre solution complète, intégrée et testée sur le robot est détaillée en 4.3. Les résultats expérimentaux obtenus sont ensuite présentés et discutés en partie 4.4.

4.1 Localisation métrique visuelle

4.1.1 Géométrie de l'image

4.1.1.1 Le modèle sténopé

Une image est la projection d'un environnement en trois dimensions sur un plan. Une manière simple de représenter cette transformation est le modèle sténopé. Ce dernier est qualifié de central car le modèle de la caméra associée à une image comporte un unique centre optique. La transformation d'un point 3D vers un pixel de l'image est alors définie au niveau de l'intersection entre le plan image de la caméra et la droite reliant son centre optique à ce point 3D.

Dans ce modèle, un repère est associé à la caméra C, dont l'origine est placée au niveau de son centre optique. Par convention, les axes $\boldsymbol{X_C}$ et $\boldsymbol{Y_C}$ sont définis dans un plan parallèle au plan image avec $\boldsymbol{X_C}$ orienté vers la droite et $\boldsymbol{Y_C}$ vers le bas. L'axe $\boldsymbol{Z_C}$ est aligné sur l'axe optique, dans la direction où regarde la caméra (Figure 4.1). Le plan image s'exprime lui dans un repère pixellique. L'origine de ce repère se situe au niveau du coin supérieur gauche de l'image, avec $\boldsymbol{u}$ orienté vers la droite et $\boldsymbol{v}$ vers le bas.

La projection m d'un point 3D M dans l'image, comme décrit en Figure 4.1, correspond alors à une opération mathématique simple lorsque les paramètres intrinsèques et extrinsèques de la caméra C sont connus.

Paramètres intrinsèques : Les paramètres intrinsèques d'une caméra sont des paramètres optiques, spécifiques à chaque caméra. Dans le modèle sténopé, ces paramètres correspondent aux distances focales (horizontale f_x et verticale f_y), ainsi qu'aux coordonnées (u_0, v_0) du centre optique projeté dans l'image. Un autre paramètre s_{uv} permet de gérer la non-orthogonalité des lignes et colonnes des cellules photosensibles du capteur utilisé. Ces paramètres sont souvent déterminés par une phase de calibration de la caméra et définissent la matrice de paramètres intrinsèques :

$$K = \begin{pmatrix} f_x & s_{uv} & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{4.1}$$

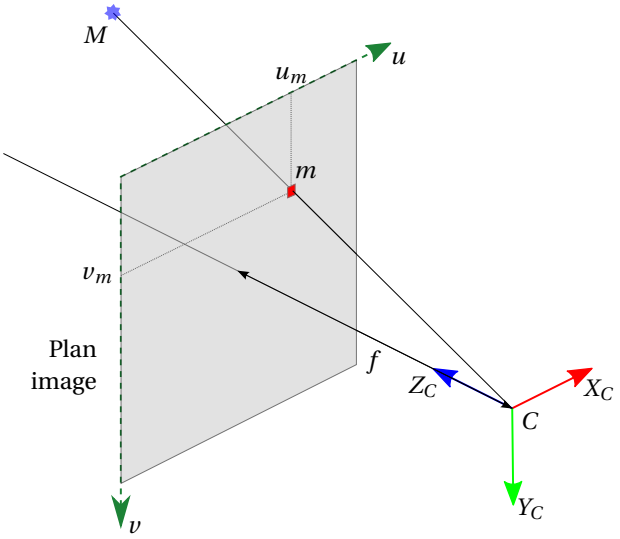


FIGURE 4.1: Projection d'un point 3D dans une image suivant le modèle sténopé.

D'autres modèles de caméras plus complexes existent pour gérer notamment les phénomènes de distorsion (STURM et al., 2011) ou modéliser des caméras catadioptriques (GEYER et DANIILIDIS, 2000).

La projection m d'un point 3D M, exprimé en coordonnées homogènes dans le repère de la caméra C, peut alors être déterminée à partir de K par la transformation :

$$\underbrace{\begin{pmatrix} su_m \\ sv_m \\ s \end{pmatrix}}_{s\tilde{m}} = K \underbrace{\begin{pmatrix} X_{M|C} \\ Y_{M|C} \\ Z_{M|C} \\ 1 \end{pmatrix}}_{\tilde{M|_C}} \tag{4.2}$$

Paramètres extrinsèques : Cependant, cette projection n'est correcte que si le point M considéré est exprimé dans le repère de la caméra C. L'équation de projection générale d'un point 3D doit donc prendre en compte la pose de la caméra dans le repère de référence utilisé. Cette pose dans l'espace correspond aux paramètres extrinsèques de la caméra, et se caractérise par une rotation $R_{3\times3}$ et une translation (t_X, t_Y, t_Z). L'équation (4.2) peut alors s'étendre pour donner :

$$\underbrace{\begin{pmatrix} su_m \\ sv_m \\ s \end{pmatrix}}_{s\tilde{m}} = \underbrace{\begin{pmatrix} f_x & s_{uv} & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{K} \begin{pmatrix} & R_{3\times3} & & \begin{matrix} t_X \\ t_Y \\ t_Z \end{matrix} \\ 0 & 0 & 0 & 1 \end{pmatrix} \underbrace{\begin{pmatrix} X_M \\ Y_M \\ Z_M \\ 1 \end{pmatrix}}_{\tilde{M}} \tag{4.3}$$

Dans la suite de ce manuscrit, le terme de matrice de projection fera référence au produit matriciel des matrices de paramètres intrinsèques et extrinsèques. Cette matrice sera identifiée par le symbole C, et constituera un raccourci pour désigner la caméra.

Notons également que lorsque nous parlerons de paramètres à optimiser dans C, ces paramètres seront toujours les paramètres extrinsèques de la caméra.

* * *

Avec le modèle sténopé, toute coordonnée (u_i, v_i) de l'image définit donc une droite de projection dans le monde réel. Cependant, aucune information de profondeur n'y est associée, et le point réel projeté sur l'image peut se situer n'importe où sur cette droite. Pour retrouver la profondeur, une solution classique est l'emploi d'un système multi-vues.

4.1.1.2 Reconstruction d'amers 3D

L'emploi de systèmes multi-vues permet de reconstruire la position 3D de points réels à partir de leurs projections dans au moins deux images. Lorsque deux caméras observent un même point 3D, ses projections dans les deux images définissent deux droites qui s'intersectent au niveau de la position du point 3D. Ainsi, il est facile de déterminer les coordonnées 3D d'un point si ce dernier est observé par au moins deux caméras dont les paramètres intrinsèques et extrinsèques sont connus.

Reconstruction à partir de deux vues : En pratique, bien qu'un point 3D M produise deux observations en m_{I_1} et m_{I_2} sur deux images différentes, les droites issues de m_{I_1} et m_{I_2} ne s'intersectent pas. Ce phénomène s'explique par divers facteurs, comme le bruit de détection mais aussi la discrétisation pixellique. Dans de telles situations, la technique du point du milieu est employée pour estimer les coordonnées d'un point 3D. Le point reconstruit est défini alors par le milieu du segment orthogonal aux droites issues de m_{I_1} et m_{I_2} (Figure 4.2).

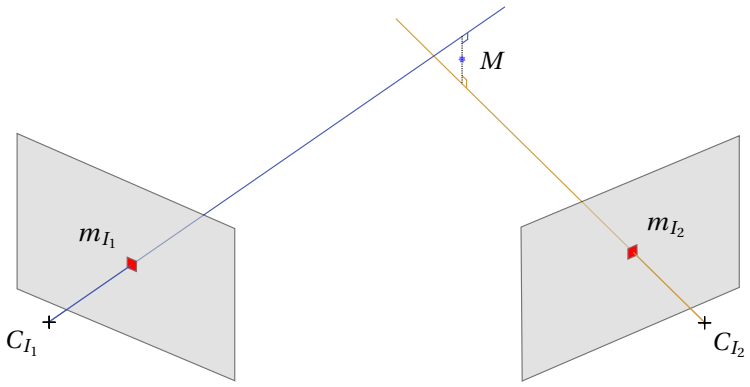


FIGURE 4.2: Triangulation par la méthode du point du milieu.

Certaines approches, cherchent également à augmenter la précision de leur reconstruction avec des méthodes sous pixelliques, dont les différents intérêts dans les tâches de SLAM ont été présentés dans (ROYER, 2006).

Association des projections : La reconstruction d'un point 3D passe donc par l'association d'au moins deux de ses projections dans différentes images dont les paramètres sont connus. Dans un premier temps, le système doit pouvoir associer ses diverses projections entre elles.

La géométrie épipolaire est souvent utilisée dans les tâches de reconstruction de points 3D. Son principe repose sur le fait que les centres optiques de deux caméras forment avec un point 3D, projeté sur leurs deux images I_1 et I_2, un plan. Ce plan intersecte chaque image en deux droites. Ainsi, comme le montre la Figure 4.3, à une position pixellique m_{I_1} dans I_1 correspond une droite d dans I_2.

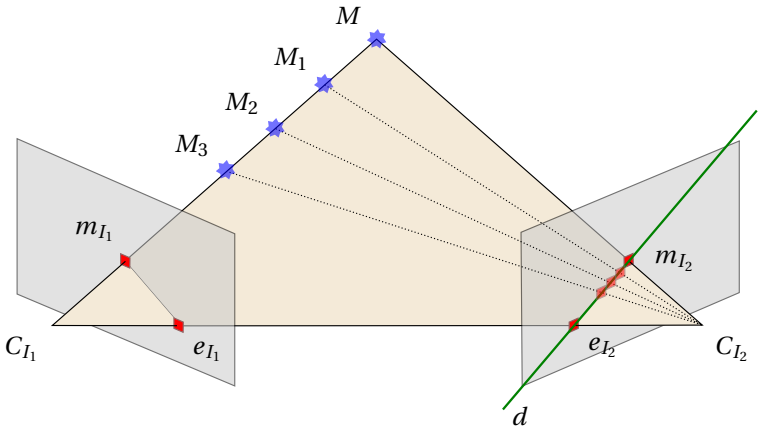


FIGURE 4.3: Géométrie épipolaire : les projections du centre optique de chaque image dans l'autre image sont appelés les épipôles e_{I_1} et e_{I_2}. Les épipôles et les centres optiques des deux images sont tous alignés sur une même droite. À toute droite de projection issue de l'image I_1 correspond une droite dans le plan image de I_2 passant par e_{I_2}.

En connaissant la transformation géométrique entre les caméras C_{I_1} et C_{I_2}, l'appariement entre les points des images I_1 et I_2 peut être accéléré en réduisant la zone de recherche. Il est également possible d'utiliser cette contrainte pour rejeter des appariements aberrants ou optimiser les paramètres d'un système.

Nuage de points multi-vues : La reconstruction d'amers 3D peut être étendue à des systèmes composés de plus de deux caméras. Dans les dispositifs composés de nombreuses caméras, la gestion d'un nuage de points 3D peut se formuler comme un problème d'optimisation. Le critère d'optimisation le plus employé dans la littérature est la minimisation de l'erreur de reprojection. Comme l'illustre la Figure 4.4, l'erreur de reprojection correspond à la distance entre la projection supposée d'un point 3D et la projection observée dans l'image.

L'optimisation cherche alors à minimiser cette erreur en modifiant les coordonnées 3D du point :

$$M^* = \underset{M}{\operatorname{argmin}} \sum_{C \in \mathscr{C}} e_v(M,C)^T \, \Omega_v \, e_v(M,C) \tag{4.4}$$

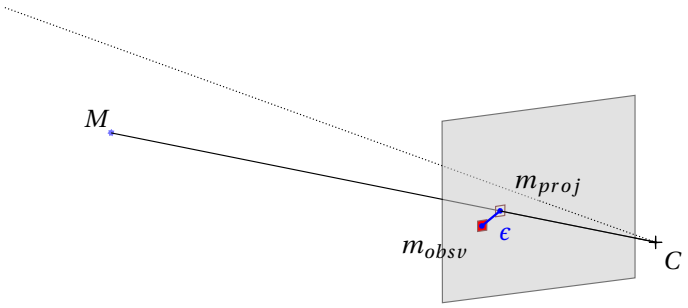


FIGURE 4.4: Erreur de reprojection : correspond à la distance entre l'observation m_{obs} et la projection m_{proj} du point M dans l'image.

où M^* est la position optimisée du point 3D M, et $\mathscr{C}$ l'ensemble des caméras observant M. La fonction d'erreur e_v retourne un vecteur en deux dimensions qui correspond à l'erreur de reprojection sur $\boldsymbol{u}$ et $\boldsymbol{v}$. Ω_v est la matrice d'information de taille 2×2 associée à la projection de M dans C. Cette matrice joue un rôle de pondération, et se révèle pertinente par exemple lorsque les images des caméras considérées sont de résolutions différentes.

* * *

Dans notre contexte, générer un nuage de points 3D nous permet de représenter métriquement l'environnement. Cette structure prend le rôle de carte, et va permettre de déterminer les paramètres extrinsèques de nouvelles caméras qui observent ce nuage.

4.1.1.3 Localisation d'une caméra

Localiser une caméra dans l'espace revient à estimer ses paramètres extrinsèques. Diverses méthodes existent. Certaines cherchent à estimer la transformation directe entre deux images comme dans (NISTER, 2003).

D'autres méthodes passent par l'utilisation et la maintenance d'un nuage de points 3D. L'estimation de la pose d'une caméra peut alors se faire en connaissant la projection de plusieurs points du nuage dans l'image (LEPETIT, MORENO-NOGUER et FUA, 2009 ; HESCH et ROUMELIOTIS, 2011 ; PENATE-SANCHEZ, ANDRADE-CETTO et MORENO-NOGUER, 2013).

Ce problème peut s'interpréter comme un problème d'optimisation. Connaissant les paramètres intrinsèques de la caméra et des associations entre des points 3D et leurs projections 2D, l'optimisation cherche la pose de la caméra qui minimise l'erreur de reprojection, soit :

$$C^* = \underset{C}{\operatorname{argmin}} \sum_{M \in \mathscr{M}} e_v(M,C)^T \, \Omega_v \, e_v(M,C) \tag{4.5}$$

où C^* est la pose optimisée de la caméra C et $\mathscr{M}$, les points 3D observés dans l'image de cette dernière.

Les performances de telles approches se retrouvent cependant fortement dégradées par la présence de données aberrantes. Pour éviter cela, l'optimisation présentée en équation (4.5), est souvent couplée à l'algorithme RANSAC [1].

Proposé dans (FISCHLER et BOLLES, 1981), l'algorithme RANSAC cherche à résoudre des problèmes de calcul de pose. L'intérêt de cette méthode est sa robustesse, face à des entrées pouvant contenir des données aberrantes. Au cours de son fonctionnement, l'algorithme RANSAC classe les données d'entrée en deux : les *inliers*, qui satisfont un critère (par exemple une erreur de reprojection inférieure à un seuil) ; et les *outliers* qui ne satisfont pas ce critère. L'algorithme RANSAC cherche itérativement la solution à un problème via la boucle suivante :

1. recherche d'une solution à partir d'un tirage aléatoire dans les données d'entrée ;
2. classification des données d'entrée en *inliers* et *outliers*, suivant la solution générée ;
3. lorsque le nombre d'*inliers* est le meilleur obtenu, la solution courante est définie comme la meilleure solution ;
4. sortie de boucle si le nombre d'*inliers* est supérieur à un seuil fixé, ou si le nombre d'itérations maximal est atteint.

De cette manière l'algorithme RANSAC est performant dans les situations où la majorité des données d'entrée concordent avec la vraie solution du problème. Les tirages successifs du RANSAC ont ainsi plus de chance de tomber rapidement sur la solution optimale. Pour ces raisons, l'algorithme RANSAC est particulièrement adapté au contexte de la localisation visuelle.

* * *

Les méthodes présentées jusqu'ici permettent : l'estimation de la position de points 3D à partir d'au moins deux caméras ; et l'estimation de la pose d'une caméra à partir de points 3D. En considérant ces deux problèmes simultanément, on retrouve le formalisme du SLAM. L'estimation des points 3D correspond alors à la phase de cartographie, et celle de la pose d'une nouvelle caméra à la phase de localisation. Étant donné le faible coût des caméras mais aussi leur présence sur la plupart des plates-formes d'électronique embarquée, la littérature associée au SLAM visuel métrique est dense et variée. Elle se divise principalement en deux grandes familles : les méthodes de filtrage et d'optimisation.

4.1.2 Filtrage et Optimisation

Le premier algorithme de SLAM visuel à avoir fonctionné en temps-réel, à partir d'une webcam reliée à un ordinateur a été présenté dans (DAVISON, 2003) et est maintenant connu sous le nom de MonoSLAM (DAVISON et al., 2007). La solution MonoSLAM

1. pour RANdom SAmple Consensus.

repose sur l'emploi d'un EKF et le suivi d'amers visuels dans l'environnement. Elle nécessite cependant l'initialisation manuelle de la position 3D de certains amers pour fonctionner. L'algorithme MonoSLAM a été appliqué dans le contexte de la robotique humanoïde, notamment dans les travaux de (STASSE et al., 2006) où la connaissance des mouvements du robot est utilisée, et permet de supprimer l'étape d'initialisation manuelle des amers.

L'utilisation de filtres pour résoudre des problématiques de SLAM visuel a donné lieu à de nombreux travaux (CIVERA et al., 2010 ; EADE et DRUMMOND, 2007). Mais ce type d'approche reste soumis au défaut principal des filtres en SLAM : comme expliqué en 2.2.2.1, le coût des calculs des distributions devient rapidement incompatible avec des usages temps-réel lorsque beaucoup d'amers sont présents dans la carte. Or, dans (STRASDAT, MONTIEL et DAVISON, 2012), les auteurs démontrent que pour obtenir des estimations plus précises, un grand nombre d'amers visuels est préférable. Pour cette raison, les méthodes de SLAM visuel les plus employées ces dernières années se sont basées sur l'optimisation par ajustement de faisceaux.

L'ajustement de faisceaux optimise conjointement les positions des caméras et des points 3D considérés. Basée sur la minimisation de l'erreur de reprojection des amers visuels dans les images qui les observent, cette optimisation se formule comme la combinaison des équations (4.4) et (4.5) :

$$\mathcal{M}^*, \mathcal{C}^* = \underset{\mathcal{M},\mathcal{C}}{\operatorname{argmin}} \sum_{M \in \mathcal{M}} \sum_{C \in \mathcal{C}} e_v(M,C)^T \, \Omega_v \, e_v(M,C) \tag{4.6}$$

avec $\mathcal{M}$ l'ensemble des points 3D et $\mathcal{C}$ l'ensemble des caméras qui les observent. La fonction e_v correspond à l'erreur de reprojection d'un point M dans l'image issue de la caméra C, et Ω_v est la matrice d'information associée à cette observation. Dans les approches de SLAM basées sur l'ajustement de faisceaux, la carte se conçoit donc comme un graphe visuel dont les sommets sont les caméras et les amers 3D.

Malgré son efficacité, l'ajustement de faisceaux a été employé tardivement en robotique pour résoudre le problème de SLAM visuel. Comme expliqué en partie 2.2.2.2, cette optimisation est coûteuse, ce qui limite son fonctionnement en temps-réel aux plates-formes performantes. Pour contourner ce problème, une solution courante proposée dans la littérature est celle des images-clés.

4.1.3 Approches par images-clés

Comme brièvement présentée en partie 2.2.2.2, pour diminuer la complexité de l'ajustement de faisceaux, les approches par images-clés cherchent à réduire le nombre de caméras impliquées dans l'optimisation en :

1. sélectionnant, et donc en limitant, les images à ajouter dans la carte ;
2. ne performant l'optimisation que sur un sous-ensemble, choisi en fonction de la requête courante.

De nombreux algorithmes se basent sur ces principes, introduits dans les travaux de (MOURAGNON et al., 2006) et (KLEIN et MURRAY, 2007). Globalement, leur boucle de fonctionnement peut se décrire comme le processus présenté en Figure 4.5.

