Cartographie 3D et localisation par vision monoculaire pour la navignation autonome d'un robot mobile

Eric Royer

N° d'ordre : D.U : 1676
EDSPIC : 354

UNIVERSITÉ BLAISE PASCAL - CLERMONT II

École Doctorale
Sciences Pour l'Ingénieur de Clermont-Ferrand

Thèse présentée par :
Eric ROYER

Formation Doctorale CSTI :
Composants et Systèmes pour le Traitement de l'Information

en vue de l'obtention du grade de

DOCTEUR D'UNIVERSITÉ

spécialité : Vision pour la robotique

Cartographie 3D et localisation par vision monoculaire pour la navigation autonome d'un robot mobile

Soutenue publiquement le 26 Septembre 2006 devant le jury :

M. François CHAUMETTE	Rapporteur et examinateur
M. Michel DHOME	Directeur de thèse
M. Seth HUTCHINSON	Rapporteur et examinateur
M. Jean-Marc LAVEST	Président du jury
M. Maxime LHUILLIER	Examinateur
M. Peter STURM	Rapporteur et examinateur

Remerciements

Je remercie tout d'abord François Chaumette, Seth Hutchinson et Peter Sturm qui ont accepté de faire partie de mon jury de thèse.

Je remercie également mes encadrants en commençant par Michel Dhome, mon directeur de thèse. En me proposant cette thèse, il m'a offert la possibilité de travailler sur un sujet plein de potentiel. Je suis également très reconnaissant à Maxime Lhuillier qui m'a initié à la vision au cours de mon stage de DEA et qui m'a permis de continuer à apprendre sur le sujet tout au long de ma thèse. Je remercie également Jean-Marc Lavest qui a toujours su me faire profiter de son expérience et me donner le bon conseil au bon moment. Enfin Thierry Chateau, Jean-Thierry Lapresté et les autres membres de l'équipe ComSee du LASMEA ont toujours été présents lorsque j'avais besoin d'aide.

J'ai eu la chance d'avoir un sujet qui mette en œuvre le savoir faire de plusieurs équipes du laboratoire. Ainsi, j'ai pu travailler avec l'équipe de commande des systèmes robotiques, principalement Philippe Martinet et Benoit Thuilot. Cela m'a permis de m'ouvrir à un domaine que je connaissais très peu au début de ma thèse.

Il ne faudrait surtout pas que j'oublie l'équipe technique du LASMEA. Réparer tous les problèmes électriques, mécaniques ou logiciels demande du temps et de la compétence. Sans leur savoir-faire, les expérimentations sur cycab auraient été bien moins nombreuses. J'ai une pensée particulière pour François Marmoiton qui n'a jamais hésité à m'accompagner pour collecter des données malgré des températures parfois bien en dessous de zéro.

Et pout finir, je voudrais remercier aussi les doctorants du laboratoire qui m'ont apporté leur aide sur un problème technique, leurs idées, leurs encouragements, bref toutes ces choses qui rendent le travail plus facile ou plus agréable. Alors merci à Lucie, Etienne, Joël, Jonathan, Guillaume, Jean, Hicham, Noël, Steve, Matthieu, Vincent, Julie, François, Fabio, et les autres ...

Résumé

Ce mémoire de thèse présente la réalisation d'un système de localisation pour un robot mobile fondé sur la vision monoculaire. L'objectif de ces travaux est de pouvoir faire naviguer un véhicule robotique sur un parcours donné en milieu urbain.

Le robot est d'abord conduit manuellement. Pendant cette phase d'apprentissage, la caméra embarquée enregistre une séquence vidéo. Après un traitement approprié hors ligne, une image prise avec le même matériel permet de localiser le robot en temps réel. Cette localisation peut être utilisée pour commander le robot et faire en sorte qu'il suive de façon autonome le même parcours que durant la phase d'apprentissage. Le principe retenu consiste à construire une carte tridimensionnelle de la zone parcourue pendant la phase d'apprentissage, puis à se servir de la carte pour se localiser. L'étape de "cartographie" est un algorithme de reconstruction 3D à partir d'une caméra en mouvement. La carte est un modèle 3D de la scène observée constitué d'un nuage de points. La localisation du robot à partir d'une image consiste à recaler l'image courante sur le modèle 3D de la scène, ce qui permet d'en déduire la pose de la caméra et donc du robot.

Même si le but est de pouvoir localiser un véhicule en milieu urbain, nous avons choisi de ne pas faire d'hypothèses restrictives sur la géométrie des scènes rencontrées. De plus le calcul de la pose du robot est fait avec six degrés de liberté, ce qui ne restreint pas l'usage du système à un véhicule évoluant sur un sol plan.

Enfin une grande partie de ce mémoire est consacré à l'étude des performances du sytème. De nombreuses expérimentations en conditions réelles ont été réalisées afin d'évaluer la robustesse. La précision a pu être mesurée en comparant la localisation calculée par l'algorithme de vision avec la vérité terrain enregistrée par un récepteur GPS différentiel.

Mots-clef : localisation, vision monoculaire, temps-réel, reconstruction 3D, robot mobile.

Abstract

This thesis presents the realization of a localization system for a mobile robot relying on monocular vision. The aim of this project is to be able to make a robot follow a path in autonomous navigation in an urban environment.

First, the robot is driven manually. During this learning step, the on board camera records a video sequence. After an off-line processing step, an image taken with the same hardware allows to compute the pose of the robot in real-time. This localization can be used to control the robot and make it follow the same trajectory as in the learning step. To do this, we build a three dimensional model of the environment (a map) and use this map to localize the robot. The map building step is a structure from motion algorithm. The map is a three dimensional model of the environment made of points. In order to localize the robot, we establish correspondences between the curent image and the 3D model. This allows to compute the pose of the camera and the robot.

Even if the goal is to be able to localize a vehicle in an urban environment, we have chosen not to make restrictive assumptions on the structure of the scenes. Additionally, the pose computation is done with six degrees of freedom which doesn't limit the usage of the system to vehicles moving on a ground plane.

Finally, a large part of this report is dedicated to the performance evaluation of the system. Many experiments in real situations were conducted to evaluate the robustness. The accuracy was measured by comparing the localization computed with the vision algorithm to the ground truth recorded with a differential GPS sensor.

Key-words : localization, monocular vision, real-time, structure from motion, mobile robot.

Table des matières

Table des figures

Liste des tableaux

Introduction

Localisation par vision

Donner la vue à une machine est une tâche difficile, alors que voir est une activité tellement naturelle pour les êtres humains et les animaux que l'ampleur de la difficulté est souvent minorée. L'invention de la caméra et de l'ordinateur ont permis de franchir un premier pas vers la vision artificielle, mais le plus dur reste à faire. La caméra est capable d'acquérir une image, un peu à la manière de ce que fait l'œil. La caméra peut même dépasser les performances de l'œil dans certaines situations. Mais pour interpréter l'image, l'ordinateur et les algorithmes mis au point par l'homme sont loin de pouvoir égaler ce dont est capable le cerveau. Voir est une tâche très simple pour nous, mais c'est aussi un processus inconscient que nous ne sommes pas capables d'expliquer et encore moins de reproduire artificiellement. Pourtant, la recherche en vision par ordinateur est active depuis une trentaine d'années et elle s'appuie sur des connaissances plus anciennes. Les ambitions premières ont dû être revues à la baisse, et plutôt que de vouloir reproduire la totalité du système visuel humain, les recherches se concentrent sur des tâches plus particulières et plus simples comme reconnaître un objet particulier, ou retrouver la structure tridimensionnelle d'une scène. Ces travaux ont d'ores et déjà donné lieu à un certain nombre d'applications dans divers domaines : surveillance, métrologie, modélisation pour la synthèse d'images, robotique, ...

La localisation d'un robot est une des tâches auxquelles la vision par ordinateur peut apporter une réponse. Se localiser est une de ces choses que nous faisons naturellement en utilisant notre vue. Cela ne nous aide pas beaucoup pour construire un algorithme utilisable par un ordinateur, mais cela nous montre tout de même qu'il existe une solution performante au problème de la localisation par vision. Se localiser est une tâche importante qu'un robot doit être capable de réaliser pour accomplir un certain nombres de missions qu'il devra réaliser de manière autonome. C'est même incontournable pour un robot mobile qui doit atteindre son but et revenir à son point de départ une fois le travail effectué. Le premier champ

d'application des robots est la réalisation de tâches dans des lieux où l'homme ne peut pas aller. Il peut s'agir de zones difficilement accessibles (exploration planétaire ou sous-marine), dangereuses (zones radioactives, robots militaires), ou à des échelles différentes (robot à l'intérieur d'une canalisation, voire robot chirurgical à l'intérieur du corps humain).

Mais au delà de ces champs d'applications très particuliers, l'automatisation de certaines tâches peut être utilisée par le plus grand nombre. Ainsi, on a vu apparaître les premiers systèmes de transports automatisés, d'abord de façon expérimentale, puis en application à grande échelle. Par exemple, la ligne 14 du métro Parisien transporte des milliers de personnes chaque jour de façon automatique. La possibilité d'automatiser les transports offre des perspectives pour réduire les nuisances causées par les transports urbains. Mais les transports en commun ne peuvent pas couvrir l'ensemble des besoins de déplacement des usagers. La création de transports individuels publics permettrait d'offrir un complément aux transports en commun pour atteindre des zones non couvertes, ou pour circuler à des horaires tardifs. Ces nouveaux modes de transport public pourraient permettre de réduire les nuisances par rapport à la voiture individuelle grâce à des véhicules spécifiquement adaptés à un contexte urbain : plus petits, plus légers, moins consommateurs d'énergie et de places de parking. Pour l'instant, le seul complément en matière de transports publics est le taxi, mais son coût est élevé. Une alternative pourrait être l'utilisation de véhicules en libre-service comme dans l'expérience Praxitèle [1]. Mais un tel système n'est accessible qu'aux personnes pouvant conduire et les personnes n'ayant pas le permis, les enfants, certaines personnes agées ou handicappées ne peuvent pas toujours en profiter. D'autre part, la mise à disposition de véhicules dans des stations impose à l'utilisateur de ramener le véhicule dans une des stations. Pour ces deux raisons, il est intéressant de pouvoir disposer de véhicules automatisés individuels. Des projets de recherche tels que Bodega [2] et MobiVip [3] ont été lancés pour étudier la faisabilité de transports individuels publics automatisés en milieu urbain non instrumenté. Dans ce cadre, le défi à relever pour l'automatisation des véhicules est de plus grande ampleur que pour le métro, car les véhicules devront être capables de circuler sur une

[1] http ://www-rocq.inria.fr/praxitele/

[2] http ://jazz.ensil.unilim.fr/bodega/ : projet s'inscrivant dans le cadre de ROBEA (ROBotique et Entités Artificielles), programme interdisciplinaire de recherche initié par le CNRS et soutenu par l'INRIA.

[3] http ://www-sop.inria.fr/mobivip/ : ce projet s'inscrit dans le cadre du programme PREDIT de recherche, d'expérimentation et d'innovation dans les transports terrestres, initié et conduit par les ministères chargés de la recherche, des transports, de l'environnement et de l'industrie, l'ADEME et l'ANVAR.

zone étendue et pas seulement sur une ligne sécurisée et équipée de capteurs. Une navette automatisée a été expérimentée en milieu ouvert à Antibes pendant deux semaines de Juin 2004 dans le cadre du projet Européen Cybermove [4]. Le système fonctionnait avec 200 aimants placés dans la chaussée pour le guidage du véhicule sur le parcours et des capteurs pour détecter les obstacles (laser et ultrasons). L'étape suivante en terme de difficulté consiste à réaliser le même type de système sans devoir équiper l'infrastructure. Les travaux présentés dans ce mémoire proposent une solution pour la localisation d'un véhicule dans un environnement non instrumenté. Cela constitue une des multiples briques nécessaires à la réalisation d'un système de transport automatisé sûr et fiable.

Objectif et méthode utilisée

L'objectif de ce travail est de réaliser un système de localisation permettant à un robot de naviguer de façon autonome en utilisant une seule caméra. La trajectoire à suivre est donnée au robot par un mécanisme d'apprentissage. Le robot est d'abord conduit manuellement. Pendant cette phase d'apprentissage, la caméra embarquée enregistre une vidéo. Après un traitement approprié, une image prise avec le même matériel permet de localiser le robot. Cette localisation peut être utilisée pour commander le robot et faire en sorte qu'il suive de façon autonome le même parcours que durant la phase d'apprentissage. L'ensemble du procédé est illustré sur la figure 1. Le principe retenu consiste à construire une carte tridimensionnelle de la zone parcourue pendant la phase d'apprentissage, puis à se servir de la carte pour se localiser. L'étape de "cartographie" est un algorithme de reconstruction 3D à partir d'une caméra en mouvement. La carte est un modèle 3D de la scène observée constitué d'un nuage de points. La localisation du robot à partir d'une image consiste à recaler l'image courante sur le modèle 3D de la scène, ce qui permet d'en déduire la pose de la caméra et donc du robot.

Ce travail s'incrit dans le cadre du projet Bodega qui a pour but le développement de solutions pour la navigation autonome et sûre en milieu urbain. La première application visée est la commande d'un véhicule électrique en milieu urbain. Même si au final les solutions développées pourront être utilisées ailleurs, la mesure des performances des algorithmes développés est faite en fonction de cette application. La mesure des erreurs de position par exemple est faite sur la position du robot donnée de façon métrique et non pas dans l'image (en pixels) comme on pourrait le faire pour un système de réalité augmentée.

[4] http ://www.cybermove.org/

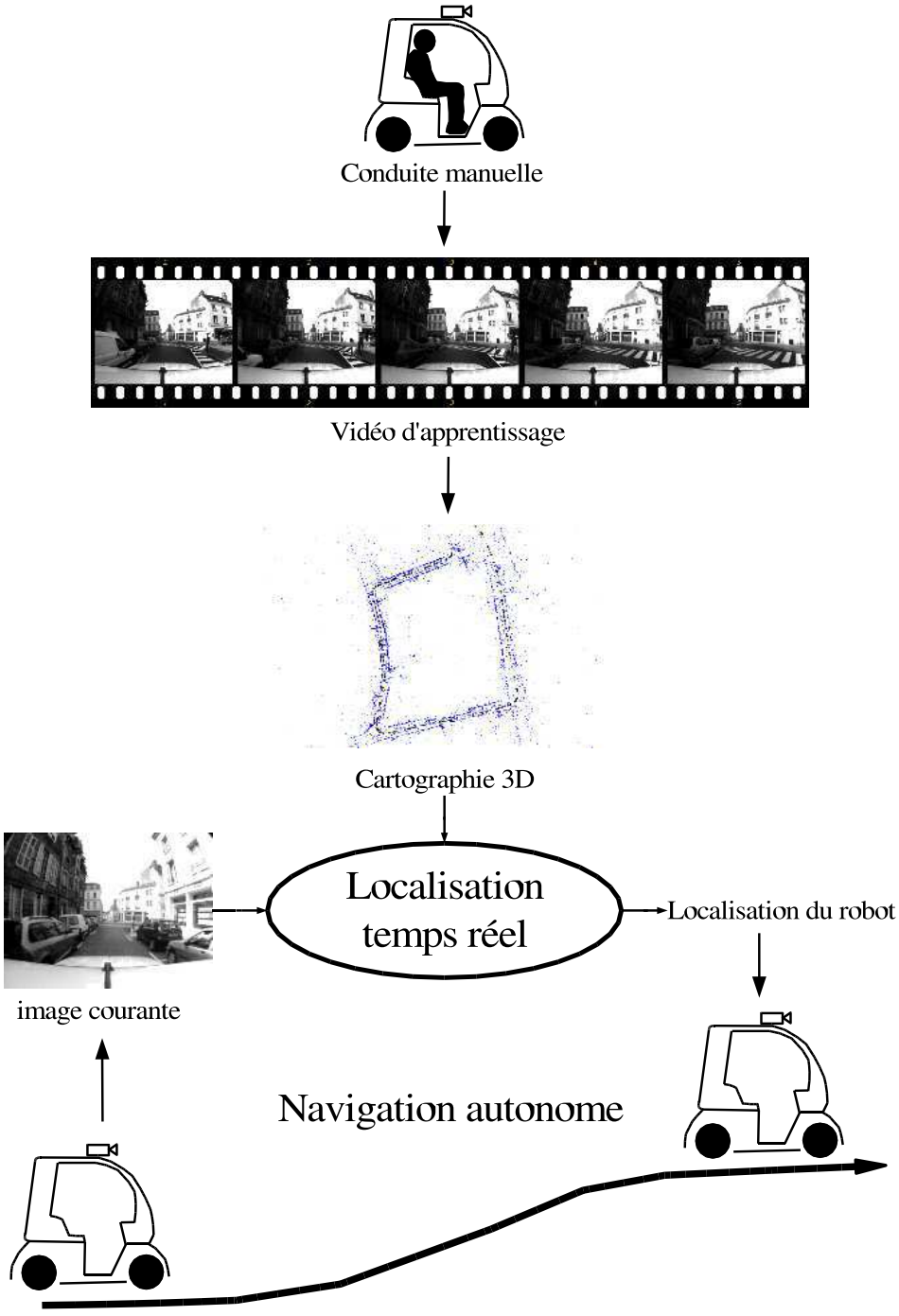

FIG. 1 – Fonctionnement global du système de cartographie et localisation

La principale motivation d'utiliser la vision au sein de cette application est de pouvoir pallier les insuffisances du GPS. En effet, on pourrait penser qu'un récepteur GPS est le capteur idéal pour connaître la position d'un véhicule en ville. Ce récepteur est le principal outil pour la navigation aérienne et maritime. Il est également souvent présent dans les voitures individuelles pour aider le conducteur à trouver sa route. Mais en réalité, les signaux en provenance des satellites peuvent être masqués par les immeubles et limiter la précision à plusieurs mètres, ce qui est insuffisant pour rester dans la voie de circulation. C'est pour cette raison, qu'il est utile de considérer l'usage d'autres capteurs. La vision est une bonne alternative au GPS en milieu urbain car les bâtiments et le mobilier urbain sont des éléments stables à long terme. Ils permettent d'obtenir des images dans lesquelles on peut trouver un grand nombre d'amers potentiels. Il est aussi possible, lorsqu'on a affaire à des images de bâtiments, d'exploiter leur structure assez spécifique : facades planes, présence de ligne verticales ou horizontales, ... Certains travaux exploitent cette particularité, mais nous avons choisi de ne pas le faire. Même si en ville on observe ce genre de structure très souvent, il existe aussi de nombreux points de vue depuis lesquels ce n'est pas le cas. La figure 2 montre quelques exemples de scènes urbaines où des hypothèses préalables sur la structure de la scène observée sont difficilement utilisables : architecture particulière, absence de bâtiments à proximité de la caméra, arbres, ... L'hypothèse d'un sol plan est souvent faite en robotique mobile en milieu intérieur et le sol est souvent plan en milieu urbain. Mais là encore, on trouve assez fréquemment des contre-exemples : ralentisseurs, passages pour piétons surélevés, trous dans la chaussée, caniveaux, ... C'est pour ces raisons que nous avons préféré ne pas faire d'hypothèses simplificatrices sur la nature des scènes rencontrées.

FIG. 2 – Exemples de scènes urbaines sans structure particulière

Contributions

Depuis les débuts de la vision par ordinateur, les problèmes fondamentaux ont été identifiés et étudiés. L'association de primitives entre images, le calcul robuste de géométrie épipolaire, le suivi et bien d'autres techniques ont fait l'objet de très nombreux travaux au cours des trente dernières années. Parallèlement à cela, la puissance de calcul des ordinateurs s'est considérablement accrue, si bien que ce qui demandait plusieurs minutes de calcul dans les années 1980, peut être réalisé aujourd'hui en une fraction de seconde. Finalement, on arrive à un point où la théorie et les algorithmes de base sont bien connus et où le matériel est devenu performant et facilement disponible. Le nouvel enjeu maintenant est de pouvoir assembler les briques de base développées ces dernières années, pour construire des systèmes de vision complets, performants et robustes. En robotique mobile, l'exemple du Grand Challenge DARPA (Defense Advanced Research Projects Agency) est intéressant. Les participants à ce concours devaient mettre au point un véhicule capable de parcourir de façon autonome environ 300 km en plein désert sur un parcours donné juste avant le départ. Lors de la première édition en 2004, la meilleure équipe a réussi à parcourir 11 km seulement, et la plupart des robots n'ont parcouru que quelques centaines de mètres. L'année suivante, cinq robots ont réussi à finir le parcours. Pourtant, en un an, la recherche sur les aspects fondamentaux de la robotique n'a pas fait des progrès extraordaires. La progression spectaculaire dans ce concours est due au choix judicieux des algorithmes parmi ceux existants dans la littérature et à une meilleure intégration de l'ensemble.

Le travail présenté dans cette thèse peut être vu sous le même aspect. Le système de localisation présenté ici résulte de l'association d'un algorithme de reconstruction 3D à partir d'une caméra en mouvement et d'un algorithme de recalage d'une image sur un modèle 3D. Les principales contributions de ce travail sont :

– La réalisation d'un système complet de localisation temps réel pour un robot mobile reposant sur l'utilisation d'une seule caméra. Le système est polyvalent car, contrairement à un certain nombre de travaux antérieurs, très peu d'hypothèses sont faites sur l'environnement du robot. Le sol n'est pas supposé plan, ce qui rend l'utilisation possible en extérieur aussi bien qu'en intérieur. La localisation est calculée avec six degrés de liberté. On ne fait pas l'hypothèse d'un environnement structuré avec des lignes ou des plans horizontaux ou verticaux ce qui limiterait l'usage à des environnements de type intérieur.

– L'évaluation des performances du système, grâce notamment à la comparaison des résultats fournis par l'algorithme de vision avec la vérité terrain enregistrée par un récepteur GPS différentiel. Un grand nombre d'expérimentations réalisées dans des lieux variés et par des conditions météo changeantes attestent de la robustesse et de la polyvalence de la méthode.

En plus de ces contributions, la réalisation de ce système a donné lieu au développement de sous-ensembles qui ont été ou qui peuvent être utilsés en dehors de l'application initialement visée. Ainsi, la première étape du travail présenté dans ce rapport a été le développement d'une bibliothèque pour la détection et l'appariement de points d'intérêt rapide. Le code a été particulièrement optimisé afin d'exploiter au mieux la parallélisation SIMD (Single Instruction Multiple Data) possible avec les processeurs actuels. D'autre part, la localisation repose sur une cartographie du voisinage de la trajectoire par vision monoculaire. Cette reconstruction 3D constitue une étape essentielle du travail présenté et pourrait éventuellement être utilisée pour d'autres applications comme la modélisation pour la synthèse d'images ou la réalité augmentée. Les travaux réalisés au cours de cette thèse ont donné lieu à plusieurs publications ([68], [70], [71], [69], [72], [73]).

Structure du document

Le travail présenté dans ce mémoire résulte de l'association de plusieurs algorithmes de vision ayant déjà fait l'objet de nombreux travaux. En effet, le problème de la reconstruction 3D tout comme celui du recalage d'une image par rapport à un modèle sont des sujets très étudiés en vision. Pour cette raison, le choix a été fait de présenter l'état de l'art associé à chaque partie en début de chapitre. Le chapitre 1, présente les travaux antérieurs concernant la localisation d'un robot mobile. Dans le chapitre 2, nous détaillons la méthode utilisée pour mettre en correspondance des primitives entre images. Ces techniques sont utilisées dans toute la suite de ce rapport. La construction de la carte par un algorithme de reconstruction 3D est présentée au chapitre 3. Le chapitre 4 est consacré à l'algorithme de localisation temps-réel. Enfin, l'utilisation de l'algorithme de localisation pour commander le cycab est présentée au chapitre 5. Dans ce chapitre, nous analysons également les performances du système en termes de précision et de robustesse.

Chapitre 1

État de l'art

1.1 Capteurs utilisés pour la localisation

Il existe deux grandes classes de capteurs qui peuvent être utilisés pour localiser un robot mobile. D'une part les capteurs extéroceptifs permettent de percevoir le monde qui entoure le robot (caméras, télémètres lasers, radars, GPS,...). Ils offrent la possibilité de déterminer la position du robot par rapport à son environnement. D'autre part, les capteurs proprioceptifs permettent de prendre des mesures sur les parties mobiles du robot et donnent des informations sur le mouvement du robot, mais pas sur sa localisation absolue. Dans cette classe de capteurs on trouve les odomètres ainsi que les capteurs inertiels.

Bien entendu, il est possible de combiner l'information de plusieurs capteurs pour améliorer la robustesse et la précision de la localisation [20]. Dans une véritable application de transport automatisé, c'est même quasiment indispensable pour assurer le bon fonctionnement et la sécurité du sytème, ne serait-ce que parce qu'un des capteurs peut tomber en panne et fournir des informations complètement erronées.

1.1.1 Le système GPS

Le capteur de position par excellence est le récepteur GPS (Global Positioning System). Ce système de positionnement est le résultat de lourds investissements réalisés par le département américain de la défense dans les années 1970. Le système repose sur 24 satellites placés sur des orbites leur permettant d'être toujours répartis tout autour de la Terre. Un récepteur GPS, placé sur un véhicule par exemple, peut calculer sa position en mesurant le temps que met un signal émis par les satellites pour arriver jusqu'à lui. A cause de la dérive possible de

l'horloge interne du récepteur, 4 satellites visibles sont nécessaires pour obtenir la position du récepteur.

Ce système est très utilisé dans les transports maritimes, aériens ou terrestres. L'énorme avantage de ce capteur est de fournir directement une position (il a été conçu pour cela). De plus on peut l'utiliser partout sur Terre et à tout moment. La précision avec un récepteur à bas coût est de l'ordre de quelques dizaines de mètres. La principale cause d'imprécision vient du calcul de la distance aux satellites à cause du trajet dans l'atmosphère. Mais il est possible de faire beaucoup mieux avec un récepteur différentiel. En utilisant une base terrestre fixe, on peut réduire le bruit de mesure et atteindre une précision de l'ordre d'un ou deux centimètres. Signalons enfin pour être complet qu'un nouveau système de positionnement par satellite, avec une précision annoncée de 1 à 10 mètres, verra le jour dans les années qui viennent. Il s'agit du système européen Galileo dont le premier satellite de test a été lancé le 28 décembre 2005.

Au vu de ces performances et de cette facilité d'utilisation, on pourrait même se demander s'il est utile de concevoir d'autres capteurs de positionnement destinés à des robots mobiles. Il faut tout de même garder à l'esprit que le GPS a certaines limites. La première est que les satellites doivent être visibles. Il est donc impossible de se localiser par GPS à l'intérieur d'un bâtiment, dans un tunnel, sous un pont,... En ville, même si les immeubles ne masquent pas totalement le ciel, la précision de la localisation n'est pas aussi bonne qu'en terrain dégagé. Même avec un récepteur GPS différentiel très coûteux, il n'est pas possible de se localiser partout et continuement au centimètre près à cause des occultations et des multitrajets (réflexion des signaux sur les façades). Des travaux visant à améliorer la précision de localisation en milieu urbain existent [12] mais on perd alors la simplicité d'un récepteur donnant directement une position précise. Pour finir, le système GPS permet de connaître la position, mais pas l'orientation du récepteur, ce qui est possible avec d'autres capteurs.

1.1.2 Perception de l'environnement local

L'utilisation du GPS est bien sûr très pratique pour localiser un robot en extérieur car le système a été spécialement développé pour cela. Mais les animaux et les êtres humains n'ont pas attendu d'avoir des satellites en orbite pour se localiser. Ils utilisent leurs capteurs naturels pour percevoir leur environnement et se localiser ainsi. Pour l'être humain, les principaux sens utilisés sont la vue et le toucher. Ces sens sont complétés par des informations proprioceptives qui nous donnent des informations sur nos accélérations (oreille interne) et sur nos déplacements. Certains animaux, comme les chauves-souris sont également capables

de se localiser selon le principe du sonar. Avec un capteur et un traitement informatique approprié, il est possible d'utiliser les mêmes principes, c'est-à-dire d'analyser l'environnement d'un robot pour reconnaitre des lieux connus et localiser le robot.

Parmi les capteurs utilisables, on peut distinguer deux grandes classes. Les capteurs actifs émettent un signal et analysent le signal renvoyé par l'environnement alors que les capteurs passifs n'émettent rien. En règle générale, les capteurs passifs sont préférables car ils ne risquent pas de perturber ce qui entoure le robot, et de plus ils consomment souvent moins d'énergie. Une caméra est un bon exemple de capteur passif, sauf si on utilise un éclairage artificiel. Parmi les capteurs actifs, on trouve les télémètres, les radars et les sonars. Tous sont utilisés dans des applications de robotique, mais pour la localisation du robot, le plus utilisé est certainement le télémètre laser. Une comparaison entre localisation par vision ou en utilisant un télémètre laser est effectuée dans [64].

1.1.3 Capteurs proprioceptifs

Les capteurs proprioceptifs sont utilisés en complément d'un ou plusieurs capteurs extéroceptifs. En effet, les capteurs proprioceptifs donnent simplement une information sur le mouvement du robot. Ils ne permettent pas de calculer la position initiale du robot lorsque celui-ci commence à fonctionner. D'autre part, les mesures fournies sont des mesures de vitesse (pour l'odomètre) ou d'accélération (capteurs inertiels) qui doivent être intégrées une ou deux fois pour fournir une position. En pratique, cela signifie que la position calculée dérive au cours du temps à cause du bruit de mesure. De plus certaines hypothèses doivent être valides pour utiliser un odomètre : les roues ne doivent pas glisser sur le sol, et le sol est supposé plan, ce qui n'est généralement pas le cas en extérieur.

Finalement, les capteurs proprioceptifs sont intéressants pour des mesures à court terme, en complément d'un capteur extéroceptif. Par exemple, on peut utiliser un odomètre, pour avoir une position mise à jour à 100 Hz alors que le capteur extéroceptif dont on dispose ne travaille qu'à 10 Hz.

1.2 Systèmes de localisation fondés sur la vision

Pour l'être humain la vision est la principale source d'informations permettant de se localiser. Cela prouve qu'un système de localisation fondé sur la vision peut parfaitement fonctionner. Partant de ce constat, il est normal d'essayer d'utiliser l'information présente dans les images pour réaliser un système de localisation

informatisé.