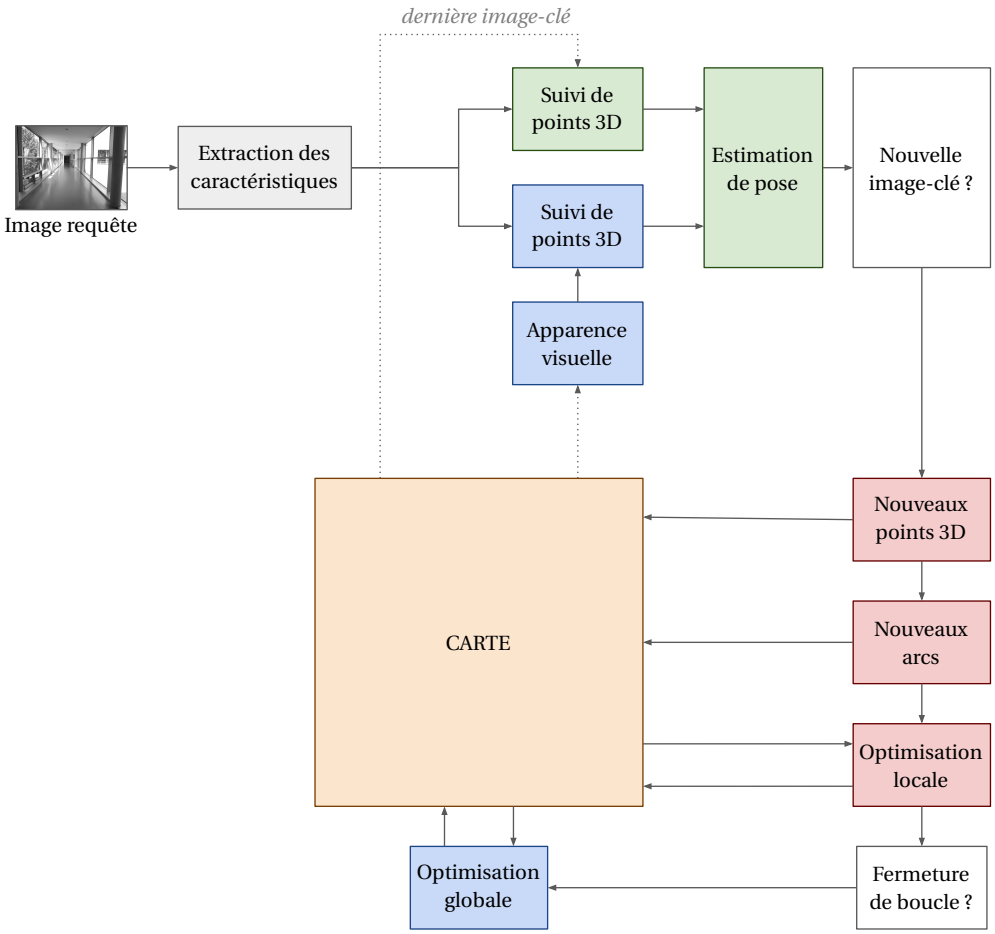


FIGURE 4.5: Boucle de fonctionnement général des approches par images-clés. Les blocs fonctionnels verts sont associés à la localisation, les rouges à la cartographie, et les bleus à la gestion des fermetures de boucle.

Considérant un flux d'images en entrée, le suivi des amers 3D permet de localiser l'image requête par rapport au nuage de points (comme expliqué en partie 4.1.1.3). Lorsque les positions de plusieurs caméras observant un même point sont estimées, ce point peut être facilement triangulé et ajouté dans la carte. Dans (KLEIN et MURRAY, 2007), les opérations de suivi et de cartographie sont séparées dans deux tâches parallèles, ce qui permet à la solution de fonctionner en temps-réel.

Ce schéma est repris par de nombreuses méthodes. Parmi elles, l'algorithme ORB-SLAM (MUR-ARTAL, MONTIEL et TARDOS, 2015) constitue aujourd'hui l'une des approches les plus populaires de SLAM visuel métrique reposant sur l'emploi d'une caméra monoculaire. Disponible en projet *open-source*[2], l'algorithme est capable de fermer les

2. https://github.com/raulmur/ORB_SLAM

boucles et emploie des caractéristiques ORB, ce qui le rend compatible avec des usages commerciaux. Ainsi, ORB-SLAM est souvent utilisé comme point de comparaison grâce à ses bonnes performances mais aussi puisque l'algorithme s'intègre facilement sur une plate-forme équipée d'une simple caméra.

Cependant, l'utilisation d'une caméra monoculaire dans un système SLAM conduit à diverses limitations, notamment liées au problème de l'échelle. Les méthodes de triangulation comme dans (KLEIN et MURRAY, 2007 ; VENTURA et al., 2014), ou les solutions basées sur la paramétrisation par inverse de profondeur comme (CIVERA, DAVISON et MONTIEL, 2008), résultent forcément en des approximations : la carte estimée n'est valide qu'à un facteur d'échelle près. Pour forcer ce facteur d'échelle, certaines stratégies s'aident de contraintes. Dans les travaux de (SCARAMUZZA et al., 2009), dont le contexte applicatif est celui de la voiture autonome, les auteurs s'appuient sur les mouvements non-holonomes de la plate-forme pour suivre des trajectoires circulaires par morceaux afin de déterminer la véritable échelle de la carte. Une autre contrainte proposée dans (LOTHE et al., 2010) consiste à supposer connues, et donc à fixer, certaines distances comme celle du sol à la caméra.

Une autre solution est d'ajouter au système une source d'information permettant de retrouver la véritable échelle de la carte. Pour supprimer l'incertitude sur la position des amers 3D, l'initialisation peut s'effectuer par un dispositif stéréo. Différents systèmes de SLAM se basent alors sur la stéréo-vision (LEMAIRE et al., 2007 ; PIRE et al., 2015 ; MORENO, BLANCO et GONZALEZ-JIMENEZ, 2016). D'autres préfèrent utiliser une caméra RGB-D comme (HENRY et al., 2010 ; ENGELHARD et al., 2011 ; ENGEL, SCHÖPS et CREMERS, 2014). Ce type de caméra initialise la profondeur d'un point grâce à des projections de motifs infrarouges ou par mesure de temps de vol. Le développement de telles solutions s'est accéléré avec la démocratisation de ces capteurs (Kinect, Asus Xtion, etc.). Dans sa deuxième version, ORB-SLAM2 (MUR-ARTAL et TARDÓS, 2017a) propose ainsi d'employer une caméra monoculaire, RGB-D, ou un dispositif stéréo [3].

D'autres systèmes utilisent une centrale inertielle. Différentes méthodes de fusion entre données inertielles et visuelles ont été étudiées, aussi bien sur la base de filtres (MOURIKIS et ROUMELIOTIS, 2007 ; LI et MOURIKIS, 2013 ; BLOESCH et al., 2015) que d'optimisations (LEUTENEGGER et al., 2015 ; MUR-ARTAL et TARDÓS, 2017b ; QIN, LI et SHEN, 2018). On retrouve ces méthodes dans la littérature sous le nom de SLAM visuel-inertiel. Le code source des plus populaires est disponible comme celui d'OKVIS [4] (LEUTENEGGER et al., 2015) ou de VINS [5] (QIN, LI et SHEN, 2018).

Les sous-parties qui suivent détaillent les stratégies les plus courantes de la littérature correspondant à chaque bloc de la boucle de fonctionnement présentée en Figure 4.5.

4.1.3.1 Déterminer sa pose dans la carte

La localisation du capteur dans la carte revient à estimer ses paramètres extrinsèques. Pour cela, le système doit :

3. https://github.com/raulmur/ORB_SLAM2
4. https://github.com/ethz-asl/okvis
5. https://github.com/HKUST-Aerial-Robotics/VINS-Mono

— extraire des points d'intérêt sur l'image requête,

— associer ces points d'intérêt aux points 3D de la carte,

— utiliser ces associations pour estimer la pose de la caméra.

Extraction des points d'intérêt : La majorité des approches par images-clés reposent sur l'emploi de points caractéristiques comme amers. Sur une image, ces points d'intérêts sont extraits et caractérisés par un couple détecteur-descripteur, comme présenté en 3.2.2.1. Ces dernières années, la nécessité d'un fonctionnement temps-réel des solutions de SLAM a favorisé l'utilisation de couples détecteur-descripteur rapides. Dans ORB-SLAM, des caractéristiques binaires ORB sont utilisées à partir d'une détection FAST à huit niveaux d'échelle ; OKVIS se base sur des points saillants extraits par un détecteur de Harris (HARRIS et STEPHENS, 1988) puis décrits par des descripteurs binaires BRISK (LEUTENEGGER, CHLI et SIEGWART, 2011) [6] ; il en va de même pour VINS qui suit des coins caractérisés par une description BRIEF (SHI et TOMASI, 1993 ; CALONDER et al., 2010).

Chacune de ces solutions joue sur les paramètres du détecteur pour assurer une détection uniforme des caractéristiques dans l'image. Dans le cas d'ORB-SLAM, la détection uniforme est assurée par la division de l'image en une grille. Dans chaque case de la grille, le seuil de détection du détecteur FAST est adapté pour extraire au moins cinq points d'intérêt.

Suivi des amers 3D : Le suivi des amers 3D consiste ensuite à établir des liens 2D-3D entre les points d'intérêt extraits sur une image requête et les points 3D cartographiés. Le risque de cette étape est d'établir de mauvaises associations 2D-3D, pouvant fausser l'estimation de la pose de la caméra.

Pour éviter cela, ORB-SLAM s'appuie sur un modèle de mouvement. Puisque les images se suivent temporellement, les coordonnées pixelliques des projections issues d'amers 3D peuvent se prédire. La recherche et l'association entre points d'intérêt et points 3D se fait donc selon ces prédictions. Lorsque le modèle de mouvement ne décrit plus suffisamment bien la transformation entre deux images, la zone de recherche de chaque point dans l'image est élargie.

La solution OKVIS associe de manière brute les points d'intérêt de l'image requête avec ceux qu'elle est supposée observer. Certaines associations sont ensuite rejetées en fonction de leur distance [7] avec la pose prédite. Via cette démarche, OKVIS profite pleinement de la rapidité d'association des descripteurs binaires.

Une question non-triviale reste l'association des points d'intérêt d'une image avec les points 3D de la carte. Cette association pourrait être purement géométrique, et considérer le point d'intérêt le plus proche de la projection supposée. Mais ce type d'approche ne permet pas d'éviter les erreurs liées à l'incertitude du capteur. Ainsi, les points 3D doivent être associées aux caractéristiques extraites sur une image selon leur ressemblance. Pour cela, à chaque point 3D doit être associé un descripteur de la même nature que ceux associés aux points d'intérêt 2D des images. Dans ORB-SLAM, le descripteur

6. Orientés selon la direction de la gravité, observable grâce à la centrale inertielle embarquée.
7. De Mahalanobis (DE MAESSCHALCK, JOUAN-RIMBAUD et MASSART, 2000).

d'un point 3D est le descripteur ORB le plus représentatif des descripteurs de toutes ses projections. Pour le déterminer, l'algorithme cherche parmi les points d'intérêts qui lui ont été associés celui dont la distance de Hamming avec tous ces autres descripteurs est la plus petite.

Estimation de la pose courante : L'estimation de la pose de la caméra dont est issue l'image courante est en général réalisée en deux temps. D'abord, un suivi des points 3D est réalisé dans l'image courante, comme expliqué précédemment. Ce suivi, un peu grossier, permet d'estimer une première pose de la caméra, par exemple grâce à une optimisation comme celle présentée dans l'équation (4.5).

Un second suivi, plus précis, se base sur cette première estimation de pose. Plus d'associations 2D-3D sont générées en considérant les amers 3D supposés être visibles depuis la pose estimée. Ces nouvelles associations sont alors utilisées pour raffiner la première estimation.

Notons que certaines méthodes utilisent l'algorithme RANSAC pour classifier les associations comme *inliers* et *outliers*. Une autre approche consiste à réaliser plusieurs optimisations successives en triant à chaque itération toutes les associations sous les étiquettes d'*inliers* et d'*outliers*. Seules les associations classées comme *inliers* sont alors utilisées dans l'itération suivante pour l'optimisation. L'Algorithme 1 résume ce fonctionnement. Remarquons que les valeurs de `thresholds` sont souvent organisées de sorte à ce que $v_1 > v_2 > v_3$... Ainsi, le critère de classification d'*inliers* est de plus en plus restrictif et seules les meilleures associations sont conservées.

Algorithme 1 Optimisation itérative

Entrées : `image`, `associations2D-3D`, `points3D`

```
pose = image.extrinsics
inliers = associations2D-3D
// Définition des seuils d'erreurs de reprojection.
thresholds [] = {v1, v2, v3, ...}
// Optimisations itératives.
for v ∈ thresholds do
  pose = optimize (image, inliers, points3D)
// Actualise les inliers.
  inliers = null
  for association ∈ associations2D-3D do
    projection = project3DPoints (pose, image.intrinsics, association.point3D)
    reprojectionError = distance (projection, association.point2D)
    if reprojectionError < v then
      inliers += association
    end if
  end for
end for
```

Sorties : `pose`, `inliers`

Détection de fermeture de boucle : Dans la plupart des cas, la détection de fermeture de boucle se fait grâce à des techniques basées sur l'apparence visuelle. Une présentation de telles méthodes a été faite dans le chapitre précédent, dans la partie 3.1.

Certaines solutions utilisent l'apparence visuelle comme stratégie de secours. Dans le cas où le suivi depuis la dernière image-clé acquise échoue, par exemple lorsque le modèle de mouvement est trop éloigné de la réalité, le système cherche à se localiser par rapport à une autre image-clé de référence. Cette dernière est alors identifiée par l'apparence. D'autres solutions emploient l'apparence visuelle pour détecter la dérive de la trajectoire estimée.

Pour optimiser le temps de calcul global du processus, la détection de fermeture de boucle peut s'effectuer dans une tâche parallèle. Une première tâche cherche à localiser la pose caméra courante par rapport à la dernière image-clé ajoutée à la carte, pendant qu'une deuxième tâche la localise par rapport à l'image-clé qui lui ressemble le plus. Comme la recherche de fermeture de boucle dépend du nombre d'images-clés dans la carte, réaliser cette recherche dans une tâche parallèle permet de ne pas perturber le suivi temps-réel.

4.1.3.2 Augmenter la carte

Ajout d'image-clé : L'intérêt des approches par images-clés est de diminuer la complexité du problème d'optimisation en limitant le nombre de paramètres qui y entrent en jeu. Pour cela, un premier filtre trie les images requêtes pour n'ajouter à la carte que les plus pertinentes : les images-clés.

Ce filtre peut s'appuyer sur différents critères, mais son principe est toujours de maximiser l'information contenue dans la carte, en y ajoutant un nombre minimal d'images-clés. Par exemple, si le robot ne bouge pas entre deux images du flux vidéo d'entrée, les deux images sont redondantes. Il suffit alors d'ajouter une seule de ces images dans la carte pour obtenir une information proche de l'optimum. Par contre, si le robot reste statique un certain temps mais que son environnement change autour de lui, l'ajout d'une nouvelle image-clé peut se révéler pertinent.

Un critère simple consiste à appliquer un seuil d'ajout, basé sur le rapport entre nouveaux points observés et points déjà dans la carte. OKVIS emploie une heuristique basée sur le recouvrement visuel : si la zone définie par les points suivis sur l'image représente 50 à 60% de la zone définie par tous les points extraits sur l'image, cette dernière est ajoutée à la carte comme une nouvelle image-clé.

En général, l'ajout d'une image-clé est cependant déterminé par la combinaison de plusieurs critères. Dans (KLEIN et MURRAY, 2007), qui constitue l'une des premières approches de SLAM métrique monoculaire par images-clés, les auteurs utilisent trois conditions :

— la qualité du suivi : un minimum de points de la carte doit être suivi pour assurer la qualité de l'estimation courante et la continuité du graphe visuel ;

— une distance temporelle minimale : ajouter deux fois la même image est inutile, ainsi les auteurs espacent leurs images-clés d'au moins 20 acquisitions ;

— la position par rapport à la carte : un critère force chaque image-clé à être à une distance minimale de l'amer 3D le plus proche.

Ces critères sont globalement repris par les différentes approches de la littérature ; par exemple, ceux utilisés par ORB-SLAM en dérivent directement.

Ajustement de faisceau local : La deuxième astuce des approches par images-clés pour diminuer la complexité algorithmique de la solution est d'éviter de réaliser de « grosses optimisations » . Pour cela, les optimisations sont de préférence réalisées sur des sous-graphes locaux. En effet, les contraintes ajoutées par une nouvelle image-clé ont un impact plus fort sur les sommets du graphes qui lui sont proches. Un ajustement de faisceau local est réalisé sur ce sous-ensemble de sommets, dont les extrémités sont fixes. Dans la littérature, ce sous ensemble est composé par :

— l'image-clé courante C_N et les points qu'elle observe $\mathcal{M}_{C_N}$;

— toutes les images-clés $\mathcal{C}_n$, observant au moins un point de $\mathcal{M}_{C_N}$;

— tous les points $\mathcal{M}_{\mathcal{C}_n}$, observés par les images-clés $\mathcal{C}_n$ et n'appartenant pas à $\mathcal{M}_{C_N}$;

— enfin, les images-clés observant des points de l'ensemble $\mathcal{M}_{\mathcal{C}_n}$, mais ne partageant pas d'observation commune avec l'image-clé courante, sont ajoutées à l'optimisation mais leurs poses restent fixes.

La Figure 4.6 illustre cette carte locale optimisée. On y constate que les images-clés, sommets au voisinage $n+2$ de l'image courante dans le graphe de co-observabilité, jouent le rôle d'ancres aux limites de la carte locale pour maintenir une continuité avec la carte globale.

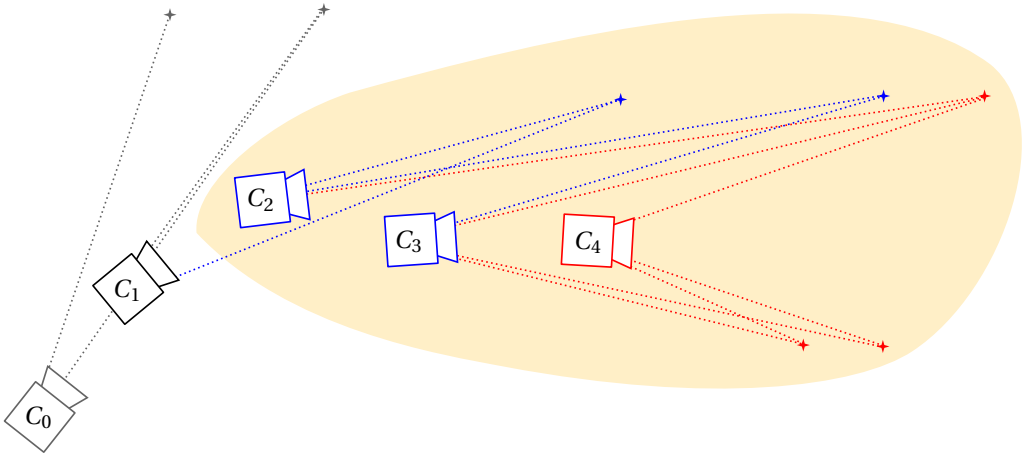


FIGURE 4.6: Variables d'optimisation locale. À l'ajout de C_4 dans la carte, l'ajustement de faisceau est réalisé sur une sous-carte locale. Les variables optimisées sont placées dans la forme jaune (caméras et amers). La caméra C_1 intervient dans le processus d'optimisation mais ses paramètres sont fixes. C_0 n'est pas utilisée.

Bien que cette optimisation soit moins coûteuse pour être utilisée à l'ajout de chaque nouvelle image-clé dans la carte, elle n'assure pas la non-dérive du système ; d'où l'importance que revêt la détection de fermeture de boucle, et la correction qui s'en suit.

Optimisation globale : Lorsque la dérive du système est avérée, une optimisation globale de la carte est nécessaire. Cette fois, l'ajustement de faisceau doit être réalisé sur la carte entière. Cependant, lors de grosses dérives du système, l'optimisation de la carte entière par un ajustement de faisceau simple peut se révéler inefficace.

Pour fonctionner, l'optimisation par ajustement de faisceaux doit partir d'un bon état initial. Plusieurs approches cherchent ainsi à « diluer » l'erreur de fermeture de boucle sur le graphe avant de réaliser une optimisation par ajustement de faisceaux. Dans (KONOLIGE et AGRAWAL, 2008), les auteurs se basent sur un *squelette*, encodant les transformations relatives entre chaque pose des caméras du graphe. ORB-SLAM a également recours à cette pré-optimisation en reprenant les travaux présentés dans (STRASDAT, MONTIEL et DAVISON, 2010). ORB-SLAM s'aide pour cela d'un *graphe essentiel*, retenant les transformations entre images-clés partageant beaucoup d'observations.