On peut distinguer plusieurs façons de considérer le problème de la localisation. On peut chercher à obtenir une localisation qualitative ou quantitative. Par exemple dans une maison, une localisation qualitative reviendrait à identifier la pièce où se trouve le robot sans plus de précision. Une localisation quantitative signifierait être capable de donner les coordonnées du robot dans un repère lié à la maison. En milieu extérieur, la localisation qualitative a été abordée par Morita *et al.* [54] en utilisant des algorithmes de reconnaissance et en identifiant les lieux selon la présence dans l'image de bâtiments, de feuillage, etc.... Une autre distinction peut être faite entre la localisation sans a priori (en anglais le problème "kidnapped robot" traité par Thrun *et al.* [81]), ou la localisation précise connaissant déjà une pose approximative du robot.

Dans la suite de cet état de l'art, nous allons plus particulièrement nous intéresser au calcul d'une position précise avec une connaissance a priori sur la position du robot. C'est ce que nous cherchons à réaliser.

1.2.1 Cartographie et localisation simultanées

De nombreux travaux sont regroupés sous le terme SLAM (Simultaneous Localization And Mapping) qu'on peut traduire par cartographie et localisation simultanées. Le principe consiste en partant d'un position initiale sans connaissance sur l'environnement, à construire une carte à partir des observations qui sont faites et aussitôt qu'un début de carte est disponible à s'en servir pour se localiser. Au fur et à mesure des observations, la carte est enrichie de nouveaux amers et la position des anciens amers est affinée en tenant compte des nouvelles observations. Le principe est très général et peut être utilisé avec de nombreux capteurs (télémètre, radar, caméra,...). Les premières publications sur le sujet datent de la deuxième moitié des années 1980, avec les travaux de Smith et Cheesman [78] et de Moutarlier et Chatila [56]. L'algorithme le plus fréquemment utilisé pour traiter ce problème est fondé sur l'utilisation d'un filtre de Kalman étendu [36]. Dans cette approche, le vecteur d'état contient à la fois la pose courante du robot et la position de l'ensemble des amers présents dans la carte. Une matrice de covariance est associée au vecteur d'état. Elle permet de quantifier la confiance qu'on a sur la position du robot et des amers. Lorsqu'une nouvelle image est disponible, certains amers présents dans l'image sont déjà répertoriés dans la carte. Cette nouvelle observation permet d'une part de localiser le robot et d'autre part la position de ces amers est mise à jour pour tenir compte de l'observation. En général, chaque nouvelle observation permet de réduire l'incertitude sur les amers. En outre, chaque nouvelle image permet d'enrichir la carte avec de nouveaux amers,

ce qui permet de cartographier des zones qui n'avaient pas été explorées. La principale limite de cet algorithme est l'accroissement du temps de calcul de la mise à jour du filtre de Kalman lorsque la taille de la carte augmente. La complexité de la mise à jour est en $O(N^2)$, où N est la taille du vecteur d'état. Cela signifie que faire du SLAM temps-réel avec l'approche décrite plus haut n'est possible que dans un environnement de taille réduite où le nombre d'amers utilisés est limité. Pour aller au delà, il est possible d'utiliser des sous-cartes. On peut se contenter de mettre à jour une carte locale de l'environnement immédiat du robot et changer de sous-carte lorsque le robot en sort. Il faut alors gérer la cohérence entre les sous-cartes comme le proposent Newman et Leonard [57].

Les travaux qui nous intéressent plus particulièrement ici sont ceux faisant appel à une ou plusieurs caméras, avec éventuellement en complément un capteur proprioceptif. Ce qui fait la spécificité de la vision dans l'algorithme classique de SLAM, c'est le fait qu'une seule observation d'un amer ne suffit pas à connaître sa position 3D. Ce cas particulier lié aux capteurs de vision, désigné sous le nom de "bearing-only SLAM" a été étudié par exemple par Lemaire *et al.* [40] et Bailey [4]. Il faut alors attendre d'avoir au moins deux observations d'un amer depuis deux positions différentes pour ajouter un nouvel amer à la carte. L'autre spécificité est liée au traitement de l'image pour détecter et apparier des amers, ce qui est à la base de bon nombre d'algorithmes de vision. Cette partie demande un temps de calcul non négligeable qui s'ajoute au temps lié à la gestion de la carte. Cela rend difficile la mise en place d'un algorithme de SLAM temps-réel à partir de la vision seulement. La plupart des travaux couplent vision (souvent stéréo) et odométrie pour obtenir des temps de calcul plus raisonnables. C'est le cas par exemple des travaux de Se *et al.* [76] dans lesquels le robot utilise un système trinoculaire et un odomètre. Toujours pour réduire le temps de calcul, le sol est souvent supposé plan ce qui limite à trois le nombre de degrés de liberté pour la pose du robot. Du SLAM temps-réel (30 Hz) utilisant une seule caméra avec six degrés de liberté a tout de même été réalisé par Davison [13]. Le système fonctionne très bien tant que la carte est de taille réduite (moins d'une centaine d'amers).

Le terme SLAM largement répandu dans la communauté robotique désigne quasiment le même problème que l'expression "Structure From Motion" plus courant dans la communauté de la vision par ordinateur. Et la difficulté à traiter une séquence vidéo monoculaire en temps réel pour obtenir une reconstruction 3D se retrouve aussi dans les travaux de vision. La seule différence entre les deux problèmes est que le SLAM doit forcément être fait de manière incrémentale alors que pour le problème de la reconstruction 3D, on peut parfois disposer dès le départ de l'ensemble des images de la séquence. Selon la communauté, les outils utilisés pour traiter le problème sont souvent différents : filtre de Kalman pour

le SLAM et ajustement de faisceaux pour la reconstruction 3D. Certains travaux de reconstruction 3D tendent à se rapprocher du SLAM en procédant de manière incrémentale et en temps réel. Ils donnent alors lieu à des algorithmes d'odométrie visuelle. C'est le cas des travaux de Nistér *et al.* [62] et de Mouragnon *et al.* [55]. Toutefois, contrairement à ce qui est proposé par Davison, les matrices de covariances liées aux points ne sont pas calculées. L'algorithme ne cherche pas à identifier et peut-être optimiser des amers qui ont déjà été cartographiés, puis qui sont sortis du champ de vision et qui redeviennent visibles au fil du temps.

1.2.2 Cartographie puis localisation

Pour contourner la difficulté, il est possible de considérer séparément le problème de la cartographie et celui de la localisation. Dans ce cas, il est possible de traiter la partie la plus complexe (la cartographie) hors-ligne pour obtenir un système de localisation rapide capable de fournir la pose du robot en temps réel. C'est l'approche retenue dans cette thèse, et nous allons à présent examiner les travaux déjà publiés à ce sujet.

Dans cette approche aussi, il est possible de faire la cartographie en utilisant la vision seule ou en couplant la vision avec un ou plusieurs autres capteurs. Par exemple, Cobzas *et al.* [11] utilisent une caméra placée sur une plateforme rotative ainsi qu'un télémètre laser pour construire un ensemble d'images panoramiques enrichies de l'information de profondeur fournie par le télémètre. En utilisant cette carte, la localisation est possible à partir d'une image 2D seulement (le télémètre n'est pas utilisé pour la localisation). Kidono *et al.* [38] utilisent les mêmes capteurs pour la cartographie et la localisation : une tête stéréoscopique et un odomètre. Comme souvent lorsqu'un odomètre est utilisé, le sol est supposé plan, ce qui est bien adapté à un environnement intérieur. La carte 3D est ensuite utilisée pour localiser le robot en temps réel. Ohya *et al.* [63] construisent également une carte 3D à partir d'un capteur trinoculaire et d'un odomètre. Cette carte contient la position des lignes verticales observées durant la phase d'apprentissage. Là encore, le sol est supposé plan et l'utilisation de lignes verticales limite l'application de la méthode à un environnement structuré. Une approche différente de construction de carte 3D a été proposée par Li et Tsuji [44]. Dans cette approche, la caméra est placée sur un véhicule de façon à voir sur le côté de la route. A partir de ces vues, une segmentation est faite fondée sur le mouvement qui permet de différencier les façades des différents bâtiments qui sont classés selon la distance à la caméra.

Dans certains cas, une carte 2D peut être suffisante : dans les travaux de Kelly [37], une mosaïque du sol est utilisée comme carte pour guider un véhi-

cule automatisé servant au transport dans un bâtiment industriel. La fiabilité du procédé a été démontré puisque le sytème a été testé pendant une longue période et en moyenne une seule intervention manuelle par semaine a été nécessaire. La même idée de construire une mosaïque du sol a été utilisée par Gracias et Santos-Victor [22] pour localiser un robot submersible aux abords d'une canalisation sous-marine.

1.3 Suivi de trajectoire sans localisation globale

Jusqu'à présent les travaux recensés dans cet état de l'art étaient liés à la localisation d'un robot mobile. Mais pour commander un robot sur une trajectoire déterminée, il n'est pas nécessaire de pouvoir se localiser. Il est également possible de suivre un chemin représenté par une séquence d'images clefs. Le principal avantage de cette méthode est de ne pas nécessiter l'étape de construction de carte qui est généralement coûteuse en temps de calcul. L'inconvénient est de ne pas pouvoir s'écarter de la trajectoire apprise, sauf à calculer en ligne une reconstruction 3D locale ce qui peut être difficile dans un contexte temps-réel. Dans cette approche, la vue courante et l'image clef la plus proche sont utilisées pour définir le mouvement du robot. Celui-ci peut être calculé à partir d'une heuristique simple comme dans les travaux de Matsumoto *et al.* [49] ou de Argyros *et al.* [3]. Le calcul du déplacement peut aussi être fait à partir de la détermination de la géométrie épipolaire entre les deux images. C'est le cas de la méthode de Jeon et Kim [33] qui est basée sur les points de fuite en environnement intérieur. L'algorithme proposé plus récement par Goedemé *et al.* [21] est plus générique car basé sur une mise en correspondance d'amers entre deux images panoramiques. Enfin, le déplacement à effectuer peut aussi être obtenu par asservissement visuel [29], comme le font Remazeilles *et al.* [67] ou Blanc *et al.* [8].

Chapitre 2

Mise en correspondance d'images

2.1 Etat de l'art

Le problème peut être posé de la façon suivante. On dispose de plusieurs images d'une même scène prises dans des conditions différentes. On souhaite trouver dans chaque image des points qui sont la projection d'un même point de la scène. Dans le cas d'une scène rigide, le terme "conditions différentes" désigne le plus souvent un changement de point de vue associé à un changement des conditions d'éclairage. Mettre en correspondance des primitives entre plusieurs images est un problème fondamental en vision par ordinateur. Pour cette raison, il a fait l'objet de nombreux travaux. Parmi ceux-ci, on peut identifier deux schémas classiques : détection puis appariement de primitives et suivi. Le but de ce chapitre n'est pas de dresser une liste exhaustive des techniques mises au point pour mettre des images en correspondance. Nous nous limitons à présenter certaines méthodes utilisables dans notre application afin de justifier et d'expliquer la méthode retenue.

2.1.1 Détection - appariement

On commence par détecter des primitives dans les images. Pour chacune des primitives détectées, on calcule un descripteur local. Ce descripteur local permet de calculer un score de ressemblance entre deux primitives. Finalement, l'appariement est réalisé par un algorithme qui utilise les scores de ressemblance et souvent des contraintes géométriques additionnelles.

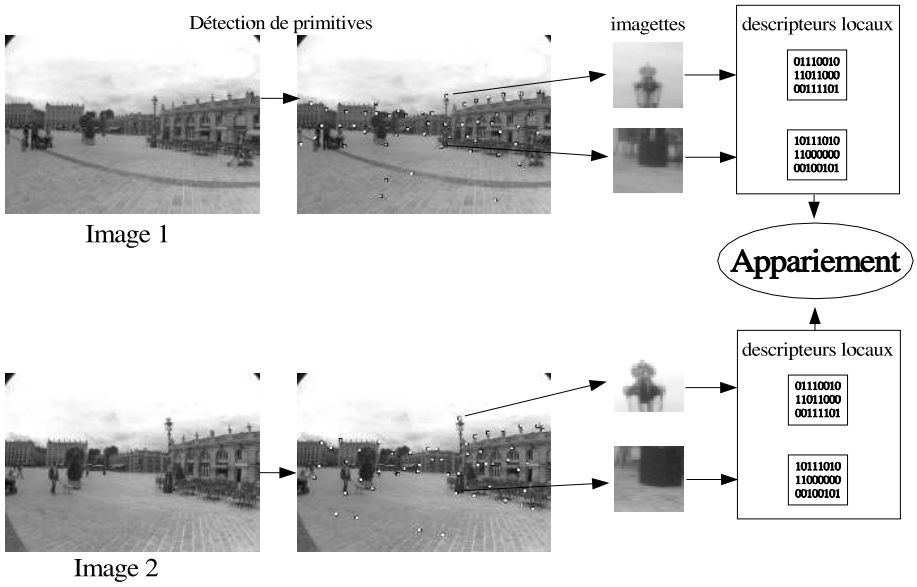

FIG. 2.1 – Appariement de primitives : détection, calcul de descripteurs locaux, appariement

Détection de primitives

On peut chercher à détecter différents types de primitives : des régions, des contours ou des points. Les points sont les plus exploités, car les algorithmes géométriques tels que le calcul de pose, de géométrie épipolaire ou l'ajustement de faisceaux travaillent le plus souvent sur des points. D'autre part, détecter des points est moins sensible aux occultations que détecter des primitives qui occupent une part significative de l'image. Mais les régions et les contours (segments ou courbes) ont également été utilisés pour la mise en correspondance d'images.

Les premiers détecteurs de points d'intérêt ont vu le jour à la fin des années 1970 avec les travaux de Beaudet [6] puis de Moravec [53]. L'idée étant de rechercher dans l'image des zones où le signal varie fortement. Harris et Stephens [25] ont amélioré l'idée en calculant les courbures principales de la fonction d'autocorrélation du signal. Une courbure importante dans les deux directions principales indique la présence d'un coin qui est retenu comme point d'intérêt. D'autres détecteurs de points basés contours ont aussi été proposés. Dans ce cas, les points retenus sont définis comme des points d'intersection de plusieurs contours ou des points d'inflexion, ou de courbure maximale. Les travaux de Schmid *et al.* [75] ont permis d'avoir une comparaison de différents détecteurs. Dans cet article, les détecteurs sont évalués du point de vue de la répétabilité lorsque les images sont

prises dans des conditions différentes : changement de luminosité ou transformation géométrique. Les auteurs montrent qu'une version améliorée du détecteur de Harris offre la meilleure répétabilité. Suite à ces travaux, Mikolajczyk et Schmid [50] ont développé un nouveau détecteur invariant aux transformations affines. Ce détecteur utilise une approche multi-échelle et permet de caractériser le voisinage du point d'intérêt pour permettre le calcul d'un descripteur local. La même idée est exploitée par Lowe [47] pour détecter des points en leur associant une échelle et une orientation. Ces informations sont utilisées dans le calcul du descripteur local SIFT.

L'utilisation des contours est beaucoup moins répandue dans la littérature. Les contours sont souvent utilisés dans le cas d'objets non texturés pour lesquels l'utilisation de points d'intérêt ne donne pas de résultats satisfaisants. C'est le cas par exemple des travaux de Mikolajczyk *et al.* [51], de Carmichael et Hébert [10] ou de Bourgeois [9].

La détection directe de régions suivie d'un appariement est délicate à réaliser car la segmentation en régions peut donner des résultats différents entre deux images proches à cause principalement des occultations mais aussi du bruit ou d'un changement de point de vue. Ceci dit, la détection et l'appariement de points ne peut se faire sans considérer le voisinage du point. C'est donc une région définie autour des points d'intérêt qui est utilisée pour calculer un descripteur local. L'utilisation directe de régions conduit souvent à des algorithmes plus coûteux en temps de calcul. Une méthode pour extraire des régions invariantes aux transformations affines a toutefois été proposée par Tuytelaars *et al.* [86] pour mettre en correspondance des images prises depuis des points de vues éloignés.

Calcul de descripteurs locaux

Lorsque les primitives sont détectées, on calcule pour chacune d'entre elles un descripteur local. Considérons le cas où les primitives sont des points. On considère une imagette, de taille fixe ou variable, dont le centre est le point détecté. Le descripteur local est un condensé de l'information présente dans l'imagette. La principale qualité souhaitée pour un descripteur local est l'invariance aux changements de luminosité et de point de vue. Mais une plus grande invariance va souvent de pair avec un temps de calcul plus élevé.

Le descripteur local le plus simple est un vecteur contenant l'intensité de chaque pixel d'une imagette de taille fixe. Dans ce cas, le score qui mesure la ressemblance entre deux primitives est la somme des différences de niveau de gris

pixel par pixel :

$$SSD(P_1,P_2) = \sum_{i=-N}^{N}\sum_{j=-N}^{N}(I_1(x_1+i,y_1+j) - I_2(x_2+i,y_2+j))^2 \tag{2.1}$$

Cette façon de calculer un score de ressemblance entre deux imagettes a l'avantage d'être très simple et très rapide à calculer mais elle n'est pas invariante aux changements d'illumination. Pour cela, il est préférable d'utiliser une corrélation centrée normée (ZNCC) :

$$ZNCC(P_1,P_2) = \frac{\sum_{d\in V}(I_1(P_1+d) - \overline{I_1}(P_1))(I_2(P_2+d) - \overline{I_2}(P_2))}{\sqrt{\sum_{d\in V}(I_1(P_1+d) - \overline{I_1}(P_1))^2}\sqrt{\sum_{d\in V}(I_2(P_2+d) - \overline{I_2}(P_2))^2}} \tag{2.2}$$

avec

$$\begin{aligned} \overline{I_i}(P_i) &= \tfrac{1}{|V|}\textstyle\sum_{d\in V} I_i(P_i+d) \\ V &= \{-N,\ldots,N\}\times\{-N,\ldots,N\} \end{aligned} \tag{2.3}$$

Cette méthode demande peu de temps de calcul. De plus elle permet de tirer facilement parti des jeux d'instruction vectoriels des processeurs récents. Elle permet donc d'utiliser de nombreuses primitives (plusieurs centaines par image) en temps réel. Son principal inconvénient est de ne pas être invariante aux changements de perspective.

Pour pallier cet inconvénient, d'autres méthodes ont été proposées. Elles sont souvent plus coûteuses en temps de calcul, mais certaines d'entre elles sont utilisées dans des applications de vision temps réel. L'idée développée par Lepetit *et al.* [41] consiste à considérer le problème de la mise en correspondance de primitives comme un problème de classification. Dans cette approche, on cherche à mettre en correspondance les points détectés sur une image avec les points d'un objet dont le modèle est connu. Le modèle de l'objet permet de générer, en faisant varier le point de vue, une collection d'imagettes associés à chaque point. La collection d'imagettes constitue alors une base d'apprentissage pour un algorithme de reconnaissance. L'intérêt de cette méthode est de faire la plus grande partie des calculs hors ligne et d'utiliser une méthode très rapide de reconnaissance en ligne.

Une autre possibilité intéressante est de ne pas chercher à obtenir un descripteur local invariant au changement de point de vue, mais plutôt à précalculer l'apparence qu'aura l'imagette en fonction du point depuis lequel elle est observée. Cela a été exploité par exemple par Molton *et al.* [52] pour du SLAM en temps réel en modélisant chaque point d'intérêt et son imagette associée par un plan texturé. Cette méthode suppose qu'on connaisse une pose approximative de la caméra avant l'appariement.

Enfin, d'autres descripteurs locaux sont invariants à des transformations géométriques plus complexes. Le plus connu d'entre eux est probablement SIFT développé par Lowe [47]. Ce descripteur est fondé sur une approche multi-échelle et sur la construction d'un histogramme de l'orientation des gradients autour du point d'intérêt. Cela lui permet d'être invariant aux changements affines d'illuminations, aux rotations, aux changements d'échelle. L'appariement tolère aussi des transformations affines modérées. Plus récement Ling et Jacobs [45] proposent d'utiliser les lignes de niveau de l'intensité lumineuse autour du point d'intérêt, pour construire un descripteur invariant aux déformations d'un objet qui peut être non rigide.

Algorithmes d'appariement

L'algorithme d'appariement est chargé de constituer une liste de couples de points à partir d'une liste de candidats potentiels. Chaque candidat est un triplet (P_1, P_2, s) où P_1 est une primitive dans la première image, P_2 une primitive dans la seconde image et s un score de ressemblance calculé à partir des descripteurs locaux associés à P_1 et P_2. La liste de couples finale doit respecter la contrainte d'unicité : chaque point de l'image i doit être apparié à au plus un point de l'image j, pour $(i, j) \in \{1,2\}^2$.

Il existe plusieurs stratégies pour obtenir une liste de couples définitive. Dans la plupart des cas, on commence par réduire le nombre de candidats en éliminant tous ceux qui ont un score inférieur à un certain seuil. Ensuite les deux méthodes les plus simples consistent soit à garder les meilleurs soit à éliminer les plus mauvais. Dans le premier cas, on choisit le candidat avec le meilleur score et on applique la contrainte d'unicité pour éliminer parmi les candidats restants ceux qui ne peuvent plus respecter la contrainte. Dans certains cas, cette méthode peut être trop brutale si un point P_1 peut être potentiellement apparié à P_2 et à Q_2 avec des scores très proches on risque de garder le mauvais appariement. Dans un cas comme celui-ci, il peut être préférable d'apparier d'abord des points qui ne présentent pas cette ambiguité. Les méthodes de relaxation, comme celle proposée par Zhang *et al.* [87], visent à résoudre ce problème.

D'autres méthodes plus élaborées ont également été proposées afin d'exploiter des contraintes de voisinage. Il est naturel de penser que deux points voisins de l'image 1 vont probablement être appariés à deux points voisins dans l'image 2. Cela se vérifie généralement si les points voisins dans l'image sont à peu près à la même distance de la caméra. Jung et Lacroix [34] ont développé un algorithme permettant de regrouper des points d'intérêt pour les apparier en exploitant des contraintes de voisinage. Cela permet de diminuer le nombre de faux apparie-

ments.

2.1.2 Suivi

Une autre façon de mettre en correspondance deux images est de faire un suivi en utilisant des images prises à des instants rapprochés et en exploitant le fait que chaque point de l'image s'est peu déplacé. Dans le cas de suivi de primitives, on peut éviter de faire une nouvelle détection de primitives à chaque image et simplement chercher à retrouver la nouvelle position des primitives dans l'image $i+1$ à partir de leur position dans l'image i.

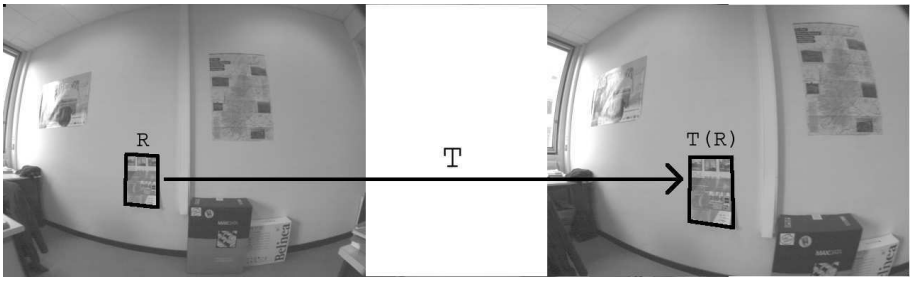

FIG. 2.2 – Principe du suivi

Pour suivre une région R en se basant sur la texture, on cherche les paramètres d'une transformation géométrique T qui minimisent la différence entre la région R dans l'image i et la région $T(R)$ dans l'image $i+1$. Le plus souvent le suivi concerne une région plane et la transformation considérée est une homographie.

Cela a été proposé par Lucas et Kanade [48] puis amélioré par Shi et Tomasi [77] pour suivre le voisinage de points d'intérêt. Le même principe peut être employé pour suivre une surface plane comme dans les travaux de Jurie et Dhome [35] ou plus récemment de Benhimane et Malis [7].

On peut également faire un suivi en utilisant les contours plutôt que la texture. L'utilisation de contours est plus fréquement employée pour suivre un objet dont on connaît le modèle comme dans les travaux de Drummond et Cipolla [15], mais cela est possible aussi sans modèle connu comme le font Akhriev et Kim [1].

2.2 Méthode choisie

Parmi l'ensemble des méthodes décrites dans la littérature, il faut en choisir une qui convienne à l'application visée. Dans notre cas, la principale contrainte

est celle de la vitesse d'exécution dans la partie localisation. Pour atteindre le temps réel vidéo, le temps disponible pour détecter et apparier les primitives entre deux images est de l'ordre de 30 ms, le reste du temps est consacré au calcul de la pose à partir des appariements obtenus. De façon générale, les méthodes les plus précises et robustes sont aussi celles qui demandent le temps de calcul le plus important. Cela signifie que si on choisit un détecteur et des descripteurs locaux rapides, on pourra traiter beaucoup de primitives sur chaque image. Au contraire, si on veut utiliser une méthode d'appariement plus précise ou plus robuste, on va devoir se contenter d'un nombre de primitives plus réduit. Selon les auteurs, les deux stratégies sont exploitées. Par exemple, Nistér [62] fait de l'odométrie visuelle en détectant jusqu'à 5000 points d'intérêt par image, alors que l'algorithme de SLAM de Davison [13] se contente de 10 points par image. Le principal intérêt d'utiliser un grand nombre de primitives est la robustesse aux occultations. En effet, si seulement quelques primitives sont utilisées sur l'image, une occultation, même de surface réduite, risque de les masquer. Dans [13], cette difficulté est résolue en choisissant de nouvelles primitives dès que celles utilisées sont masquées. Pour notre algorithme, nous avons préféré utilisé un grand nombre de points d'intérêt (1500 environ détectés dans chaque image) pour plus de robustesse. D'autre part, entre apprentissage et localisation, il peut s'écouler plusieurs jours, ce qui signifie que certains éléments présents dans la scène vont être modifiés. On voit donc dès le départ, qu'une partie des points mémorisés ne seront plus utilisables et qu'il faut donc détecter plus de points que le minimum nécessaire pour se localiser. Ceci nous a conduit à utiliser le détecteur de coins de Harris qui est un détecteur rapide. Il est associé à un calcul de corrélation centré normé, qui est lui aussi rapide à calculer même si le nombre de points est grand.

2.2.1 Points d'intérêt de Harris

L'idée de la méthode de Harris [25] est d'étudier la courbure de la fonction d'auto-corrélation du signal. Si en un point les deux courbures principales sont élevées alors on a affaire à un coin. Si une seule des deux courbures est grande, alors il s'agit d'un bord. Le critère de Harris est donc une fonction qui est maximale quand les deux courbures sont grandes et faible si une des courbures au moins est faible. Avant de calculer les dérivées du signal, il faut lisser l'image qui peut être bruitée. Plusieurs variantes sont possibles au niveau des lissages. La variante choisie est légèrement différente de celle donnée dans [75]. Nous avons préféré utiliser un lissage avec des coefficients binômiaux plutôt qu'un lissage Gaussien pour réduire les temps de calcul. Le filtre binômial de taille n est une approximation du filtre Gaussien obtenu pour $\sigma = \frac{1}{2}\sqrt{n-1}$. De plus, pour les filtres

de taille réduite dont nous avons besoin, les coefficients sont simples ($\frac{1}{2}, \frac{1}{4}, \ldots$). cela nous permet d'utiliser des instructions SIMD très efficaces.

On commence par un lissage de l'image dans les deux directions en convoluant avec le noyau $\frac{1}{4}(\ 1 \quad 2 \quad 1\ )$ qui est une approximation du filtre Gaussien avec $\sigma = \frac{1}{2}\sqrt{2}$. Puis on calcule la matrice de Harris, qui est symétrique, en chaque pixel :

$$M_H(x,y) = \begin{pmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \left(\frac{\partial I}{\partial y}\right)^2 \end{pmatrix} \tag{2.4}$$

On obtient trois images contenant les trois coefficients de la matrice. Chaque image est lissée en x et en y avec un masque $\frac{1}{16}(\ 1 \quad 4 \quad 6 \quad 4 \quad 1\ )$ ce qui est une approximation du filtre Gaussien obtenu pour $\sigma = 1$. Finalement on évalue le critère de Harris par :

$$H(x,y) = det(M_H) - k.trace(M_H)^2 \tag{2.5}$$

avec k un coefficient bien choisi ($k = 0.04$ dans la version actuelle du code). Un pixel (x,y) est un maximum local de H si $H(x,y) \geq H(x+i, y+j)$ pour tout $(i,j) \in \{-1,0,1\}^2$. Finalement on conserve les plus grands maxima locaux comme points d'intérêt.

Les points sont d'abord détectés au pixel près. Ensuite on affine la position de chaque point par la méthode suivante : si un maximum local de H est détecté en (x,y) (pixel à coordonnées entières), alors on recherche la position du maximum indépendemment en x et en y. L'abscisse du maximum est obtenue comme l'abscisse x^* du maximum de $P(u)$ où P est le polynôme de degré 2 tel que $P(x-1) = H(x-1,y)$, $P(x) = H(x,y)$ et $P(x+1) = H(x+1,y)$. L'ordonnée du maximum est obtenue de la même façon. Ceci est illustré sur la figure 2.3. L'apport de la méthode subpixellique a pu être mesuré en réalisant la détection et l'appariement de points d'intérêt sur des images de synthèse pour lesquelles une déformation connue a été appliquée (rotation, changement d'échelle ou homographie générée aléatoirement). La proportion de points correctement appariés à moins de ε pixels près est donnée dans le tableau 2.1. On constate une nette amélioration de la précision pour un coût supplémentaire minime en temps de calcul.