L'optimisation globale de la carte se conçoit donc comme deux optimisations successives. Au cours de la première, seules les poses des images-clés sont optimisées, en prenant en compte leurs transformations relatives. Après cette optimisation, le nuage de point 3D est repositionné suivant le mouvement des images-clés. Puis un ajustement de faisceaux global prenant en compte l'ensemble des images-clés $\mathscr{C}$ et l'ensemble des points 3D $\mathscr{M}$ de la carte est réalisé. Remarquons que, au cours de chacune de ces optimisations, et pour des raisons de stabilité numérique, un sommet du graphe est souvent fixé. La majorité des approches choisit arbitrairement de fixer la première image-clé ajoutée au graphe, qui définit, de manière directe ou relative, le repère monde.

* * *

Les approches par images-clés présentent ainsi aujourd'hui de bonnes performances sur les systèmes embarqués. Néanmoins, leur utilisation dans notre contexte n'est pas si simple. La sous-partie suivante met en lumière les lacunes de telles approches sur notre plate-forme dans nos cas d'usages.

4.1.4 Limites pratiques sur Pepper

Une solution de SLAM métrique visuel est envisageable sur Pepper, d'autant plus que le robot est équipé d'une caméra stéréo [8]. Les deux caméras monoculaires, montées de manière rigide dans chaque œil du robot, sont synchronisées et calibrées sur les lignes de production pour constituer un dispositif stéréo. Les champs de vue vertical et horizontal de ce dispositif sont illustrés en Figure 4.7. L'écart entre les deux caméras, gauche et droite est de 7 cm.

Bien que les approches par images-clés aient été utilisées avec succès dans le contexte de la navigation pour drones (NIKOLIC et al., 2014 ; CONCHA et al., 2016), leur fonctionnement sur Pepper est soumis à des contraintes différentes qui dégradent leurs performances.

8. http://doc.aldebaran.com/2-5/family/pepper_technical/video_Stereo_pep.html

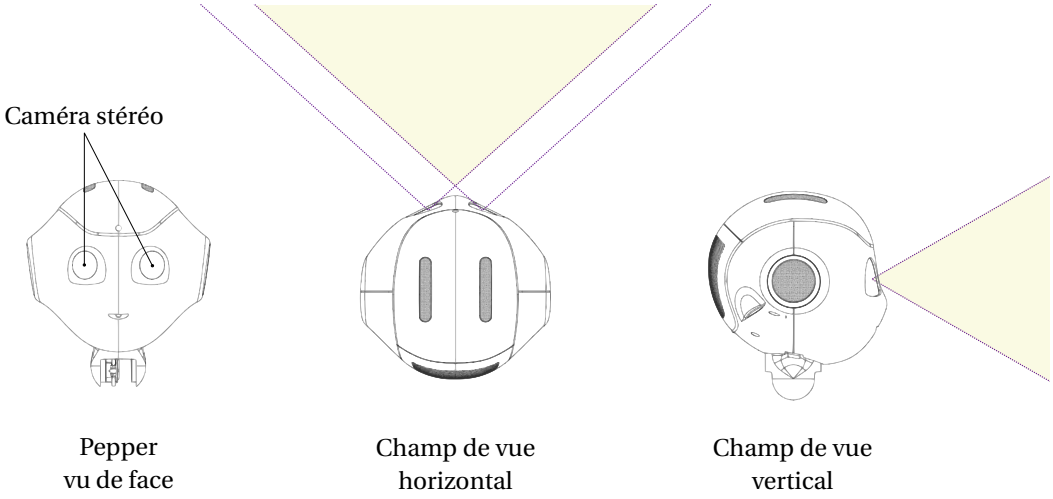


FIGURE 4.7: Caméra stéréo de Pepper. Le dispositif est composé de deux caméras monoculaires placées dans chaque œil dont les champs de vue se recoupent. Pour chaque caméra monoculaire le champ de vue horizontal est de 96°et de 60°pour le vertical.

Tout d'abord, certaines contraintes viennent directement de la plate-forme. Les robots Pepper sont équipés d'IMUs [9] et CPUs entrée de gamme :

— les données issues des IMUs des robots Pepper ne permettent pas d'envisager leur utilisation pour résoudre le problème du facteur d'échelle ;

— les CPUs de ces robots empêchent un traitement des images à haute fréquence.

En effet, Pepper est équipé d'un processeur quatre cœurs Atom E3845, tournant à 1,91 GHz. Sur ces quatre cœurs, trois sont dédiés à des calculs obligatoires provenant de tâches interactives comme le contrôle et la planification de mouvements, la détection d'humains et la reconnaissance vocale. Toute solution de SLAM, créée pour fonctionner industriellement sur Pepper est donc limitée à l'utilisation d'un seul cœur du CPU. À titre de comparaison, les travaux présentés dans (CONCHA et al., 2016) sont testés sur un processeur Intel Core i7-3770K tournant à 3,5 GHz dans trois tâches parallèles. Ceux de (NIKOLIC et al., 2014) disposent de deux processeurs ARM Cortex A9 intégrés dans un circuit FPGA XILINX Zynq 7020.

Alors que les systèmes de SLAM actuels fonctionnent aux alentours de 20 à 30 Hz, la puissance computationnelle allouée aux tâches de localisation sur Pepper ne permet pas d'envisager de telles fréquences de traitement.

D'autres contraintes viennent de la fonction sociale du robot. Par exemple, le champ visuel de Pepper est susceptible de se retrouver régulièrement obstrué par des interactions avec ses utilisateurs. Remarquons également que les mouvements de tête du robot génèrent un flou de bougé important puisque les rotations sur les axes de lacet et tangage au niveau du cou sont autorisées à monter jusqu'à une vitesse de 7 radians par seconde en fonctionnement normal.

9. Unité de Mesure Inertielle.

Ainsi, le robot est susceptible de tourner rapidement la tête et d'interagir avec des gens. Ces différentes contraintes conduisent à l'échec d'une étape cruciale de la chaîne algorithmique du SLAM visuel : celle de suivi des amers 3D. En effet, les méthodes de SLAM basées sur la vision utilisent la co-observabilité des amers comme contrainte. Les poses des caméras sont indirectement reliées par leurs observations communes. L'échec du suivi des amers 3D entre plusieurs images conduit ainsi à la scission de la carte générée en de multiples graphes visuels, qui ne sont reliés par aucune contrainte. Ce problème est en pratique inévitable. Même avec une plate-forme plus performante, un environnement visuellement peu texturé ou très dynamique risque de tenir le suivi d'amers en échec.

Pour compenser le problème des environnements peu texturés, certaines approches se basent sur l'emploi de caractéristiques visuelles à plus haut niveau. Dans (GOMEZ-OJEDA et GONZALEZ-JIMENEZ, 2016), les auteurs proposent par exemple de suivre des points d'intérêt mais également des segments. D'autres solutions cherchent même à s'abstraire de l'utilisation de caractéristiques visuelles et à reposer directement sur l'information portée par l'intensité des pixels (SILVEIRA, MALIS et RIVES, 2008 ; ENGEL, SCHÖPS et CREMERS, 2014 ; FORSTER, PIZZOLI et SCARAMUZZA, 2014). Concernant la gestion des environnements dynamiques, le raisonnement le plus couramment employé consiste à identifier les objets dynamiques dans les images pour ne pas les prendre en compte lors des différentes estimations. À cet effet, le système présenté dans (YU et al., 2018) utilise un réseau de segmentation combiné à une vérification de la cohérence de mouvement pour identifier les objets dynamiques et les classifier comme *outliers*. De la même façon, dans (BESCÓS et al., 2018) les auteurs se basent sur des critères de cohérence multi-vue, ainsi que l'emploi d'apprentissage profond pour rejeter les objets mobiles, mais proposent aussi de reconstruire les parties soustraites des images par *inpainting*[10].

De telles solutions restent trop coûteuses sur notre matériel. Mais une information pertinente permettant de combler l'absence de contraintes visuelles entre plusieurs prises de vue est disponible sur Pepper : l'odométrie. Sur les robots mobiles à roues, l'odométrie constitue une source d'information facile à calculer et relativement fiable sur de courts trajets. Dans notre cas, l'utilisation de contraintes odométriques permettrait alors de maintenir des arêtes entre les différents graphes visuels constituant une carte détériorée par de multiples échecs de suivi, comme l'illustre la Figure 4.8.

La partie suivante détaille ainsi notre solution, qui consiste à introduire des arêtes odométriques dans un graphe visuel, et notre approche pour maintenir ce graphe composé de contraintes de natures différentes.

10. L'*inpainting* est une technique de reconstruction d'images détériorées qui cherche à remplir les parties manquantes d'une image.

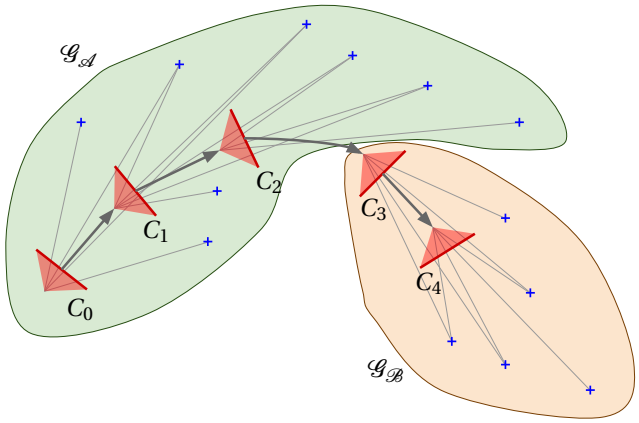


FIGURE 4.8: Conséquences de la perte de suivi d'amers 3D. Les images-clés (triangles rouges) et les points 3D (croix bleues) forment la carte. Le suivi des amers est perdu entre C_2 et C_3, ce qui scinde la carte en deux graphes visuels indépendants $\mathcal{G}_{\mathcal{A}}$ et $\mathcal{G}_{\mathcal{B}}$, reliés par une arête odométrique (flèche grise).

4.2 Odométrie et ajustement de faisceaux

Contrairement à un dispositif visuel porté à la main, une information odométrique forte est disponible sur Pepper. Il est facile d'estimer le déplacement du capteur stéréo entre deux acquisitions : premièrement en estimant le déplacement de la base de Pepper, puis en remontant la chaîne articulaire du robot partant de la base jusqu'au capteur stéréo (Figure 4.9). Grâce à des encodeurs magnétiques, les positions de toutes les articulations sont connues à chaque instant. Le déplacement de la base est estimé par combinaison de l'information issue de compte-tours sur chaque roue et d'un gyroscope, placé dans la base du robot, fournissant une information de vitesse angulaire instantanée sur l'axe $\boldsymbol{Z_{Base}}$.

L'utilisation de l'odométrie dans des systèmes de SLAM visuel est pertinente et a conduit à de nombreux travaux. En effet, les travaux de (CUMANI et al., 2004) emploient une information odométrique pour forcer le déplacement estimé entre plusieurs observations visuelles et ainsi résoudre le problème du facteur d'échelle. Une autre approche est proposée dans (JOHANNSSON et al., 2013) et cherche à maintenir un graphe de poses réduit, qui incorpore des données odométriques là où les transitions visuelles ne sont pas assurées. Les auteurs de (KAESS et DELLAERT, 2006) fusionnent les informations issues de l'odométrie et de la vision dans une optimisation cherchant à expliquer au mieux les mesures de ces deux sources d'information.

L'odométrie permet donc de pallier les lacunes d'un système de SLAM purement visuel. En particulier, l'odométrie conduit à l'estimation d'une carte plus cohérente en résolvant le problème du facteur d'échelle et en compensant les pertes de suivi visuel en cours de fonctionnement (EUDES, 2011 ; JOHANNSSON et al., 2013).

Nous combinons ici les approches présentées dans (KAESS et DELLAERT, 2006) et (JOHANNSSON et al., 2013). Notre solution se base sur une optimisation prenant à la fois en compte les contraintes issues de l'observation d'amers visuels et les contraintes odométriques. En effet, l'optimisation séparée de ces arêtes de natures différentes ne peut

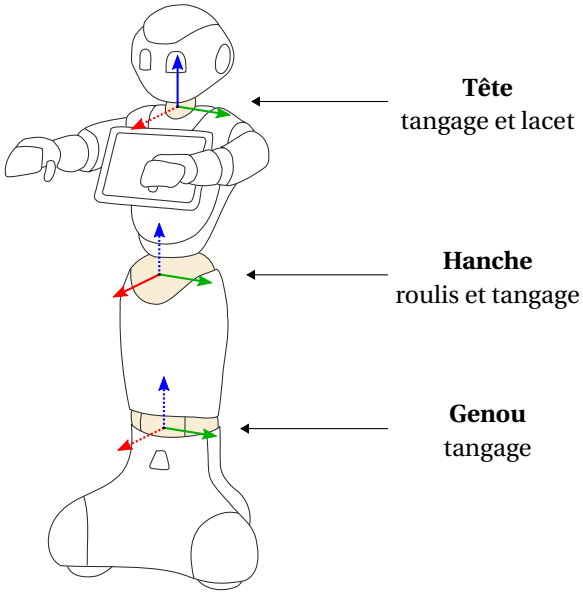


FIGURE 4.9: Chaîne articulaire depuis la base jusqu'au capteur stéréo. Pour chaque articulation, le repère associé est dessiné avec l'axe $\boldsymbol{X}$ en rouge, $\boldsymbol{Y}$ en vert et $\boldsymbol{Z}$ en bleu. Pour plus de détails : http://doc.aldebaran.com/2-5/family/pepper_technical/joints_pep.html.

assurer la cohérence de la carte en cas de perte de suivi visuel. Une unique optimisation mêlant information visuelle et odométrique est ainsi préférable. L'équation (4.6) est dans ce but augmentée pour prendre en compte ces contraintes odométriques :

$$\begin{aligned}\mathcal{M}^*, \mathcal{C}^* = \underset{\mathcal{M},\mathcal{C}}{\operatorname{argmin}} & \sum_{M\in\mathcal{M}} \sum_{C\in\mathcal{C}} e_v(M,C)^T \, \Omega_V \, e_v(M,C) \\ & + \sum_{C_i\in\mathcal{C}} e_\Delta(C_i, C_{i_{Parente}})^T \Omega_{D_i} \, e_\Delta(C_i, C_{i_{Parente}})\end{aligned} \tag{4.7}$$

où $\mathcal{M}$ est l'ensemble des amers 3D observés et $\mathcal{C}$ l'ensemble des poses des images-clés. Les fonctions e_v et e_Δ sont les fonctions de coût évaluant respectivement les arêtes visuelles et odométriques. Les matrices Ω_V et Ω_D évaluent quant à elles l'information portée par l'observation courante, qu'elle soit issue du domaine visuel ou odométrique.

On retrouve bien dans l'équation (4.7) un premier terme, sous la forme d'une double somme, correspondant à l'ajustement de faisceau classique, et un second terme, impliquant les arêtes odométriques. Les données odométriques sont mesurées et conservées entre l'acquisition de deux images-clés. Elles traduisent ainsi le déplacement de l'image-clé parente $C_{Parente}$ vers l'image-clé courante C. Malgré la nature différente des mesures effectuées (en pixel pour la vision et sous la forme d'un déplacement pour l'odométrie), l'équation (4.7) reste cohérente. Les sommes sont en effet adimensionnelles, grâce à l'emploi des matrices Ω_V et Ω_{D_i}. Dans ce contexte, il devient cependant nécessaire de bien caractériser la matrice d'information associée à chaque arête.

4.2.1 Information odométrique

On peut se représenter l'information odométrique comme la confiance que l'on a dans la mesure du déplacement entre deux images-clés. Pour un robot à roues, la mesure d'un déplacement court est plus certaine que celle associée à un déplacement plus long. Cela s'explique par le fait que plus le robot parcourt de chemin, plus il accumule de petites erreurs. La variance de sa position sous chaque dimension du plan dans lequel il se déplace augmente alors ; à l'inverse de l'information qui diminue.

Sur Pepper, le déplacement du capteur stéréo entre l'acquisition de deux images-clés peut s'exprimer par une translation et une rotation en trois dimensions. Grâce aux capteurs de position placés sur chaque articulation, la proprioception du robot ne dérive pas. L'information associée à la chaîne articulaire est donc constante. L'information associée au déplacement de la base du robot est quant à elle dépendante de différents facteurs.

La base de Pepper comporte trois roues motrices omnidirectionnelles. Chaque roue est composée d'un axe fort, qui est motorisé, et d'un axe faible selon lequel la roue peut rouler librement. Aucune commande, ni aucune mesure, ne peut être effectuée sur l'axe faible. Sur la Figure 4.10, l'axe motorisé de chaque roue est représenté par une flèche noire. Grâce à ce design, Pepper est une plate-forme holonome. Par exemple, en appliquant deux vitesses opposées aux roues R_g et R_d et aucune à R_a, le robot avance en ligne droite. De même, lorsque toutes les roues tournent à la même vitesse, Pepper tourne sur lui-même.

Un modèle cinématique linéaire J permet de transformer les vitesses mesurées des roues en vitesse de la base, s'exprimant par v_X et v_Y, respectivement les vitesses de la base selon les axes $\boldsymbol{X}$ et $\boldsymbol{Y}$, et v_θ, la vitesse de rotation autour de l'axe $\boldsymbol{Z}$ (Figure 4.10) :

$$\begin{pmatrix} v_X \\ v_Y \\ v_\theta \end{pmatrix} = J . \begin{pmatrix} v_{R_g} \\ v_{R_d} \\ v_{R_a} \end{pmatrix} \tag{4.8}$$

Un déplacement D de la base dans le plan défini par $\boldsymbol{X}$ et $\boldsymbol{Y}$, peut ainsi se décrire par les trois incréments : dX, dY et $d\theta$.

Le déplacement de la base estimé à partir des mesures sur les roues est noté D_R. Pour quantifier l'information qui lui est associée, on considère ce déplacement comme suivant un phénomène de marche aléatoire, avec la distance parcourue comme variable d'intégration. Plus la distance parcourue est grande, plus les erreurs sur chacune des composantes de D augmente. L'écart-type σ_R associé au mouvement mesuré de chaque roue est donc directement proportionnel à la racine carré de la valeur w de la rotation de la roue :

$$\sigma_R = \alpha \times \sqrt{w_R} \tag{4.9}$$

En considérant que l'erreur de mesure sur chaque roue est indépendante, on peut définir

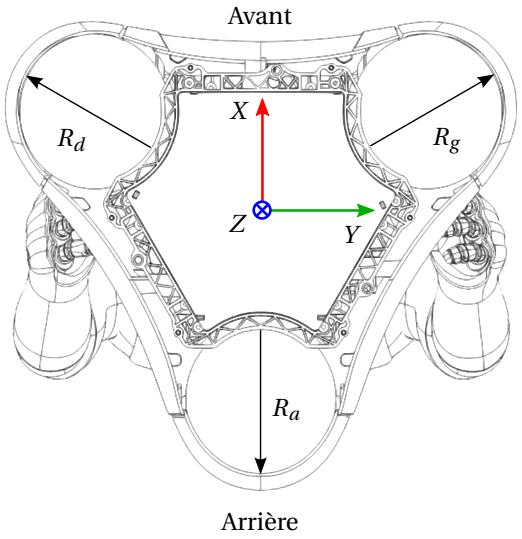


FIGURE 4.10: Schéma de la base de Pepper vue du dessous.

la matrice de covariance associée aux mesures de mouvements des roues comme :

$$cov_R = \begin{pmatrix} \sigma^2_{R_g} & 0 & 0 \\ 0 & \sigma^2_{R_d} & 0 \\ 0 & 0 & \sigma^2_{R_a} \end{pmatrix} \tag{4.10}$$

En reprenant le modèle cinématique défini dans l'équation (4.8), la matrice de covariance associée à un déplacement D_R, estimée par les mesures sur chaque roue, s'exprime comme :

$$cov_{D_R} = J.cov_R.J^T \tag{4.11}$$

Et l'information associée à D_R correspond alors à l'inverse de sa covariance, soit :

$$\Omega_{D_R} = cov^{-1}_{D_R} \tag{4.12}$$

Remarquons ici une particularité de ce modèle : si aucun mouvement n'est mesuré sur les roues, alors $\sigma^2_R = 0$ et donc l'information associée est infinie. En pratique cela peut se concevoir comme le fait que l'on sait lorsque le robot n'a pas bougé : notre confiance en D_R est donc très grande. Cependant, ces cas particuliers peuvent détériorer notre système d'un point de vue numérique. Pour éviter cela, le pas du capteur de position des roues est employé comme limite inférieure de σ_R.

L'odométrie du robot découle cependant d'une fusion entre les mesures effectuées sur le mouvement des roues, et celles d'un gyroscope placé dans la base. L'information

apportée par ce dernier doit donc également être prise en compte. De la même manière, on considère l'erreur de ce capteur comme suivant un phénomène de marche aléatoire, mais cette fois, dépendant du temps. En effet, même en restant statique, les mesures du gyroscope s'accumulent et l'erreur sur l'orientation du robot augmente. L'erreur σ_G sur d_θ, issue des données du gyroscope, est donc directement proportionnelle à la racine carré du temps écoulé entre deux mesures :

$$\sigma_G = \beta \times \sqrt{\Delta_t} \tag{4.13}$$

où Δ_t est la durée écoulée entre deux mesures du gyroscope. La matrice d'information sur D_G, déplacement estimé par le gyroscope, peut donc s'écrire :

$$\Omega_{D_G} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sigma_G^2} \end{pmatrix} \tag{4.14}$$

En considérant que les mesures sur les roues et celles du gyroscope sont des sources d'information indépendantes, l'information associée à D correspond à la somme de celles associées à ces deux sources, soit :

$$\Omega_D = \Omega_{D_R} + \Omega_{D_G} \tag{4.15}$$

Pour finir, remarquons que les deux facteurs α et β introduits dans les équations (4.9) et (4.13), sont dans notre cas calibrés et déterminés sur un jeu d'explorations d'apprentissage. Ici, ces deux termes sont évalués à partir du jeu d'expériences présenté en partie 3.4.2.

4.2.2 Information visuelle

L'information visuelle Ω_V caractérise la confiance que l'on peut avoir dans une observation visuelle. Dans l'équation (4.7), l'information visuelle Ω_V est directement liée à l'incertitude de notre détecteur. Son évaluation est réalisée sur des paires d'images stéréo acquises depuis Pepper. Les images stéréo de chaque paire ont exactement les mêmes paramètres (intrinsèques et extrinsèques) mais correspondent à deux acquisitions distinctes. Les points d'intérêts extraits sur une image sont associés à leur plus proche voisin dans la seconde image en suivant leurs coordonnées (u_i, v_i). Notons que pour éviter les biais dus aux changements de l'environnement, nous veillons à ce que l'environnement ne subisse pas de variations visuelles importantes. Les associations sont ensuite contrôlées manuellement et celles aberrantes rejetées. Le lot d'associations final permet alors de calculer la matrice de covariance associée à l'erreur du détecteur utilisé puis d'en déterminer l'information qui lui est associée.

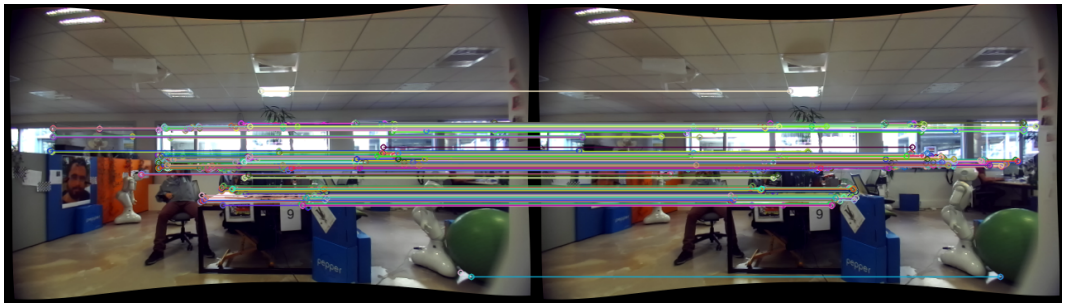


FIGURE 4.11: Exemple d'images utilisées pour la calibration de l'erreur du détecteur.

* * *

De cette manière, l'odométrie et la vision peuvent être utilisées et combinées dans une même optimisation. Cette optimisation nous permet de corriger la trajectoire estimée par notre système de SLAM. Basé sur la stéréo-vision et l'odométrie de Pepper, notre algorithme de SLAM cherche à profiter pleinement de ces deux capteurs tout en respectant les contraintes computationnelles liées à la plate-forme. Cette solution de SLAM est détaillée dans la partie qui suit.

4.3 Solution de SLAM visuel métrique sur Pepper

Cette partie présente notre solution complète de SLAM métrique, combinant données visuelles et odométriques. Contrairement aux méthodes de la littérature, notre solution doit être compatible avec les contraintes de notre plate-forme, notamment une puissance CPU limitée conduisant à une fréquence globale de fonctionnement bien inférieure aux standards actuels. Notre travail reprend les grandes lignes de la solution de SLAM visuel stéréo ORB-SLAM2 (MUR-ARTAL et TARDÓS, 2017a) en adaptant certaines étapes pour les rendre fonctionnelles sur Pepper. L'enchaînement des étapes de notre système est résumé sur la Figure 4.12.

Cette partie se divisera en quatre temps. D'abord, en 4.3.1, notre attention se portera sur les caractéristiques visuelles utilisées par notre système. Ensuite, les structures de données de notre solution seront présentées en 4.3.2. Et pour finir, les étapes des processus de localisation et de cartographie seront détaillées en 4.3.3 et 4.3.4. Leurs différences avec les solutions de l'état-de-l'art seront mises en avant.

4.3.1 Image stéréo et caractéristiques visuelles

Le capteur stéréo de Pepper permet de récupérer une image stéréo, correspondant à la concaténation des caméras monoculaires placées dans l'œil gauche et droit du robot. Les caméras sont calibrées industriellement et synchronisées. Une distorsion peut être observée sur les images issues des caméras, mais cette dernière est rectifiée par logiciel, comme le montre la Figure 4.13.

Grâce à cette rectification, les lois de projection de points 3D sur le plan image correspondent à celles du modèle sténopé présenté en 4.1.1.1. De plus, les droites épipo-

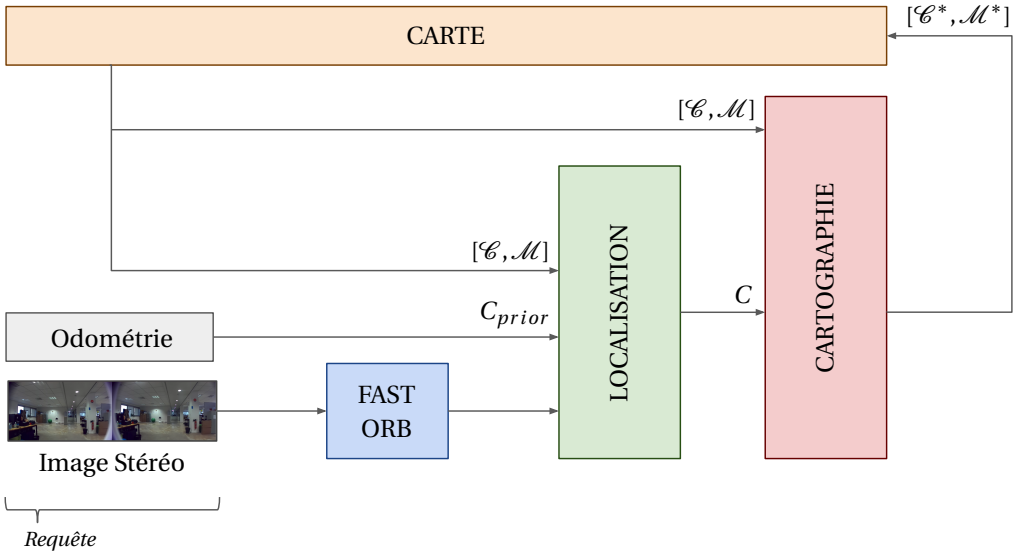


FIGURE 4.12: Notre processus de SLAM : entrées et sorties.

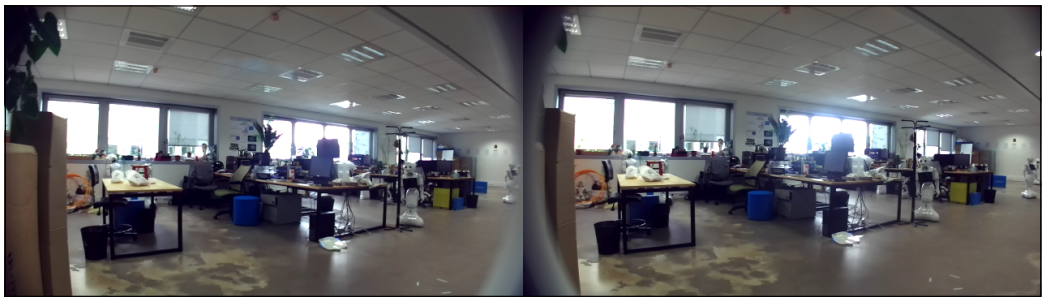


Image stéréo acquise

Image stéréo rectifiée

FIGURE 4.13: Rectification de l'image stéréo : l'image stéréo est rectifiée à partir des informations de calibration du dispositif. Cette rectification supprime la distorsion et permet au système de suivre les lois de projection du modèle sténopé.

laires se retrouvent alignées de sorte à ce qu'un point réel M soit projeté dans les images gauche et droite sur la même ligne, soit $v_{m_{gauche}} = v_{m_{droite}}$. En pratique, ce critère est utilisé pour rejeter les associations aberrantes entre les images gauche et droite.

Notre solution utilise un détecteur FAST (ROSTEN et DRUMMOND, 2006) et une description ORB (RUBLEE et al., 2011), pour les mêmes raisons que celles développées

en 3.2.2.1 mais également dans un soucis de compatibilité avec notre solution de relocalisation (Chapitre 3). Dans notre solution actuelle, la relocalisation par apparence visuelle est réalisée via l'algorithme FAB-MAP 2.0 (CUMMINS et NEWMAN, 2011), adapté aux caractéristiques binaires ORB. Remarquons qu'ici, la localisation d'apparence utilisée n'emploie pas de données Wi-Fi, comme dans la solution présentée dans le Chapitre 3), mais se base seulement sur des données visuelles.

Notre solution se différencie d'ORB-SLAM, OKVIS ou VINS car notre détection de points d'intérêt n'est pas contrôlée par un critère d'uniformité sur la répartition des caractéristiques visuelles dans l'image. Ce choix s'explique par plusieurs raisons (DYMCZYK et al., 2016). Tout d'abord, nous évitons ainsi le risque d'ajout de bruit dans les zones peu texturées de l'image, mais surtout, nous cherchons à favoriser les caractéristiques plus redétectables. En effet, la re-détectabilité des caractéristiques visuelles contribue à la qualité de la solution de SLAM.

Les points d'intérêt visuels sont donc extraits sur l'image stéréo et associés entre les images gauche et droite avant de passer à travers notre algorithme de SLAM. Dans la suite, pour éviter une confusion avec les points 3D de la carte, ces points seront nommés points 2D, ou projections, bien qu'on puisse leur attribuer une position 3D dans l'espace par rapport au capteur.

Notre solution utilise ainsi ce dispositif pour remplir sa carte d'amers 3D, et ces derniers sont utilisés en retour pour localiser les nouvelles acquisitions stéréo.

4.3.2 Représentation et structures de données

Notre solution de SLAM se base sur la caméra stéréo de Pepper, couplée à l'odométrie du robot. Les valeurs de calibration du dispositif stéréo, correspondant aux paramètres intrinsèques et à la position relative des deux caméras, sont disponibles pour chaque robot. Une requête de notre algorithme se constitue donc :

— d'une image stéréo, et ses paramètres intrinsèques ;

— d'une indication odométrique, fournissant un *a priori* sur la pose du capteur, donc sur ses paramètres extrinsèques.

Notre solution produit une carte composée d'*images-clés* et de *points 3D*. Comme l'illustre la Figure 4.8, ces images-clés et points 3D sont reliés entre eux par des arêtes de différentes natures, qui forment alors un graphe hybride. Cette carte s'exprime dans un repère de référence qui est arbitrairement initialisé à la pose du robot au départ d'une acquisition.

Les images-clés : Chaque image-clé est caractérisée par plusieurs informations :

- ses paramètres intrinsèques, issus de la calibration industrielle des caméras,
- ses paramètres extrinsèques, correspondant à sa pose dans le repère de référence,
- ses points caractéristiques, extraits sur l'image par une détection FAST et associés à leur descripteur binaire ORB,

- ses contraintes stéréo (associations de chaque point dans l'image gauche et droite, ainsi que la pose relative des deux caméras monoculaires),
- et finalement, son apparence visuelle, permettant la détection de fermeture de boucle par l'algorithme FAB-MAP (CUMMINS et NEWMAN, 2008).

Chaque image-clé peut être reliée à deux autres images-clés de la carte :

1. Une appelée *image-clé parente*, qui correspond à la dernière image-clé ajoutée à la carte lors d'une même acquisition. Une mesure odométrique constitue alors le lien entre ces deux images-clés.
2. Une autre appelée *image-clé de localisation*, qui correspond à l'image-clé de référence sélectionnée par notre algorithme pour déterminer la pose de l'image courante dans la carte (plus de détails sont donnés en partie 4.3.3.1).

Les points 3D : Les points 3D sont issus des caractéristiques visuelles extraites sur les images-clés. Un point 3D est caractérisé par :

- ses coordonnées 3D dans le repère de référence,
- les liens vers ses projections dans les images-clés qui l'observent,
- un lien vers l'image-clé depuis laquelle il a été initialisé.

Notons que, contrairement à (MUR-ARTAL, MONTIEL et TARDOS, 2015), aucun descripteur n'est associé à nos points 3D.

Aux images-clés et points 3D s'ajoutent deux autres structures de données :

- un graphe de co-visibilité, indiquant quelles images-clés possèdent des observations en commun ;
- un index inversé, utilisé par FAB-MAP pour accélérer l'étape de détection de fermeture de boucle, tel que présenté dans (CUMMINS et NEWMAN, 2011).

Ces deux structures de données peuvent être intégralement reconstruites à partir des informations caractérisant nos images-clés et points 3D. Leur rôle est d'accélérer le traitement d'une requête en optimisant les recherches qui lui sont nécessaires.

Les deux sous-parties qui suivent présentent respectivement les étapes de localisation et de cartographie de notre solution. Les explications s'attarderont en particulier sur les différences avec les méthodes de la littérature.

4.3.3 Localisation d'une image stéréo

Notre processus de localisation prend en entrée une requête (image stéréo et un *a priori* odométrique) et une carte de l'environnement [11]. Les caractéristiques visuelles FAST-ORB sont extraites en amont de notre processus et ont déjà été associées entre les

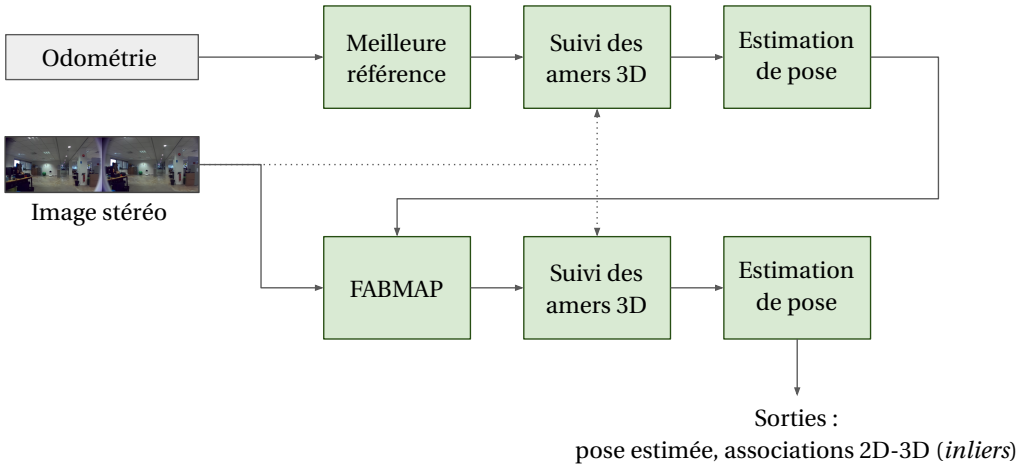


FIGURE 4.14: Étapes de localisation. Trois étapes sont identifiables : le choix de la meilleure référence, le suivi d'amers 3D et l'estimation de la pose dans la carte.

images gauche et droite. Notre localisation se divise en plusieurs étapes dont l'enchaînement est présenté en Figure 4.14.

Rappelons ici qu'un seul cœur est disponible pour faire tourner notre système de SLAM sur Pepper, donc aucune parallélisation n'est intéressante. Chacune de ces étapes est détaillée ci-dessous et l'Algorithme 2 résume leur enchaînement. Ainsi, les étapes de la Figure 4.14 correspondent aux fonctions suivantes dans l'Algorithme 2 (p. 119) :

— « Meilleure référence », correspond à la fonction `findClosestKeyframe`. Celle-ci prend en entrée une pose et renvoie l'image-clé cartographiée qui en est la plus proche dans la carte.

— « FABMAP », correspond à la fonction `fabmap`. Cette fonction consiste à associer à la requête l'image-clé cartographiée dont l'apparence visuelle est la plus similaire.

— « Suivi des amers 3D », correspond à la fonction `track3DPoints`. L'objectif de cette fonction est d'associer aux points d'intérêt de l'image requête des points 3D de la carte. Ainsi `track3DPoints`, génère des associations 2D-3D.

— « Estimation de pose », correspond à la fonction `optimizePose`. La fonction optimise la pose de la requête à partir des associations entre les points 3D de la carte et leur projection dans l'image requête. `optimizePose` retourne deux valeurs : un booléen indiquant la réussite ou l'échec de l'estimation, et la pose estimée.

4.3.3.1 Image-clé de localisation

Les amers 3D de notre système ne possèdent pas directement de descripteurs. Pour pouvoir les associer avec les caractéristiques visuelles extraites sur la requête, il faut donc passer par les images-clés de la carte. La première étape de notre localisation consiste ainsi à identifier une image de référence dans la carte, à partir de laquelle le traitement de notre requête va se baser. Nous appelons cette référence l'*image-clé de localisation*.

11. Cette carte peut évidemment être vide au départ d'une acquisition.

Pour l'identifier, notre algorithme utilise deux stratégies, visibles sur la Figure 4.14, et basées respectivement sur l'odométrie et l'apparence visuelle.

Utilisation de l'odométrie : Pour se localiser à l'aide de la vision, la requête doit partager des observations avec des images-clés cartographiées. L'idée est donc de chercher quelles images de la carte sont susceptibles de partager le plus d'observations avec la requête courante.

Un raisonnement consiste à comparer la pose supposée du capteur dans la carte avec celle de toutes les images-clés. Diverses métriques peuvent être utilisées, comme des critères de recouvrement volumique considérant les cônes de visibilités des diverses images stéréo. Notre solution recherche ainsi la meilleure référence en comparant les poses des images-clés à celle supposée de notre requête par une heuristique simple, basée sur la position du capteur et son orientation. L'emploi de cette heuristique conduit à la fois à des temps de calcul très raisonnables, mais aussi à la sélection de références pertinentes et cohérentes avec l'acquisition en cours. Cette sélection de la meilleure référence par l'odométrie est identifiée par la fonction `findClosestKeyframe` dans l'Algorithme 2.

Utilisation de l'apparence visuelle : L'objectif de l'apparence visuelle dans notre système est double :

1. détecter la dérive de la trajectoire estimée et introduire de nouvelles contraintes dans la carte permettant la correction de cette dernière,
2. récupérer une erreur de suivi ou d'estimation odométrique.

L'apparence visuelle peut alors être considérée comme une solution de secours, venant combler les lacunes de l'estimation visuelle basée sur l'odométrie. Notre idée est donc d'utiliser l'apparence visuelle pour détecter une fermeture de boucle que l'odométrie aurait manqué. Pour éviter que l'apparence visuelle sélectionne la même référence que celle choisie à l'odométrie, les images-clés partageant une information de covisibilité directe avec l'image-clé de référence identifiée par l'odométrie, sont soustraites de la recherche par apparence visuelle (ligne 10 de l'Algorithme 2).

Cette stratégie s'est révélée pertinente. En particulier dans les lignes droites où elle évite des calculs redondants, mais aussi lors de grosses dérives odométriques. En effet, l'apparence visuelle a tendance à choisir la dernière image-clé ajoutée, quand elles ont des champs de vue similaires, ce qui empêche souvent de détecter que le robot est revenu dans un lieu qu'il a déjà visité et qui est associé à une image-clé plus ancienne.

Dans l'Algorithme 2, cette recherche de référence basée sur l'apparence visuelle est identifiée par la fonction `fabmap`.