Sur une image de taille 512×384, on obtient généralement de l'ordre de 5000 maxima locaux de H. Parmi ceux-ci on en retient seulement une partie pour éviter de passer trop de temps dans l'algorithme d'appariement. Les maxima locaux sont classés en fonction de la valeur de $H(x,y)$. La première idée a été de retenir les N valeurs les plus élevées, mais sur certaines images tous les points étaient

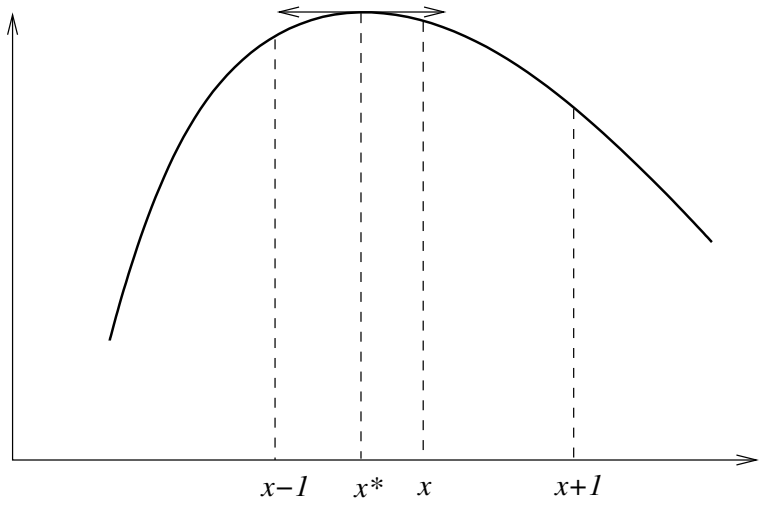

FIG. 2.3 – Détection subpixellique des points d'intérêt

ε	détection au pixel	détection subpixellique
0.2	9 %	47 %
1.0	88 %	93 %

TAB. 2.1 – Nombre de points appariés à mieux que ε pixels.

concentrés dans une même zone. Il a donc fallu améliorer le critère de sélection pour obtenir des points mieux répartis sur l'image. Pour cela l'image est divisée en 64 baquets rectangulaires (8×8). La méthode finalement retenue consiste à retenir les N meilleurs points globalement sur l'image et à rajouter les M meilleurs points de chaque baquet. Habituellement on prend $N = 500$ et $M = 20$. On obtient ainsi au maximum 1780 points d'intérêt par image mais souvent un peu moins car certains baquets dans des zones uniformes ont moins de 20 maxima locaux. Le résultat obtenu par les deux méthodes est illustré sur la figure 2.4. Pour cette image, les points avec la valeur $H(x,y)$ la plus élevée sont presque tous concentrés dans les branches de l'arbre, et on en trouve très peu sur le sol alors que celui-ci est quand même texturé. L'utilisation de baquets permet d'avoir des points d'intérêt bien répartis dans l'image. Ceci est une caractéristique importante car si tous les points sont concentrés dans une petite zone, une occultation de cette zone risque de faire échouer la localisation.

FIG. 2.4 – Détection de points d'intérêt. A gauche, l'image originale, au milieu les 1500 meilleurs points d'intérêt, à droite les points détectés en utilisant la sélection par baquets. L'utilisation de baquets permet d'obtenir des points mieux répartis dans l'image.

2.2.2 Mise en correspondance

Méthode standard

A partir des points détectés dans deux images, on cherche à former des appariements (P_1, P_2, s) avec P_i point d'intérêt de l'image i et s un score de corrélation d'autant plus élevé que le voisinage des points P_1 et P_2 est semblable. Pour chaque point P_1 de l'image 1, on définit une zone de recherche (ou Region Of Interest, ROI) rectangulaire centrée en les coordonnées de P_1 dans l'image 2 comme sur la figure 2.5. On forme tous les appariements possibles (P_1, P_2, s) avec P_2 point d'intérêt détecté dans la région de recherche. Ils sont retenus si leur score dépasse un certain seuil.

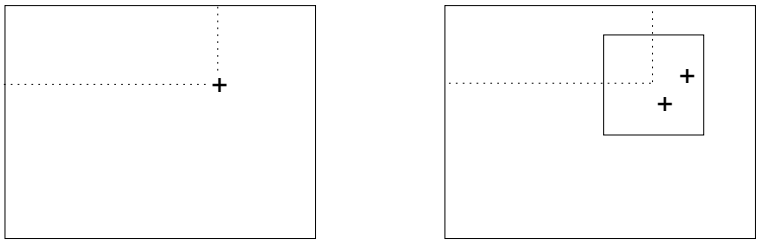

FIG. 2.5 – Un point détecté dans l'image gauche et la région d'intérêt correspondante dans l'image de droite

Pour satisfaire la contrainte d'unicité (un point ne peut pas être apparié à deux points différents), il faut supprimer un certain nombre d'appariements candidats. Pour cela, on extrait l'appariement (P_1, P_2, s) avec le score maximal, et on élimine

tous les autres couples contenant P_1 ou P_2. On répète cette opération jusqu'à ce qu'il n'y ait plus de candidats disponibles.

Le score de corrélation utilisé est le ZNCC (corrélation croisée centrée et normée) sur un voisinage de 11×11 pixels. Le seuil minimum pour retenir le couple est fixé à 0.8.

$$zncc_{11}(P_1,P_2) = \frac{\sum_{d\in V_{11}}(I_1(P_1+d)-\overline{I_1}(P_1))(I_2(P_2+d)-\overline{I_2}(P_2))}{\sqrt{\sum_{d\in V_{11}}(I_1(P_1+d)-\overline{I_1}(P_1))^2}\sqrt{\sum_{d\in V_{11}}(I_2(P_2+d)-\overline{I_2}(P_2))^2}}$$

avec

$$\overline{I_i}(P_i) = \frac{1}{11^2}\sum_{d\in V_{11}} I_i(P_i+d)$$

et V_{11} le voisinage 11×11 centré en $(0,0)$.

Cette méthode permet d'obtenir une liste de points appariés entre les deux images. Parmi les couples obtenus, certains ne sont pas formés de points homologues : les deux points du couple ne sont pas deux projections d'un même point de l'espace. Les algorithmes de calcul de pose ou de la géométrie épipolaire qui utilisent ces appariements doivent donc être robustes aux faux appariements.

Définition de la zone de recherche

Dans l'algorithme donné dans le paragraphe précédent, le choix de la région de recherche est primordial pour obtenir de bonnes performances. Si la zone de recherche associée à P_1 est trop petite, ou mal placée sur l'image 2, le risque est que le point qui devrait être apparié à P_1 ne se trouve pas dans la zone de recherche. Si la zone de recherche est trop grande, le temps de calcul augmente car on doit considérer plus de couples potentiels et calculer un plus grand nombre de scores de corrélation. De plus, le risque de faux appariements augmente si les images comportent des motifs répétitifs (plusieurs fenêtres identiques sur une façade de bâtiments par exemple). Il faut donc selon les informations dont on dispose définir au mieux les zones de recherche.

Les schémas A,B,C de la figure 2.6 illustrent les trois méthodes utilisées. Dans le cas où on n'a pas d'information à priori (schéma A) la seule méthode consiste à associer au point $P_1(x,y)$ une zone de recherche dans l'image 2 centrée en (x,y) et de taille moyenne (120×80 pixels). C'est ce qui est fait au départ dans l'algorithme de reconstruction 3D. Par contre dès qu'on dispose d'une information, même approximative sur la position relative des deux caméras, on peut mieux définir la zone de recherche. Deux situations sont possibles. Si on connaît la contrainte

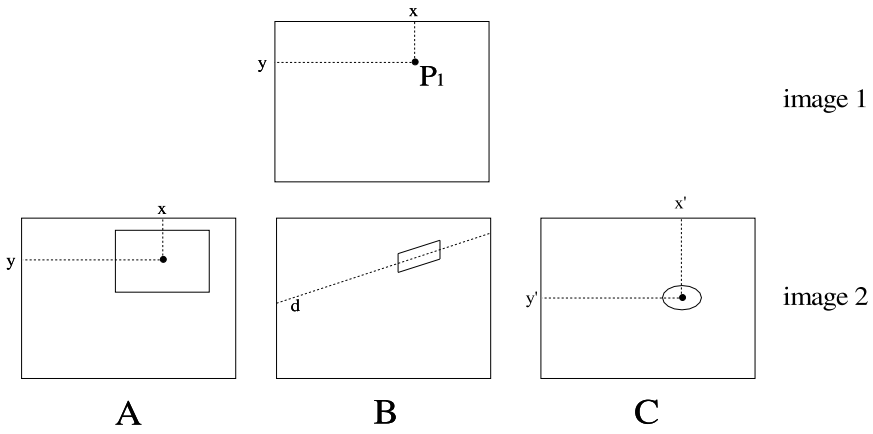

FIG. 2.6 – Trois méthodes pour définir la zone de recherche

épipolaire entre les deux images (schéma B), alors la zone de recherche peut être définie comme une bande de quelques pixels de large autour de la droite épipolaire issue de P_1 (on peut se contenter d'un segment plutôt que de toute la droite). Si on connaît une pose approximative de la caméra et les coordonnées 3D du point correspondant à P_1 (schéma C), alors il est possible de calculer les coordonnées dans l'image 2 de l'endroit où doit se projeter le point. C'est ce qui se produit dans l'algorithme de localisation. On dispose aussi d'une matrice de covariance associée à la pose de la caméra. On peut donc en déduire un ellipsoïde de confiance pour la position du point P_2 dans l'image 2 et ainsi avoir une position et une taille de zone de recherche adaptée. En pratique, plutôt que de considérer une région elliptique de l'image, on prend plutôt le rectangle tangent à l'ellipse.

Si l'objectif utilisé présente une forte distorsion, les régions de recherche telles qu'elles sont définies sur la figure 2.6 ne sont pas parfaitement adaptées. Ainsi, sous l'effet de la distorsion, les droites épipolaires deviennent des courbes. Les ellipses devraient aussi subir la distorsion. Nous nous sommes contentés de contourner ce problème en augmentant un peu la taille des zones de recherche sans changer leur forme. Il pourrait être intéressant de prendre en compte la distorsion de l'objectif pour définir des zones de recherche plus petites et accélérer ainsi l'appariement des points d'intérêt.

Chapitre 3

Cartographie 3D

3.1 Reconstruction 3D à partir d'une caméra en mouvement

3.1.1 Formulation du problème

Un capteur quel qu'il soit nous donne une information limitée sur le monde extérieur. A un instant donné, la caméra produit une image qui résulte de l'observation des objets placés devant elle. Le processus de formation de l'image peut être formalisé assez simplement. Le modèle le plus courant pour une caméra est le modèle sténopé représenté sur la figure 3.1. A partir de ce modèle, il est facile de déterminer la position dans l'image de la projection d'un point dont la position 3D est connue. Le problème inverse (retrouver la position d'un point 3D connaissant sa projection) ne peut pas être résolu si on ne dispose que d'une seule image. Il faut utiliser la projection du point dans deux images au minimum pour obtenir une solution. Dans ce cas, à condition que la pose des caméras soit connue, la solution peut être obtenue avec la construction géométrique simple de la figure 3.2. Un problème se pose lorsqu'on cherche à faire la même chose sans connaître la pose de la caméra. Dans la pratique, c'est un cas très courant car la pose d'une caméra est souvent difficile à mesurer. En particulier, pour une caméra en mouvement tenue à la main ou montée sur un véhicule la pose de la caméra est inconnue, tout comme le sont les positions respectives des objets constituant la scène observée. Dans ce cas, on doit mettre en œuvre des algorithmes plus complexes pour retrouver la structure tridimensionnelle de la scène ainsi que la position de la caméra au moment de chaque prise de vue.

Le problème de la reconstruction 3D à partir d'une caméra en mouvement est désigné en anglais par l'expression "Structure From Motion". Le problème peut

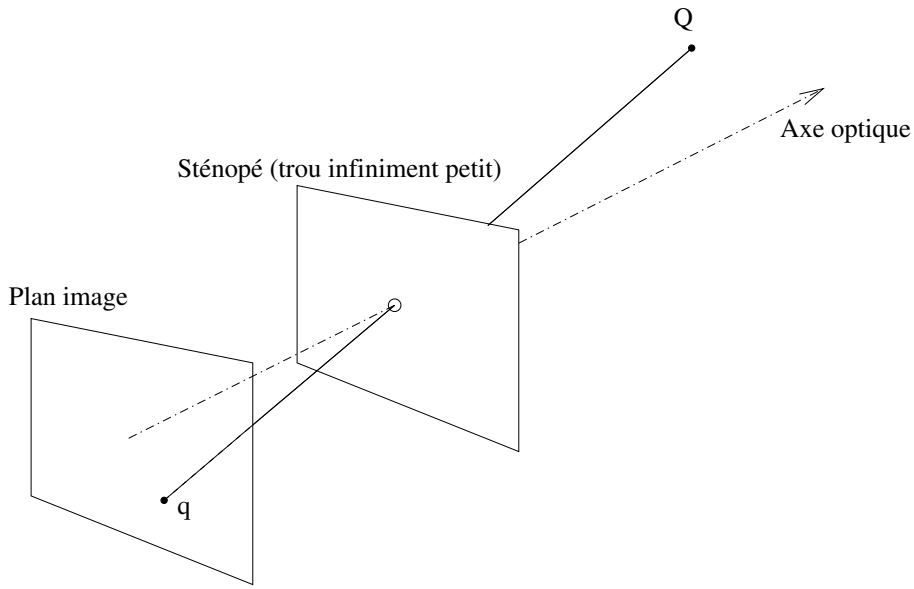

FIG. 3.1 – Projection d'un point grâce au modèle sténopé d'une caméra

être formulé comme suit. Etant donné un ensemble d'images d'une même scène, on souhaite trouver les paramètres des caméras ayant pris ces images ainsi qu'un modèle tridimensionnel de la scène. Selon les besoins, le modèle tridimensionnel peut être un modèle complexe qui décrit la surface des objets ou un modèle constitué d'un simple nuage de points 3D. Dans notre cas, le modèle que nous voulons construire est le modèle le plus simple qui permette de localiser correctement le robot dans la phase de navigation autonome. Un nuage de points est suffisant pour cette application. Le calcul de la position de la caméra pour chaque image et celui du modèle 3D ne sont pas deux problèmes indépendants : les paramètres des caméras et les coordonnées des points sont des inconnues qui doivent être calculées simultanément. Une analyse rapide pour une caméra correspondant au modèle sténopé montre que la connaissance de la projection d'un point 3D dans une image donne deux équations scalaires. Si on dispose de m images et n points se projetant dans chacune des images, alors on peut écrire $2mn$ équations. D'un autre côté, on peut dénombrer le nombre de paramètres à déterminer : 4 paramètres intrinsèques de la caméra (dans sa modélisation la plus simple), 6 paramètres extrinsèques pour chaque position de la caméra et 3 coordonnées pour chaque point, soit au total $4+6m+3n$ inconnues. A condition que m et n soient suffisamment grands (et en dehors de certaines surfaces ou mouvements critiques [80]), il est possible de résoudre le problème. De nombreuses méthodes ont été étudiées pour cela et elles

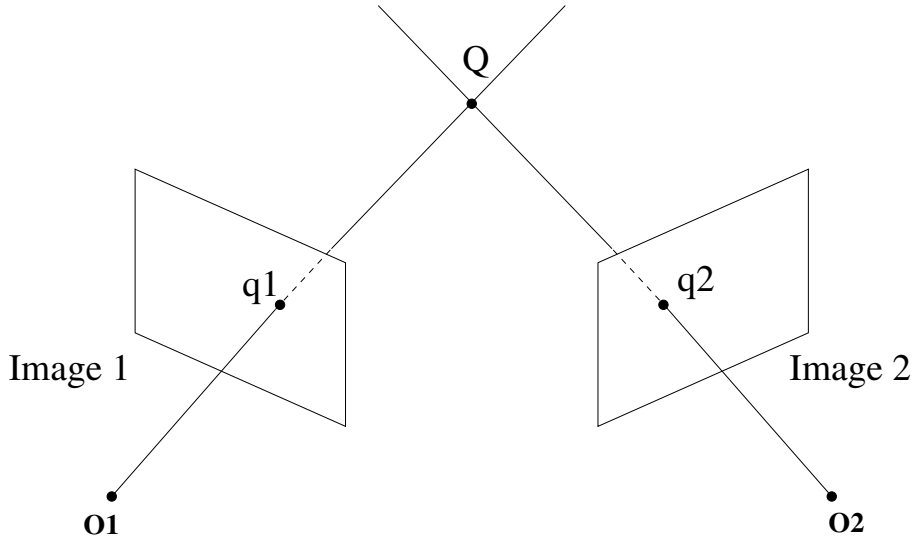

FIG. 3.2 – Triangulation d'un point

vont être présentées dans les paragraphes suivants. Ce sera également l'occasion d'introduire les notations qui seront utilisées dans la suite.

3.1.2 Travaux antérieurs

L'obtention d'un modèle tridimensionnel de la scène à partir d'images est un problème fondamental en vision par ordinateur. Les premiers algorithmes complets capables de produire un modèle 3D de façon totalement automatique ont vu le jour à la fin des années 1990, mais les bases de ces travaux sont bien plus anciennes que cela. Le cadre naturel pour étudier la formation des images est la géométrie projective, dont les premiers éléments étaient connus dès l'antiquité mais qui a été plus affinée à partir du 17^e siècle. Son exploitation pour faire de la mesure à partir d'images a été l'œuvre des photogrammètres au 20^e siècle afin de pouvoir établir des cartes à partir de photographies aériennes. C'est aussi l'usage des appareil photographiques pour des usages photogrammétriques qui a rendu nécessaire la modélisation précise des caméras. L'arrivée des ordinateurs a permis de nouvelles applications pour lesquelles l'extraction d'informations tridimensionnelles à partir d'images était devenue importante : modélisation pour la synthèse d'image, robotique, ... Pour ces applications l'automatisation des processus est devenue nécessaire. L'association automatique de primitives entre images permet cette automatisation mais au prix d'une plus grande complexité car elle produit forcément de faux appariements. Les algorithmes de calcul géo-

métrique qui suivent doivent donc prendre en compte explicitement la présence de données erronées. Enfin, des recherches ont également été menées pour utiliser des caméras dont on ne connaît pas les paramètres intrinsèques ou dont les paramètres peuvent varier au cours du temps. Tous ces travaux ont permis d'abord de formaliser la géométrie de deux ou plusieurs images, puis de mettre au point des algorithmes capables de construire un modèle 3D à partir d'images. Ces étapes vont être présentées dans les paragraphes qui suivent après une introduction des notations utilisées.

Modèle de caméra et notations

Projection d'un point Le modèle le plus simple pour une caméra est le modèle sténopé. Ce modèle correspond à une caméra dont l'objectif est un trou infiniment petit situé à une distance f du plan où se forme l'image. On appelle distance focale la longueur f. C'est ce modèle que nous allons utiliser pour écrire les équations qui décrivent la projection d'un point de l'espace. Les principales notations sont introduites ici et résumées dans le tableau 3.1.

Paramètres intrinsèques	
f_x, f_y	focale en x et en y
(u_0, v_0)	position du point principal (en pixels)
K	matrice des paramètres intrinsèques
$(X_{p_{centre}}, Y_{p_{centre}})$	coordonnées du centre de l'image
$(a_0, a_1, a_2, a_3, a_4)$	coefficients de distorsion radiale
Notation des coordonnées	
(X_w, Y_w, Z_w)	point de la scène dans le repère du monde $\mathbf{R_w}$
(X_c, Y_c, Z_c)	point de la scène dans le repère caméra $\mathbf{R_c}$
(x_i, y_i)	projection du point dans le plan image
(x_{pu}, y_{pu})	projection du point (en pixels) pour une caméra sans distorsion et (u_0, v_0) ramené au centre de l'image
(x_p, y_p)	projection du point ayant subi la distorsion géométrique (en pixels)
Paramètres extrinsèques	
$T = (T_x, T_y, T_z)$	Coordonnées du centre optique de la caméra dans $\mathbf{R_w}$
R	Matrice de rotation donnant l'orientation de la caméra
(α, β, γ)	Angles d'Euler paramétrant R
Matrice de projection	
P	Matrice 3×4 de projection ou matrice de caméra

TAB. 3.1 – Notations utilisées pour les caméras et les points

Les points observés par la caméra sont définis par leurs coordonnées dans le repère du monde $\mathbf{R_w}$. La caméra elle-même est positionnée dans ce repère. Il faut au minimum six paramètres, appelés paramètres extrinsèques, pour définir sa position et son orientation. Les trois premiers paramètres extrinsèques sont les coordonnées du centre optique de la caméra dans le repère du monde $T = (T_x, T_y, T_z)$. L'orientation peut être donnée soit par une matrice de rotation R, soit par les trois angles d'Euler (α, β, γ). Les deux façons de paramétrer la rotation seront utilisées selon la situation. La matrice de rotation est obtenue à partir des angles d'Euler par :

$$R = \begin{bmatrix} \cos(\beta)cos(\gamma) & cos(\gamma)sin(\beta)sin(\alpha) - sin(\gamma)cos(\alpha) & cos(\gamma)sin(\beta)cos(\alpha) + sin(\gamma)sin(\alpha) \\ sin(\gamma)cos(\beta) & sin(\gamma)sin(\beta)sin(\alpha) + cos(\gamma)\cos(\alpha) & sin(\gamma)sin(\beta)cos(\alpha) - cos(\gamma)sin(\alpha) \\ -sin(\beta) & cos(\beta)sin(\alpha) & cos(\beta)cos(\alpha) \end{bmatrix} \tag{3.1}$$

Soit un point $Q(X_w, Y_w, Z_w)$ dont les coordonnées sont données dans le repère du monde. Les étapes suivantes sont nécessaires pour calculer les coordonnées de la projection du point dans l'image. Il faut commencer par déterminer ses coordonnées (X_c, Y_c, Z_c) dans le repère caméra $\mathbf{R_c}$ en utilisant la relation écrite en coordonnées homogènes (le signe $\equiv$ désigne une égalité à un facteur multiplicatif près) :

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \equiv \begin{bmatrix} R & -RT \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.2}$$

Il reste à écrire les équations de projection :

$$\begin{cases} x_i = \frac{X_c}{Z_c} \\ y_i = \frac{Y_c}{Z_c} \end{cases} \tag{3.3}$$

ou de façon équivalente en coordonnées homogènes :

$$\begin{bmatrix} sx_i \\ sy_i \\ s \end{bmatrix} \equiv \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{3.4}$$

Puis le calcul de la position du point image sur le capteur en pixels noté (x_{pu}, y_{pu}) :

$$\begin{bmatrix} x_{pu} \\ y_{pu} \\ 1 \end{bmatrix} \equiv \begin{bmatrix} f/dx & 0 & u_0 \\ 0 & f/dy & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \equiv K \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{3.5}$$

où f est la focale de l'objectif exprimé en pixels. Dans le cas où les pixels du capteur sont carrés, le rapport dx/dy est très proche de 1. On note aussi $f_x = f/dx$ et $f_y = f/dy$. On note K la matrice des paramètres intrinsèques.

L'ensemble de ces opérations peut être écrite en coordonnées homogènes sous la forme d'un simple produit de matrices :

$$\begin{bmatrix} x_{pu} \\ y_{pu} \\ 1 \end{bmatrix} \equiv P \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.6}$$

dans cette équation P est de taille 3×4, on l'appelle la matrice de projection associée à la caméra, ou parfois matrice de la caméra.

Modélisation de la distorsion Les objectifs réellement utilisés ne correspondent pas exactement au modèle sténopé. Ils sont le plus souvent composés de plusieurs lentilles afin de corriger au mieux les aberrations d'une lentille simple. Malgré tout certaines aberrations demeurent. Certaines affectent la netteté de l'image, mais celle qui nous intéresse plus particulièrement est la distorsion géométrique. En raison de cette aberration, un point ne se projette pas à la même position sur le plan image qu'avec un sténopé. De ce fait l'image d'une droite du monde réel n'est pas une droite dans le plan focal. Cette aberration peut être modélisée et compensée. Ce paragraphe détaille le modèle de distorsion utilisé. Il n'est pas toujours indispensable de corriger la distorsion de façon logicielle si on utilise un objectif de bonne qualité et de focale assez grande. La distorsion est d'autant plus visible que le champ angulaire de l'objectif est grand. Dans notre cas, l'objectif utilisé est un objectif à très grand angle (fish-eye). Une image prise à travers cet objectif est visible sur la figure 3.3. L'image résultat, après la correction de la distorsion apparaît sur la même figure. Même s'il est possible de corriger toute l'image comme dans cet exemple, il est souvent plus rapide de ne corriger que les coordonnées des points utilisés.

Une caméra réelle projette un point 3D en un point de coordonnées (x_p, y_p), alors qu'une caméra de même focale placée au même endroit et suivant parfaitement le modèle sténopé projetterait le point en (x_{pu}, y_{pu}). La correction de la distorsion permet de passer de (x_p, y_p) à (x_{pu}, y_{pu}). La distorsion se manifeste selon une composante radiale et une composante tangentielle. Cette dernière étant beaucoup moins importante, nous avons considéré uniquement dans notre modèle la distorsion radiale. La modélisation de la distorsion radiale pour un objectif grand-angle tel que celui que nous utilisons nécessite d'exploiter un polynôme de degré 5. Les coefficients de distorsion radiale sont notés $(a_1, a_2, a_3, a_4, a_5)$. On commence par calculer la distance entre le point (x_p, y_p) et le point principal, l'unité

FIG. 3.3 – Une image prise avec un objectif fish-eye à gauche et l'image après correction de la distorsion à droite.

est choisie de façon à valoir 1 pour une distance en pixels égale à la focale :

$$U = \frac{x_p - u_0}{f_x} \tag{3.7}$$

$$V = \frac{y_p - v_0}{f_y} \tag{3.8}$$

$$r^2 = U^2 + V^2 \tag{3.9}$$

La distance au point principal après correction est donnée par :

$$d_r = a_5 r^{10} + a_4 r^8 + a_3 r^6 + a_2 r^4 + a_1 r^2 \tag{3.10}$$

Ce qui nous permet d'obtenir les coordonnées du point après correction. On notera que le point principal de l'image corrigée a été ramené au centre de l'image :

$$x_{pu} = U(1 + d_r) f_x + X_{p_{centre}} \tag{3.11}$$

$$y_{pu} = V(1 + d_r) f_x + Y_{p_{centre}} \tag{3.12}$$

Les coefficients de distorsion, ainsi que la focale et la position du point principal sont obtenus en calibrant la caméra. Le calibrage est effectué en prenant plusieurs images d'une mire et en cherchant les paramètres de la caméra qui minimisent l'erreur de reprojection de chaque point de la mire dans chaque image (ajustement de faisceaux). La solution la plus simple au niveau algorithmique consiste à utiliser une mire 3D dont les dimensions sont parfaitement connues. Mais cela impose d'avoir recours à des méthodes de fabrication ou de mesure coûteuses pour obtenir une mire précise. Il est également possible d'utiliser une mire dont les dimensions sont connues seulement grossièrement. Dans ce cas, les

paramètres à optimiser comprennent non seulement les paramètres de la caméra mais aussi les dimensions de la mire : c'est ce qu'on appelle l'autocalibrage. La méthode que nous avons employée est une méthode d'autocalibrage utilisant une mire plane selon le procédé donné par Lavest *et al.* [39]. La mire utilisée apparaît en figure 3.4. C'est une plaque métallique peinte en noir sur laquelle des pastilles rondes photoréfléchissantes ont été placées. Une douzaine d'images sont utilisées pour le calibrage. Elles sont obtenues en déplaçant la mire de façon à couvrir toute la surface de l'image. Les paramètres optimisés dans la minimisation sont les paramètres intrinsèques (f_x, f_y, u_0, v_0), les coefficients de distorsion radiale $(a_1, a_2, a_3, a_4, a_5)$, les paramètres extrinsèques pour chaque pose et les coordonnées des centres des pastilles de la mire.

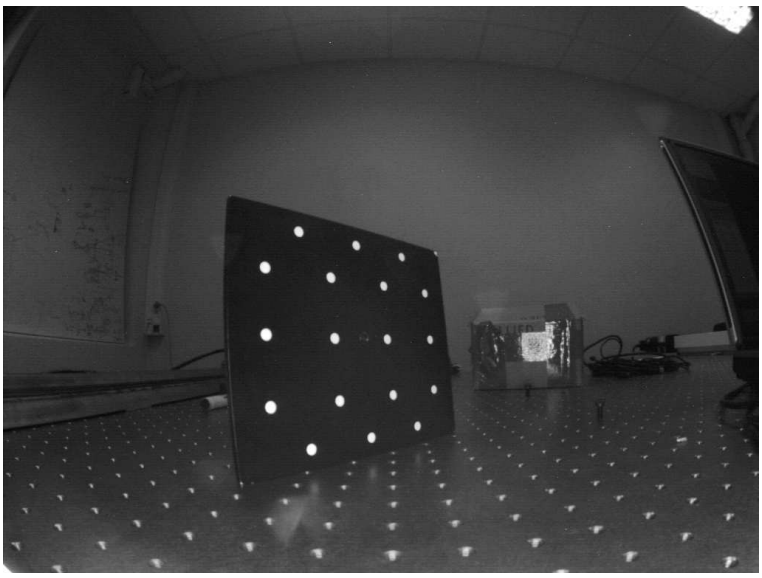

FIG. 3.4 – Mire utilisée pour le calibrage de la caméra

Géométrie épipolaire

La modélisation de la caméra permet d'écrire les équations qui régissent la formation d'une image. Dès lors qu'on dispose de plusieurs images d'une même scène, il devient intéressant d'établir des relations liant le déplacement de la caméra au déplacement de la projection des points de la scène dans les images. L'essentiel des concepts mis en œuvre et des méthodes de calcul est maintenant bien connu et figure dans les livres de Faugeras [16] ou de Hartley et Zisserman [28].

Les paragraphes suivants ont pour but de simplement rappeler les principales méthodes tout en introduisant les notations qui seront utilisées dans la suite.

Calcul robuste de la géométrie épipolaire Le problème qu'on cherche à résoudre est le suivant. On connaît la projection de n points de l'espace dans deux ou trois images. Pour deux points pris dans deux images, on dit que ce sont des points homologues s'ils sont l'image du même point de la scène. On suppose qu'on dispose d'un ensemble de points homologues. On cherche à déterminer la pose relative des caméras à partir de cette information. Une propriété essentielle permet de faire ce calcul : la géométrie épipolaire est indépendante de la structure de la scène. Plusieurs situations peuvent être considérées : les paramètres intrinsèques peuvent être connus ou non, on peut faire le calcul à partir de deux ou trois images. Dans le cas où les paramètres intrinsèques sont inconnus, on va calculer la matrice fondamentale (pour deux images) ou le tenseur trifocal (pour trois images). Si on dispose de deux images et d'une caméra calibrée, on peut calculer la matrice essentielle. Dans chaque cas, il existe un nombre minimum de points homologues à utiliser pour mener à bien le calcul. Ensuite, on peut utiliser le résultat obtenu pour calculer la pose relative de la caméra pour chaque image.