* * *

Une fois l'image de référence identifiée, notre processus de localisation peut suivre les points 3D de la carte dans la requête et estimer sa pose. Ces tâches (de suivi et d'estimation de pose) s'effectuent donc deux fois : à partir des références estimées par l'odo-

métrie et par l'apparence visuelle. L'*image-clé de localisation* est ensuite déterminée en fonction du succès de l'estimation :

— si la pose estimée à partir de la référence déterminée par l'apparence visuelle est considérée comme bonne, cette référence devient l'image-clé de localisation ;

— sinon, et si la pose estimée à partir de la référence déterminée par l'odométrie est bonne, cette dernière devient l'image-clé de localisation.

Ces conditions sont visibles dans l'Algorithme 2. Dans le cas où le système a échoué à localiser visuellement la requête, aucune image-clé de localisation ne lui est associée, et la pose de la requête est simplement celle estimée par l'odométrie (détails en 4.3.3.3).

4.3.3.2 Associations 2D-3D

Généralement, l'association entre des caractéristiques visuelles se fait à l'aide d'une distance entre leurs descripteurs. On retrouve ainsi de nombreuses applications basées sur l'association de points d'intérêt visuels entre deux images. Dans notre cas, nous souhaitons déterminer des associations entre les amers 3D de la carte et leur projection potentielle dans notre image requête.

Contrairement à ORB-SLAM, dans notre système les points 3D ne sont pas directement associés avec des descripteurs. Ce choix nous évite de devoir associer un descripteur à nos points 3D. En effet, un point 3D peut être associé à un grand nombre de descripteurs ORB, pouvant être remarquablement différents selon le point de vue des images-clés qui l'observent. Notre système passe alors par l'utilisation des images-clés pour suivre les amers 3D dans une requête, comme illustré en Figure 4.15

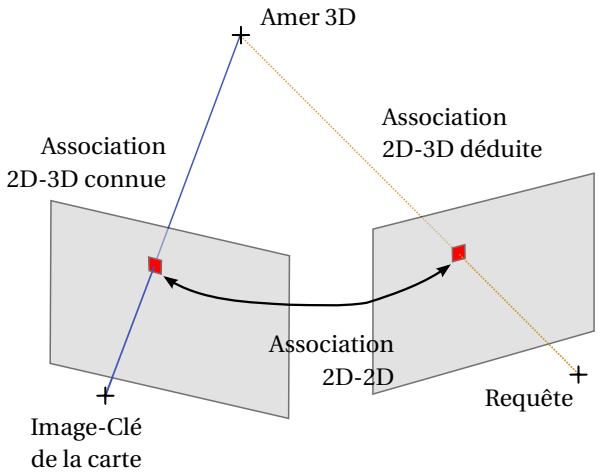


FIGURE 4.15: Association indirecte entre un point d'intérêt de la requête et un amer 3D. La déduction d'un lien entre un point 3D et sa projection dans la requête passe par l'association entre les points d'intérêt visuels de la requête et d'une image-clé de la carte.

Grâce aux arêtes conservées entre les amers 3D et leurs projections dans les images-clés, il est possible de suivre les points de la carte dans une requête. Pour cela, une association « brute-force » basée sur la distance de Hamming est réalisée entre les points d'intérêt de l'image requête et ceux d'une ou plusieurs images-clés.

Dans certains cas l'utilisation d'une unique image-clé référence se révèle être insuffisante. Par exemple, lorsque le champ de vue de l'image de requête chevauche ceux de deux images-clés, comme illustré en Figure 4.16, ou que les références contiennent des occultations.

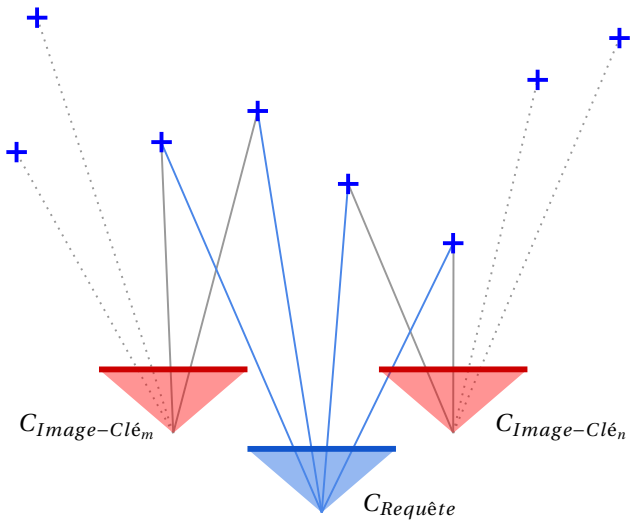


FIGURE 4.16: Intérêt de l'utilisation de plusieurs références : par exemple dans les cas où les images-clés ne se chevauchent pas, générer des associations à partir de plusieurs références peut se révéler pertinent.

La construction de l'ensemble des références utilisées pour le suivi suit les liens de co-observabilité directs entre les images-clés. L'image-clé de départ servant à déterminer ce sous-ensemble est identifiée par l'odométrie ou l'apparence visuelle, comme cela a été présenté plus haut (4.3.3.1). Toutes les images-clés partageant au moins une observation avec cette dernière sont ensuite récupérées grâce à la fonction `getReferences` dans l'Algorithme 2.

L'étape d'association, entre les points d'intérêt de la requête et ceux des images-clés identifiées par la fonction `getReferences`, cherche à tirer partie du fait qu'une connaissance *a priori* sur la pose du capteur dans la carte est disponible. Les images de référence extraites par `getReferences` sont classées grâce à l'heuristique utilisée dans `findClosestKeyframe`, de la plus proche à la plus lointaine. Les points d'intérêt de l'image requête sont ensuite associés itérativement avec ceux des images-clés de référence, en prenant en compte l'ordre dans lesquelles ces dernières ont été classées.

Remarquons ici que dans le cas où l'ensemble des références est construit à partir du résultat de `fabmap`, l'image de requête n'a pas d'*a priori* odométrique valide pour cette méthode. La pose supposée du capteur est alors fixée comme la pose de l'image-clé identifiée par l'apparence visuelle.

Dans l'Algorithme 2, toute cette étape de suivi correspond à la fonction `track3DPoints`.

4.3.3.3 Raffinement visuel de la pose

Grâce aux associations 2D-3D entre points d'intérêts de la requête et amers 3D de la carte, la pose de la caméra peut être estimée dans la carte. Pour cela, notre processus fait appel à l'optimisation présentée dans l'équation (4.5), qui cherche à minimiser l'erreur de reprojection des amers dans l'image requête en modifiant ses paramètres extrinsèques. Comme le système est rigide, l'optimisation peut se réaliser à partir de l'image issue d'une caméra monoculaire (gauche ou droite). Mais il est également possible de considérer le système stéréo complet. Dans notre cas, seule l'image gauche est utilisée durant la phase de localisation.

La bibliothèque d'optimisation utilisée est g^{2}o, présentée dans (KÜMMERLE et al., 2011), qui permet une intégration facile de ce type d'optimisation. Afin d'obtenir une meilleure solution, plusieurs optimisations sont réalisées successivement en supprimant à chaque itération les *outliers*. Cette approche est utilisée dans d'autres solutions comme ORB-SLAM, et est détaillée dans l'Algorithme 1.

En pratique, nous obtenons de meilleures estimations de poses en supprimant les associations aberrantes avant la première optimisation. Pour cela, nous utilisons la pose supposée de l'image dans la carte et appliquons un seuil très haut sur chaque reprojection afin d'écarter les *outliers* évidents. Quatre optimisations sont ensuite réalisées successivement, de la même manière que celle présentée dans l'Algorithme 1. Les critères de définition *inliers* / *outliers* deviennent de plus en plus restrictifs entre chacune de ces itérations.

Après ces quatre optimisations, les *inliers* sont à nouveau identifiés pour la pose estimée avec un critère plus permissif. Si plus d'un tiers des associations spécifiées en entrée est classé en tant qu'*inliers*, l'estimation de pose est considérée comme correcte et seules les associations correctes sont conservées. Si l'estimation de pose est considérée comme mauvaise, la localisation retourne la pose estimée par l'odométrie, et une valeur booléenne indiquant l'échec de l'estimation visuelle.

Cette étape d'optimisation complète correspond à la fonction `OptimizePose` dans l'Algorithme 2. La valeur booléenne retournée `Localized` renseigne sur le succès de l'estimation.

4.3.3.4 Processus complet

Le processus de localisation complet considère donc en entrée une requête, correspondant à une image stéréo (`Image`), une information *a priori* sur la pose du capteur apportée par l'odométrie (`Prior`), et la carte courante (`Map`). En sortie, la localisation renvoie la pose estimée C_{final} du dispositif stéréo dans la carte et les associations entre points d'intérêt dans l'image et amers 3D de la carte considérées comme correctes (les *inliers*, notées `CorrectAssociations`). L'enchaînement des différentes étapes participant à ce processus est détaillé dans l'Algorithme 2.

Le processus de localisation est entièrement indépendant du processus de cartographie : seule la carte est requise. Ce design a été, entre autre, motivé par les applications de Pepper en boutique. En effet, dans certaines situations il peut être avantageux d'in-

Algorithme 2 Processus de localisation

Entrées : Image, Prior, Map

// Initialisation des sorties.
1: $\texttt{Localization}_{\texttt{kf}}$ = null
2: CorrectAssociations = null
3: C_{final} = Prior
// Tentative de localisation initialisée par le prior odométrique.
4: $\texttt{Odometry}_{\texttt{kf}}$ = findClosestKeyframe (Prior, Map)
5: $\texttt{References}_{\texttt{o}}$ = getReferences ($\texttt{Odometry}_{\texttt{kf}}$, Map)
6: $\texttt{Associations}_{\texttt{o}}$ = track3DPoints (Image, $\texttt{References}_{\texttt{o}}$, Map)
7: [$\texttt{Localized}_{\texttt{o}}$, C_o^*] = optimizePose ($\texttt{Associations}_{\texttt{o}}$, Map)
// Prépare la carte d'apparence en fonction de la réussite de l'estimation.
8: ApparenceMap = Map
9: **if** $\texttt{Localized}_{\texttt{o}}$ **then**
// Suppression des références sélectionnées par l'odométrie.
10: ApparenceMap = Map $\setminus$ $\texttt{References}_{\texttt{o}}$
11: CorrectAssociations += $\texttt{Associations}_{\texttt{o}}$.*inliers*
12: $\texttt{Localization}_{\texttt{kf}}$ = $\texttt{Odometry}_{\texttt{kf}}$
13: $C_{final} = C_o^*$
14: **end if**
// Tentative de localisation initialisée par l'apparence visuelle.
15: $\texttt{Visual}_{\texttt{kf}}$ = fabmap (Image, ApparenceMap)
16: $\texttt{References}_{\texttt{v}}$ = getReferences ($\texttt{Visual}_{\texttt{kf}}$, Map)
17: $\texttt{Associations}_{\texttt{v}}$ = track3DPoints (Image, $\texttt{References}_{\texttt{v}}$, Map)
18: [$\texttt{Localized}_{\texttt{v}}$, C_v^*] = optimizePose ($\texttt{Associations}_{\texttt{v}}$, Map)
// Mise-à-jour des sorties.
19: **if** $\texttt{Localized}_{\texttt{v}}$ **then**
20: CorrectAssociations += $\texttt{Associations}_{\texttt{v}}$.*inliers*
21: $\texttt{Localization}_{\texttt{kf}}$ = $\texttt{Visual}_{\texttt{kf}}$
22: $C_{final} = C_v^*$
23: **end if**

Sorties : $\texttt{Localization}_{\texttt{kf}}$, CorrectAssociations, C_{final}

terdire au robot de cartographier son environnement en désactivant le processus de cartographie. Dans ce cas, le traitement de localisation peut être légèrement accéléré en ne considérant qu'une seule image du dispositif stéréo.

La création d'une carte reste néanmoins une étape essentielle. Sans carte, il est impossible de se localiser. Les différentes étapes de notre processus de cartographie sont présentées dans la sous-partie suivante.

4.3.4 Cartographie et optimisation

Une fois localisée, une requête peut être ajoutée à la carte, et devenir de cette manière une nouvelle image-clé. Pour cela, elle suit notre processus de cartographie, illustré en Figure 4.17.

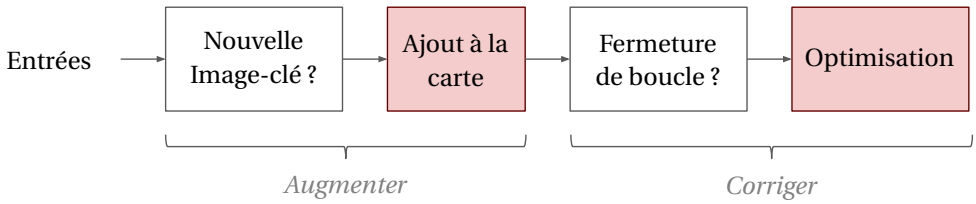


FIGURE 4.17: Étapes de cartographie. La cartographie est divisée en deux parties : la première ajoute de l'information à la carte ; la seconde corrige la carte courante.

Les entrées de ce processus de cartographie sont :

— la requête courante (l'image stéréo, ses points d'intérêts stéréo...),

— les résultats de la localisation (sa pose optimisée dans la carte, les liens entre projections et amers 3D *inliers*, son image-clé de localisation, ...),

— la transition odométrique estimée depuis la dernière image-clé ajoutée à la carte,

— la carte courante.

Le processus complet de cartographie se divise en deux parties : la première dont le rôle est d'augmenter la carte, et la seconde qui cherche à corriger les erreurs accumulées dans la carte.

4.3.4.1 Augmenter la carte

Le point de départ du processus de cartographie consiste à déterminer si la requête courante apporte une information supplémentaire à la carte, et du coup, si il est intéressant de la y ajouter. Pour cela, différents critères peuvent être mis en place, comme ceux présentées en 4.1.3.2 (qualité du suivi et distance temporelle). Notre solution reprend ces critères en les adaptant à notre contexte spécifique [12], notamment en considérant les requêtes où le suivi a été complètement perdu.

12. En particulier, le critère de distance temporelle minimale est adapté en prenant en compte notre fréquence d'acquisition plus basse.

Lorsqu'une requête est ajoutée à la carte, elle modifie cette carte de plusieurs façons : en éditant des liens avec des objets déjà présents dans la carte, et en ajoutant de nouveaux objets dedans. Ces nouveaux objets sont : la requête, transformée en une nouvelle image-clé, et les nouveaux points 3D observés par cette dernière.

Nouveaux points 3D : Contrairement aux approches monoculaires qui doivent se servir d'une mémoire temporaire pour initialiser la position d'amers 3D, le dispositif stéréo permet d'initialiser directement la position 3D des points d'intérêt par rapport au capteur. Ainsi, si une projection stéréo n'est associée à aucun point de la carte au cours de la localisation, elle initialise un nouvel amer 3D. Cet amer conserve alors une référence vers son image-clé d'initialisation.

Les projections stéréo qui ont été associées à des amers de la carte après la phase de localisation (4.3.3.3), éditent quant à elles de nouveaux liens avec les amers déjà cartographiés.

Création de nouvelles arêtes : En suivant les résultats de la localisation, de nouvelles arêtes visuelles sont éditées entre les amers de la carte et leur projection dans la requête courante. Mais d'autres types d'arêtes sont également présents dans la carte entre les images-clés.

Chaque nouvelle image-clé peut être potentiellement associée à deux autres images-clés de la carte :

— son image-clé parente, correspondant à la dernière image-clé ajoutée à la carte ;

— son image-clé de localisation, à partir de laquelle la requête a été localisée.

Dans les deux cas, les liens renseignent sur les poses relatives entre ces images et la nouvelle image-clé. Ces transformations sont estimées d'une part via l'odométrie (pour l'image-clé parente) et d'autre part, via l'estimation de pose visuelle (pour l'image-clé de localisation)

* * *

Le simple ajout d'images-clés n'est pas suffisant pour obtenir une carte cohérente. En pratique, de petites erreurs s'accumulent entre chaque image-clé conduisant à une erreur plus significative lors de longues trajectoires. De plus, il arrive que le suivi des amers échoue, notamment lors de fortes rotations du capteur, qui génèrent du flou ou des images dont les champs de vue ne se recouvrent pas. Dans ces cas, l'estimation de pose de la caméra stéréo ne repose que sur l'odométrie.

Pour maintenir une certaine cohérence de la carte, cette dernière doit être corrigée lorsque le système détecte une erreur trop importante.

4.3.4.2 Correction et maintenance

Afin d'obtenir une carte métrique cohérente, notre système emploie l'optimisation présentée en partie 4.2 pour corriger sa trajectoire. Cependant, cette optimisation reste coûteuse ; il est donc nécessaire de limiter ces appels. Pour cela, nous mettons en place différents critères permettant d'évaluer l'incohérence de la carte vis-à-vis de l'information apportée par une nouvelle requête.

Détecter une fermeture de boucle : Notre système considère qu'une optimisation est nécessaire lorsqu'une boucle est fermée. Ainsi, selon les résultats de la phase de localisation, une optimisation est réalisée si :

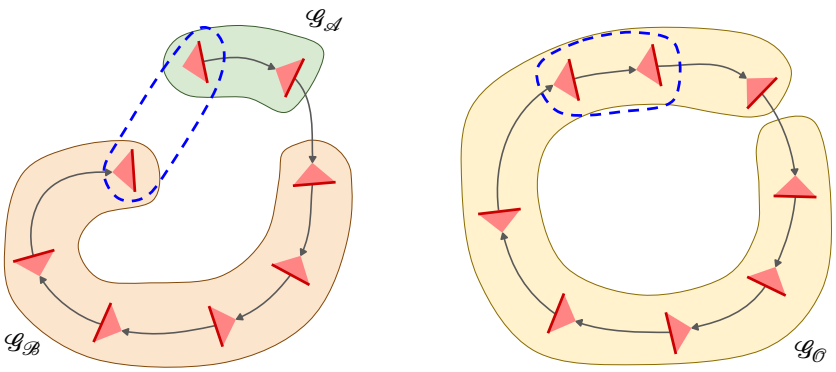
1. l'image-clé de localisation a été initialisée par FAB-MAP,

   ou,

2. les images-clés parente et de localisation ne partagent aucun lien de co-visibilité, même indirects.

La première condition gère les fermetures de boucle identifiées par FAB-MAP, et donc les cas où le système a fortement dérivé, mais aussi lorsque le suivi a été perdu ponctuellement.

La seconde condition correspond à la situation représentée en Figure 4.18, et gère les cas où le suivi visuel a été perdu, mais où la position estimée par l'odométrie reste bonne. En effet, dans cette situation, notre système basé sur l'heuristique de la fonction `findClosestKeyframe` dans l'Algorithme 2 choisit bien la meilleure référence, et l'utilisation de l'apparence ne permet pas d'estimer une autre solution.



Détection d'une fermeture de boucle

Graphe optimisé

FIGURE 4.18: Intérêt de la fermeture de boucle. Le suivi visuel est perdu au cours du processus de SLAM, générant deux graphes indépendants $\mathcal{G}_\mathcal{A}$ et $\mathcal{G}_\mathcal{B}$. Une fermeture de boucle est détectée quand une requête avec un parent dans $\mathcal{G}_\mathcal{B}$ est localisée par rapport à une image-clé de $\mathcal{G}_\mathcal{A}$. Des contraintes visuelles sont alors ré-introduites entre les deux graphes visuels pour les unifier.

Lors de la détection d'une fermeture de boucle, le système prend conscience que certains points 3D ont été observés en double. En effet, certains points d'intérêt 2D des images-clés se retrouvent associés à plusieurs points 3D. Notre système fusionne alors

ces points pour éviter les doublons dans la carte. Cette fusion permet ainsi de réintroduire des contraintes visuelles entre les différentes parties du graphe qui ne partageaient pas d'arête visuelle jusqu'alors.

Optimisation globale : Au cours d'une optimisation par ajustement de faisceaux, les contraintes entre images-clés sont induites par les observations qu'elles partagent. Cependant, si certaines parties du graphe n'ont aucune observation en commun, la carte est divisée en de multiples graphes visuels disjoints. Aucune connexion cohérente entre eux ne peut être assurée pendant l'optimisation.

Notre solution consiste à reconnecter ces graphes grâce aux mesures de l'odométrie. Arêtes odométriques et visuelles sont mêlées dans une seule et même optimisation, comme introduit en partie 4.2. Cependant, lors de grosses dérives de la trajectoire, l'optimisation présentée dans l'équation (4.7) peut ne pas converger vers la solution optimale. Pour bien fonctionner, cette dernière nécessite un bon état initial. L'optimisation globale de notre carte se divise donc en deux temps.

Premièrement, notre système cherche à s'approcher d'un état plus proche de la solution optimale en ne considérant que les arêtes entre images-clés. Chaque image-clé C est potentiellement reliée à deux autres images-clés : avec son image-clé parente $C_{Parente}$ par un lien odométrique ; avec son image-clé de localisation C_{Loc} par rapport à laquelle elle a été localisée. Ces différentes arêtes sont donc utilisées lors d'une pré-optimisation, se formulant par :

$$\begin{aligned}\mathscr{C}^* = \underset{\mathscr{C}}{\operatorname{argmin}} \sum_{C_i \in \mathscr{C}} & e_\Delta(C_i, C_{i_{Parente}})^T \Omega_{D_i} e_\Delta(C_i, C_{i_{Parente}}) \\ & + e_\Delta(C_i, C_{i_{Loc}})^T \Omega_{\Phi_i} e_\Delta(C_i, C_{i_{Loc}})\end{aligned} \tag{4.16}$$

où $\mathscr{C}$ est l'ensemble des positions des images-clés de la carte, Ω_D et Ω_Φ sont les matrices d'information respectivement associées aux transformations odométrique et visuelle estimées, et e_Δ la fonction d'erreur entre deux poses caméras. e_Δ retourne dans notre intégration un vecteur 7D, qui traduit les différences entre les {translation ;rotations} mesurées et attendues [13]. Chaque point 3D de la carte est ensuite repositionné en fonction du mouvement de son image-clé d'initialisation. Comme dans (KONOLIGE et AGRAWAL, 2008), cette étape permet à notre système de générer une meilleure approximation de la solution avant de lui appliquer une seconde optimisation.

La seconde optimisation est plus globale et réintroduit les contraintes issues des points 3D. Cette optimisation cherche à minimiser à la fois les erreurs associées aux transitions odométriques mesurées entre image-clé et image-clé parente, et les erreurs de reprojections associées aux amers 3D cartographiés. Ce problème a été présenté en partie 4.2 et correspond au problème d'optimisation introduit par l'équation (4.7) (p. 106).

13. La rotation étant exprimée par un quaternion.

Ces deux optimisations sont réalisées grâce à la librairie g^{2}o (KÜMMERLE et al., 2011), projet *open-source* d'outils d'optimisation dédiés aux problèmes de SLAM [14].

4.4 Évaluation expérimentale

Dans cette partie, nous présentons une évaluation de notre solution de SLAM métrique visuel. Dans un premier temps, les conditions d'acquisition seront détaillées en 4.4.1. Une évaluation qualitative sera ensuite réalisée en 4.4.2 et les temps de calcul mesurés seront discutés en 4.4.3.

4.4.1 Conditions expérimentales

Notre algorithme a été évalué à partir de données acquises par plusieurs robots Pepper. Ces robots sont équipés d'un dispositif stéréo, composé de deux caméras monoculaires placées dans chaque œil avec une distance focale et une distance de mise au point fixes. Chaque dispositif stéréo est calibré sur les lignes de production de Pepper, et ses paramètres sont accessibles sur le robot.

Les images monoculaires utilisées sont de dimensions 640×360. L'image stéréo résultante est la concaténation de ces deux images monoculaires. Dans notre intégration, pour limiter la charge computationnelle, les images stéréo sont acquises toutes les 0,5 s. Avec une telle fréquence d'acquisition, le suivi visuel est souvent perdu [15] (Figure 4.19).

Nos acquisitions sont réalisées dans les *open-spaces* des locaux de SoftBank Robotics Europe. Aucune contrainte spécifique n'est appliquée pendant ces expériences. Ainsi, on retrouve dans les données collectées des problèmes typiques tels que des occultations, des objets déplacés ou en déplacement, ainsi que des variations d'intensité lumineuse.

Plusieurs acquisitions sont réalisées dans cet environnement, totalisant 950 m parcourus. Ces acquisitions couvrent de nombreux scénarios comme : différentes vitesses du robot, les variations des conditions lumineuses selon l'heure d'acquisition, et des mouvements du robot variés. Chaque acquisition contient au moins une fermeture de boucle.

Dans toutes nos expériences, notre solution a réussi à fermer la boucle et optimiser la carte de façon cohérente. Les deux sous-sections qui suivent présentent les résultats obtenus lors d'une expérience représentative, pour donner au lecteur une évaluation qualitative de notre solution.

4.4.2 Résultats qualitatifs

Nous n'avons pas été capables de comparer quantitativement notre système aux méthodes classiques de SLAM visuel (ou visuel-inertiel) à cause des spécificités de nos acquisitions. En effet, aucun algorithme prêt-à-l'emploi trouvé dans la littérature n'a pu délivrer des performances suffisamment bonnes sur les données acquises par Pepper

14. https://github.com/RainerKuemmerle/g2o

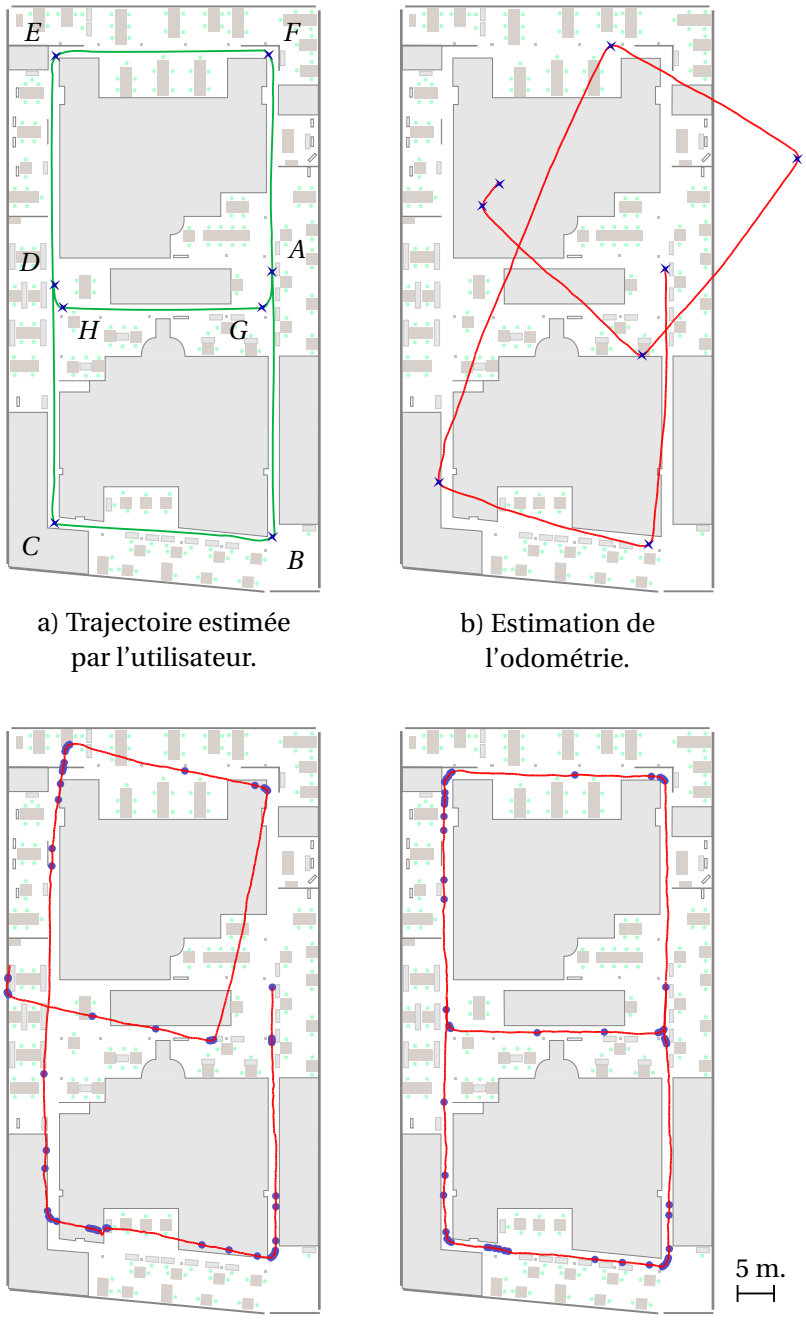
15. À titre indicatif, les applications de SLAM visuel actuelles se permettent de monter jusqu'à 30 Hz en embarqué. La base de données du challenge KITTI (GEIGER et al., 2013), souvent utilisée dans ce domaine, propose des données synchronisées à 10 Hz, soit cinq fois plus rapidement que dans notre contexte.

FIGURE 4.19: Images stéréo (issues de la caméra gauche) acquises à 2 Hz lors de rotations du robot. Les champs de vue de deux images consécutives se recouvrent peu. Le flou de bougé induit par la rotation entraîne une difficulté supplémentaire pour suivre les points d'intérêt.

pour présenter une comparaison pertinente avec notre travail. Les cas critiques dont souffrent ces algorithmes sont principalement liés aux nombreuses pertes de suivi causées par notre acquisition à basse fréquence des capteurs visuel et inertiel, ainsi que les limitations en puissance computationnelle.

Les performances de notre solution sont montrées en Figure 4.20, pour une acquisition représentative. La trajectoire de cette expérience fait 227 m de long et contient deux fermetures de boucle. Déplacé à une vitesse moyenne de 0,4 $m.s^{-1}$, 885 images stéréo sont acquises. Nous traçons les différentes trajectoires estimées sur un plan d'architecte du bâtiment afin de fournir une appréciation qualitative de nos résultats (Figure 4.20).

Une nette amélioration de la trajectoire estimée est visible pour le chemin prenant



a) Trajectoire estimée par l'utilisateur.

b) Estimation de l'odométrie.

c) Odométrie et stéréo-vision sans fermeture de boucle.

d) Odométrie et stéréo-vision avec fermeture de boucle et optimisation.

FIGURE 4.20: Évaluation qualitative des performances de notre algorithme. a) montre le chemin estimé par l'utilisateur ; quelques positions clés sont annotées par des lettres. La trajectoire réalisée par le robot suit : $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow A \rightarrow G \rightarrow H \rightarrow D$. Deux fermetures de boucle sont présentes en A et D. b) présente la trajectoire estimée par l'odométrie. c) est le résultat de notre algorithme en désactivant la recherche de fermeture de boucle. d) correspond aux performances de notre algorithme complet : combinant données visuelles et odométriques, avec la gestion de fermeture de boucle (recherche et correction). Dans c) et d), des cercles bleus mettent en évidence les positions où le suivi visuel est perdu.

en compte l'information visuelle par rapport à celui estimé par l'odométrie seule. La trajectoire estimée par l'odométrie seule est particulièrement déformée dans les lignes droites, ce qui génère une carte non-cohérente avec les dimensions métriques réelles de l'environnement visité. L'utilisation des amers visuels a ainsi permis d'obtenir des lignes plus droites et une trajectoire plus proche de celle vraiment réalisée.

Cependant, de nombreuses pertes de suivi surviennent et affectent la qualité de la carte. Dans notre expérience, le suivi est perdu 62 fois. Ces échecs de suivi sont particulièrement localisés au niveau de rotations rapides du robot. En effet, la fréquence d'acquisition ne permet pas d'assurer que les champs de vue de deux images consécutives se recouvrent dans de telles situations. Cela s'observe sur les Figures 4.20 c) et 4.20 d) : le suivi est perdu au niveau de chaque virage important de la trajectoire. À cause de ces pertes, l'erreur accumulée devient suffisamment importante pour perturber la forme générale de la carte.

Les deux fermetures de boucle en A et D sont bien détectées par notre algorithme. Ces détections déclenchent notre processus d'optimisation pour corriger la carte. Dans la Figure 4.20 d), les optimisations successives retournent une carte finale cohérente avec la forme de la trajectoire réalisée.

Comme évaluation de la cohérence de la carte générée, nous calculons les erreurs relatives sur les distances CE et BE, comparées à la vérité terrain établie à l'aide d'un télémètre laser. Ces erreurs sont présentées dans le Tableau 4.1.

Le choix de ces distances est motivé par le fait que chacune met en lumière différents types de correction. CE correspond ainsi à une ligne droite, l'erreur qui lui est associée indique alors comment notre algorithme a géré une correction dans un trajet linéaire. BE est une diagonale du trajet parcouru, elle est alors également associée à la correction des angles dans notre exploration.

	$erreur(CE)$ (%)	$erreur(BE)$ (%)
b) Odométrie Seule	1,4	21,1
c) Vision et Odométrie **sans** fermeture de boucle	0,4	2,1
d) Vision et Odométrie **avec** fermeture de boucle	0,1	0,5

TABLEAU 4.1: Comparaison des erreurs sur les distances dans la carte estimée par : b) l'odométrie seule, c) la combinaison de données visuelles et odométriques sans activer la fermeture de boucle, d) la combinaison de données visuelles et odométriques avec la détection de fermeture de boucle et la correction de la carte activées.

4.4.3 Temps de calcul sur Pepper

Tous les algorithmes présentés dans ce chapitre ont été testés directement sur le CPU de Pepper, un processeur quatre cœurs Atom E3845, tournant à 1,91 GHz. Pepper est vendu comme un robot interactif social et est utilisé quotidiennement dans de nom-

breux magasins. Dans une utilisation typique, au moins trois cœurs du CPU sont réquisitionnés pour le calcul de tâches nécessaires comme le contrôle et la planification de mouvements, la détection d'humains ou la reconnaissance vocale. Un seul cœur est alors disponible pour toute solution logicielle de localisation destinée à être déployée sur les robots.

Nous avons estimé empiriquement qu'acquérir et traiter les images stéréo toutes les 0,5 s constitue un bon compromis entre la qualité de la carte estimée et l'empreinte computationnelle de l'algorithme. Sur Pepper, une telle fréquence d'acquisition permet un traitement de localisation et cartographie en temps-réel.

Dans l'expérience présentée en Figure 4.20, le temps de traitement du processus complet de localisation (Figure 4.14) ne dépasse jamais 346 ms pour 800 images-clés. En ajoutant les phases d'augmentation de la carte et de détection de fermeture de boucle présentées en Figure 4.17, la durée maximale de traitement d'une requête sur cette expérience passe à 380 ms, pour une durée moyenne de 200 ms. L'acquisition d'images requêtes à 2 Hz permet donc à l'algorithme de fonctionner avec une marge de sécurité.

90 % de ce temps de calcul est pris par le processus de localisation. Cependant, toutes les fonctions présentées dans l'Algorithme 2 n'ont pas le même impact et n'évoluent pas de la même façon au cours de l'expérience. Nous classons ici ces étapes, de la plus consommatrice à la plus légère en temps de calcul, suivant les mesures faites pendant l'expérience illustrée en Figure 4.20 :

- `fabmap` : 78,22 % du processus de localisation.

  La fonction la plus gourmande en temps de calcul est la fonction de détection de fermeture de boucle basée sur l'apparence visuelle. FAB-MAP, présenté en partie 3.2, se divise en deux blocs : le premier qui génère le vecteur d'apparence, et le second qui associe à chaque image-clé un score de vraisemblance. La durée de création du vecteur d'apparence est proportionnelle au nombre de points d'intérêt détectés dans l'image requête [16] et dure en moyenne 31,97 ms dans cet environnement. Le calcul des vraisemblances augmente, lui, de manière logarithmique avec le nombre de lieux présents dans la carte, comme expliqué dans (CUMMINS et NEWMAN, 2011). Dans notre expérience, pour 800 images-clés, il requiert 69,15 ms.

- `track3DPoints` : 2× 6,18 % du processus de localisation.

  Le suivi des points dans l'image dépend de nombreux facteurs si bien qu'il est compliqué de prédire son comportement. En effet, il dépend à la fois du nombre de références utilisées, du nombre points d'intérêt extraits sur l'image courante, de s'ils ressemblent fortement à des points d'intérêt d'images-clés estimées proches, etc. Dans notre expérience, la durée la plus longue associée à cette fonction est de 32,72 ms.

- `optimizePose` : 2× 2,53 % du processus de localisation.

  Le raffinement de la pose correspond à quatre optimisations successives (4.3.3.3) faites par g^{2}o (KÜMMERLE et al., 2011). Le durée maximale passée dans cette fonc-

16. Notons que pour éviter une redondance de description, seule l'image gauche est utilisée pour construire ce vecteur d'apparence.

tion est de 12,92 ms.

- `getReferences` : 2× 1,92 % du processus de localisation.

  Déterminer l'ensemble des références à utiliser est rapide grâce à la maintenance d'un graphe de co-observabilité et ne dépasse jamais 16,63 ms.

- `findClosestKeyframe` : 0,52 % du processus de localisation.

  Cette fonction croît linéairement avec le nombre d'images-clés dans la carte. Cependant, la simplicité calculatoire de l'heuristique utilisée permet de ne jamais excéder 1,74 ms dans notre expérience.

L'optimisation de la carte est plus consommatrice en temps de calcul. Dans l'expérience présentée, l'optimisation de la carte prend en effet jusqu'à 17,49 s. La fermeture de boucle détectée en *A* sur la Figure 4.20 est corrigée en 13,07 s, et 17,49 s sont nécessaires pour celle détectée en *D*. Des deux optimisations nécessaires à la correction globale de la carte, celle présentée dans l'équation (4.16) (p. 123), pouvant s'apparenter à l'optimisation de squelette de (KONOLIGE et AGRAWAL, 2008), prend 19 % du processus complet d'optimisation. Au cours du processus complet d'optimisation, 98 % du temps de calcul est passé dans les fonctions d'optimisation de g^{2}o (KÜMMERLE et al., 2011).

L'optimisation complète de la carte est donc trop lourde pour tourner en temps-réel sur le robot. Pour éviter au robot de devoir s'arrêter, la correction de la carte est réalisée dans une tâche parallèle quand des ressources sont disponibles. En attendant, le robot continue de se localiser dans sa carte sans correction de fermeture de boucle.

4.5 Conclusion

Dans ce chapitre, nous avons présenté notre solution de SLAM métrique sur Pepper, utilisant la caméra stéréo et l'odométrie du robot. Contrairement aux approches de l'état-de-l'art qui peuvent se permettre des fréquences de fonctionnement tournant aux alentours de 20 à 30 Hz, les contraintes computationnelles de notre plate-forme ne permettent pas de traiter sereinement plus de deux requêtes par seconde. Cette fréquence basse de fonctionnement tient en échec les algorithmes classiques de SLAM par ajustement de faisceaux en nuisant à la qualité de suivi des amers 3D. De plus, même avec une fréquence de fonctionnement plus élevée, l'environnement non-contraint ne peut assurer une présence d'amers visuels dans chaque image. Ainsi, dans des cas d'environnements peu texturés, d'occultations causées par des interactions ou d'un important flou de bougé, le suivi des amers 3D est inévitablement perdu.

Notre contribution consiste en la création d'un processus de SLAM complet qui prend en compte l'information issue de l'odométrie du robot pour compenser ces pertes de suivi visuel. Notre approche détecte les fermetures de boucle et corrige la carte générée grâce à une optimisation combinant données visuelles et odométriques.

La solution de SLAM créée a été testée sur Pepper et s'est montrée compatible avec un usage temps-réel sur le robot. L'optimisation globale du graphe reste néanmoins coûteuse mais peut se permettre d'être réalisée quand des ressources sont disponibles.

Chapitre 5

Conclusion et perspectives

Le déploiement massif de robots mobiles dans différents secteurs comme la vente, la finance, le tourisme, ou même chez les particuliers, soulève de nombreuses problématiques. Parmi elles, celle de la localisation autonome devient essentielle pour permettre à ces robots de se déplacer dans l'espace sans risquer de se perdre.

Étudiée depuis les années 80, la question de la localisation dans un environnement inconnu non-contraint va de pair avec celle de la cartographie. De nombreuses solutions ont été proposées dans la littérature et regroupées sous l'acronyme de SLAM pour *Simultaneous Localization And Mapping*, soit la localisation et cartographie simultanées.

Cette thèse a cherché à donner au robot Pepper la capacité de se localiser dans son environnement. Réalisée dans le cadre d'un dispositif CIFRE avec l'entreprise SoftBank Robotics Europe, elle a proposé des solutions aux problèmes que les approches de l'état-de-l'art rencontrent dans une utilisation classique de Pepper. Ainsi, nos travaux ont été guidés par un contexte applicatif et les contraintes d'industrialisation d'une solution logicielle de localisation pour ces robots.

Pour compenser le faible champ perceptuel des capteurs lasers présents sur Pepper, nos recherches se sont tournées vers l'emploi de la vision pour résoudre le problème de localisation et cartographie simultanées. Cependant, l'utilisation de la vision seule s'est révélée être insuffisante dans notre contexte et nos travaux ont cherché à tirer parti des différentes sources d'information présentes sur le robot.