Dans la pratique, il faut tenir compte du fait que si l'appariement est fait de façon automatique, les points ne sont pas tous bien appariés. Il existe parmi les n appariements disponibles, une proportion non négligeable de faux appariements. Pour cette raison, le calcul de la géométrie épipolaire (par le biais de la matrice fondamentale ou essentielle, ou par le tenseur trifocal) doit être fait en utilisant une méthode robuste comme RANSAC proposé par Fischler et Bolles [18]. Cette technique, utilisée par exemple pour calculer la matrice essentielle, consiste à faire plusieurs fois le calcul en prenant à chaque fois aléatoirement un échantillon de taille minimale parmi l'ensemble des points appariés disponibles. Chaque fois qu'un échantillon est considéré, on obtient une solution et on peut vérifier pour chaque appariement s'il est cohérent avec la solution qui vient d'être calulée. Finalement, c'est la solution qui donne le plus grand nombre d'appariements cohérents qui est retenue.

D'autre part, la géométrie épipolaire n'est pas définie si la caméra tourne autour d'un axe passant par le centre optique. Dans ce cas, il n'est pas possible de faire une reconstruction 3D des points. Ce cas ne se produit pas dans notre application car la caméra est montée sur un véhicule de type voiture qui ne peut pas tourner sur place. Cependant, dans le cas de virages très serrés, on se rapproche d'une configuration dégénérée et le processus de reconstruction devient plus sensible aux erreurs d'appariement, de calibrage ou au bruit dans la position des points détectés.

Matrice essentielle La matrice essentielle E permet de décrire la géométrie épipolaire de deux images pour une caméra calibrée. Elle a été introduite par Longuet-Higgins [46]. La relation qui lie deux points homologues q et q' vus par les caméras C et C' est la suivante :

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix}^T E \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0 \tag{3.13}$$

où x_i, y_i et x'_i, y'_i sont les coordonnées dans le plan image (et non pas les coordonnées en pixels) comme indiqué dans le tableau 3.1.

Cette relation étant indépendante de la scène, il est possible de calculer E à partir de la connaissance des coordonnées des points dans les images seulement. Pour cela, et grâce à l'utilisation de contraintes sur les valeurs singulières de la matrice, il suffit de connaître 5 points homologues. Alors que la plupart des méthodes de calcul concernant la géométrie épipolaire sont bien connues depuis le début des années 1990, une méthode de calcul efficace pour déterminer la matrice essentielle n'a été publiée qu'en 2003 par Nistér [60] sous le nom d'algorithme des 5 points. Lorsqu'on connaît E, il est possible de calculer la pose de la caméra C' relativement à la caméra C. Pour cela, il est usuel de considérer que la pose des caméras C et C' sont données par les matrices :

$$\begin{bmatrix} I_3 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} R & -RT \\ 0 & 1 \end{bmatrix}$$

Dans ce cas, la matrice essentielle et la pose relative des caméras sont liées par la relation suivante :

$$E = [T]_\times R \tag{3.14}$$

où $[T]_\times$ désigne la matrice antisymétrique construite à partir de T. La reconstruction qu'on peut obtenir à partir du calcul de matrice essentielle est une reconstruction Euclidienne, ce qui signifie qu'elle est définie à une similitude près. Le choix de la position de l'origine, de l'orientation et de l'échelle sont définies de façon arbitraires. L'algorithme de reconstruction 3D développé dans cette thèse utilise le calcul de matrice essentielle tel qu'il est présenté ici. Le détail des calculs est donné dans la section 3.2.

Matrice fondamentale La matrice fondamentale F est définie par une relation analogue à celle donnée plus haut pour la matrice essentielle. Pour deux points homologues q et q', la relation est la suivante :

$$\begin{bmatrix} x'_{pu} \\ y'_{pu} \\ 1 \end{bmatrix}^T F \begin{bmatrix} x_{pu} \\ y_{pu} \\ 1 \end{bmatrix} = 0 \tag{3.15}$$

Cette fois ci, les coordonnées des points x et x' utilisées sont directement les coordonnées dans l'image exprimée en pixels (après correction de la distorsion s'il y a lieu). Pour écrire cette relation, les paramètres intrinsèques de la caméra ne sont pas utilisés. La relation entre matrice fondamentale et matrice essentielle est facile à écrire :

$$E = K'^T F K \tag{3.16}$$

où K et K' sont les matrices de paramètres intrinsèques des deux caméras.

Cette relation écrite pour 8 points homologues permet de calculer la matrice fondamentale (algorithme des 8 points [26]). En utilisant une contrainte sur le rang d'une matrice fondamentale, on peut aussi calculer F à partir de 7 points homologues seulement ce qui peut donner jusqu'à 3 solutions (algorithme des 7 points [84]). Lorsqu'on connaît F, on peut retrouver les matrices de caméra et reconstruire les points à une transformation projective près. En particulier cela signifie que la reconstruction obtenue ne respecte pas les angles, ni les rapports de distance.

Caméras et nuage de points reconstruits

Lorsqu'on dispose d'un ensemble de points homologues donnés dans une séquence d'images, la détermination de la localisation des caméras et la détermination de la position 3D des points observés sont deux problèmes intimement liés. En effet, dès que le positionnement relatif des caméras est connu il devient possible de calculer la position 3D des points par simple triangulation. De même, si l'ensemble des points 3D est connu, alors le calcul de la pose de chaque caméra est un problème simple. Chacun de ces problèmes peut être traité par de multiples méthodes et variantes. Les algorithmes utilisés dans la suite seront présentés plus en détail au paragraphe 3.2.

On a vu au paragraphe précédent que le calcul de la géométrie épipolaire menait d'abord à la pose des caméras, ce qui permettait par la suite de trianguler les points. Cependant, ce procédé ne conduit pas à la solution optimale qui peut être définie comme celle où l'erreur de reprojection moyenne de tous les points

dans toutes les images est minimale. Après avoir calculé une solution initiale, il est donc souvent utile de passer par un processus de minimisation itératif pour se rapprocher de la solution optimale. Cette minimisation est connue sous le nom d'ajustement de faisceaux. De nombreux détails à son propos sont donnés dans l'ouvrage de Triggs *et al.* [85]. Le plus souvent la méthode de minimisation utilisée est celle de Levenberg-Marquardt qui est une méthode du deuxième ordre. Dans le cas de l'ajustement de faisceaux, le nombre de variables à gérer est très important : de l'ordre de $6m+3n$ lorsqu'il y a m caméras et n points. En pratique, il est fréquent de travailler sur des séquences de plusieurs centaines d'images et des dizaines de milliers de points. Cela conduirait à des temps de calcul excessifs si la minimisation ne pouvait être gérée de façon astucieuse. Heureusement, tous les points ne sont pas visibles dans toutes les caméras, mais seulement dans une petite partie d'entre elles. Cela signifie que la matrice Jacobienne utilisée dans la minimisation est creuse. En exploitant sa structure, il est possible de réduire fortement le nombre d'opérations nécessaires et la place mémoire utilisée.

D'une séquence d'images à un modèle tridimensionnel

A partir de ces algorithmes de base, des systèmes complets de reconstruction 3D ont été mis au point et publiés. Certains de ces algorithmes sont capables de calculer un modèle tridimensionnel texturé de la scène à partir d'une séquence d'images et sans aucune autre information. Dans ce domaine, on peut par exemple citer les travaux de Nistér [58], de Fitzgibbon et Zisserman [19], de Pollefeys *et al.* [65], de Beardsley *et al.* [5] ou encore de Lhuillier et Quan [43]. En plus des calculs décrits dans les paragraphes précédents, ces travaux apportent des méthodes pour gérer la cohérence de la géométrie calculée entre paires ou triplets d'images tout au long de la séquence. D'autre part, la reconstruction 3D à partir d'une séquence d'images provenant d'une caméra vidéo a fait apparaître le problème de la sélection des images à utiliser. En effet, il n'est pas judicieux d'utiliser toutes les images de la vidéo dans la reconstruction. Cela augmenterait inutilement le temps de calcul et de plus, le calcul de la géométrie épipolaire est mal conditionné si les caméras sont trop proches. La sélection d'un ensemble d'images clef pour la reconstruction 3D a été étudiée par exemple par Nistér [59] et Torr *et al.* [83]. Enfin, lorsque le but est la construction d'un modèle surfacique, il faut développer une méthode de triangulation pour reconstruire des facettes à partir du nuage de points reconstruit.

3.2 Algorithme de cartographie 3D pour la navigation autonome

3.2.1 Aperçu de la méthode

Pour construire une cartographie 3D de l'environnement, nous disposons d'une séquence d'images de luminance ainsi que des paramètres intrinsèques de la caméra (y compris les coefficients de distorsion radiale). De plus, comme la caméra est montée sur un véhicule de type voiture, nous pouvons faire l'hypothèse que le mouvement entre deux images n'est jamais une rotation sur un axe passant par le centre optique de la caméra.

L'objectif est d'obtenir la pose de la caméra pour chaque image de la séquence ainsi qu'un nuage de points reconstruits. Le nuage de points devra être aussi riche que possible, car ce sont ces points qui serviront à localiser le robot.

L'algorithme mis au point pour cela peut être résumé de la façon suivante :

1. sélection d'un ensemble d'images clef parmi toutes les images de la séquence vidéo,
2. calcul des paramètres extrinsèques des caméras correspondantes aux images clef et recontruction du nuage de points associé,
3. localisation des images non clef.

L'étape 2 permet d'obtenir la cartographie nécessaire pour la localisation du robot en ligne. L'étape 3 permet de connaître précisément la trajectoire suivie par le robot entre les images clef. Dans le cas où on veut suivre exactement la trajectoire d'apprentissage, cela permet de donner au robot le chemin à suivre plus précisément que d'interpoler entre les positions des images clef.

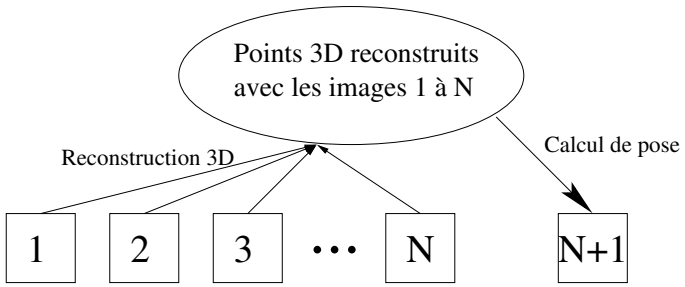

FIG. 3.5 – Calcul d'une solution initiale pour la pose de la caméra N+1

De toutes ces étapes, c'est l'étape 2 qui constitue le cœur de l'algorithme de reconstruction 3D. Le calcul du mouvement de la caméra et de la structure de la

scène est fait de manière incrémentale : c'est le nuage de points reconstruit entre les images clef 1 à N qui permet de calculer la pose de la caméra $N+1$. Les nouveaux points présents dans les images N et $N+1$ peuvent alors être reconstruits et enrichissent le nuage de points 3D ce qui permettra de calculer la pose de l'image suivante. Ceci est illustré sur la figure 3.5. Cette façon de faire a été choisie pour deux raisons. D'une part, la méthode utilisée pour le calcul de pose n'est pas perturbée par des configurations dégénérées de la scène (une scène où tous les points sont soit sur un même plan soit sur une quadrique réglée ne permet pas de faire un calcul de matrice fondamentale). D'autre part, il n'est pas nécessaire de disposer à l'avance de toutes les images, ce qui a permis à Mouragnon *et al.* [55] de généraliser la méthode à un processus de reconstruction 3D temps réel. Cette façon de faire est utilisée pour générer une solution initiale de la structure de la scène. Cette solution initiale est optimisée grâce à un ajustement de faisceaux. L'ajustement est géré de manière hiérarchique : une longue séquence est découpée en deux parties (avec une petite partie commune). Chaque sous-séquence est optimisée indépendamment, puis les sous-séquences sont fusionnées puis le résultat de la fusion est optimisé.

La manière dont s'articulent le calcul de la solution initiale de façon incrémentale et l'optimisation hiérarchique n'est pas facile à saisir au premier abord. La figure 3.6 permet de mieux comprendre le fonctionnement de l'algorithme. On considère sur le schéma le cas de la reconstruction d'une séquence composée de 6 images clef (chaque image est désignée par son numéro). On effectue les étapes suivantes :

- A l'étape 1, on divise la séquence en deux sous-séquences avec 2 caméras communes dans chaque sous-séquence.
- A l'étape 2, on divise la première sous-séquence en deux parties. On continue à diviser jusqu'à obtenir des sous-séquences composées de trois images.
- A l'étape 3, on calcule une valeur initiale de la pose des caméras et la position 3D des points associés pour le premier triplet d'images, puis on utilise un ajustement de faisceaux sur le triplet (123).
- A l'étape 4, on fait un calcul de pose incrémental pour la caméra 4 en utilisant le nuage de points 3D reconstruits à partir des caméras 1,2,3. Puis on fait un ajustement de faisceaux sur le triplet (234).
- A l'étape 5, on fusionne les sous-séquences (123) et (234) pour obtenir la séquence (1234). Puis on fait un ajustement de faisceaux sur le résultat de la fusion.
- A l'étape 6, on commence le traitement de la sous-séquence (3456). On la divise en deux parties. Pour le triplet (345), on calcule la pose de la caméra 5 grâce au nuage de points reconstruit à partir des caméras (1234). Puis on

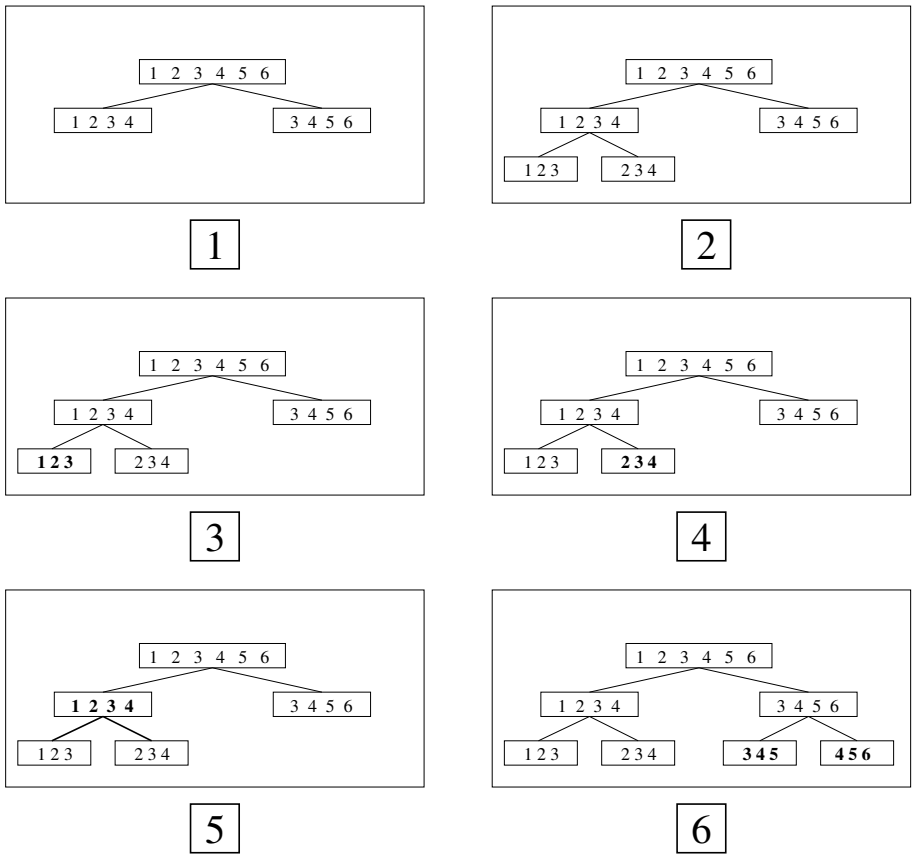

FIG. 3.6 – Utilisation d'un algorithme de calcul de pose incrémental associé à un ajustement de faisceaux hiérarchique

fait un ajustement de faisceaux sur le triplet (345).

– Le processus se poursuit ainsi jusqu'à ce qu'on ait reconstruit la séquence complète. L'algorithme se termine par un ajustement de faisceaux global sur toute la séquence.

Nous allons maintenant voir plus en détail la méthode utilisée dans chaque partie de l'algorithme. La mise en correspondance de points entre images, qui est à la base de tous les calculs qui suivent, a été décrite au chapitre 2.

3.2.2 Sélection d'images clef

Sélectionner des images clef plutôt que d'utiliser toutes les images de la séquence est nécessaire pour deux raisons. D'abord, traiter toutes les images serait beaucoup trop coûteux en temps de calcul. Ensuite, si le mouvement de la caméra

entre deux images clef est trop faible, alors le calcul de la géométrie épipolaire est mal conditionné. Il faut donc faire en sorte que le déplacement de la caméra entre deux images clef soit le plus grand possible tout en étant toujours capable de faire une mise en correspondance entre les images. La méthode utilisée est une méthode heuristique. Nous n'avons pas cherché à avoir une sélection optimale des images clef (la notion d'optimalité serait d'ailleurs à définir). Contrairement à l'algorithme proposé par Nistér [59], la méthode présentée ici ne nécessite pas de connaître à l'avance toute la vidéo.

La première image de la séquence est toujours choisie comme image clef, elle est notée I_1. La deuxième image clef I_2 est choisie la plus éloignée possible de I_1 dans le flux vidéo en respectant la contrainte qu'il y ait au moins M points d'intérêt communs entre I_1 et I_2. Une fois que les images clef I_1 à I_n sont choisies ($n > 1$), on choisit I_{n+1} pour qu'il y ait au moins M points d'intérêt communs entre I_{n+1} et I_n et au moins N points d'intérêt communs entre I_{n+1} et I_{n-1}. Cela nous assure qu'il y a suffisamment de points en correspondance entre trois images clef succesives pour calculer le mouvement de la caméra. Dans nos expériences nous détectons environ 1500 points d'intérêt par image et les seuils sont fixés à $M = 400$ et $N = 300$.

3.2.3 Calcul de la géométrie épipolaire pour le premier triplet d'images

Pour le premier triplet d'images, il n'est pas possible de calculer la pose des caméras à partir d'un nuage de points reconstruits puisque ces derniers n'existent pas encore. Il faut donc une méthode pour obtenir la géométrie épipolaire à partir des coordonnées dans les images d'un ensemble de points homologues. Comme la caméra est calibrée, la solution la plus appropriée est d'utiliser un calcul de matrice essentielle comme proposé par Nistér [60] et amélioré par la suite dans [61]. On calcule une matrice essentielle entre la première et la dernière image du triplet, la pose de la caméra du milieu étant obtenue ensuite à l'aide d'un calcul de pose. Le calcul de matrice essentielle est effectué grâce à l'algorithme des 5 points que nous allons décrire ici. On peut noter qu'il existe de nombreuses autres méthodes utilisables pour initialiser la géométrie des trois premières caméras. On aurait aussi bien pu calculer la matrice fondamentale entre les deux premières vues ou le tenseur trifocal entre les trois premières images, puis utiliser les paramètres intrinsèques connus et reconstruire les points à partir de là.

Algorithme des 5 points pour deux vues

On suppose connus 5 points homologues $(q^i, q'^i), i = 1..5$. Chaque point q^i est la projection d'un point Q^i de l'espace dans l'image 1. Sa projection dans l'image 2 est notée q'^i. On note de la même façon le point q et le vecteur 3×1 de ses coordonnées homogènes dans l'image : $q^i = [q_1, q_2, q_3]^T$. On cherche les matrices essentielles E vérifiant :

$$q'^{i^T} E q^i = 0, \forall i \in \{1..5\} \tag{3.17}$$

Les équations 3.17 sont la traduction de la contrainte épipolaire. Cela nous donne cinq équations linéaires en E. On peut écrire cela plus simplement en notant les coefficients de E en colonne : $\widetilde{E} = [E_{11}\ E_{12}\ E_{13}\ E_{21}\ E_{22}\ E_{23}\ E_{31}\ E_{32}\ E_{33}]^T$ et $\widetilde{q} = [q_1q'_1\ q_2q'_1\ q_3q'_1\ q_1q'_2\ q_2q'_2\ q_3q'_2\ q_1q'_3\ q_2q'_3\ q_3q'_3]^T$ pour chacun des cinq points homologues. On a alors les cinq relations :

$$\widetilde{q}^T \widetilde{E} = 0 \tag{3.18}$$

Cela signifie que $\widetilde{E}$ est un vecteur du noyau d'une matrice 5×9 notée N obtenue à partir des coordonnées des cinq points dans les deux images. En dehors de certains cas particuliers, le noyau de cette matrice est de dimension 4. Ainsi on peut écrire :

$$\widetilde{E} = x\widetilde{X} + y\widetilde{Y} + z\widetilde{Z} + \widetilde{W} \tag{3.19}$$

où $(\widetilde{X}, \widetilde{Y}, \widetilde{Z}, \widetilde{W})$ forme une base du noyau de N et (x, y, z) trois coefficients à déterminer (la matrice E étant définie à un coefficient multiplicatif près, le coefficient de $\widetilde{W}$ a été fixé à 1). A ce stade, on peut noter qu'il est possible d'utiliser plus de cinq points pour déterminer $(\widetilde{X}, \widetilde{Y}, \widetilde{Z}, \widetilde{W})$, il suffit de conserver les vecteurs associés aux quatre plus petites valeurs singulières de N.

On obtient une sixième relation en exprimant le fait que E est une matrice essentielle. En effet toute matrice essentielle est de rang 2 et ses deux valeurs singulières non nulles sont égales [16]. E doit donc vérifier l'équation :

$$EE^TE - \frac{1}{2} trace(EE^T)E = 0 \tag{3.20}$$

En injectant l'équation 3.19 dans 3.20, on obtient un système non linéaire de 9 équations à 3 inconnues noté (S). Chacune de ces neuf équations est une équation polynômiale de degré 3 en x, y et z. La première étape consiste à calculer z. Une fois z trouvé, le calcul de x et y est assez simple.

A partir du système de neuf équations évoqué plus haut, on va chercher une équation ne faisant intervenir que z. L'élimination de x et y commence par l'utilisation de l'algorithme de Gauss-Jordan avec pivot partiel pour éliminer les termes

de plus haut degré. On obtient le système suivant (le symbole $\times$ et les lettres L à S représentent des coefficients réels quelconques) :

A	x^3	y^3	x^2y	xy^2	x^2z	y^2z	x^2	y^2	xyz	xy	xz^2	xz	x	yz^2	yz	y	z^3	z^2	z	1
(a)	1	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
(b)	0	1	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
(c)	0	0	1	0	0	0	0	0	0	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
(d)	0	0	0	1	0	0	0	0	0	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
(e)	0	0	0	0	1	0	0	0	0	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
(f)	0	0	0	0	0	1	0	0	0	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
(g)	0	0	0	0	0	0	1	0	0	L	$\times$	$\times$	$\times$	M	N	O	$\times$	$\times$	$\times$	$\times$
(h)	0	0	0	0	0	0	0	1	0	P	Q	R	S	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
(i)	0	0	0	0	0	0	0	0	1	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

On continue l'élimination des termes de plus haut degré en x et en y en posant les nouvelles équations :

$$\begin{array}{ll} (j) \equiv (e) - z(g) & \text{(on elimine } x^2z\text{)} \\ (k) \equiv (f) - z(h) & \text{(on elimine } y^2z\text{)} \\ (l) \equiv (d) - x(h) + P(c) + zQ(e) + R(e) + S(g) & \text{(on elimine } xy^2\text{)} \\ (m) \equiv (c) - y(g) + L(d) + zM(f) + N(f) + O(h) & \text{(on elimine } x^2y\text{)} \end{array} \tag{3.21}$$

On obtient alors quatre équations qui ne contiennent aucun terme en x^3, y^3, x^2 ni y^2. En ajoutant l'équation (i) et en notant $[\alpha]$ un polynôme de degré α en z, on obtient :

$$\begin{array}{ll} (i) & xy[1] + x[2] + y[2] + [3] = 0 \\ (j) & xy[1] + x[3] + y[3] + [4] = 0 \\ (k) & xy[1] + x[3] + y[3] + [4] = 0 \\ (l) & xy[2] + x[3] + y[3] + [4] = 0 \\ (m) & xy[2] + x[3] + y[3] + [4] = 0 \end{array} \tag{3.22}$$

Ce système peut s'écrire comme

$$\left\{ \begin{array}{ll} B(z) \cdot X = 0 & (1) \\ C(z) \cdot X = 0 & (2) \end{array} \right. \text{ avec } X = [xy\ x\ y\ 1]^T \tag{3.23}$$

Si le triplet (x, y, z) est une solution du système (S) alors il existe z tel que (1) et (2) ont une solution en X, c'est-à-dire que z est solution de :

$$\left\{ \begin{array}{l} \det B(z) = 0 \\ \det C(z) = 0 \end{array} \right. \tag{3.24}$$

Chaque déterminant est un polynôme de degré 11 en z.

$$\begin{array}{l} \det B(z) = B_{11}z^{11} + B_{10}z^{10} + \cdots + B_1 z + B_0 \\ \det C(z) = C_{11}z^{11} + C_{10}z^{10} + \cdots + C_1 z + C_0 \end{array} \tag{3.25}$$

On peut éliminer le terme de plus haut degré entre les deux équations. Il nous reste alors à chercher les racines réelles d'un polynôme de degré 10 en z.

$$P_{10}(z) = B_{11}\det C(z) - C_{11}\det B(z) \tag{3.26}$$

Le calcul effectif des coefficients de P_{10} est assez fastidieux. Il a été effectué grâce au logiciel de calcul formel Mupad [1].

La méthode utilisée ici pour calculer les racines de ce polynôme est celle de la chaîne de Sturm [79]. Une fois celles-ci calculées, on obtient au plus 10 solutions pour z. Pour chacune de ces solutions on calcule alors x et y en résolvant le système 3.27 comme si xy,x,y étaient trois inconnues indépendantes.

$$\begin{cases} (i)\ \ xy[1]+x[2]+y[2]+[3]=0 \\ (j)\ \ xy[1]+x[3]+y[3]+[4]=0 \\ (k)\ \ xy[1]+x[3]+y[3]+[4]=0 \end{cases} \tag{3.27}$$

Lorsque x,y,z sont connus, on obtient E à partir de l'équation 3.19. On a donc au plus 10 solutions pour E.

A partir de E, il est possible de calculer le déplacement relatif de la caméra entre les deux images. On note ce déplacement (R,T), il est défini par :

$$P' = P\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \tag{3.28}$$

où P et P' sont les matrices de projection de la première et de la deuxième caméra. La solution pour (R,T) n'est pas unique. Tout d'abord la reconstruction (et donc le calcul de T) n'est possible qu'à un facteur d'échelle près. Ensuite, le calcul fournit quatre solutions. On choisit celle qui permet de reconstruire les cinq points devant les deux caméras. Le calcul de R et T est fait à l'aide d'une décomposition en valeurs singulières [16]. La SVD de E est :

$$E = U\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}V^T \tag{3.29}$$

[1] http ://www.sciface.com/

avec U et V tels que $\det U > 0$ et $\det V > 0$. On note :

$$T_0 = [u_{13}\ u_{23}\ u_{33}]^T \tag{3.30}$$

$$R_a = UDV^T \tag{3.31}$$

$$R_b = UD^TV^T \tag{3.32}$$

$$D = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.33}$$

$$\tag{3.34}$$

On obtient 4 solutions pour (R,T) : (R_a,T_0), $(R_a,-T_0)$, (R_b,T_0), $(R_b,-T_0)$.

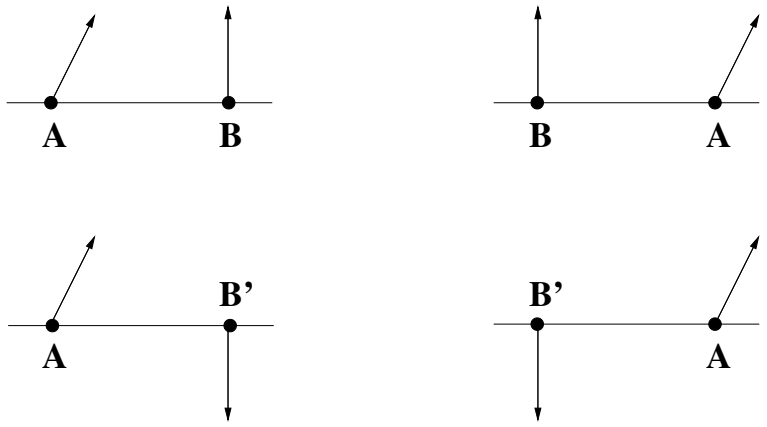

FIG. 3.7 – Les quatre solutions pour la position des caméras

Parmi ces solutions, une seule permet de reconstruire les cinq points devant les deux caméras simultanément. Les quatre solutions obtenues sont représentées sur la figure 3.7. La première caméra A étant donnée, les 4 solutions possibles pour la caméra B sont représentées. Le passage de R_a à R_b correspond à un demi-tour de l'une des caméras autour de la droite qui joint les centres des deux caméras : c'est ce qui se produit lorsqu'on passe de la ligne supérieure à la ligne inférieure de la figure. En fait pour choisir la bonne solution, il suffit de reconstruire un seul point en faisant l'hypothèse que la solution (R_a,T_0) est la bonne. Si le point reconstruit est devant les deux caméras, on a la bonne solution. Si le point est derrière les deux caméras, alors la bonne solution est $(R_a,-T_0)$. Si le point reconstruit est devant une seule des deux caméras, alors on fait faire un demi-tour à une des caméras autour de la droite qui joint le centre des deux caméras. Cela revient à passer de R_a à R_b. Ensuite, on choisit entre (R_b,T_0) et $(R_b,-T_0)$ de la même façon que précédemment.

Algorithme des 5 points pour trois vues

On pourrait utiliser directement l'algorithme des 5 points sur deux vues avec RANSAC. Mais il est préférable d'utiliser trois vues. En effet, à partir de deux images seulement, la solution n'est pas unique dans le cas où la scène est plane et où tous les points sont plus proches d'une des deux caméras. Pour éviter d'être confrontés à cette ambiguïté, il est préférable d'utiliser l'algorithme des 5 points pour calculer la géométrie de trois vues. Cela est fait de la manière suivante, illustrée sur la figure 3.8.

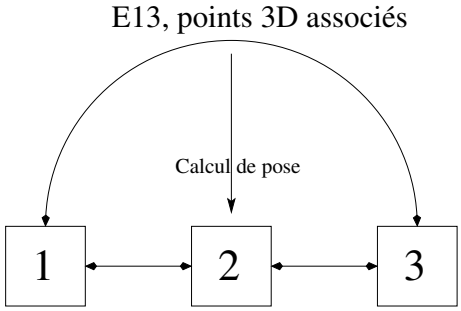

FIG. 3.8 – Algorithme des 5 points sur 3 vues

L'algorithme est basé sur l'utilisation de RANSAC. A partir d'un échantillon de 5 points en correspondance dans les images 1,2 et 3, on commence par calculer la matrice essentielle entre les caméras 1 et 3 avec l'algorithme des 5 points. On en déduit le déplacement de la caméra entre ces deux vues et on triangule les 5 points pour obtenir leur position 3D. Ces points 3D sont ensuite utilisés pour calculer la pose de la caméra 2. Finalement chaque tirage aléatoire de 5 points nous a donné une hypothèse pour la pose des trois caméras. Pour chaque hypothèse, on reconstruit l'ensemble des points appariés et on compte ceux qui sont compatibles avec la géométrie qui vient d'être calculée. L'hypothèse pour laquelle on obtient le plus grand nombre de points cohérents est retenue. La méthode utilisée pour calculer la pose de la caméra 2 est la méthode de Grunert décrite dans l'article de Haralick *et al.* [24]. Cette méthode sera décrite plus en détail au chapitre 4.

Une méthode rapide est utilisée pour trianguler les points. Elle est couramment appelée méthode du point milieu [27]. La méthode est utilisée ici pour reconstruire les points appariés entre les images 1 et 3. Elle sera également utilisée dans toute la suite. La figure 3.9 permet de préciser les notations : on note O_1 et O_2 les centres optiques des deux caméras et q_1 et q_2 la position des points dans chaque image. L'idée de la méthode est de chercher la perpendiculaire commune aux droites (O_1q_1) et (O_2q_2). La perpendiculaire commune est dirigée par $\vec{v} = \overrightarrow{O_1q_1} \wedge$

$\overrightarrow{O_2q_2}$ (le symbole $\wedge$ désigne le produit vectoriel). Le point Q_1 est défini comme l'intersection de la droite (O_1q_1) et du plan Π_2. Le plan Π_2 est le plan passant par O_2 et dirigé par les vecteurs $\overrightarrow{O_2q_2}$ et $\vec{v}$. Le calcul donne :

$$Q_1 = O_1 + \alpha\overrightarrow{O_1q_1} \tag{3.35}$$

avec

$$\alpha = \frac{\overrightarrow{O_1O_2}\cdot\overrightarrow{O_2q_2}\ \overrightarrow{O_1q_1}\cdot\overrightarrow{O_2q_2} - \overrightarrow{O_1O_2}\cdot\overrightarrow{O_1q_1}\ \|\overrightarrow{O_2q_2}\|^2}{\left(\overrightarrow{O_1q_1}\cdot\overrightarrow{O_2q_2}\right)^2 - \|\overrightarrow{O_1q_1}\|\|\overrightarrow{O_2q_2}\|} \tag{3.36}$$

Pour répartir l'erreur de reprojection entre les deux images, on définit de la même façon le point (Q_2) comme l'intersection de la droite (O_2q_2) et du plan Π_1 qui est le plan contenant O_1 et qui est dirigé par les vecteurs $\overrightarrow{O_1q_1}$ et $\vec{v}$. Au final, le point triangulé est le point Q, milieu du segment $[Q_1Q_2]$.

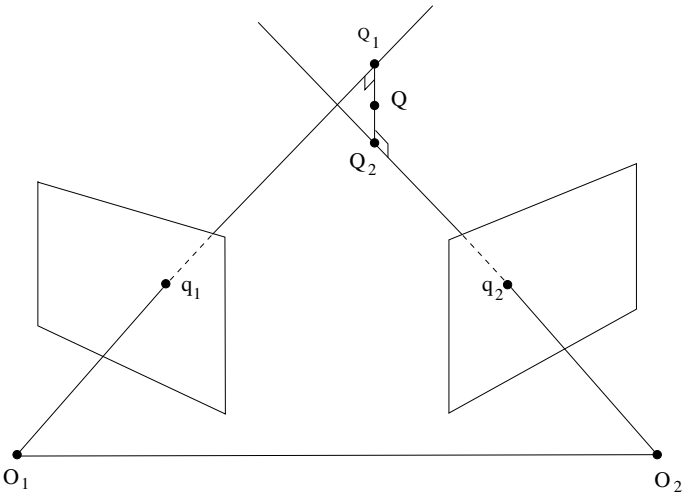

FIG. 3.9 – Triangulation de points vus dans deux images

3.2.4 Reconstruction 3D par calcul de pose incrémental

Le calcul de la matrice essentielle pour le triplet initial permet de reconstruire un nuage de points 3D qui sert à initialiser un processus de reconstruction incrémental. En effet, comme on l'a vu au paragraphe 3.1.2, la connaissance des points 3D permet de localiser une caméra, et de nouveaux points vus dans une caméra pourront être reconstruits pour enrichir le nuage de points 3D.

Supposons qu'on connaisse la pose des caméras C_1 à C_N ($N \geq 3$). On cherche à déterminer la pose de la caméra C_{N+1}. Pour cela, on commence par détecter des points d'intérêt dans l'image $N+1$ et on les apparie avec les points qui ont été détectés dans l'image N. On obtient une liste d'appariements (q_N^i, q_{N+1}^i) (q_j^i désigne le point d'intérêt numéro i dans l'image j) de deux types différents. Dans la première catégorie on trouve les appariements pour lesquels q_N^i est apparié à un point q_{N-1}^i dans l'image $N-1$. Dans la seconde, ce sont les points qui n'étaient pas visibles dans l'image $N-1$. Les points de la première catégorie ont déjà été triangulés : leur position 3D est connue. Avec l'ensemble de ces points on calcule la pose de la caméra C_{N+1}. A ce stade, on pourrait trianguler les points de la seconde catégorie, mais pour obtenir un plus grand nombre de points, on recommence une étape d'appariement entre l'image $N+1$ et l'image N. Cette fois ci, la zone de recherche des points dans les images est simplement une bande de quelques pixels de large le long de la ligne épipolaire. Compte tenu de la distorsion introduite par l'objectif, la ligne épipolaire n'est pas une droite, mais la bande choisie est suffisament large pour que la ligne épipolaire reste à l'intérieur de la bande. Cette deuxième phase d'appariements tient compte de la géométrie épipolaire qui a été calculée et elle produit environ 20 % d'appariements corrects supplémentaires. Le calcul de la pose de la caméra C_{N+1} est répété avec ces nouveaux appariements. Ensuite, les points de la deuxième catégorie sont triangulés en utilisant les vues N et $N+1$. Ainsi la reconstruction partielle a été enrichie avec une caméra supplémentaire et de nouveaux points 3D reconstruits. Le processus se répète jusqu'à la fin de la séquence.

3.2.5 Ajustement de faisceaux

Le calcul du mouvement de la caméra décrit précédemment ne donne pas une solution optimale. De plus, le calcul de la pose de la caméra C_{N+1} dépend des résultats obtenus pour les caméras précédentes, et les erreurs de calcul peuvent s'accumuler tout au long de la séquence. Pour limiter grandement ce problème, on utilise un ajustement de faisceaux pour affiner la solution obtenue. L'ajustement de faisceaux est un processus de minimisation basé sur l'algorithme de Levenberg-Marquardt. La fonction de coût est $f(C_E^1, \cdots, C_E^N, Q^1, \cdots, Q^M)$ où C_E^i désigne les paramètres extrinsèques de la caméra i, et Q^j désigne les coordonnées 3D du point j. La fonction de coût est la somme des carrés des erreurs de reprojection de tous les points considérés comme corrects dans toutes les images :

$$f(C_E^1, \cdots, C_E^N, Q^1, \cdots, Q^M) = \sum_{1 \leq i \leq N} \sum_{1 \leq j \leq M, j \in A_i} \left\| q_i^j - \pi(P_i Q^j) \right\|^2 \qquad (3.37)$$

où $\left\| q_i^j - \pi(P_i Q)^j \right\|^2$ est le carré de la distance euclidienne entre $\pi(P_i Q^j)$ projection du point Q^j par la caméra i, et q_i^j est le point d'intérêt correspondant ; P_i est la matrice de projection 3×4 construite d'après les valeurs contenues dans C_E^i et les paramètres intrinsèques connus de la caméra ; A_i est l'ensemble des points considérés comme corrects dans l'image i.

Pour rendre l'algorithme de minimisation robuste aux faux appariements, on maintient une liste des reprojections correctes dans chaque image : c'est l'ensemble A_i dans l'équation 3.37 ci-dessus. Au début de la minimisation, on considère comme correctes les reprojections pour lesquelles l'erreur de reprojection est inférieure à un seuil (3 pixels). On fait une série d'itérations à A_i constant, puis on recalcule A_i et on recommence la minimisation pour une nouvelle série d'itérations. On répète ce processus jusqu'à ce que le nombre de reprojections correctes se stabilise.

L'algorithme utilisé pour la minimisation est celui de Levenberg-Marquardt [66] qui combine les avantages d'une méthode du second ordre (Newton-Raphson qui offre une convergence rapide) et d'une méthode du premier ordre (descente de gradient qui converge même si la solution initiale est éloignée du minimum). On note ε le vecteur composé des erreurs de reprojection de tous les points dans toutes les images :

$$\varepsilon = \begin{pmatrix} \cdots \\ q_i^j - \pi(P_i Q^j) \\ \cdots \end{pmatrix} \tag{3.38}$$

et U le vecteur de paramètres formé avec les paramètres extrinsèques des caméras et les coordonnées 3D des points reconstruits :

$$U = \begin{pmatrix} \cdots \\ C_E^i \\ \cdots \\ \cdots \\ Q^j \\ \cdots \end{pmatrix} \tag{3.39}$$

La fonction f associe à un vecteur de paramètres U, un vecteur de mesure des erreurs de reprojection :

$$\varepsilon = f(U) \tag{3.40}$$

Une approximation de f au premier ordre donne :

$$f(U + \delta U) \approx f(U) + J\delta U \tag{3.41}$$

où J est la matrice Jacobienne de f. Afin de minimiser $\|\varepsilon\|^2$ avec la méthode de Newton-Raphson, à chaque itération on cherche δU tel que $f(U+\delta U)=0$, ce qui revient à chercher δU vérifiant les équations normales :

$$J^T J\delta U = -J^T\varepsilon \tag{3.42}$$

Cette façon de faire offre une convergence rapide si la solution initiale donnée pour U est suffisament proche du minimum. L'inconvénient est que la convergence n'est pas garantie. Dans l'algorithme de Levenberg-Marquardt, les équations normales sont remplacées par les équations normales augmentées. On remplace $J^T J = \left(\alpha_{ij}\right)$ de l'équation 3.42 par la matrice $L = \left(\alpha'_{ij}\right)$ définie de la façon suivante :

$$\begin{cases} \alpha'_{ii} = \alpha_{ii}(1+\lambda) \\ \alpha'_{ij} = \alpha_{ij} \qquad (i \neq j) \end{cases} \tag{3.43}$$

On cherche donc δU vérifiant :

$$L\delta U = -J^T\varepsilon \tag{3.44}$$

Cette modification des termes diagonaux permet de se rapprocher du fonctionnement d'une descente de gradient lorsque λ est grand. La valeur de λ est mise à jour à chaque itération. Elle diminue lorsqu'on se rapproche du minimum et elle augmente si la résolution de 3.44 tend à s'en éloigner.

La fonction de coût donnée au dessus est une fonction de $6N+3M$ variables, où N est le nombre d'images et M le nombre de points. Le calcul fait intervenir l'ensemble des reprojections des points dans toutes les images. Lorsque M et N sont grands, une implémentation directe conduirait à des temps de calcul extrèmement longs pour obtenir $J^T J$ et résoudre les équations normales. Heureusement, la plupart des points ne sont pas visibles dans toutes les images. Ceci implique que la matrice J est une matrice creuse. En utilisant cette particularité, les temps de calcul deviennent raisonnables.

Dans la phase de localisation en ligne, il est important de pouvoir disposer d'une mesure de l'incertitude associée à la pose courante du robot. Pour cela, il faut au préalable avoir calculé l'incertitude associée aux données qui constituent la carte. L'incertitude associée à la reconstruction 3D peut être caclulée à partir de la pseudo-inverse de $J^T J$ à l'étape finale de l'ajustement de faisceaux. L'incertitude obtenue peut alors être propagée au calcul de pose dans l'étape de localisation. Cependant, pour pouvoir calculer l'incertitude associée à la localisation en temps réel, certaines hypothèses simplificatrices ont du être faites. Cela conditionne la manière dont l'incertitude sur les points 3D reconstruits est calculée. Pour cette raison, l'ensemble des calculs d'incertitude est regroupé au paragraphe 4.5.

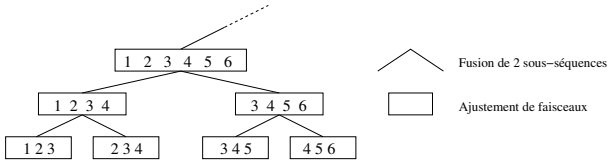

FIG. 3.10 – Pyramide d'ajustement de faisceaux hiérarchique

3.2.6 Fusion de deux sous-séquences

L'utilisation de l'ajustement de faisceaux hiérarchique est illustré sur la figure 3.10. Cette façon de traiter la reconstruction de la séquence impose de fusionner des sous-séquences. Pour cela, la division d'une séquence en deux sous-séquences est faite de manière à avoir toujours deux images en commun entre les deux sous-séquences.

Pour fusionner deux séquences S^1 et S^2, on utilise les deux dernières caméras S^1_{N-1} et S^1_N de S^1 et les deux premières caméras S^2_1 et S^2_2 de S^2. Comme les images sont les mêmes, les caméras associées après la fusion doivent être les mêmes. Pour cela, on applique une rotation et une translation à S^2 pour que S^1_N et S^2_2 aient la même position et orientation. Ensuite un facteur d'échelle est appliqué pour que $d(S^1_{N-1}, S^1_N) = d(S^2_1, S^2_2)$, où $d(S^i_n, S^j_m)$ est la distance euclidienne entre les centres optiques des caméras associées à S^i_n et S^j_m. Cela n'assure pas que S^1_{N-1} et S^2_1 sont les mêmes, mais un ajustement de faisceaux est utilisé sur le résultat de la fusion pour réduire l'imprécision de la reconstruction. On fusionne ainsi jusqu'à ce que la séquence complète soit reconstruite. D'autres méthodes de fusion sont possibles, l'essentiel étant d'appliquer un ajustement de faisceaux sur le résultat de la fusion.

3.2.7 Passage à un repère lié au sol

Le processus de reconstruction 3D utilise seulement des images. La reconstruction est donc faite à un facteur d'échelle près. De plus l'origine et l'orientation du repère sont choisies de façon arbitraire (dans notre cas, la caméra du milieu de la séquence est à l'origine du repère). Cela est gênant lorsqu'on veut utiliser la cartographie construite pour commander un robot. En effet, la commande du robot est faite dans un plan horizontal et l'échelle est importante. Il faut donc, une fois la reconstruction 3D effectuée, replacer le modèle obtenu dans un repère lié au sol et avec le mètre comme unité de longueur.

Le facteur d'échelle peut être déterminé très simplement en utilisant l'odo-

mètre du robot (le cycab décrit au paragraphe 5.1.3). Nous calculons simplement la longueur de la trajectoire à partir de l'enregistrement des vitesses de rotation des deux roues arrières tout au long de la trajectoire. La caméra a été placée sur le toit du cycab de façon que l'axe optique de la caméra soit horizontal. On peut se replacer aisément dans un repère lié au sol (avec un axe vertical) avec des dimensions en mètres. C'est suffisant pour la navigation autonome du robot. On peut noter que la précision de la longueur du trajet n'est pas critique. Il nous est arrivé lors d'une panne d'odomètre de mesurer la longueur de la trajectoire en marchant à coté du cycab et en comptant les pas. Cette mesure très grossière est suffisante pour avoir un comportement acceptable du robot lors de la phase de navigation.

Il est également possible d'utiliser un enregistrement du parcours par le GPS différentiel centimétrique mais cela n'est pas nécessaire pour faire naviguer le robot. Ce recalage est tout de même très utile pour pouvoir comparer les résultats de localisation par vision à la vérité terrain donnée par le GPS. La procédure de recalage utilisée dans ce cas sera développée au chapitre 5.

3.2.8 Exemples de reconstructions

Avant de passer à une analyse détaillée des performances de l'algorithme de reconstruction au chapitre 5, ce paragraphe donne quelques illustrations des cartes obtenues avec la procédure de reconstruction qui vient d'être décrite. Les figures 3.12, 3.13, 3.14 montrent trois exemples de reconstructions 3D. Pour chaque exemple, on a représenté la reconstruction 3D en vue de dessus. Les carrés noirs représentent la position du centre optique des caméras qui correspondent aux images clef. Les points reconstruits sont également représentés (sous forme de points). En comparant avec des images extraites de la séquence vidéo d'apprentissage au dessous, il est souvent possible d'identifier certains éléments du décor. Par exemple, sur la figure 3.13, on voit clairement apparaître les façades des immeubles qui délimitent la rue. Sur la figure 3.12, on peut également reconnaître des façades ainsi que le bord de la route. Sur la figure 3.14, les arbres sont visibles comme des concentrations de points de forme circulaire.

On peut également noter que les images clef sont d'autant plus proches que la courbure de la trajectoire est grande. Ceci est compréhensible car les points suivis sortent plus vite du champ de la caméra si la vitesse angulaire est grande. Pour la reconstruction Campus de la figure 3.12, on a établi un histogramme montrant le nombre d'images clef sur lesquelles les points sont suivis. Cet histogramme est représenté en figure 3.11. On constate que peu de points sont suivis sur plus de quatre images clef. Les histogrammes calculés pour d'autres reconstructions sont approximativement les mêmes. Le tableau 3.2 donne pour chacune de ces

Séquence	Images clef	Points reconstruits	Longueur approximative	Taille d'image
Campus	244	25042	150 m	512×384
Antibes	250	24188	500 m	640×480
Polydome	195	26459	300 m	640×480

TAB. 3.2 – Données numériques pour quelques exemples de reconstruction

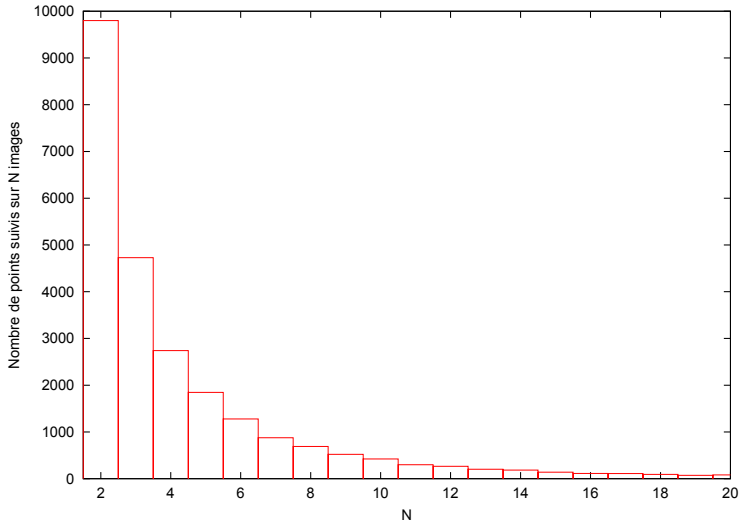

FIG. 3.11 – Pour une valeur de N, nombre de points suivis sur N images clef.

séquences le nombre d'images clef et le nombre de points 3D reconstruits ainsi que la longueur approximative de la trajectoire.

Pour la navigation autonome du robot, nous avons utilisé une caméra pointée dans la direction d'avancement du véhicule. Pour d'autres applications, il peut être intéressant de placer la caméra différemment. Par exemple, pour étudier la faisabilité d'un guidage automatique d'une voiture lors d'un créneau, nous avons utilisé le même processus de reconstruction avec une caméra dirigée sur le côté du véhicule (l'axe optique faisait un angle de 30° avec l'axe d'avancement du véhicule). La reconstruction faite durant la manœuvre de parking permet de bien identifier les véhicules garés et l'emplacement disponible, comme on peut le voir sur la figure 3.15.

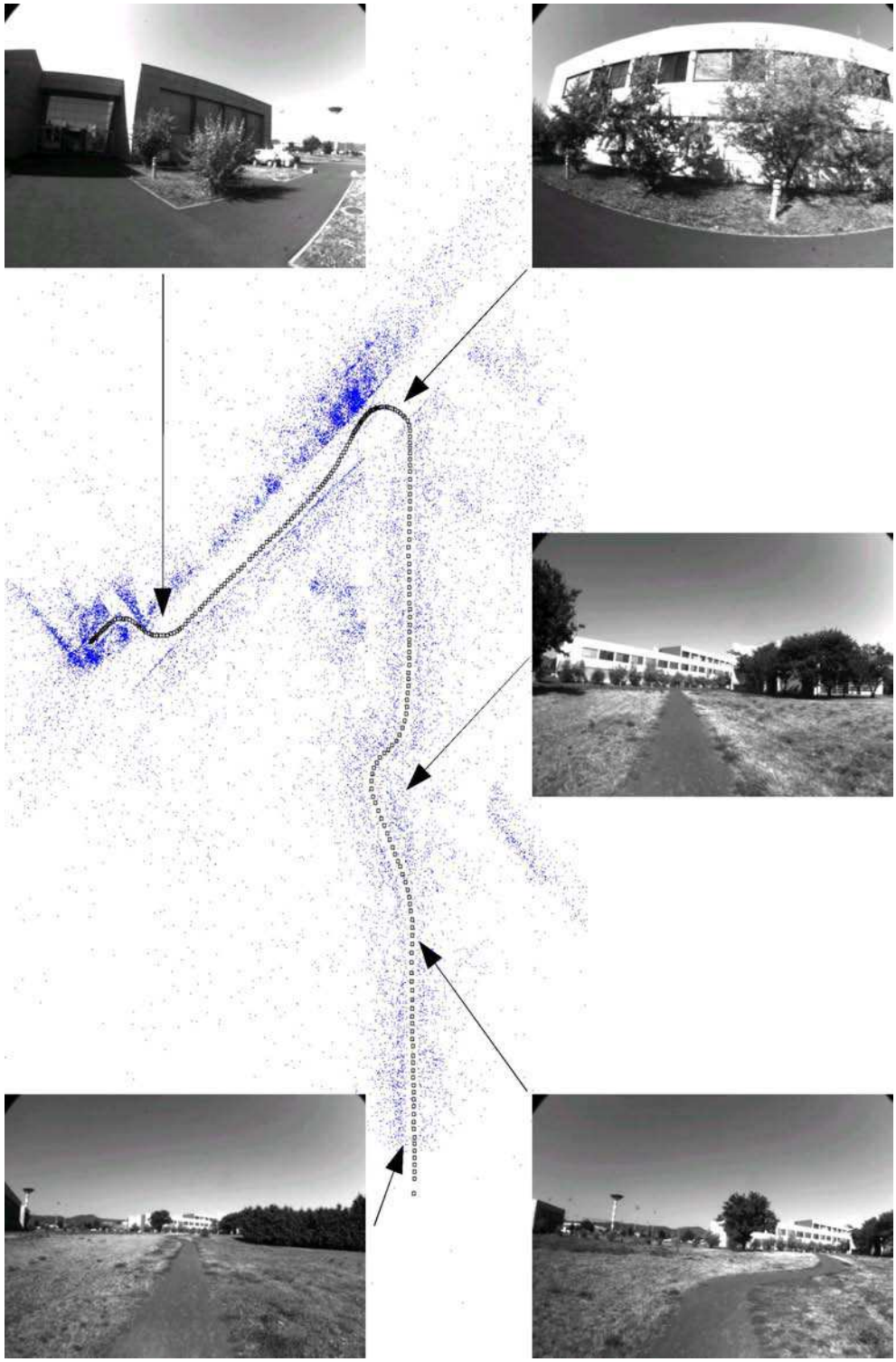

FIG. 3.12 – Reconstruction 3D devant le LASMEA sur le campus des Cézeaux (longueur : 150 m).

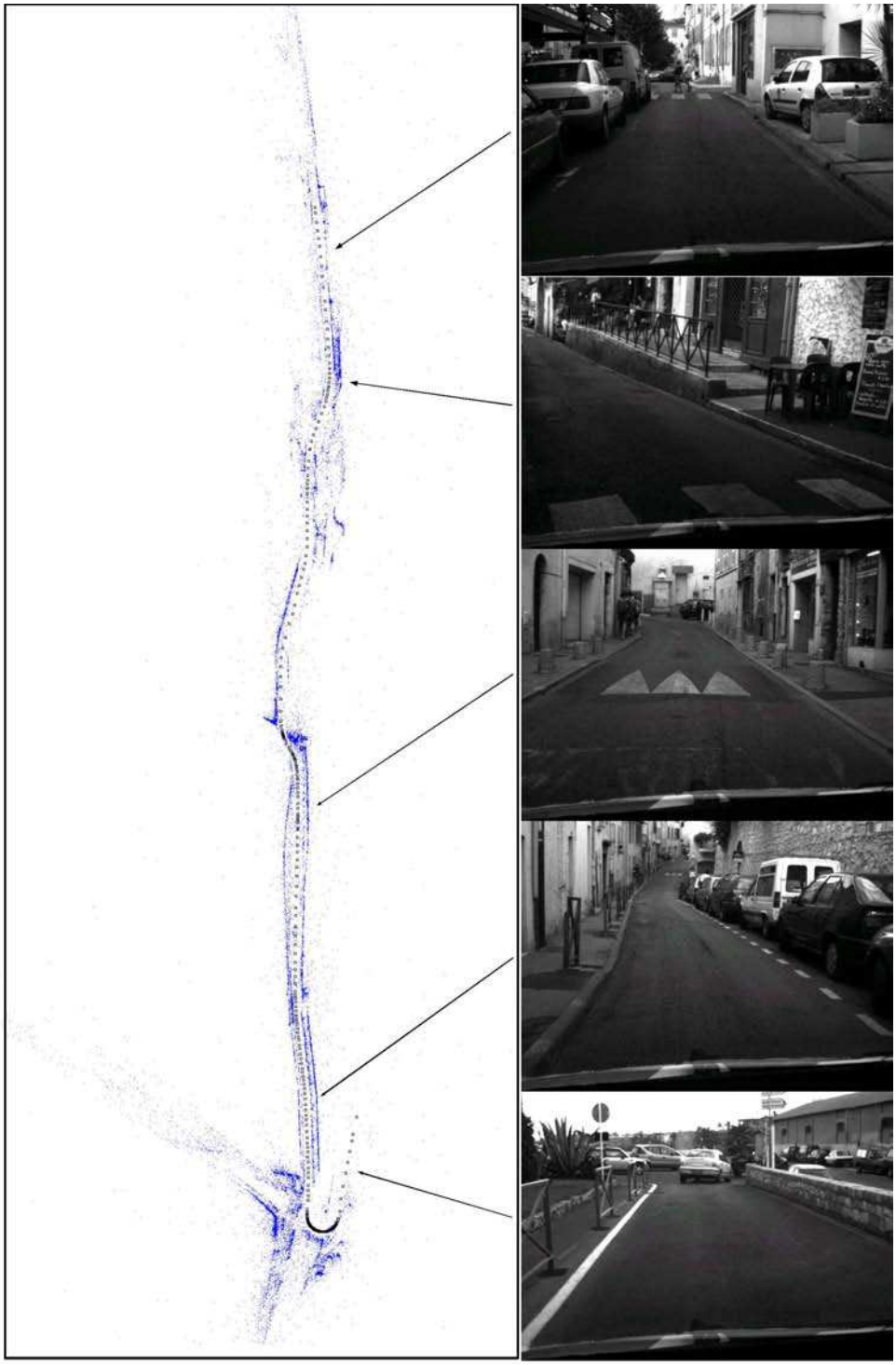

FIG. 3.13 – Reconstruction 3D dans une rue du centre ville d'Antibes (longueur : 500 m).