Dans un premier temps, nous nous sommes intéressés au problème de la relocalisation du robot dans une carte, sans connaissance *a priori* sur sa pose. Ce problème est bien connu dans la littérature et les méthodes de relocalisation basée sur l'apparence visuelle sont nombreuses. Cependant, ces méthodes sont tenues en échec par les environnements visuellement redondants. Une autre approche de la littérature repose sur l'utilisation d'empreintes Wi-Fi qui caractérisent les signaux Wi-Fi perçus à diverses poses de l'environnement. Si l'unicité des points d'accès Wi-Fi permet de délimiter l'erreur de ce type de localisation, la précision de ces approches reste insuffisante dans notre contexte.

Notre solution cherche à profiter de la synergie entre la localisation visuelle, précise mais pouvant être induite en erreur, et la localisation Wi-Fi, dont l'erreur est bornée mais la précision moindre. Nous proposons de combiner données Wi-Fi et visuelles dans une approche probabiliste. Diverses stratégies de fusion sont envisagées et une stratégie de fusion précoce est proposée. Ces stratégies ont été évaluées expérimentalement dans plusieurs environnements. Dans nos expériences, notre stratégie de fusion précoce a été la seule approche à toujours avoir amélioré les performances de relocalisation visuelle sans les détériorer de façon significative.

Dans un second temps, la mise en place d'une solution complète de SLAM métrique a été étudiée. Notre solution s'est basée sur l'utilisation de la caméra stéréo présente sur Pepper. Ce type de capteur a déjà été employé dans diverses méthodes de SLAM visuel de la littérature. Ces méthodes reposent sur le suivi de primitives visuelles pour estimer la position d'une image dans l'espace. Dans notre contexte, plusieurs contraintes nuisent à la qualité de ce suivi. Les rotations rapides du robot, les occultations causées par des interactions et les limites de la puissance computationnelle disponible s'additionnent et conduisent à l'acquisition d'images floues ou qui ne se recouvrent pas. Il est alors impossible d'assurer un suivi de primitives visuelles, ce qui rend inutilisables les solutions courantes de l'état-de-l'art.

Nous proposons d'adapter l'approche des solutions de SLAM visuel basées sur l'optimisation par ajustement faisceaux à notre contexte. Ces méthodes maintiennent une carte sous la forme d'un graphe d'images-clés reliées entre elles par les observations qu'elles ont en commun. L'échec de suivi entre plusieurs images-clés génèrent donc des graphes visuels disjoints ne partageant aucune observation. Pour reconnecter ces graphes, notre solution s'aide des données odométriques collectées par le robot. Cette stratégie permet à notre système de fonctionner à basse fréquence et de supporter de nombreuses pertes de suivi. Notre solution a été testée à travers différentes acquisitions contenant chacune, au moins une fermeture de boucle. Dans toutes nos expériences, le robot s'est montré capable de détecter et de fermer les différentes boucles. L'utilisation de cette solution basée sur la stéréo-vision et l'odométrie du robot a ainsi démontré une nette amélioration de la cohérence et de la précision de la carte construite par rapport à l'utilisation de l'odométrie seule.

Les travaux réalisés dans cette thèse ont ainsi permis la création d'une solution de localisation et cartographie compatible avec les contraintes d'utilisation du robot Pepper. Ces travaux ont montré l'intérêt de considérer différentes sources de données pour résoudre le problème de SLAM sur Pepper et de profiter des synergies entre chacune d'entre elles.

Plusieurs perspectives d'évolution de ces travaux sont envisageables. Pour commencer, les solutions de relocalisation et de construction de la carte présentées dans ce manuscrit ont été étudiées séparément. Une première étape consisterait donc à inclure la relocalisation basée sur la combinaison des données visuelles et Wi-Fi dans notre solution complète de SLAM.

Une autre réflexion pourrait s'orienter sur la question du SLAM long-terme. Dans le contexte applicatif du robot Pepper, une solution de SLAM long-terme permettrait de ne plus avoir à dissocier les phases d'exploration et de fonctionnement. Le robot serait alors constamment utilisé dans un mode SLAM. Cette utilisation nécessite de réfléchir à la manière dont doit être gérée la carte. En effet, les changements cycliques ou causés par l'évolutivité de l'environnement génèrent des variations qui s'accumulent et rendent les cartes long-terme difficilement maintenables. Une piste intéressante de recherche serait d'associer à chaque lieu une mémoire, qui n'encoderait que les variations pertinentes de l'environnement.

Bibliographie

ADARVE, Juan David et al. (2012). « Computing occupancy grids from multiple sensors using linear opinion pools ». In : *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, p. 4074–4079.

AGARWAL, Sameer et Keir MIERLE (2012). « Ceres solver: Tutorial & reference ». In : *Google Inc* 2, p. 72.

ANGELI, Adrien et al. (2008). « Real-time visual loop-closure detection ». In : *International Conference on Robotics and Automation*, p. 1842–1847.

APARICIO, Sofía et al. (2008). « A fusion method based on bluetooth and wlan technologies for indoor location ». In : *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*. IEEE, p. 487–491.

ARROYO, Roberto et al. (2016). « Fusion and binarization of CNN features for robust topological localization across seasons ». In : *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, p. 4656–4663.

BABINEC, Andrej et al. (2014). « Visual localization of mobile robot using artificial markers ». In : *Procedia Engineering* 96, p. 1–9.

BADINO, Hernán, Daniel HUBER et Takeo KANADE (2012). « Real-time topometric localization ». In : *2012 IEEE International Conference on Robotics and Automation*. IEEE, p. 1635–1642.

BAILEY, Tim et al. (2006). « Consistency of the EKF-SLAM algorithm ». In : *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, p. 3562–3568.

BARFOOT, Timothy D (2005). « Online visual motion estimation using fastslam with sift features ». In : *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, p. 579–585.

BAY, Herbert, Tinne TUYTELAARS et Luc VAN GOOL (2006). « Surf: Speeded up robust features ». In : *European conference on computer vision*. Springer, p. 404–417.

BAZEILLE, Stéphane et David FILLIAT (2011). « Incremental topo-metric slam using vision and robot odometry ». In : *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, p. 4067–4073.

BESCÓS, Berta et al. (2018). « DynSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes ». In : *arXiv preprint arXiv:1806.05620*.

BISWAS, Joydeep et Manuela VELOSO (2010). « Wifi localization and navigation for autonomous indoor mobile robots ». In : *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, p. 4379–4384.

BISWAS, Joydeep et Manuela VELOSO (2013). « Multi-sensor mobile robot localization for diverse environments ». In : *Robot Soccer World Cup*. Springer, p. 468–479.

BLAER, Paul et Peter ALLEN (2002). « Topological mobile robot localization using fast vision techniques ». In : *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*. T. 1. IEEE, p. 1031–1036.

BLOESCH, Michael et al. (2015). « Robust visual inertial odometry using a direct EKF-based approach ». In : *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, p. 298–304.

BOONSRIWAI, Sujittra et Anya APAVATJRUT (2013). « Indoor WIFI localization on mobile devices ». In : *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2013 10th International Conference on*. IEEE, p. 1–5.

CALONDER, Michael et al. (2010). « Brief: Binary robust independent elementary features ». In : *European conference on computer vision*. Springer, p. 778–792.

CARLEVARIS-BIANCO, Nicholas et Ryan M EUSTICE (2012). « Learning temporal coobservability relationships for lifelong robotic mapping ». In : *IROS Workshop on Lifelong Learning for Mobile Robotics Applications*, p. 15.

CASTELLANOS, Jose A et al. (1999). « The SPmap: A probabilistic framework for simultaneous localization and map building ». In : *IEEE Transactions on robotics and Automation* 15.5, p. 948–952.

CHATILA, Raja et Jean-Paul LAUMOND (1985). « Position referencing and consistent world modeling for mobile robots ». In : *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*. T. 2. IEEE, p. 138–145.

CHATILA, Raja et Philippe MOUTARLIER (1989). « Stochastic multisensor data fusion for mobile robot location and environment modelling ». In : *Proceedings of the 5th International Symposium of Robotics Research*, p. 207–216.

CHOW, C et Cong LIU (1968). « Approximating discrete probability distributions with dependence trees ». In : *IEEE transactions on Information Theory* 14.3, p. 462–467.

CIVERA, Javier, Andrew J DAVISON et JM Martinez MONTIEL (2008). « Inverse depth parametrization for monocular SLAM ». In : *IEEE transactions on robotics* 24.5, p. 932–945.

CIVERA, Javier et al. (2010). « 1-Point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry ». In : *Journal of Field Robotics* 27.5, p. 609–631.

COLLETT, Matthew (2010). « How desert ants use a visual landmark for guidance along a habitual route ». In : *Proceedings of the National Academy of Sciences*, p. 201001401.

CONCHA, Alejo et al. (2016). « Visual-inertial direct SLAM ». In : *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, p. 1331–1338.

COSTANTE, Gabriele et al. (2016). « Exploring Representation Learning With CNNs for Frame-to-Frame Ego-Motion Estimation. » In : *IEEE robotics and automation letters* 1.1, p. 18–25.

CSORBA, Michael (1998). « Simultaneous localisation and map building ». Thèse de doct. University of Oxford.

CUI, Youjing et Shuzhi Sam GE (2003). « Autonomous vehicle positioning with GPS in urban canyon environments ». In : *IEEE transactions on robotics and automation* 19.1, p. 15–25.

CUMANI, Aldo et al. (2004). « Integrating monocular vision and odometry for SLAM. » In : *WSEAS Transactions on Computers* 3.3, p. 625–630.

CUMMINS, Mark et Paul NEWMAN (2008). « FAB-MAP: Probabilistic localization and mapping in the space of appearance ». In : *The International Journal of Robotics Research* 27.6, p. 647–665.

— (2011). « Appearance-only SLAM at large scale with FAB-MAP 2.0 ». In : *The International Journal of Robotics Research* 30.9, p. 1100–1123.

DAVISON, Andrew J (2003). « Real-time simultaneous localisation and mapping with a single camera ». In : *null*. IEEE, p. 1403.

DAVISON, Andrew J et al. (2007). « MonoSLAM: Real-time single camera SLAM ». In : *IEEE Transactions on Pattern Analysis & Machine Intelligence* 6, p. 1052–1067.

DE MAESSCHALCK, Roy, Delphine JOUAN-RIMBAUD et Désiré L MASSART (2000). « The mahalanobis distance ». In : *Chemometrics and intelligent laboratory systems* 50.1, p. 1–18.

DEANS, Matthew et Martial HEBERT (2001). « Experimental comparison of techniques for localization and mapping using a bearing-only sensor ». In : *Experimental Robotics VII*. Springer, p. 395–404.

DELLAERT, Frank (2012). *Factor graphs and GTSAM: A hands-on introduction*. Rapp. tech. Georgia Institute of Technology.

DISSANAYAKE, MWM Gamini et al. (2001). « A solution to the simultaneous localization and map building (SLAM) problem ». In : *IEEE Transactions on robotics and automation* 17.3, p. 229–241.

DOUCET, Arnaud, Simon GODSILL et Christophe ANDRIEU (2000). « On sequential Monte Carlo sampling methods for Bayesian filtering ». In : *Statistics and computing* 10.3, p. 197–208.

DURRANT-WHYTE, Hugh et Tim BAILEY (2006). « Simultaneous localization and mapping: part I ». In : *IEEE robotics & automation magazine* 13.2, p. 99–110.

DURRANT-WHYTE, Hugh et al. (2003). « A bayesian algorithm for simultaneous localisation and map building ». In : *Robotics Research*. Springer, p. 49–60.

DURRANT-WHYTE, Hugh F (1988). « Uncertain geometry in robotics ». In : *IEEE Journal on Robotics and Automation* 4.1, p. 23–31.

DYMCZYK, Marcin et al. (2015). « Keep it brief: Scalable creation of compressed localization maps ». In : *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, p. 2536–2542.

DYMCZYK, Marcin et al. (2016). « Map summarization for tractable lifelong mapping ». In : *RSS Workshop*.

EADE, Ethan et Tom DRUMMOND (2007). « Monocular SLAM as a graph of coalesced observations ». In : *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, p. 1–8.

ELFES, Alberto (1989). « Using occupancy grids for mobile robot perception and navigation ». In : *Computer* 6, p. 46–57.

ENGEL, Jakob, Thomas SCHÖPS et Daniel CREMERS (2014). « LSD-SLAM: Large-scale direct monocular SLAM ». In : *European Conference on Computer Vision*. Springer, p. 834–849.

ENGEL, Jakob, Jörg STÜCKLER et Daniel CREMERS (2015). « Large-scale direct SLAM with stereo cameras ». In : *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, p. 1935–1942.

ENGELHARD, Nikolas et al. (2011). « Real-time 3D visual SLAM with a hand-held RGB-D camera ». In : *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*. T. 180, p. 1–15.

EUDES, Alexandre (2011). « Localisation et cartographie simultanées par ajustement de faisceaux local: propagation d'erreurs et réduction de la dérive à l'aide d'un odomètre ». Thèse de doct. Université Blaise Pascal-Clermont-Ferrand II.

FARAGHER, Ramsey et Robert HARLE (2015). « Location fingerprinting with bluetooth low energy beacons ». In : *IEEE journal on Selected Areas in Communications* 33.11, p. 2418–2428.

FARKAS, Károly, Árpád HUSZÁK et Győző GÓDOR (2013). « Optimization of Wi-Fi access point placement for indoor localization ». In : *Journal IIT (Informatics & IT Today)* 1.1, p. 28–33.

FILLIAT, David (2007). « A visual bag of words method for interactive qualitative localization and mapping ». In : *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, p. 3921–3926.

FISCHLER, Martin A et Robert C BOLLES (1981). « Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography ». In : *Communications of the ACM* 24.6, p. 381–395.

FLEPS, Michael et al. (2011). « Optimization based IMU camera calibration ». In : *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, p. 3297–3304.

FORSTER, Christian, Matia PIZZOLI et Davide SCARAMUZZA (2014). « SVO: Fast semi-direct monocular visual odometry ». In : *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, p. 15–22.

GÁLVEZ-LÓPEZ, Dorian et Juan D TARDOS (2012). « Bags of binary words for fast place recognition in image sequences ». In : *IEEE Transactions on Robotics* 28.5, p. 1188–1197.

GEIGER, Andreas et al. (2013). « Vision meets robotics: The KITTI dataset ». In : *The International Journal of Robotics Research* 32.11, p. 1231–1237.

GEORGE, Laurent et Alexandre MAZEL (2013). « Humanoid robot indoor navigation based on 2D bar codes: Application to the NAO robot ». In : *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*. IEEE, p. 329–335.

GEYER, Christopher et Kostas DANIILIDIS (2000). « A unifying theory for central panoramic systems and practical implications ». In : *European conference on computer vision*. Springer, p. 445–461.

GLOVER, Arren et al. (2012). « OpenFABMAP: An open source toolbox for appearance-based loop closure detection ». In : *Robotics and automation (ICRA), 2012 IEEE international conference on*. IEEE, p. 4730–4735.

GLOVER, Arren J et al. (2010). « FAB-MAP+ RatSLAM: Appearance-based SLAM for multiple times of day ». In : *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, p. 3507–3512.

GOMEZ-OJEDA, Ruben et Javier GONZALEZ-JIMENEZ (2016). « Robust stereo visual odometry through a probabilistic combination of points and line segments ». In : *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, p. 2521–2526.

GRANA, Costantino et al. (2013). « A fast approach for integrating ORB descriptors in the bag of words model ». In : *Multimedia Content and Mobile Devices*. T. 8667. International Society for Optics et Photonics, p. 866709.

GRISETTI, Giorgio, Cyrill STACHNISS et Wolfram BURGARD (2005). « Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling ». In : *ICRA*, p. 2432–2437.

GRISETTI, Giorgio et al. (2010). « A tutorial on graph-based SLAM ». In : *IEEE Intelligent Transportation Systems Magazine* 2.4, p. 31–43.

GUIVANT, Jose E et Eduardo Mario NEBOT (2001). « Optimization of the simultaneous localization and map-building algorithm for real-time implementation ». In : *IEEE transactions on robotics and automation* 17.3, p. 242–257.

HAHNEL, Dirk et al. (2003). « An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements ». In : *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*. T. 1. IEEE, p. 206–211.

HARRIS, Chris et Mike STEPHENS (1988). « A combined corner and edge detector. » In : *Alvey vision conference*. T. 15. 50. Citeseer, p. 10–5244.

HARTLEY, Richard et Andrew ZISSERMAN (2003). *Multiple view geometry in computer vision*. Cambridge university press.

HARTMANN, Jan et al. (2012). « Real-time visual slam using fastslam and the microsoft kinect camera ». In : *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*. VDE, p. 1–6.

HE, Suining et S-H Gary CHAN (2016). « Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons ». In : *IEEE Communications Surveys & Tutorials* 18.1, p. 466–490.

HENRY, Peter et al. (2010). « RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments ». In : *In the 12th International Symposium on Experimental Robotics (ISER*. Citeseer.

HESCH, Joel A et Stergios I ROUMELIOTIS (2011). « A direct least-squares (DLS) method for PnP ». In : *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, p. 383–390.

JIANG, Wenchao et Zhaozheng YIN (2015). « Indoor localization by signal fusion ». In : *18th International Conference on Information Fusion*.

JIRKU, Michal, Vladimír KUBELKA et Michal REINSTEIN (2016). « WiFi localization in 3D ». In : *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, p. 4551–4557.

JOHANNSSON, Hordur et al. (2013). « Temporally scalable visual SLAM using a reduced pose graph ». In : *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, p. 54–61.

JOLY, Cyril, François PEYRET et al. (2008). « Étude comparative des techniques de filtrage non-linéaire appliquées à la localisation 2D d'un véhicule en temps réel ». In :

JULIER, Simon J et Jeffrey K UHLMANN (2001). « Simultaneous localisation and map building using split covariance intersection ». In : *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*. T. 3. IEEE, p. 1257–1262.

KAESS, Michael et Frank DELLAERT (2006). *Visual slam with a multi-camera rig*. Rapp. tech. Georgia Institute of Technology.

KENDALL, Alex, Matthew GRIMES et Roberto CIPOLLA (2015). « Posenet: A convolutional network for real-time 6-dof camera relocalization ». In : *Proceedings of the IEEE international conference on computer vision*, p. 2938–2946.

KLEIN, Georg et David MURRAY (2007). « Parallel tracking and mapping for small AR workspaces ». In : *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, p. 225–234.

KNEIP, Laurent, Margarita CHLI et Roland Y SIEGWART (2011). « Robust real-time visual odometry with a single camera and an IMU ». In : *Proceedings of the British Machine Vision Conference 2011*. British Machine Vision Association.

KONDA, Kishore Reddy et Roland MEMISEVIC (2015). « Learning Visual Odometry with a Convolutional Network. » In : *VISAPP (1)*, p. 486–490.

KONOLIGE, Kurt et Motilal AGRAWAL (2008). « FrameSLAM: From bundle adjustment to real-time visual mapping ». In : *IEEE Transactions on Robotics* 24.5, p. 1066–1077.

KONOLIGE, Kurt et James BOWMAN (2009). « Towards lifelong visual maps ». In : *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, p. 1156–1163.

KONOLIGE, Kurt et al. (2010). « View-based maps ». In : *The International Journal of Robotics Research* 29.8, p. 941–957.

KOSTAVELIS, Ioannis et Antonios GASTERATOS (2015). « Semantic mapping for mobile robotics tasks: A survey ». In : *Robotics and Autonomous Systems* 66, p. 86–103.

KRAJNÍK, Tomáš et al. (2014). « Long-term topological localisation for service robots in dynamic environments using spectral maps ». In : *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, p. 4537–4542.

KÜMMERLE, Rainer et al. (2011). « g 2 o: A general framework for graph optimization ». In : *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, p. 3607–3613.

LAFAYE, Jory, David GOUAILLIER et Pierre-Brice WIEBER (2014). « Linear model predictive control of the locomotion of Pepper, a humanoid robot with omnidirectional wheels ». In : *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE, p. 336–341.

LARNAOUT, Dorra et al. (2013). « Vehicle 6-dof localization based on slam constrained by gps and digital elevation model information ». In : *Image Processing (ICIP), 2013 20th IEEE International Conference on*. IEEE, p. 2504–2508.