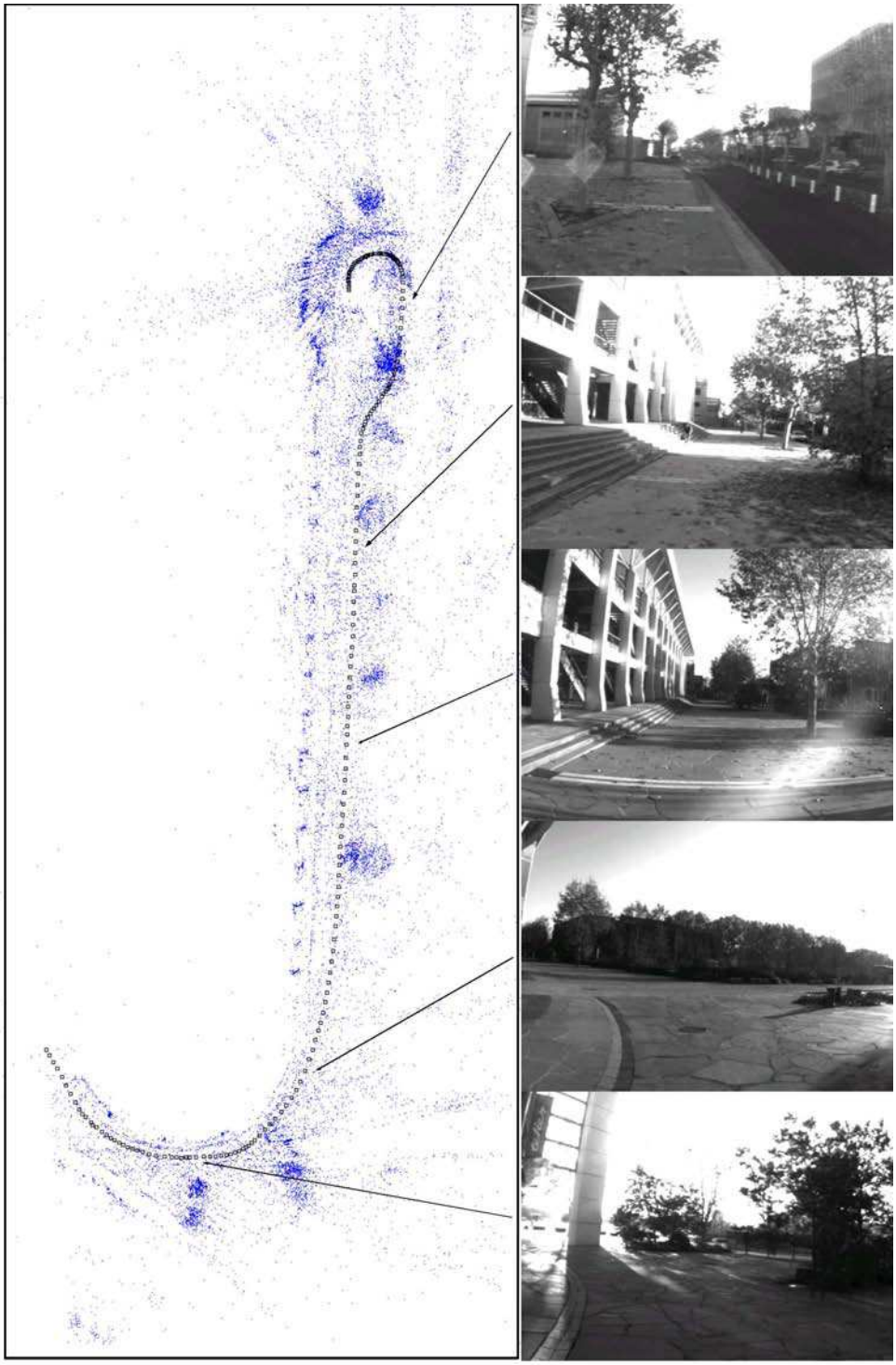

FIG. 3.14 – Reconstruction 3D autour de Polydome, Clermont-Ferrand (longueur : 300 m).

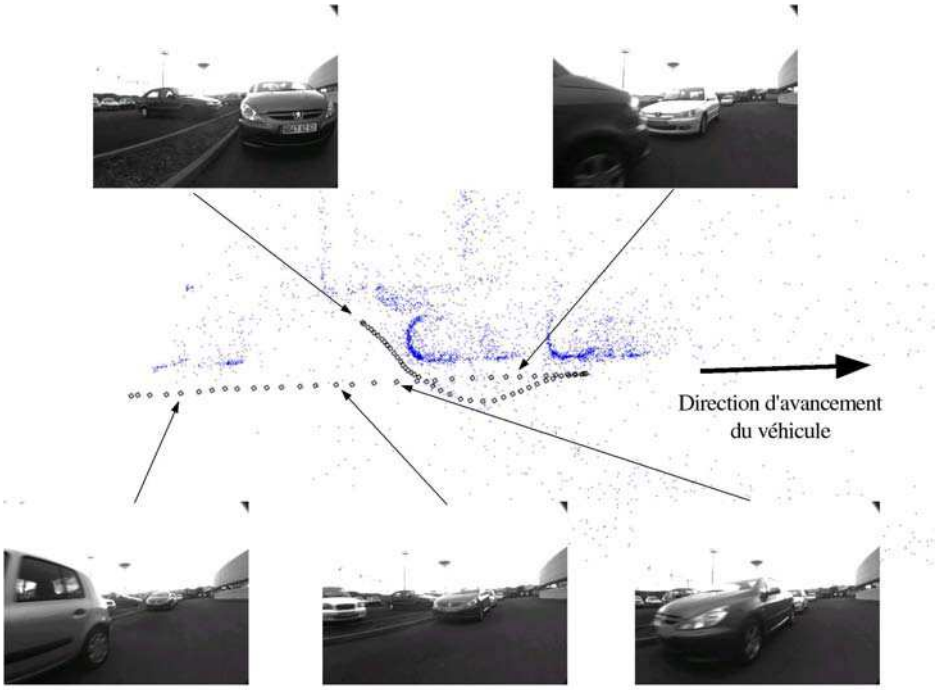

FIG. 3.15 – Reconstruction 3D lors d'une manœuvre de créneau.

3.3 Perspectives

Le processus de reconstruction 3D décrit dans ce chapitre peut être amélioré sur de nombreux points. Tout d'abord la sélection des images clef pourrait être mieux intégrée avec le calcul du mouvement de la caméra. On pourrait reconstruire le mouvement jusqu'à l'image n et choisir seulement à ce moment là l'image clef suivante. Cela permettrait de définir un critère de sélection des images clef basé sur le mouvement 3D de la caméra ou sur la structure de la scène et non pas simplement sur un critère 2D. Un autre point à améliorer par rapport à la méthode actuelle est liée à la fusion de sous-séquences. Lorsqu'on fusionne deux sous-séquences, le calcul du recalage des deux parties est fait sur seulement deux caméras communes. Ainsi une petite erreur angulaire sur ce recalage initial entraîne de grandes erreurs au bout de chacune des deux sous-séquences. L'ajustement de faisceaux qui est réalisé juste après corrige ce problème, mais demande de nombreuses itérations et peut occasionnellement diverger. Il serait plus judicieux d'améliorer le recalage initial. Deux pistes sont possibles. La première consiste à utiliser plus de caméras communes pour recaler de longues séquences. La seconde

consiste à faire quelques itérations d'un ajustement de faisceaux juste après la fusion en considérant chacune des deux sous-séquences comme des blocs rigides. Dans ce cas, l'optimisation porterait seulement sur les paramètres de la similitude appliquée à la deuxième sous-séquence pour se recoller sur la première. Après quelques itérations, un ajustement de faisceaux ordinaire pourrait être utilisé.

La critique qui peut être faite à notre méthode de reconstruction est le temps de calcul nécessaire. En effet, l'ajustement de faisceaux demande beaucoup de de temps lorsque les séquences à optimiser sont longues. Dans la méthode hiérarchique, cela se produit aux niveaux les plus élevés de la pyramide. En plus du temps de calcul, la place mémoire nécessaire augmente avec la taille de la séquence si bien qu'il devient difficile de traiter des séquences de l'ordre du kilomètre avec un PC standard. Pour résoudre ces problèmes, une autre approche est actuellement développée au laboratoire par Mouragnon *et al.* [55]. Dans cette méthode, l'ajustement de faisceaux n'est plus fait de manière hiérarchique mais de façon incrémentale. A chaque nouvelle image clef, un ajustement est utilisé pour optimiser la position des n dernières caméras et des points associés en tenant compte des reprojections des points sur les m dernières images ($n = 3$ et $m = 10$ sont des valeurs typiques). Ainsi le nombre de points et de caméras considérés dans l'ajustement de faisceaux reste plafonné quelle que soit la longueur de la séquence considérée. Cet algorithme a été utilisé avec succès pour faire des reconstructions 3D en temps réel sur le flux vidéo avec une cadence de l'ordre de 7 images par secondes.

La reconstruction 3D temps-réel permet de nouvelles applications. Mais la carte produite n'est pas aussi riche qu'une carte produite hors ligne. Cela n'est pas trop gênant si la carte est utilisée tout de suite. Par exemple on peut envisager de faire naviguer plusieurs véhicules en convoi, le premier véhicule étant conduit manuellement et les véhicules suiveurs étant en mode automatique. Pour cela, le véhicule de tête fait une reconstruction 3D et la transmet aux véhicules suiveurs qui s'en servent pour se localiser. En revanche, si la carte doit être utilisée plus longtemps après, il faut prendre en compte le fait que des changements vont se produire dans l'environnement entrainant la disparition de points préalablement reconstruits. Il faut donc que la carte contienne beaucoup plus de points que ce qui est réellement nécessaire pour se localiser. Pour cela, le calcul de la carte hors ligne reste intéressant. On pourrait envisager par exemple d'utiliser une méthode d'appariement quasi-dense pour reconstruire plus de points. On pourrait aussi, après une reconstruction selon la méthode donnée dans ce chapitre, refaire une passe d'appariements sur toute la séquence en tenant compte de la géométrie qui a été calculée pour apparier plus de points. Enfin, le problème des séquences en boucle n'a pas du tout été abordé mais il est intéressant de le traiter. Un algorithme

hors ligne pourrait chercher à retrouver au fur et à mesure si la caméra repasse dans une zone de la scène déjà reconstruite. Cela introduirait plus de contraintes dans l'ajustement de faisceaux et devrait permettre d'obtenir une carte plus précise.

D'autres méthodes pour augmenter la précision et la stabilité de la reconstruction sont envisageables. La première idée est d'intégrer dans l'ajustement de faisceaux la contrainte de non-holonomie du véhicule sur lequel est montée la caméra. En effet, les positions des caméras ne sont pas indépendantes et la prise en compte de la contrainte de non-holonomie permettrait de réduire le nombre de paramètres à optimiser dans l'ajustement de faisceaux. Cela pourrait améliorer la robustesse de l'algorithme de reconstruction 3D. Dans le cas où le véhicule est équipé d'un capteur GPS en plus de la caméra, il peut être intéressant d'utiliser les positions GPS pour localiser précisément certaines images de la séquence d'apprentissage. En effet, en zone urbaine dense, le récepteur GPS peut donner des informations précises ponctuellement même si la réception n'est pas possible de façon continue. Dans cette situation, il est possible de fixer la position des caméras pour lesquelles la position GPS est enregistrée et optimiser seulement les autres variables. En plus d'une probable amélioration de la précision de la reconstruction, cela permettrait d'obtenir directement une carte géoréférencée.

Chapitre 4

Localisation

4.1 Principe général

Une fois l'étape de cartographie effectuée, une image est suffisante pour localiser le robot. A partir de la pose du robot et de la trajectoire à suivre, on calcule la commande à envoyer aux actionneurs du robot. Dans ce chapitre, nous allons commencer par voir le calcul de la pose de la caméra dans le repère de la carte construite au chapitre précédent. Le passage de la pose de la caméra à celle du robot et l'utilisation d'un modèle de mouvement du véhicule seront étudiés au paragraphe 4.6.

Dès lors qu'une pose initiale a été calculée, la mise à jour de la pose est faite à la cadence vidéo (15 Hz dans notre cas). Le principe général peut être résumé simplement :

1. Trouver l'image clef la plus proche de l'image courante.
2. Mettre en correspondance des points entre image courante et image clef. Les points 2D de l'image courante sont ainsi associés aux points 3D de la carte.
3. Calculer la pose courante à partir des appariements 3D/2D.

La méthode utilisée pour calculer la pose à partir d'appariements 3D/2D est donnée au paragraphe 4.2. Les étapes 1 et 2 ci-dessus sont faites de façon différente selon qu'il s'agit de la localisation initiale décrite au paragraphe 4.3 ou de la mise à jour de la pose détaillée au paragraphe 4.4.

4.2 Calcul de la pose à partir des correspondances 3D/2D

A partir du moment où des correspondances ont été établies entre les points 2D de l'image courante et les points 3D du modèle, il est possible de déterminer la pose de la caméra. La méthode que nous avons employée pour cela suit un schéma classique : une pose initiale est calculée avec une méthode robuste (RANSAC) puis quelques itérations d'un algorithme de minimisation itérative sont utilisées pour affiner la solution obtenue.

Le calcul de la pose initiale avec RANSAC n'est pas toujours nécessaire. Si on connaît déjà une pose approchée de la caméra, on peut directement utiliser l'algorithme itératif. C'est ce qui est fait lorsque la pose est mise à jour.

4.2.1 Solution initiale

Il faut au minimum 3 appariements 3D/2D pour calculer la pose de la caméra. Plusieurs méthodes de calcul peuvent être utilisées. Les principales sont détaillées et comparées par Haralick *et al.* [24]. Parmi celles présentées, nous avons utilisé celle de Grunert [23] qui est l'une des plus stables numériquement. Cette méthode est basée sur des calculs trigonométriques (formule d'Al-Kashi) dans le tétrahèdre formé par le centre optique de la caméra et les 3 points 3D considérés. Le calcul de la pose de la caméra se ramène à la résolution d'une équation polynômiale de degré 4. Il peut donc y avoir quatre solutions mais le plus souvent il n'y en a que deux. L'utilisation de RANSAC permet de faire le tri parmi les solutions.

4.2.2 Calcul robuste

Le calcul robuste de la pose initiale est faite en utilisant RANSAC. Plusieurs échantillons de 3 appariements 3D/2D sont choisis aléatoirement. Pour chaque échantillon, on calcule la pose correspondante par la méthode de Grunert (on obtient éventuellement plusieurs solutions). On compte le nombre d'appariements 3D/2D cohérents avec la pose qui vient d'être calculée (inliers). Un appariement est cohérent avec la pose calculée si son erreur de reprojection est inférieure à un seuil (3 pixels). La pose pour laquelle on compte le plus grand nombre d'appariements cohérents est la pose finalement retenue. Ceci ne donne bien sûr qu'une pose approximative de la caméra car elle est calculée à partir de seulement trois appariements. La méthode d'optimisation donnée au paragraphe suivant permettra d'obtenir une solution plus précise.

L'utilisation d'une méthode de calcul de la pose à partir de 3 points seulement permet de limiter le nombre de tirages aléatoires nécessaires. En effet, on souhaite généralement avoir une probabilité de 0.99 d'avoir au moins un tirage ne comportant que des appariements corrects. Si la proportion de faux appariements est ε et le nombre d'appariements qui constituent un tirage est s, alors le nombre N de tirages à faire sera :

$$N = \frac{log(1-p)}{log(1-(1-\varepsilon)^s)}$$

Par exemple si on suppose qu'il y a 40% d'outliers, le nombre de tirages de s appariements nécessaires pour avoir une probabilité $p = 0.99$ d'avoir un tirage sans faux appariement est donné dans le tableau 4.1. Avec seulement trois appariements nécessaires, le nombre de tirages nécessaires est inférieur à 20, ce qui peut être fait en quelques millisecondes.

s	3	4	5	6	7
N	19	34	57	97	163

TAB. 4.1 – Nombre N de tirages aléatoires nécessaires pour $\varepsilon = 40\%$ d'outliers et une probabilité $p = 0.99$ d'avoir un tirage sans faux appariements, compte tenu du nombre s d'appariements dans un échantillon.

4.2.3 Optimisation

Lorsqu'une valeur approchée de la pose a été calculée (par la méthode de Grunert plongée dans RANSAC ou en utilisant un modèle du mouvement du véhicule), on peut utiliser une méthode de minimisation itérative pour affiner la pose de la caméra. Pour cela, on utilise une méthode du second ordre (Newton-Raphson) pour minimiser l'erreur de reprojection de tous les appariements corrects dans l'image en faisant varier les 6 paramètres qui définissent la pose. Cette méthode a été publiée par Araújo *et al.* [2]. On note ε le vecteur composé des erreurs de reprojection des points dans l'image :

$$\varepsilon = \begin{pmatrix} \cdots \\ q^j - PQ^j \\ \cdots \end{pmatrix} \tag{4.1}$$

où $q^j = \begin{pmatrix} x^j_{pu} \\ y^j_{pu} \end{pmatrix}$ désigne les coordonnées pixel (corrigées de la distorsion) du point numéro j, Q^j ses coordonnées 3D dans le repère du monde et P la matrice

de projection de la caméra. On note U le vecteur de paramètres formé avec les paramètres extrinsèques de la caméra :

$$U = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ T_x \\ T_y \\ T_z \end{pmatrix} \tag{4.2}$$

La fonction f associe à un vecteur de paramètres U, un vecteur de mesure des erreurs de reprojection :

$$\varepsilon = f(U) \tag{4.3}$$

Une approximation de f au premier ordre donne :

$$f(U + \delta U) \approx f(U) + J\delta U \tag{4.4}$$

où J est la matrice Jacobienne de f. Afin de minimiser $\|\varepsilon\|^2$ avec la méthode de Newton-Raphson, à chaque itération on cherche δU tel que $f(U + \delta U) = 0$, ce qui revient à chercher δU vérifiant les équations normales :

$$J^T J\delta U = -J^T \varepsilon \tag{4.5}$$

Le calcul explicite des coefficients de J est donné dans l'article d'Araújo *et al.* [2]. La seule modification apportée par rapport à la méthode décrite dans l'article concerne la gestion des faux-appariements. Dans notre implémentation, ils sont recalculés à chaque itération, et seuls les appariements corrects interviennent dans le calcul de ε et de J pour l'itération en cours.

4.3 Localisation initiale

Lorsque l'algorithme de localisation est lancé, la seule hypothèse qui est faite est que le robot est au voisinage de la trajectoire de référence. La première étape consiste donc à déterminer la position du robot le long de la trajectoire d'apprentissage. Pour cela, on cherche à déterminer parmi les images clef laquelle est la plus ressemblante à l'image courante. On détecte une liste de points d'intérêt dans l'image courante. Puis on apparie cette liste successivement avec chacune des images clef de la séquence de référence. A chaque fois, on fait un calcul de pose avec RANSAC pour éliminer les faux appariements. L'image clef qui donne le plus grand nombre d'appariements corrects est considérée comme la plus proche

de l'image courante. Ce critère est très sélectif : il n'y a généralement pas d'ambiguïté entre deux images. Par exemple, la figure 4.1 montre le nombre d'appariements obtenus lorsqu'on cherche à localiser une image dans l'ensemble de la séquence "Campus" (figure 3.12, page 55, 244 images clef). Lorsque les points de vue sont totalement différents, le nombre d'appariements corrects obtenus ne dépasse pas 5 ou 6, alors que pour l'image la plus proche, le maximum atteint une valeur souvent comprise entre 200 et 400. La pose obtenue avec RANSAC est ensuite optimisée pour obtenir la pose de la caméra.

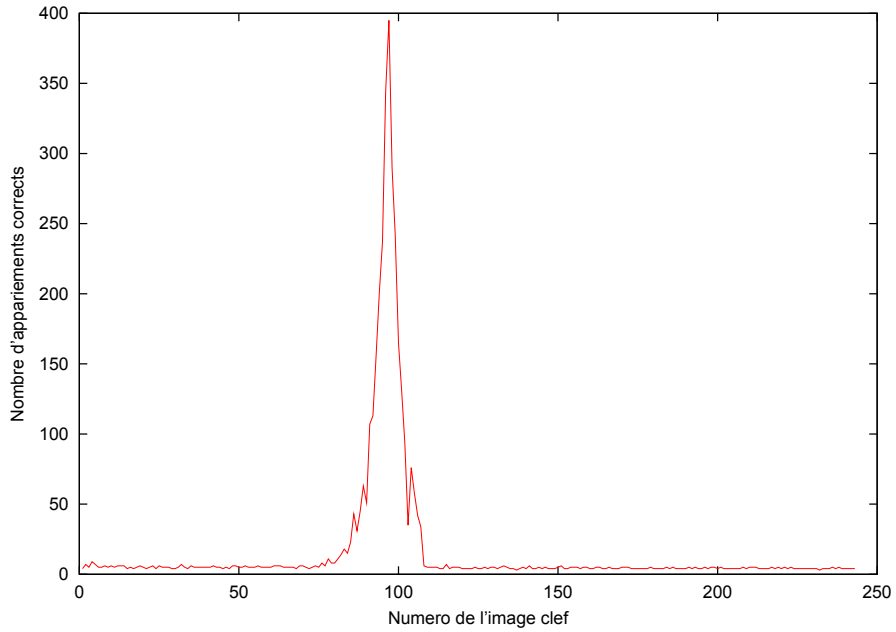

FIG. 4.1 – Nombre d'appariements corrects pour une image appariée avec chacune des images clef d'une séquence

Cette méthode pour calculer la pose initiale est plutôt lente car il faut apparier l'image courante avec la totalité des images de référence. Il existe dans la littérature des méthodes beaucoup plus efficaces pour rechercher l'image la plus ressemblante parmi les images d'une base. Nous n'avons pas cherché à mettre en œuvre une méthode plus rapide car pour l'application visée, on peut penser qu'à terme un capteur tel qu'un GPS à bas coût peut donner une pose initiale à une dizaine de mètres près. Avec cette information pour restreindre le nombre d'images clef à considérer, notre méthode d'initialisation devient tout à fait utilisable.

4.4 Mise à jour de la pose

Le calcul d'une nouvelle pose est fait à chaque nouvelle image, c'est-à-dire 15 fois par seconde. Entre l'image i et l'image $i+1$ le robot s'est peu déplacé, ce qui permet de prédire la pose approximative de la caméra à l'image $i+1$. Cela permet de savoir dès le début quelle image clef doit être utilisée. De plus, la mise en correspondance des images est simplifiée et donc le temps de calcul nécessaire est fortement réduit par rapport au calcul de la pose initiale. L'algorithme peut s'écrire en quelques étapes simples :

1. prédiction de la pose approchée de la caméra
2. choix de l'image clef la plus proche
3. détection des points d'intérêt dans l'image courante
4. appariement avec les points d'intérêt de l'image clef, ce qui donne des appariements 3D/2D
5. calcul robuste de la pose

Les paragraphes suivants donnent plus de précision sur chacune de ces étapes. La figure 4.2 montre comment s'enchaînent les opérations nécessaires à la localisation.

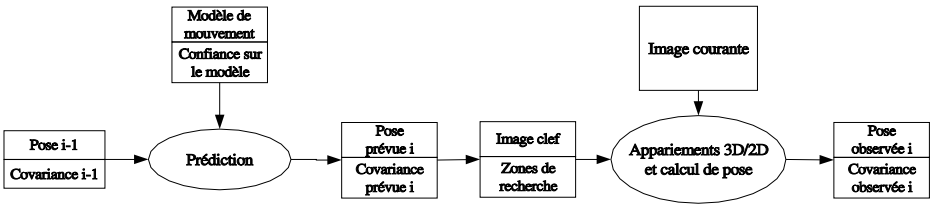

FIG. 4.2 – Prédiction de la pose approchée de la caméra, appariement des points et calcul de pose à partir de l'image courante

Prédiction de la pose approchée Si on dispose d'un modèle de mouvement du robot (ce qui sera détaillé au paragraphe 4.6) ou si on peut utiliser l'information provenant d'un capteur proprioceptif, on peut faire une prédiction de la pose de la caméra à l'image $i+1$. Même si on ne dispose pas de ces informations, on peut considérer que la pose de la caméra à l'image $i+1$ est, en première approximation, identique à ce qu'elle était à l'image i. Dans tous les cas, on associe à la

pose prédite un ellipsoïde de confiance qui est déduit de l'ellipsoïde de confiance associé à la pose de l'image i et de la confiance qu'on accorde au modèle utilisé pour la prédiction. La méthode de calcul exacte dépend du robot ou du véhicule qui supporte la caméra.

Choix de l'image clef la plus proche On note $K_1, \ldots, K_n$ l'ensemble des caméras correspondant aux images clef. Supposons que l'image clef utilisée pour localiser l'image i est K_p. La pose approchée de la caméra à l'image $i+1$ est C_{i+1}. L'image clef qui sera utilisée pour localiser l'image $i+1$ est définie comme la caméra la plus proche de C_{i+1} (en terme de distance euclidienne entre les centres optiques) parmi les cinq caméras comprises entre K_{p-2} et K_{p+2} (voir la figure 4.3). Cette méthode est bien adaptée pour un parcours le long du trajet appris dans un sens ou dans l'autre. Limiter la recherche à 5 images clef autour de la position courante permet d'avoir des trajectoires qui se recoupent (trajectoires en forme de 8 par exemple).

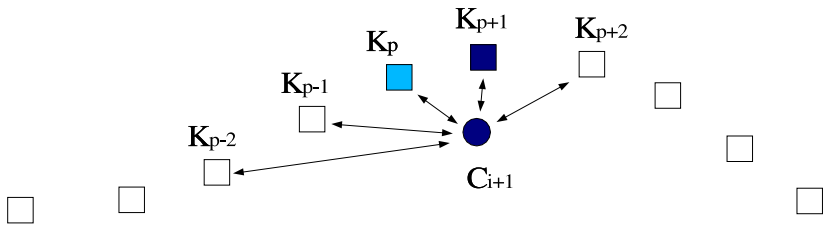

FIG. 4.3 – Choix de l'image clef la plus proche

Appariements 3D/2D L'image clef la plus proche permet de savoir quels points 3D de la carte sont potentiellement visibles dans l'image courante : ce sont tous les points visibles dans l'image clef (on les note Q_j pour $j = 1 \ldots n$). A partir de la prédiction faite sur la pose de la caméra courante, on obtient une matrice de projection approchée $\tilde{P}$ pour la caméra courante C_{i+1}. Pour chaque point 3D on obtient sa position approchée dans l'image en calculant :

$$\tilde{q_j} = \tilde{P} Q_j \tag{4.6}$$

De plus l'utilisation de la matrice de covariance $Cov_{\tilde{P}}$ associée à la pose prédite $\tilde{P}$ permet de déterminer un ellipsoïde de confiance dans l'image autour de $\tilde{q_j}$. Cette zone de confiance permet de définir la région de recherche utilisée pour

apparier les points détectés dans l'image courante avec les points de l'image clef. L'appariement des points permet d'obtenir des correspondances entre les points 2D de l'image courante et les points 3D de la carte.

Le calcul de la zone de recherche est très simple. La matrice de covariance associée à la position de q_j est simplement $J_f Cov_{\tilde{P}} J_f^T$, avec J_f la matrice Jacobienne de l'équation de projection. Cette matrice est calculée au paragraphe 4.5.1 : c'est le bloc A de la matrice J_G.

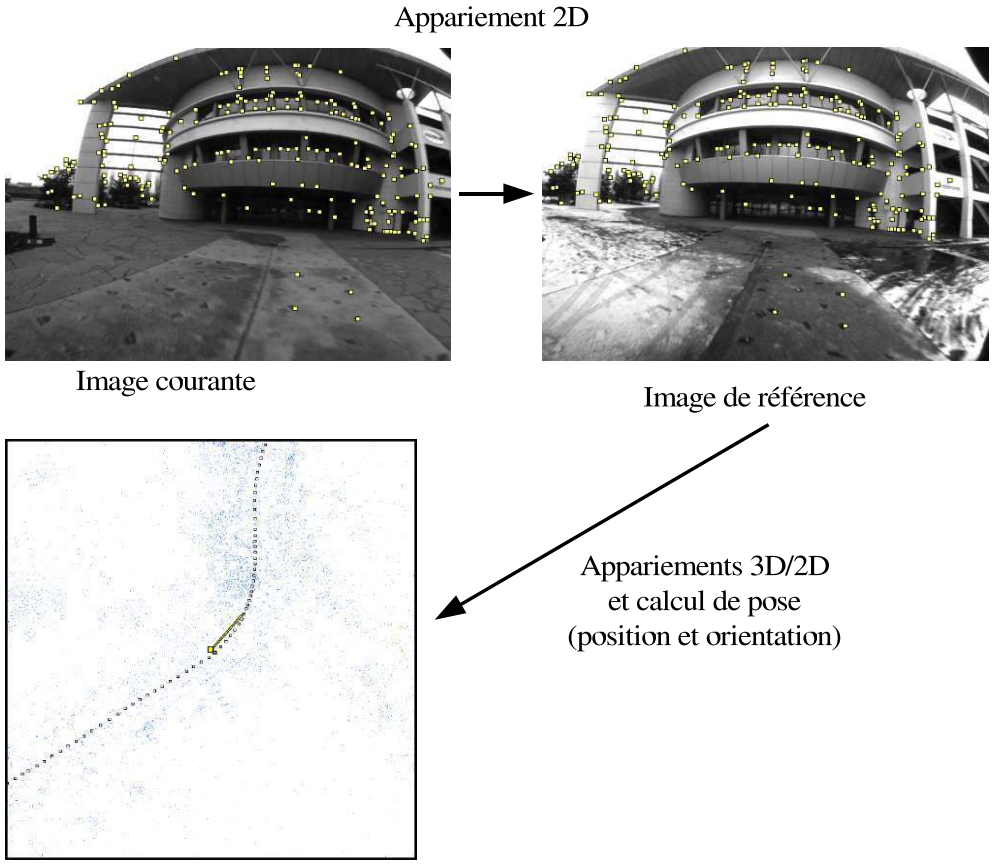

FIG. 4.4 – Appariements entre image courante et image de référence (seuls les appariements corrects sont affichés) et calcul de pose

Calcul robuste de la pose Les appariements 3D/2D permettent de calculer la pose de la caméra courante comme on l'a vu au paragraphe 4.2. La figure 4.4 donne un exemple de points appariés entre l'image courante et l'image de référence ainsi que le résultat de calcul de pose sur la carte. Deux variantes peuvent

être utilisées. La première variante consiste à faire un calcul complet en utilisant RANSAC pour obtenir une solution initiale puis la raffiner avec l'algorithme d'optimisation itératif. L'autre possibilité est d'utiliser la pose prédite par un modèle de mouvement pour initialiser l'optimisation itérative sans passer par RANSAC.

Les deux méthodes ont été utilisées. L'intérêt d'utiliser RANSAC est de pouvoir calculer la pose de la caméra $i+1$ sans dépendre du résultat du calcul de pose pour l'image i. Si pour une raison quelconque, le calcul de la pose donne un résultat aberrant, le processus de localisation est peu perturbé. Par contre, utiliser RANSAC nécessite plus de temps de calcul. De plus la pose obtenue avec RANSAC est parfois plus éloignée de la solution que la prédiction. Finalement, il est préférable de ne pas utiliser RANSAC lorsqu'on dispose d'un modèle de mouvement du véhicule. On pourrait aussi envisager de détecter les cas où la localisation se passe mal pour pouvoir réinitialiser le processus.

4.5 Calcul de l'incertitude associée à la localisation

Pour commander un robot, il est intéressant de disposer, en plus de la pose courante, d'une mesure de l'incertitude sur la pose calculée. Cette incertitude est d'abord utilisée par l'algorithme de localisation lui-même pour pouvoir prédire où se trouvera le robot à l'image suivante, ce qui permet d'avoir la position approximative des points qui devront être appariés dans l'image. D'autre part, l'incertitude associée à la pose permet de fusionner des informations de position provenant de capteurs différents pour améliorer la précision de la localisation du robot. Le calcul de l'incertitude peut aussi être utilisé directement pour la commande du robot. On peut par exemple décider de réduire la vitesse d'avancement ou même stopper complètement le véhicule si l'incertitude devient trop importante.

On pourrait calculer la matrice de covariance simplement à partir de la minimisation effectuée au paragraphe 4.2.3. Il suffirait d'inverser la matrice $J^T J$ à la solution. Mais faire cela reviendrait à considérer que tous les points 3D ont été reconstruits avec la même précision, ce qui n'est pas le cas. Certains points qui ont été suivis sur un grand nombre d'images sont généralement mieux reconstruits que des points triangulés à partir de seulement 3 images. Le calcul présenté ici tient compte de l'incertitude sur les points 3D reconstruits. Malgré tout, la propagation complète des incertitudes depuis l'étape de reconstruction 3D jusqu'à la localisation est très coûteuse en temps de calcul. Pour pouvoir calculer l'incertitude de localisation en temps-réel, certains compromis ont dû être faits. Ils sont expliqués dans les paragraphes qui suivent.

4.5.1 Calcul de la matrice de covariance associée à la pose

Le calcul de la pose de la caméra est fait à partir de n correspondances 3D/2D et on souhaite calculer la matrice de covariance associée à la pose de la caméra. Les deux sources d'incertitude que nous considérons sont d'une part l'incertitude sur la position des points détectés dans l'image et d'autre part l'incertitude qui existe sur les coordonnées des points 3D de la carte. Le calcul de l'incertitude sur les points 3D reconstruits sera fait au paragraphe 4.5.2, on suppose pour l'instant que la matrice de covariance des points 3D est connue. On va écrire le problème du calcul de la pose comme un problème de minimisation dans lequel les inconnues sont :

- la pose de la caméra (6 paramètres extrinsèques)
- la position 3D des points de la carte autour de leur position calculée dans l'étape de reconstruction 3D ($3n$ paramètres)

Idéalement, pour obtenir une estimation au maximum de vraisemblance, la minimisation à faire lors du calcul de pose devrait porter sur la totalité de ces paramètres. Mais pour cela, il faudrait répéter la résolution des équations normales avec une matrice Jacobienne J de taille $5n \times (6+3n)$ à chaque itération (et n de l'ordre de 100 à 300). Cela demanderait trop de temps de calcul. C'est pourquoi on se contente du calcul de pose avec la méthode donnée au paragraphe 4.2.3. Pour le calcul d'incertitude, par contre on considère le problème dans sa globalité pour prendre en compte toutes les sources d'incertitude. L'inversion de $J^T J$ n'est faite alors qu'une seule fois, ce qui permet de réduire significativement le temps de calcul nécessaire.

Les points 3D ayant servi à calculer la pose de la caméra sont notés Q_j pour $j = 1 \ldots n$. Le nombre n de points 3D retenus pour le calcul de la pose est nettement inférieur au nombre total de points visibles dans l'image clef. Habituellement, n est compris entre 100 et 300.

La projection du point Q_j est détectée dans l'image à la position 2D q_j. L'erreur de reprojection pour ce point est $\varepsilon_j = \pi(CQ_j) - q_j$, où $\pi(CQ_j)$ désigne la projection du point Q_j avec les paramètres de caméra C calculés pour l'image courante. On suppose que ε_j suit la loi normale $\mathcal{N}(0, \Lambda_j)$. Le vecteur composé des points 3D visibles dans l'image courante est noté $Q = (\cdots, Q_j, \cdots)^T$. Ce vecteur suit une loi normale de moyenne $Q^0 = (\cdots, X_j^0, \cdots)^T$ et de matrice de covariance Cov_{3d}. Calculer la pose de la caméra au maximum de vraisemblance signifie trou-

ver les paramètres $(C, Q_1, \cdots, Q_n)$ qui minimisent $\|G(C, Q_1, \cdots, Q_n)\|^2$ avec :

$$\|G(C,Q_1,\ldots,Q_n)\|^2 = \sum_{j=1}^{n} \varepsilon_j^T \Lambda_j^{-1} \varepsilon_j + \begin{pmatrix} \cdots \\ Q_j - Q_j^0 \\ \cdots \end{pmatrix}^T Cov_{3d}^{-1} \begin{pmatrix} \cdots \\ Q_j - Q_j^0 \\ \cdots \end{pmatrix} \tag{4.7}$$

Dans cette expression, on fait l'hypothèse que les $n+1$ vecteur aléatoires $\varepsilon_1, \ldots, \varepsilon_n, Q$ sont indépendants. Avec ces notations, on a :

$$G(C,Q_1,\ldots,Q_n) = (\varepsilon_1^T \Lambda_1^{-\frac{1}{2}}, \ldots, \varepsilon_n^T \Lambda_n^{-\frac{1}{2}}, Q^T Cov_{3d}^{-\frac{1}{2}})^T \tag{4.8}$$

Une fois le minimum trouvé, il est possible de calculer $(J_G^T J_G)^{-1}$ (avec J_G la matrice jacobienne de G) pour obtenir la matrice de covariance.

Le calcul de J_G donne la matrice creuse suivante :

$$J_G = \left(\begin{array}{c|cccc} A_1 & B_1 & 0 & \ldots & 0 \\ A_2 & 0 & B_2 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ A_n & 0 & 0 & 0 & B_n \\ \hline 0 & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & C_{i,j} & \ldots \\ \vdots & \ldots & \ldots & \ldots & \ldots \end{array}\right) \tag{4.9}$$

avec $A_j = \frac{\partial \varepsilon_j}{\partial C}$ un bloc 2×6, $B_j = \frac{\partial \varepsilon_j}{\partial Q_j}$ un bloc 2×3, $C_{i,j}$ un bloc 3×3. $Cov_{3d}^{-\frac{1}{2}}$ est une matrice $3n \times 3n$ définie par les blocs $C_{i,j}$.

La matrice de covariance Cov_{cam} associée à la pose courante est le bloc 6×6 situé en haut à gauche de la matrice $(J_G^T J_G)^{-1}$. La calcul de $J_G^T J_G$ donne :

$$J_G^T J_G = \begin{pmatrix} U & W \\ W^T & V \end{pmatrix} \tag{4.10}$$

avec U un bloc 6×6, W un bloc $6 \times 3n$ et V un bloc $3n \times 3n$. Ces blocs sont obtenus par :

$$U = A_1^T A_1 + A_2^T A_2 + \cdots + A_n^T A_n \tag{4.11}$$

$$W = \begin{pmatrix} A_1^T B_1 & A_2^T B_2 & \ldots & A_n^T B_n \end{pmatrix} \tag{4.12}$$

$$V = \begin{pmatrix} B_1^T B_1 & 0 & 0 & 0 \\ 0 & B_2^T B_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & B_n^T B_n \end{pmatrix} + Cov_{3d}^{-1} \tag{4.13}$$

Finalement, la matrice de covariance associée à la pose de la caméra est :

$$Cov_{cam} = (U - WV^{-1}W^T)^{-1} \tag{4.14}$$

Dans le cas général, V n'est pas une matrice creuse et le calcul de $WV^{-1}W^T$ peut prendre beaucoup de temps. Pour pouvoir calculer l'incertitude sur la localisation en temps-réel, il est nécessaire de faire une hypothèse supplémentaire. Si on suppose que les positions des n points 3D sont des variables aléatoires indépendantes, alors Cov_{3d} devient une matrice diagonale par blocs et la matrice V peut être réécrite sous la forme suivante :

$$V = \begin{pmatrix} B_1^T B_1 + C_{1,1}^T C_{1,1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & B_n^T B_n + C_{n,n}^T C_{n,n} \end{pmatrix} \tag{4.15}$$

Avec cette hypothèse, l'inversion de V se résume au calcul de n inverses de matrices 3×3 et Cov_{cam} peut être calculée très rapidement. De plus, le calcul de la matrice de covariance des points 3D peut aussi être simplifiée : dans ce cas seuls les blocs diagonaux sont nécessaires. Le paragraphe 4.5.2 détaille le calcul de la matrice de covariance associée aux points 3D, puis les calculs fondés sur les différentes hypothèses sont comparés au paragraphe 4.5.3.

4.5.2 Calcul de l'incertitude sur les points 3D

Nous allons présenter trois méthodes pour calculer l'incertitude associée aux points 3D. La première méthode est la méthode générale qui est normalement utilisée à la suite d'un ajustement de faisceaux. Elle permet de calculer une matrice de covariance pour l'ensemble des paramètres minimisés, c'est-à-dire la pose de chaque caméra et la position des points 3D. Comme on l'a vu au paragraphe 4.5.1, cette matrice n'est pas utilisable directement pour un calcul temps-réel. Pour cela, la deuxième méthode consiste à négliger les covariances couplées entre les points et à ne calculer que les blocs diagonaux de cette matrice. Enfin si on fait l'hypothèse que l'incertitude sur les caméras reconstruites est nulle, on peut considérablement simplifier le calcul des blocs diagonaux, ce qui constitue une troisième méthode.

Méthode générale

Les deux premières méthodes n'ont pas été développées au cours de cette thèse et s'appuient sur les travaux de Lhuillier et Perriollat [42]. Le calcul de la

matrice de covariance liée à un ajustement de faisceaux peut être fait en calculant la pseudo-inverse de la matrice Hessienne à la fin de la minimisation. La solution étant définie à une similitude près, la matrice Hessienne n'est pas inversible et il faut définir le repère de la reconstruction en fixant sept paramètres (choix de la jauge [85]). Pour fixer ce repère, on choisit d'utiliser la contrainte symétrique sur la position des caméras définie dans [42] et présentée dans le paragraphe suivant.

A la fin de l'ajustement de faisceaux, on obtient une reconstruction définie à une similitude près. On choisit une reconstruction vérifiant

$$\sum_{i=1}^{n} A_i^0 = 0 \tag{4.16}$$

avec $A_i^0, i \in \{1,\ldots,n\}$ les coordonnées des centres des caméras. Cela fixe l'origine du repère de la reconstruction. Le facteur d'échelle et l'orientation du repère sont choisis de façon arbitraire. On peut par exemple choisir le facteur d'échelle pour avoir une reconstruction métrique.

On suppose maintenant qu'à partir de la solution donnée par l'ajustement de faisceaux (et définie par A_i^0), on applique une perturbation aléatoire sur les erreurs de reprojection. Dans ce cas, une nouvelle minimisation donne une solution dont les centres des caméras sont notés A_i définie elle aussi à une similitude près. On fixe alors le repère de la nouvelle reconstruction par les contraintes suivantes :

$$\sum A_i = 0 \tag{4.17}$$

$$\sum \|A_i\|^2 = \sum \|A_i^0\|^2 \tag{4.18}$$

$$\sum A_i^0 \wedge A_i = 0 \tag{4.19}$$

L'équation 4.17 fixe l'origine du repère, l'équation 4.18 fixe l'échelle et l'équation 4.19 fixe l'orientation. On pourrait de cette façon, en effectuant un grand nombre de perturbations aléatoires des erreurs de reprojection, obtenir l'ellipsoïde de confiance associé à chaque donnée de la reconstruction. Le calcul n'est pas fait de cette manière, mais en appliquant la contrainte définie par les équations 4.17, 4.18, 4.19 au calcul de la matrice de covariance. Ceci n'est pas trivial car la contrainte ne se résume pas à fixer la position d'un point particulier ou les paramètres extrinsèques d'une caméra donnée. De plus, le calcul fondé sur la pseudo-inverse de la matrice Hessienne pour une grande reconstruction (plusieurs centaines de caméras et dizaines de milliers de points) doit être fait par blocs pour ne pas saturer la mémoire de l'ordinateur. La méthode utilisée pour mener à bien ce calcul est donnée dans [42]. Avec cette contrainte, aucune caméra ne joue un rôle particulier, ce qui permet de répartir l'incertitude de façon uniforme sur les caméras. Le calcul permet d'obtenir la matrice complète Cov_{3d} de covariance des

points 3D pour chaque image clef. On peut l'utiliser telle quelle ou conserver seulement les blocs diagonaux si on néglige les covariances couplées.

Méthode simplifiée

On peut aussi utiliser une méthode plus simple si on fait l'hypothèse additionnelle que l'incertitude sur les caméras reconstruites est nulle. Dans ce cas, la matrice de covariance peut être calculée de manière indépendante pour chaque point 3D. On considère le point Q_j vu par n cameras C_i, $i \in \{1,\ldots,n\}$. On souhaite calculer sa matrice de covariance $C_{j,j}$.

Pour cela, on suppose que les n erreurs de reprojection de ce point (notées ε_i dans l'image i) suivent des lois normales indépendantes $\mathcal{N}(0,\Lambda_i)$. De plus, on suppose que $\Lambda_i = \sigma^2 I_2$, avec σ^2 un paramètre inconnu. Dans ces conditions, la densité de probabilité pour Q_j peut s'écrire :

$$f(Q_j,\sigma^2) = \frac{1}{(2\pi\sigma^2)^n} e^{-\frac{\sum_i \|\varepsilon_i\|^2}{2\sigma^2}} \tag{4.20}$$

L'estimation au maximum de vraisemblance de Q_j et σ^2 sont $\hat{Q}_j$ qui minimise $Q_j \mapsto \sum_i \|\varepsilon_i\|^2$ et $\hat{\sigma^2} = \frac{\sum_i \|\varepsilon_i\|^2}{2n}$. On remplace l'estimation de σ^2 par l'estimateur non biaisé $\hat{\sigma^2} = \frac{\sum_i \|\varepsilon_i\|^2}{2n-3}$, avec $2n$ le nombre d'observations, et 3 le nombre de degrés de liberté de Q_j. Finalement, on obtient l'estimation de Λ_i, et $C_{j,j}$ en calculant l'inverse de $J_F^T J_F$:

$$C_{j,j} = (J_F^T J_F)^{-1} \tag{4.21}$$

où J_F est la matrice jacobienne du vecteur d'erreur F avec $F = (\varepsilon_1^T \Lambda_1^{-\frac{1}{2}}, \ldots, \varepsilon_n^T \Lambda_n^{-\frac{1}{2}})^T$.

La figure 4.5 montre les ellipsoïdes de confiance calculés avec cette méthode pour les points visibles dans une image clef de la séquence en boucle (voir Fig. 5.11, page 97).

4.5.3 Comparaison

Afin de savoir si les compromis faits dans le calcul de l'incertitude de la localisation sont acceptables, nous avons comparé quatre méthodes de calcul de la matrice de covariance fondées sur différentes hypothèses.

- Dans la méthode 1, Cov_{cam} est calculée avec la matrice complète Cov_{3d}.
- Dans la méthode 2, Cov_{3d} est calculée avec la méthode générale, mais seuls les blocs diagonaux sont utilisés pour calculer Cov_{cam}.
- Dans la méthode 3, Cov_{3d} est calculée avec la méthode simplifiée (en faisant l'hypothèse que les cameras reconstruites ont une incertitude nulle).

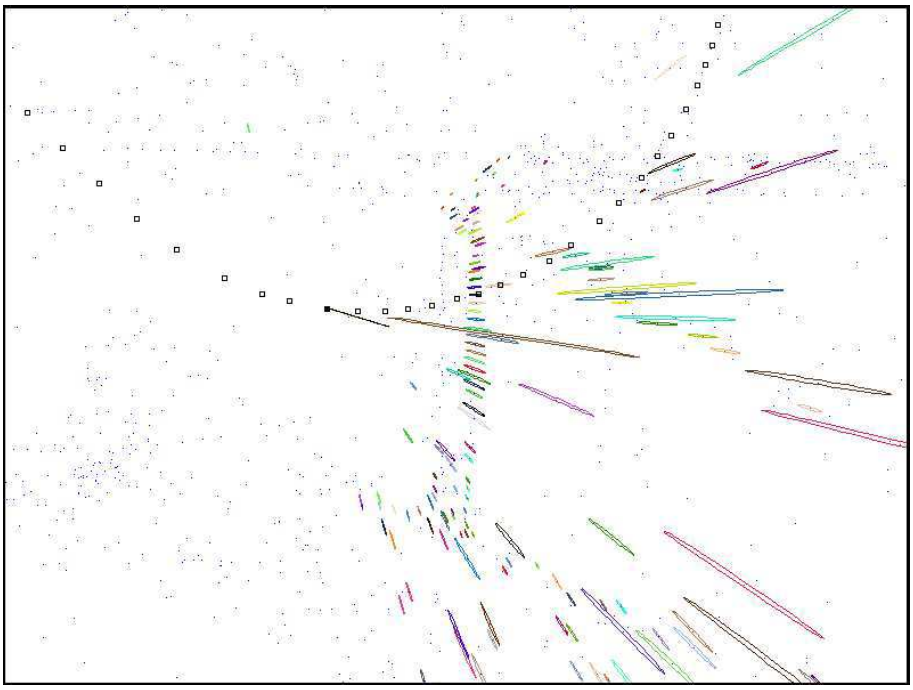

FIG. 4.5 – Ellipsoïdes de confiance à 99% calculés pour les points visibles dans une image clef (vue de dessus).

– Dans la méthode 4, on néglige l'incertitude sur les points 3D. La seule source d'incertitude considérée est l'incertitude sur la position des points 2D dans l'image courante.

La méthode 1 demande trop de temps de calcul pour être utilisée en temps réel, mais elle est incluse dans ce comparatif pour savoir si l'utilisation des blocs non diagonaux de Cov_{3d} est indispensable. On calcule la longueur du demi grand axe de l'ellipsoïde de confiance à 90% associé à la position de la caméra pour les quatre méthodes. Le résultat apparaît sur la figure 4.6 pour quelques images de la vidéo utilisée à la section 5.4.6, page 105.

Comme on pouvait s'y attendre, prendre en compte l'incertitude des points 3D augmente la taille des ellipsoïdes. La prise en compte de l'incertitude sur les caméras dans la reconstruction a le même effet. Cependant la différence entre les méthodes 1 et 2 est faible comparée à la différence avec les autres méthodes. Comme la méthode 2 est utilisable en temps réel, cela semble être un bon compromis entre précision et temps de calcul. Le temps de calcul pour les méthodes 2 et 3 est le même, la différence tient à la difficulté d'implémenter le calcul des blocs diagonaux de Cov_{3d}. Avec la simplication de la méthode 3, la taille de l'ellipsoïde

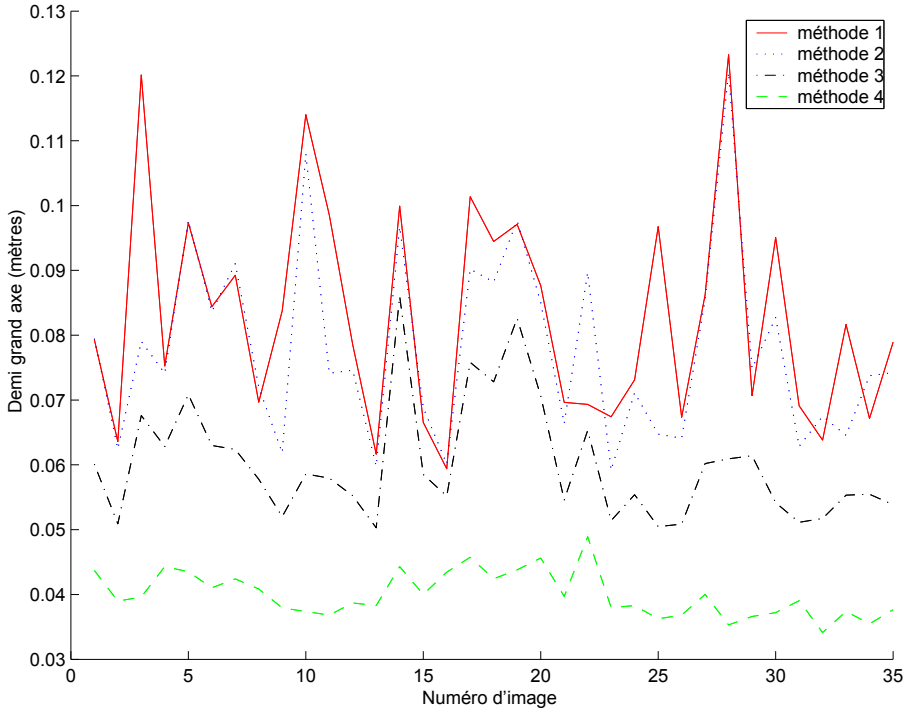

FIG. 4.6 – Demi grand axe (en mètres) de l'ellipsoïde de confiance à 90% avec quatre méthodes de calcul pour quelques images localisées par rapport à une image clef donnée. Les variations d'une image à l'autre sont essentiellement dues à des différences dans les correspondances 3D/2D.

est sous-estimée mais le résultat est tout de même nettement plus proche du calcul complet qu'en négligeant l'incertitude sur les points 3D (méthode 4). Pour compléter ces données, une comparaison de la taille des ellipsoïdes avec l'erreur de localisation mesurée grâce au GPS différentiel est faite à la section 5.4.6.

4.6 Prise en compte du modèle de mouvement du véhicule

Comme on l'a vu précédemment, il est intéressant de pouvoir prédire la pose approximative de la caméra à l'image i en utilisant le passé. Ceci peut être fait si la caméra est montée sur un véhicule. Nous l'avons fait dans le cas où la caméra est montée sur le cycab. Le résultat de la prédiction du mouvement et la pose résultant de l'observation sont alors utilisés dans un filtre de Kalman comme illustré par la

figure 4.7.

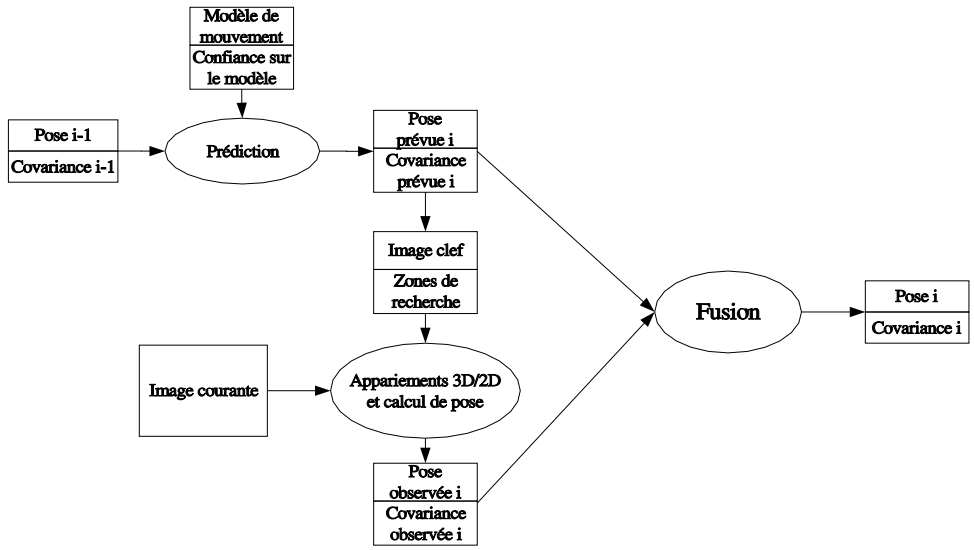

FIG. 4.7 – Filtre de Kalman prenant en compte le modèle de mouvement du véhicule

Seul un modèle simplifié a été utilisé afin d'illustrer comment un modèle de mouvement pouvait être pris en compte dans le processus de localisation. Un modèle mieux adapté au cycab, comme le modèle tricycle décrit au paragraphe 5.3.1, permettrait sans doute d'améliorer encore les performances du système. On pourrait aussi prendre en compte les mesures odométriques (vitesse de rotation de chaque roue et angle de braquage) ou la commande envoyée au cycab (calculée au moment de la localisation de l'image précédente). Pour utiliser le modèle de mouvement du cycab, la transformation rigide permettant de passer du repère caméra au repère du véhicule a été mesurée directement sur le véhicule.

Le modèle utilisé est représenté sur la figure 4.8. Connaissant la pose de la caméra à l'instant t et à des instants passés, on en déduit la pose du véhicule et on calcule le vecteur vitesse instantané du véhicule $\overrightarrow{v}$. Comme le calcul de l'orientation de la caméra est souvent plus précis que le calcul de la position, la direction du vecteur $\overrightarrow{v}$ est obtenue à partir de l'orientation de la caméra calculée à l'instant t. On note $\overrightarrow{v_0}$ le vecteur unitaire orienté selon la direction d'avancement du véhicule. La norme du vecteur vitesse est ensuite obtenue par :

$$\| \overrightarrow{v} \| = \frac{\overrightarrow{v_0} \cdot \overrightarrow{O_R(t-\Delta_t)O_R(t)}}{\Delta_t} \tag{4.22}$$

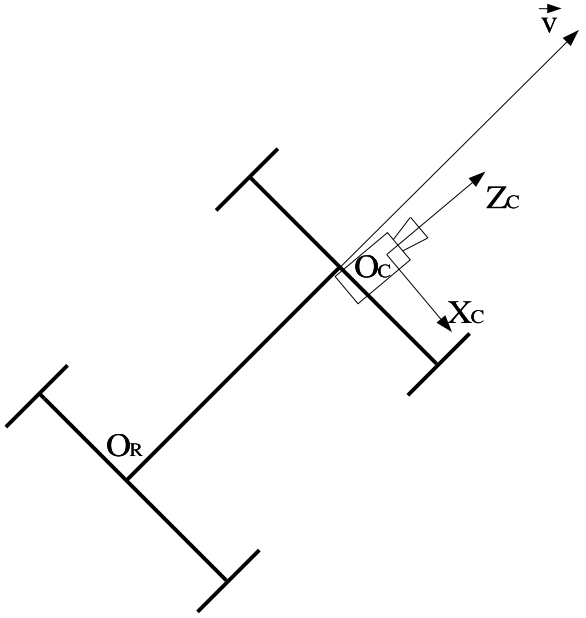

FIG. 4.8 – Modèle de mouvement utilisé dans le filtre de Kalman. Le point O_R situé au milieu de l'essieu arrière est l'origine du repère robot. Le mouvement considéré est une translation selon le vecteur vitesse.

où $O_R(t)$ désigne la position du robot à l'instant t. Dans ce modèle simplifié, on considère que l'angle de braquage du véhicule est nul (les roues sont supposées droites) et que le sol est plan. Le déplacement du véhicule durant un intervalle de durée Δ'_t est donc $\Delta'_t \overrightarrow{v}$. On applique donc cette translation au centre optique de la caméra pour obtenir la pose de la caméra à l'image suivante.

Le calcul de la pose s'accompagne du calcul de la matrice de covariance associée. La matrice de covariance à l'instant $t+\Delta'_t$ s'obtient à partir de la matrice de covariance à l'instant t en augmentant la variance sur chacune des variables :

$$Cov(t+\Delta'_t) = Cov(t) + diag(\sigma^2_{rot}, \sigma^2_{rot}, \sigma^2_{rot}, \sigma^2_{pos}, \sigma^2_{pos}, \sigma^2_{pos}) \tag{4.23}$$

où σ_{rot} et σ_{pos} sont deux constantes choisies en fonction de la confiance qu'on accorde à la prédiction pour la rotation et pour la position. Nous utilisons $\sigma_{pos} = 0.05\ m$ et $\sigma_{rot} = 0.25°$.

La prédiction qui vient d'être calculée permet de définir les zones de recherche pour l'appariement des points entre image courante et image de référence. La

pose de la caméra et la matrice de covariance associée sont calculées par vision (elles sont notées Pose observée et Covariance observée sur la figure 4.7). On fusionne ensuite la pose observée et la pose prédite pour obtenir la pose finale qui est utilisée pour commander le véhicule. La fusion est faite en utilisant les formules classiques du filtre de Kalman. On note X le vecteur d'état de la caméra formé par les 6 paramètres de pose. Le gain K est calculé par :

$$K = Cov_{prediction}(Cov_{prediction} + Cov_{observation})^{-1} \tag{4.24}$$

Et le résultat de la fusion est obtenu par :

$$X_{fusion} = X_{prediction} + K(X_{observation} - X_{prediction}) \tag{4.25}$$

$$Cov_{fusion} = Cov_{prediction} - KCov_{prediction} \tag{4.26}$$

4.7 Perspectives

L'amélioration des performances en localisation est certainement possible en couplant plus étroitement l'algorithme de vision, le modèle du véhicule, la loi de commande et les éventuels capteurs proprioceptifs. Mise à part au paragraphe 4.6, la caméra associée à l'algorithme de localisation se comporte comme un capteur de position indépendant du robot sur lequel il est installé. C'est une propriété intéressante, car cela permet une utilisation dans un grand nombre d'applications : la caméra peut être fixée rigidement sur un véhicule (équipé ou non d'autres capteurs), ou simplement tenue à la main. Par contre, cela impose à l'algorithme de vision de calculer la pose de la caméra sans utiliser certaines informations qui peuvent être disponibles et qui permettraient d'améliorer les performances du systèmes. Par exemple, la connaissance d'un modèle de mouvement du véhicule a été considérée au paragraphe 4.6. Mais d'autres informations pourraient également être prises en compte pour prévoir la pose de la caméra avant de procéder à l'appariement entre image courante et image de référence. Avec une meilleure prédiction, il est possible de réduire la taille des zones de recherche et donc d'accélérer la mise en correspondance tout en réduisant le nombre d'appariements erronés.

Pour pouvoir intégrer des informations provenant du véhicule ou d'autres capteurs, il devient nécessaire de connaître avec précision le changement de repère entre le repère caméra et le repère robot. Il serait donc utile de développer une méthode spécifique de calibrage. D'autre part, les informations odométriques et la modélisation du véhicule ne sont facilement utilisables que si on fait l'hypothèse d'un sol plan. Or il est nécessaire de prendre en compte l'inclinaison du sol

ainsi que le tangage et le roulis du véhicule pour obtenir la pose complète de la caméra avec six degrés de liberté. Pour résoudre ce problème, on pourrait imaginer modéliser la déclivité du sol le long de la trajectoire d'apprentissage à partir de la reconstruction 3D calculée.