LEMAIRE, Thomas et al. (2007). « Vision-based slam: Stereo and monocular approaches ». In : *International Journal of Computer Vision* 74.3, p. 343–364.

LEONARD, John J et Hugh F DURRANT-WHYTE (1991). « Mobile robot localization by tracking geometric beacons ». In : *IEEE Transactions on robotics and Automation* 7.3, p. 376–382.

LEPETIT, Vincent, Francesc MORENO-NOGUER et Pascal FUA (2009). « Epnp: An accurate o (n) solution to the pnp problem ». In : *International journal of computer vision* 81.2, p. 155.

LEUTENEGGER, Stefan, Margarita CHLI et Roland Y SIEGWART (2011). « BRISK: Binary robust invariant scalable keypoints ». In : *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, p. 2548–2555.

LEUTENEGGER, Stefan et al. (2015). « Keyframe-based visual–inertial odometry using nonlinear optimization ». In : *The International Journal of Robotics Research* 34.3, p. 314–334.

LEVENBERG, Kenneth (1944). « A method for the solution of certain non-linear problems in least squares ». In : *Quarterly of applied mathematics* 2.2, p. 164–168.

LI, Mingyang et Anastasios I MOURIKIS (2013). « High-precision, consistent EKF-based visual-inertial odometry ». In : *The International Journal of Robotics Research* 32.6, p. 690–711.

LIU, Hongbo et al. (2012). « Push the limit of WiFi based localization for smartphones ». In : *Proceedings of the 18th annual international conference on Mobile computing and networking*. ACM, p. 305–316.

LIU, Mengyun et al. (2017). « Scene recognition for indoor localization using a multi-sensor fusion approach ». In : *Sensors* 17.12, p. 2847.

LIU, Yang et Hong ZHANG (2012). « Visual loop closure detection with a compact image descriptor ». In : *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, p. 1051–1056.

LOTHE, Pierre et al. (2010). « Real-time vehicle global localisation with a single camera in dense urban areas: Exploitation of coarse 3d city models ». In :

LOVEGROVE, Steven, Alonso PATRON-PEREZ et Gabe SIBLEY (2013). « Spline Fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. » In : *BMVC*.

LOWE, David G (1999). « Object recognition from local scale-invariant features ». In : *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. T. 2. Ieee, p. 1150–1157.

LOWRY, Stephanie et al. (2016). « Visual place recognition: A survey ». In : *IEEE Transactions on Robotics* 32.1, p. 1–19.

LU, Feng et Evangelos MILIOS (1997). « Globally consistent range scan alignment for environment mapping ». In : *Autonomous robots* 4.4, p. 333–349.

LYNEN, Simon et al. (2014). « Placeless place-recognition ». In : *3D Vision (3DV), 2014 2nd International Conference on*. T. 1. IEEE, p. 303–310.

MARQUARDT, Donald W (1963). « An algorithm for least-squares estimation of nonlinear parameters ». In : *Journal of the society for Industrial and Applied Mathematics* 11.2, p. 431–441.

MATAS, Jiri et al. (2004). « Robust wide-baseline stereo from maximally stable extremal regions ». In : *Image and vision computing* 22.10, p. 761–767.

MCCORMAC, John et al. (2017). « Semanticfusion: Dense 3d semantic mapping with convolutional neural networks ». In : *2017 IEEE International Conference on Robotics and automation (ICRA)*. IEEE, p. 4628–4635.

MEI, Christopher (2007). « Laser-augmented omnidirectional vision for 3D localisation and mapping ». Thèse de doct. École Nationale Supérieure des Mines de Paris.

MEI, Christopher, Gabe SIBLEY et Paul NEWMAN (2010). « Closing loops without places ». In : *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, p. 3738–3744.

MEI, Christopher et al. (2009). « A Constant-Time Efficient Stereo SLAM System. » In : *BMVC*, p. 1–11.

MELEKHOV, Iaroslav et al. (2017). « Relative camera pose estimation using convolutional neural networks ». In : *International Conference on Advanced Concepts for Intelligent Vision Systems*. Springer, p. 675–687.

MIKSIK, Ondrej et Krystian MIKOLAJCZYK (2012). « Evaluation of local detectors and descriptors for fast feature matching ». In : *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, p. 2681–2684.

MILFORD, Michael J et Gordon F WYETH (2008). « Mapping a suburb with a single camera using a biologically inspired SLAM system ». In : *IEEE Transactions on Robotics* 24.5, p. 1038–1053.

— (2012). « SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights ». In : *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, p. 1643–1649.

MILFORD, Michael J, Gordon F WYETH et David PRASSER (2004). « RatSLAM: a hippocampal model for simultaneous localization and mapping ». In : *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. T. 1. IEEE, p. 403–408.

MIROWSKI, Piotr et al. (2013). « SignalSLAM: Simultaneous localization and mapping with mixed WiFi, Bluetooth, LTE and magnetic signals ». In : *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*. IEEE, p. 1–10.

MONTEMERLO, Michael et Sebastian THRUN (2007). « FastSLAM 2.0 ». In : *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*, p. 63–90.

MONTEMERLO, Michael et al. (2002). « FastSLAM: A factored solution to the simultaneous localization and mapping problem ». In : *Aaai/iaai* 593598.

MORAVEC, Hans P (1988). « Sensor fusion in certainty grids for mobile robots ». In : *AI magazine* 9.2, p. 61.

MORENO, Francisco-Angel, Jose-Luis BLANCO et Javier GONZALEZ-JIMENEZ (2016). « A constant-time SLAM back-end in the continuum between global mapping and submapping: application to visual stereo SLAM ». In : *The International Journal of Robotics Research* 35.9, p. 1036–1056.

MOURAGNON, Etienne et al. (2006). « Real time localization and 3d reconstruction ». In : *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. T. 1. IEEE, p. 363–370.

MOURIKIS, Anastasios I et Stergios I ROUMELIOTIS (2007). « A multi-state constraint Kalman filter for vision-aided inertial navigation ». In : *Robotics and automation, 2007 IEEE international conference on*. IEEE, p. 3565–3572.

MUR-ARTAL, Raul, Jose Maria Martinez MONTIEL et Juan D TARDOS (2015). « ORB-SLAM: a versatile and accurate monocular SLAM system ». In : *IEEE Transactions on Robotics* 31.5, p. 1147–1163.

MUR-ARTAL, Raúl et Juan D TARDÓS (2014). « Fast relocalisation and loop closing in keyframe-based SLAM ». In : *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, p. 846–853.

MUR-ARTAL, Raul et Juan D TARDÓS (2017a). « Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras ». In : *IEEE Transactions on Robotics* 33.5, p. 1255–1262.

MUR-ARTAL, Raúl et Juan D TARDÓS (2017b). « Visual-inertial monocular SLAM with map reuse ». In : *IEEE Robotics and Automation Letters* 2.2, p. 796–803.

MURILLO, Ana Cris et Jana KOSECKA (2009). « Experiments in place recognition using gist panoramas ». In : *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, p. 2196–2203.

MURPHY, Kevin P (2000). « Bayesian map learning in dynamic environments ». In : *Advances in Neural Information Processing Systems*, p. 1015–1021.

NEUBERT, Peer, Niko SÜNDERHAUF et Peter PROTZEL (2015). « Superpixel-based appearance change prediction for long-term navigation across seasons ». In : *Robotics and Autonomous Systems* 69, p. 15–27.

NEWMAN, Paul (1999). « On the structure and solution of the simultaneous localisation and map building problem ». In : *Doctoral diss., University of Sydney* 41.

NICOSEVICI, Tudor et Rafael GARCIA (2009). « On-line visual vocabularies for robot navigation and mapping ». In : *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, p. 205–212.

NIKOLIC, Janosch et al. (2014). « A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM ». In : *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, p. 431–437.

NISTER, David (2003). « An efficient solution to the five-point relative pose problem ». In : *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. T. 2. IEEE, p. II–195.

NISTÉR, David, Oleg NARODITSKY et James BERGEN (2004). « Visual odometry ». In : *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. T. 1. Ieee, p. I–I.

NISTER, David et Henrik STEWENIUS (2006). « Scalable recognition with a vocabulary tree ». In : *Computer vision and pattern recognition, 2006 IEEE computer society conference on*. T. 2. Ieee, p. 2161–2168.

NOWAKOWSKI, M. et al. (2017). « Topological localization using Wi-Fi and vision merged into FABMAP framework ». In : *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 3339–3344.

NOWICKI, Michał (2014). « WiFi-guided visual loop closure for indoor navigation using mobile devices ». In : *Journal of Automation Mobile Robotics and Intelligent Systems* 8.3, p. 10–18.

OCAÑA, Manuel et al. (2005). « Indoor robot navigation using a POMDP based on WiFi and ultrasound observations ». In : *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, p. 2592–2597.

OLIVA, Aude (2005). « Gist of the scene ». In : *Neurobiology of attention*. Elsevier, p. 251–256.

OLIVA, Aude et Antonio TORRALBA (2006). « Building the gist of a scene: The role of global image features in recognition ». In : *Progress in brain research* 155, p. 23–36.

OLIVERA, Vicente Matellán, José María Cañas PLAZA et Oscar Serrano SERRANO (2006). « WiFi localization methods for autonomous robots ». In : *Robotica* 24.4, p. 455–461.

PENATE-SANCHEZ, Adrian, Juan ANDRADE-CETTO et Francesc MORENO-NOGUER (2013). « Exhaustive linearization for robust camera pose and focal length estimation ». In : *IEEE transactions on pattern analysis and machine intelligence* 35.10, p. 2387–2400.

PERROLLAZ, Mathias et al. (2010). « Using the disparity space to compute occupancy grids from stereo-vision ». In : *International Conference on Intelligent Robots and Systems (IROS)*.

PHILBIN, James et al. (2007). « Object retrieval with large vocabularies and fast spatial matching ». In : *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, p. 1–8.

PIRE, Taihú et al. (2015). « Stereo parallel tracking and mapping for robot localization ». In : *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, p. 1373–1378.

POTTER, Mary C (1975). « Meaning in visual search ». In : *Science* 187.4180, p. 965–966.

QIN, Tong, Peiliang LI et Shaojie SHEN (2018). « Vins-mono: A robust and versatile monocular visual-inertial state estimator ». In : *IEEE Transactions on Robotics* 34.4, p. 1004–1020.

QUATTONI, Ariadna et Antonio TORRALBA (2009). « Recognizing indoor scenes ». In : *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, p. 413–420.

QUIGLEY, Morgan et al. (2010). « Sub-meter indoor localization in unmodified environments with inexpensive sensors ». In : *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, p. 2039–2046.

RAMADASAN, Datta, Marc CHEVALDONNÉ et Thierry CHATEAU (2014). « SLAM contraint en environnement de grande taille ». In : *Reconnaissance de Formes et Intelligence Artificielle (RFIA) 2014*.

ROBERTS, Brian et Kaveh PAHLAVAN (2009). « Site-specific RSS signature modeling for WiFi localization ». In : *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. IEEE, p. 1–6.

ROHRIG, Christof et Frank KIINEMUND (2007). « Mobile robot localization using WLAN signal strengths ». In : *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2007. IDAACS 2007. 4th IEEE Workshop on*. IEEE, p. 704–709.

ROSTEN, Edward et Tom DRUMMOND (2006). « Machine learning for high-speed corner detection ». In : *European conference on computer vision*. Springer, p. 430–443.

ROYER, Eric (2006). « Cartographie 3D et localisation par vision monoculaire pour la navignation autonome d'un robot mobile ». Thèse de doct. Université Blaise Pascal-Clermont-Ferrand II.

RUBLEE, Ethan et al. (2011). « ORB: An efficient alternative to SIFT or SURF ». In : *Computer Vision (ICCV), 2011 IEEE international conference on*. IEEE, p. 2564–2571.

RUIZ-RUIZ, Antonio J et al. (2011). « Using SIFT and WiFi signals to provide location-based services for smartphones ». In : *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Springer, p. 37–48.

SALTON, Gerard et Christopher BUCKLEY (1988). « Term-weighting approaches in automatic text retrieval ». In : *Information processing & management* 24.5, p. 513–523.

SCARAMUZZA, Davide et al. (2009). « Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints ». In : *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, p. 1413–1419.

SCHÖNBERGER, Johannes L et al. (2018). « Semantic Visual Localization ». In : *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*.

SCHWIEGELSHOHN, Fynn, Theresa NICK et Jürgen GÖTZE (2013). « Localization based on fusion of rfid and stereo image data ». In : *Positioning Navigation and Communication (WPNC), 2013 10th Workshop on*. IEEE, p. 1–6.

SHI, Jianbo et Carlo TOMASI (1993). *Good features to track*. Rapp. tech. Cornell University.

SIAGIAN, Christian et Laurent ITTI (2009). « Biologically inspired mobile robot vision localization ». In : *IEEE Transactions on Robotics* 25.4, p. 861–873.

SILVEIRA, Geraldo, Ezio MALIS et Patrick RIVES (2008). « An efficient direct approach to visual SLAM ». In : *IEEE transactions on robotics* 24.5, p. 969–979.

SIMONITE, Tom (2017). *Self-Driving Cars' Spinning-Laser Problem*.

SIVIC, Josef et Andrew ZISSERMAN (2003). « Video Google: A text retrieval approach to object matching in videos ». In : *null*. IEEE, p. 1470.

SMITH, Randall, Matthew SELF et Peter CHEESEMAN (1990). « Estimating uncertain spatial relationships in robotics ». In : *Autonomous robot vehicles*. Springer, p. 167–193.

SMITH, Randall C et Peter CHEESEMAN (1986). « On the representation and estimation of spatial uncertainty ». In : *The international journal of Robotics Research* 5.4, p. 56–68.

SNAVELY, Noah, Steven M SEITZ et Richard SZELISKI (2006). « Photo tourism: exploring photo collections in 3D ». In : *ACM transactions on graphics (TOG)*. T. 25. 3. ACM, p. 835–846.

SOO PARK, Hyun et al. (2013). « 3d point cloud reduction using mixed-integer quadratic programming ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, p. 229–236.

SOUMAN, Jan L et al. (2009). « Walking straight into circles ». In : *Current biology* 19.18, p. 1538–1542.

STACHNISS, Cyrill, Dirk HAHNEL et Wolfram BURGARD (2004). « Exploration with active loop-closing for FastSLAM ». In : *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. T. 2. IEEE, p. 1505–1510.

STASSE, Olivier et al. (2006). « Real-time 3d slam for humanoid robot considering pattern generator information ». In : *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, p. 348–355.

STEINER, Ted J, Guoquan HUANG et John J LEONARD (2015). « Location utility-based map reduction ». In : *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, p. 479–486.

STRASDAT, Hauke, J MONTIEL et Andrew J DAVISON (2010). « Scale drift-aware large scale monocular SLAM ». In : *Robotics: Science and Systems VI* 2.

STRASDAT, Hauke, José MM MONTIEL et Andrew J DAVISON (2012). « Visual SLAM: why filter? » In : *Image and Vision Computing* 30.2, p. 65–77.

STUMM, Elena, Christopher MEI et Simon LACROIX (2013). « Probabilistic place recognition with covisibility maps ». In : *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, p. 4158–4163.

STURM, Peter et al. (2011). « Camera models and fundamental concepts used in geometric computer vision ». In : *Foundations and Trends® in Computer Graphics and Vision* 6.1–2, p. 1–183.

SÜNDERHAUF, Niko et Peter PROTZEL (2011). « Brief-gist-closing the loop by simple means ». In : *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, p. 1234–1241.

— (2012). « Switchable constraints for robust pose graph SLAM ». In : *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, p. 1879–1884.

SÜNDERHAUF, Niko et al. (2015). « Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free ». In : *Proceedings of Robotics: Science and Systems XII*.

TAMAAZOUSTI, Mohamed et al. (2011). « Nonlinear refinement of structure from motion reconstruction by taking advantage of a partial knowledge of the environment ». In :

THRUN, Sebastian et Michael MONTEMERLO (2006). « The graph SLAM algorithm with applications to large-scale mapping of urban structures ». In : *The International Journal of Robotics Research* 25.5-6, p. 403–429.

THRUN, Sebastian et al. (2002). « Robotic mapping: A survey ». In : *Exploring artificial intelligence in the new millennium* 1.1-35, p. 1.

TRIGGS, Bill et al. (1999). « Bundle adjustment—a modern synthesis ». In : *International workshop on vision algorithms*. Springer, p. 298–372.

ULRICH, Iwan et Illah NOURBAKHSH (2000). « Appearance-based place recognition for topological localization ». In : *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. T. 2. Ieee, p. 1023–1029.

VENTURA, Jonathan et al. (2014). « Global localization from monocular slam on a mobile phone ». In : *IEEE transactions on visualization and computer graphics* 20.4, p. 531–539.

VERMEIREN, Vadim et al. (2018). « How a Cutting-Edge Technology Can Benefit the Creative Industries: The Positioning System at Work ». In : *Drones and the Creative Industry*. Springer, p. 115–127.

WANG, Sen et al. (2017). « Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks ». In : *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, p. 2043–2050.

WEI, Ting et Scott BELL (2011). « Indoor localization method comparison: Fingerprinting and Trilateration algorithm ». In : *University of Saskatchewan. Accessed March* 24, p. 2015.

WERNER, Martin, Moritz KESSEL et Chadly MAROUANE (2011). « Indoor positioning using smartphone camera ». In : *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*. IEEE, p. 1–6.

WIETRZYKOWSKI, Jan, Michał NOWICKI et Piotr SKRZYPCZYŃSKI (2016). « Adopting the FAB-MAP algorithm for indoor localization with WiFi fingerprints ». In : *arXiv preprint arXiv:1611.02054*.

WIRBEL, Émilie (2014). « Localisation et navigation d'un robot humanoïde en environnement domestique ». Thèse de doct. Ecole Nationale Supérieure des Mines de Paris.

WIRBEL, Emilie et al. (2013). « Humanoid robot navigation: From a visual SLAM to a visual compass ». In : *Networking, Sensing and Control (ICNSC), 2013 10th IEEE International Conference on*. IEEE, p. 678–683.

YU, Chao et al. (2018). « DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments ». In : *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, p. 1168–1174.

ZENDER, Hendrik et al. (2008). « Conceptual spatial representations for indoor mobile robots ». In : *Robotics and Autonomous Systems* 56.6, p. 493–502.

RÉSUMÉ

Après la démocratisation des robots industriels, la tendance actuelle est au développement de robots sociaux dont la fonction principale est l'interaction avec ses utilisateurs. Le déploiement de telles plate-formes dans des boutiques, des musées ou des gares relance différentes problématiques dont celle de la localisation pour les robots mobiles. Cette thèse traite ainsi de la localisation du robot Pepper en milieu intérieur non-contraint. Présent dans de nombreuses boutiques au Japon, Pepper est utilisé par des personnes non-expertes et doit donc être le plus autonome possible. Cependant, les solutions de localisation autonome de la littérature souffrent des limitations de la plate-forme.

Les travaux de cette thèse s'articulent autour de deux grands axes. D'abord, le problème de la relocalisation dans un environnement visuellement redondant est étudié. La solution proposée consiste à combiner la vision et le Wi-Fi dans une approche probabiliste basée sur l'apparence. Ensuite, la question de la création d'une carte métrique cohérente est approfondie. Pour compenser les nombreuses pertes de suivi d'amers visuels causées par une fréquence d'acquisition basse, des contraintes odométriques sont ajoutées à une optimisation par ajustement de faisceaux. Ces solutions ont été testées et validées sur plusieurs robots Pepper à partir de données collectées dans différents environnements intérieurs sur plus de 7 km.

MOTS CLÉS

SLAM basse fréquence, fusion de données multi-capteurs, robot humanoïde.

ABSTRACT

After the democratization of industrial robots, the current trend is the development of social robots that create strong interactions with their users. The deployment of such platforms in shops, museums or train stations raises various issues including the autonomous localization of mobile robots. This thesis focuses on the localization of Pepper robots in a non-constrained indoor environment. Pepper robots are daily used in many shops in Japan and must be as autonomous as possible. However, localization solutions in the literature suffer from the limitations of the platform.

This thesis is split into two main themes. First, the problem of relocalization in a visually redundant environment is studied. The proposed solution combines vision and Wi-Fi in a probabilistic approach based on the appearance. Then, the question of a consistent metrical mapping is examined. In order to compensate the numerous losses of tracking caused by the low acquisition frequency, odometric constraints are added to a bundle adjustment optimization. These solutions have been tested and validated on several Pepper robots, from data collected in different indoor environments over more than 7 km.

KEYWORDS

Low frequency SLAM, Multisensor data fusion, humanoid robot.