Chapitre 5

Mise en œuvre et performances

5.1 Matériel utilisé

Le matériel utilisé pour les expérimentations est composé d'un ordinateur portable et d'une caméra. C'est le minimum pour faire fonctionner le système de localisation. Les expérimentations qui intègrent la commande du robot ont été faites sur le cycab. Afin de comparer les résultats obtenus par vision à la vérité terrain, un récepteur GPS différentiel a été utilisé. Le matériel a évolué au cours du temps. La présentation qui en est faite ici concerne le matériel le plus récent. Certaines expérimentations plus anciennes ont été réalisées avec un matériel légèrement différent. Cela sera précisé au cas par cas si nécessaire.

5.1.1 Matériel informatique

Nous avons successivement utilisé des PC portables équipés de processeurs Pentium 4 à 3 GHz ou 3.4 GHz, puis de processeurs Pentium M à 2 GHz. Les temps de calcul étaient très proches sur les deux types de machines.

Au niveau logiciel, les acquisitions provenant des différents capteurs sont gérées par l'architecture D-Bite. Cela permet d'enregistrer en même temps que la donnée capteur, une date exprimée en millisecondes qui permet de synchroniser les données entre la caméra, l'odomètre et le GPS.

5.1.2 La caméra

La caméra est une caméra CMOS noir et blanc. Elle produit des images de résolution 1024×768 pixels. Traiter des images de cette taille serait trop lent. Aussi, elles sont rééchantillonnées par logiciel à une taille de 512×384. La procédure

de rééchantillonnage a été programmée en utilisant le jeu d'instructions SSE. Le niveau de gris d'un pixel de l'image d'arrivée est le résultat de la moyenne des niveaux de gris de 4 pixels contigus de l'image originale. Le rééchantillonnage prend moins de 2 *ms*. La caméra fournit des images à une fréquence de 15 images par seconde.

L'objectif utilisé est un objectif à très grand angle. Sa focale est de 3.5 *mm*, avec une caméra disposant d'un capteur dont la diagonale mesure $1/2$" soit 12.7 *mm*. Si on néglige la distorsion, le champ angulaire (dans la diagonale) est obtenu par $\alpha = 2tan^{-1}(\frac{12.7}{3.5\times 2})$ soit $\alpha = 122°$. En réalité, à cause de la distorsion, le champ réel est plus grand que cela. Le champ atteint 130° dans la largeur de l'image.

L'utilisation d'un objectif à très grand angle est très intéressante car cela permet de réduire les risques d'occultation. Par exemple, une personne située à deux ou trois mètres de la caméra occupe une toute petite partie de l'image. Dans ce cas, la personne masque très peu de points d'intérêt, ce qui est généralement sans conséquence sur le résultat de la localisation.

Signalons enfin que la principale difficulté lorsqu'on utilise une caméra en extérieur est le manque de dynamique des caméras actuelles. Il est fréquent de ne pas pouvoir trouver un couple diaphragme et temps de pose pour lequel la totalité de l'image soit correctement exposée. Par exemple, si les façades des bâtiments sont en plein soleil d'un côté de la rue et dans l'ombre de l'autre côté, il faut souvent faire un choix entre surexposer un côté de l'image ou sous-exposer l'autre côté, ce qui revient dans les deux cas à se priver d'une partie de l'information utile pour se localiser. Des caméras avec une plus grande dynamique permettraient certainement d'améliorer la robustesse des algorithmes de vision en extérieur.

5.1.3 Le cycab

Le robot sur lequel ont été réalisées les expérimentations est le cycab (voir figure 5.1). C'est un véhicule électrique de petite taille (1.90 *m* de long et 1.20 *m* de large). Il est capable de transporter deux personnes à une vitesse maximale de 18 km/h. La caméra est placée sur le toit, à environ 1.80m au dessus du sol. L'axe optique est dirigé vers l'avant. Du point de vue de la commande, le cycab est un véhicule non holonome dont la modélisation sera abordée plus en détail au paragraphe 5.3.1.

FIG. 5.1 – Le cycab

5.2 Temps de calcul

Pour pouvoir commander le robot de manière convenable, il est nécessaire de pouvoir calculer sa position et son orientation à une fréquence suffisante. Avant le début de ces travaux, le récepteur GPS était utilisé pour calculer le vecteur d'état du véhicule, ce qui était fait à 10 Hz. Comme la caméra fournit 15 images par seconde, nous avons cherché à calculer la pose de la caméra en moins de 66 *ms* afin de traiter les images à la même cadence que le flux vidéo. Pour cela, le code de détection des points d'intérêt et d'appariement a été particulièrement optimisé. Il fait usage de la parallélisation SIMD (Single Instruction Multiple Data) possible avec les processeurs récents. En l'occurence, sur les processeurs Pentium 4 et Pentium M, les jeux d'instructions utilisés sont ceux du MMX, SSE et SSE2 ([31],[32],[30]). Ces instructions permettent de traiter en parallèle plusieurs données du même type (entier ou flottant) en leur appliquant la même opération. Par exemple, on peut en une seule instruction additionner deux vecteurs de 16 entiers codés sur 8 bits. Les opérations disponibles regroupent toutes les opérations arithmétiques et logiques de base ainsi que certaines opérations plus spécialisées. Elles portent sur des opérandes vectorielles dont la taille totale est de 128 bits divisées en composantes de 8, 16, 32 ou 64 bits. Elles ont à l'origine été introduites pour accélérer les programmes dédiés au multimédia : traitement de l'image ou du son. L'utilisation de ces instructions est maintenant devenue incontournable pour

exploiter au mieux la puissance des processeurs. En effet, le code optimisé pour le SIMD est nettement plus rapide qu'un code séquentiel classique. Par exemple, le détecteur de points d'intérêt que nous avons développé est 4 fois plus rapide avec les instructions SSE2 qu'avec du code C classique. C'est dans les opérations de traitement d'image que les gains sont le plus facile à obtenir. Même s'il est possible d'obtenir un gain de vitesse pour les calculs géométriques de la partie vision, cela n'a pas été utilisé dans ces travaux car cela allonge le temps de développement et cela réduit nettement la lisibilité et l'évolutivité du code pour un gain de vitesse bien moins important. Finalement, le temps de calcul pour chacune des étapes de la localisation est réparti comme indiqué dans le tableau 5.1 pour des images de taille 512×384 pixels et un processeur Pentium 4 à 3 GHz.

Opération	temps de calcul
Détection des points et calcul des descripteurs locaux	35 ms
Appariement	15 ms
Calcul de pose	10 ms
Calcul de la matrice de covariance	5 ms

TAB. 5.1 – Répartition du temps de calcul pour la localisation

Etant donné qu'elle est faite hors ligne, la partie reconstruction 3D n'a pas fait l'objet d'autant d'attention pour réduire le temps de calcul. Pour une séquence comportant 300 images clef, la reconstruction 3D demande environ une heure, l'essentiel du temps étant passé dans l'ajustement de faisceaux.

5.3 Navigation autonome du cycab

La loi de commande a été développée par l'équipe commande du LASMEA. Nous l'avons simplement utilisée comme une boîte noire pour mener à bien les expérimentations. Elle est brièvement présentée ici mais on trouvera plus de détails dans les publications de Thuilot *et al.* [82].

5.3.1 Modélisation du véhicule

Sur le cycab, les quatre roues peuvent être directrices, mais pour l'ensemble des expérimentations présentées dans ce manuscrit, seules les roues avant sont utilisées pour la direction. Dans cette configuration, le cycab se déplace comme une voiture. Un modèle cinématique classique, le modèle tricycle dans lequel les

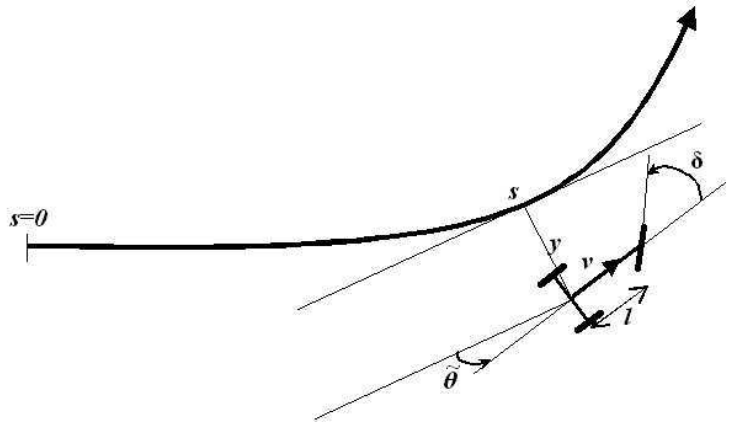

FIG. 5.2 – Modèle tricycle

deux roues avant sont fusionnées en une seule roue virtuelle, est utilisé. On peut voir une illustration de ce modèle sur la figure 5.2. La configuration du véhicule peut être décrite sans ambiguïté par un vecteur d'état de dimension 3. Ce vecteur est composé de :

- l'abscisse curviligne le long de la trajectoire de référence : s,
- l'écart latéral à la trajectoire de référence : y,
- l'écart angulaire par rapport à la trajectoire de référence : $\tilde{\theta}$.

D'un autre côté, le vecteur de commande est constitué de :

- la vitesse d'avancement du véhicule : v,
- l'angle de braquage des roues avant : δ.

Le modèle du véhicule est donné par (voir [14]) :

$$\begin{cases} \dot{s} = & v\frac{\cos\tilde{\theta}}{1-yc(s)} \\ \dot{y} = & v\sin\tilde{\theta} \\ \dot{\tilde{\theta}} = & v\left(\frac{\tan\delta}{l} - \frac{c(s)\cos\tilde{\theta}}{1-yc(s)}\right) \end{cases} \tag{5.1}$$

où l est la longueur du véhicule (distance entre essieu avant et arrière), $c(s)$ est la courbure de la trajectoire de référence à l'abscisse curviligne s. On suppose que $y \neq \frac{1}{c(s)}$ (le véhicule n'est pas au centre de courbure de la trajectoire) et que $\tilde{\theta} \neq \frac{\pi}{2}[\pi]$ (le véhicule n'est pas orienté à angle droit par rapport à la trajectoire de référence). En pratique, ces situations ne se produisent jamais.

5.3.2 Loi de commande

L'objectif de la loi de commande est d'amener et de réguler les variables d'état y et $\tilde{\theta}$ à zéro. Pour cela, la commande agit seulement sur l'angle de braquage δ. La vitesse d'avancement est considérée comme un paramètre variable. Le vecteur d'état est obtenu à partir des informations calculées par l'algorithme de localisation par vision. Par une transformation inversible de l'état et de la commande, le modèle non-linéaire (5.1) peut être converti de manière exacte, sous forme chaînée (voir [74]). Le vecteur d'état chaîné est (a_1, a_2, a_3) avec :

$$\begin{cases} a_1 = & s \\ a_2 = & y \\ a_3 = & (1 - yc(s)) \tan\tilde{\theta} \end{cases} \tag{5.2}$$

et le vecteur de commande chaîné est $M = (m_1, m_2)^T$ avec :

$$\begin{cases} m_1 = & \frac{da_1}{dt} \\ m_2 = & \frac{da_3}{dt} \end{cases} \tag{5.3}$$

De cette forme chaînée, une grande partie de la théorie de l'automatique linéaire peut être employée (comme les transformations sont exactes, il n'est pas nécessaire que l'état du véhicule soit dans une configuration particulière). Plus précisément, on peut noter que le suivi de chemin (c'est-à-dire la commande de a_2 et a_3) peut être réalisé indépendamment en contrôlant uniquement m_2 comme un contrôleur proportionnel dérivé.

L'expression de la variable de commande δ peut être obtenue en inversant la transformation chaînée de la loi de commande. Le calcul, détaillé dans [82], donne :

$$\begin{aligned} \delta(y,\tilde{\theta}) = \quad & \arctan\Big(l\Big[\tfrac{\cos^3\tilde{\theta}}{(1-c(s)y)^2}\Big(\tfrac{dc(s)}{ds}y\tan\tilde{\theta} \\ & - K_d(1-c(s)y)\tan\tilde{\theta} - K_p y \\ & + c(s)(1-c(s)y)\tan^2\tilde{\theta}\Big) + \tfrac{c(s)\cos\tilde{\theta}}{1-c(s)y}\Big]\Big) \end{aligned} \tag{5.4}$$

où K_p et K_d sont les gains proportionnels dérivatifs ($K_d > 0$ et $K_p > 0$).

Couplage entre localisation et commande

L'algorithme de localisation fournit la pose de la caméra avec six degrés de liberté. Or le vecteur d'état du robot comporte seulement trois degrés de liberté car le véhicule est contraint de rester en contact avec le sol. Pour le fonctionnement

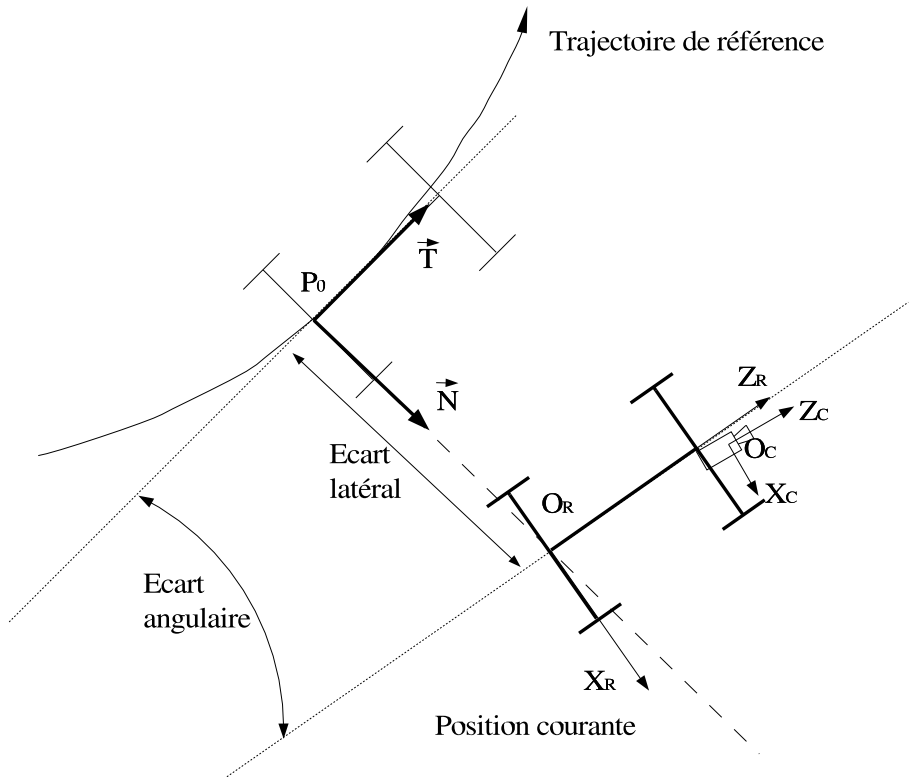

FIG. 5.3 – Calcul de l'écart latéral et angulaire par rapport à la trajectoire de référence

de l'algorithme de vision, il est important de toujours garder la pose complète de la caméra afin de pouvoir suivre efficacement les points d'intérêt dans l'image. A partir de la pose de la caméra, on applique le changement de repère entre le repère caméra et le repère robot pour obtenir la pose du robot dans l'espace. On projette ensuite la position et les axes du repère robot sur le plan du sol (supposé horizontal). La figure 5.3 permet de préciser les notations. La position du robot est repérée par le milieu de l'essieu arrière noté O_R (origine du repère robot).

La pose du robot permet de calculer l'écart latéral et l'écart angulaire par rapport à la trajectoire de référence. Pour cela, on commence par calculer la position du point P_0 le point le plus proche de la position courante sur la trajectoire de référence. On détermine la tangente $\vec{T}$, la normale $\vec{N}$ à la trajectoire de référence au point P_0 ainsi que la courbure $c(s)$ en ce point. L'écart latéral est la distance $\|\overrightarrow{P_0O_R}\|$. L'écart angulaire est l'angle formé par $\vec{T}$ et la direction d'avancement du robot. Finalement les trois informations utilisées dans la loi de commande sont l'écart latéral, l'écart angulaire et la courbure.

5.4 Précision

5.4.1 Ordre de grandeur de la précision de localisation

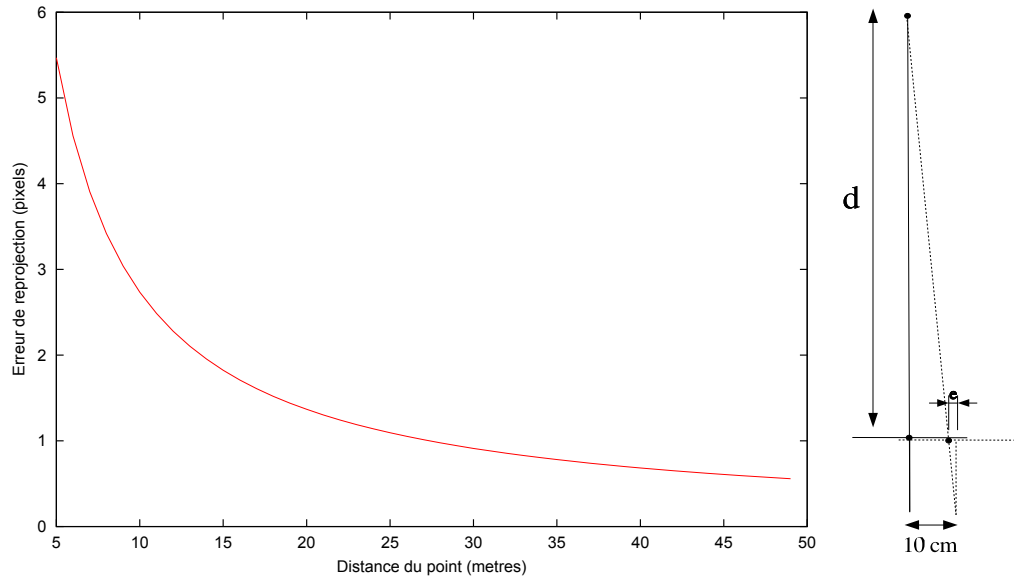

FIG. 5.4 – Erreur de reprojection e d'un point situé à une distance d donnée de la caméra pour une erreur de localisation latérale de 10 *cm*

La précision d'un algorithme de vision est souvent donnée dans l'image comme une erreur de reprojection exprimée en pixels. C'est tout à fait pertinent pour une application de réalité augmentée où la qualité de l'intégration d'un objet virtuel va être jugée sur l'image. En revanche la précision de la localisation pour un robot mobile doit être mesurée dans le monde réel et non pas dans l'image. Notre objectif est d'atteindre une précision du même ordre que celle d'un GPS différentiel, soit quelques centimètres. Mais cela n'est possible qu'à condition que les amers observés soient situés suffisamment proches de la caméra. Un calcul simple permet d'avoir un ordre de grandeur des performances qu'on peut attendre d'un système de localisation tel que le nôtre. Le graphique de la figure 5.4 montre l'erreur de reprojection e (en pixels) d'un point situé à une distance d donnée si l'erreur de localisation est de 10 *cm* perpendiculairement à l'axe optique. Le calcul a été fait pour la caméra que nous avons utilisée (images de 512×384 pixels et un angle de vue de 130° dans la largeur). On peut estimer que la position des

points d'intérêt dans l'image peut être connue au pixel près (ou un peu moins avec des techniques subpixelliques). Avec les conditions évoquées ci-dessus, cela correspond à l'erreur de reprojection d'un point situé à environ 30 mètres de la caméra.

Ce calcul simpliste ne prend pas en compte l'erreur sur la position du point 3D qui a elle-même été calculée par vision. On devrait aussi considérer le fait que plusieurs points sont utilisés pour calculer la pose de la caméra. Pour prendre en compte ces paramètres, il faut recourir aux calculs rigoureux sur les ellipsoïdes de confiance. Malgré tout, cet ordre de grandeur (précision de 10 *cm* lorsque les points observés sont à 30 *m*) a pu être approximativement vérifié lors des expérimentations.

Ce qu'il faut retenir de ce calcul c'est que la précision de la localisation dépend des points qui pourront être cartographiés et retrouvés par la suite. Du coup, il est bien difficile de donner des critères précis permettant de prévoir si le système de localisation va donner satisfaction dans un lieu donné. Pour cela, il faudrait évaluer la distance aux façades des bâtiments, savoir si des points d'intérêt pourront être détectés sur le sol, identifier les éléments permanents de la scène, prévoir quelles zones d'image risquent d'être masquées par des piétons ou des véhicules, etc ... Les résultats donnés dans le reste de ce chapitre ne peuvent pas être transposés directement à n'importe quel lieu. Mais nous nous sommes efforcés, dans la limite des contraintes matérielles, d'effectuer des expérimentations en des lieux aussi variés que possible pour évaluer les performances du système de localisation par vision.

5.4.2 Comparaison avec la vérité terrain

Géoréférencement de la cartographie 3D

Lorsqu'on souhaite comparer les résultats de localisation obtenus par vision aux données enregistrées par un récepteur GPS, il est nécessaire de calculer la pose du robot dans le même repère. Pour cela, on doit appliquer au modèle 3D de la scène obtenu à l'étape de reconstruction un changement de repère pour que le modèle soit exprimé dans le repère du GPS. C'est l'objectif de la méthode de recalage détaillée ici mais d'autres applications pourraient bénéficier du géoréférencement de la carte obtenue par vision. Même si cela n'a pas été traité dans cette thèse, on pourrait envisager de mettre en correspondance les cartes visuelles avec un plan de ville. Cela permettrait de sélectionner les données utiles pour réaliser un parcours d'une adresse à une autre.

Dans le cas où les signaux en provenance des satellites peuvent être reçus, la

position fournie par le récepteur GPS peut être considérée comme juste (avec un écart-type de 1 à 2 *cm* selon le nombre de satellites visibles) et servir de vérité terrain. Le récepteur GPS fournit la position du robot à une fréquence de 10 Hz.

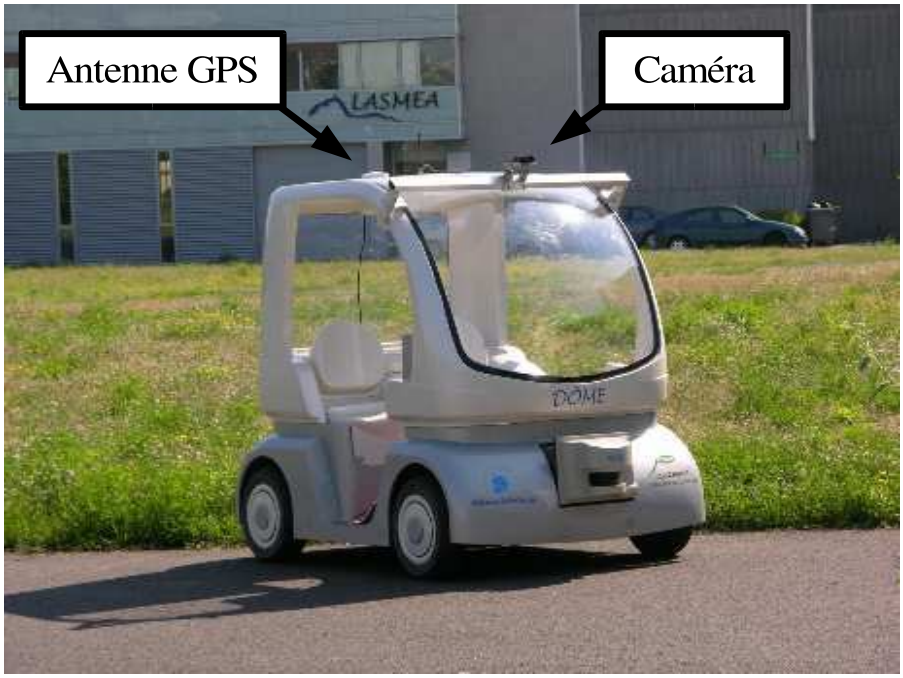

FIG. 5.5 – Position de la caméra et de l'antenne GPS sur le cycab

Le calcul du changement de repère entre la reconstruction 3D et les coordonnées GPS est un peu délicat car les deux capteurs ne sont pas placés au même endroit sur le cycab (voir figure 5.5). Ils n'enregistrent donc pas la même trajectoire. L'antenne GPS est située à l'aplomb du milieu de l'essieu arrière, ce qui correspond à l'origine du repère standard du cycab utilisé pour la commande. La caméra a été placée à l'avant pour que le toit du cycab ne vienne pas masquer une partie de la scène dans l'image. Le changement de repère entre repère caméra et repère robot a été mesuré grossièrement.

La figure 5.6 permet d'introduire les notations qui définissent les différents repères utilisés. Le repère terrestre noté $\mathbf{R}_T = (O_T, X_T, Y_T, Z_T)$ est un repère lié au sol. L'axe Y_T est vertical et dirigé vers le haut. Le récepteur GPS donne des positions exprimées dans ce repère. Le cycab évolue dans le repère $\mathbf{R}_T$, sa position est définie comme la position du point milieu de l'essieu arrière noté O_R. Un repère attaché au cycab permet de définir la position des capteurs sur le véhicule. Ce re-

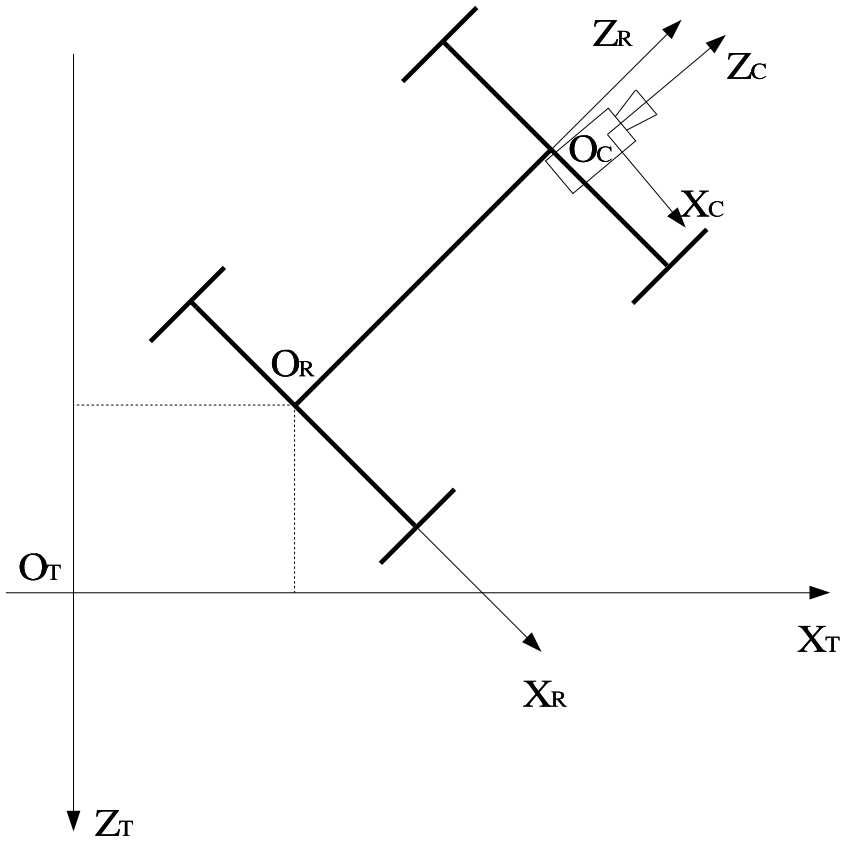

FIG. 5.6 – Repères Terrestre $\mathbf{R}_T$, Robot $\mathbf{R}_R$, Caméra $\mathbf{R}_C$

père est appelé repère robot et noté $\mathbf{R}_R = (O_R, X_R, Y_R, Z_R)$. Son origine est aussi le milieu de l'essieu arrière. L'axe Z_R est donné par la direction d'avancement du véhicule. L'antenne GPS est placée à la verticale de l'origine du repère robot. Le repère caméra $\mathbf{R}_C$ est défini de façon classique en vision. L'origine de $\mathbf{R}_C$ est au centre optique de la caméra O_C et il est orienté pour avoir Z_C selon l'axe optique de la caméra. Les axes X_C et Y_C correspondent aux axes de la matrice CCD. Le passage du repère caméra au repère robot est régi par l'équation suivante :

$$P_C = R_{RC}(P_R - T_{RC}) \tag{5.5}$$

avec P_C les coordonnées d'un point dans le repère caméra, P_R les coordonnées du même point dans le repère robot et (R_{RC},T_{RC}) la matrice de rotation et la translation qui permettent le changement de repère entre $\mathbf{R}_C$ et $\mathbf{R}_R$.

L'ensemble des points 3D reconstruits pendant l'étape de cartographie sont exprimés dans un repère arbitraire noté $\mathbf{R}_{recons}$. Le but du recalage est de trouver

la similitude qui permet de passer du repère $\mathbf{R}_{recons}$ au repère terrestre $\mathbf{R}_T$. Pour cela, on connaît :

- la trajectoire du point O_R dans le repère $\mathbf{R}_T$ qui est obtenue avec le GPS,
- la pose de la caméra dans le repère $\mathbf{R}_{recons}$ pour chaque image,
- la transformation rigide (R_{RC}, T_{RC}) entre le repère robot et le repère caméra.

La première opération consiste à retrouver l'orientation du robot à partir de l'information de position fournie par le GPS. La première idée pour cela consiste à utiliser les champs V_x et V_z renvoyés par le GPS qui indiquent la vitesse du véhicule dans un plan horizontal. Malheureusement ces données sont trop bruitées pour être utilisées directement. Nous avons préféré calculer la tangente à la trajectoire calculée à partir de la suite des positions enregistrées. Cela nous permet d'avoir la pose complète du robot toutes les 100 ms.

Ensuite, il faut passer de la pose du robot à la pose de la caméra, tout en restant dans le repère terrestre $\mathbf{R}_T$. Cela revient à générer la trajectoire qu'enregistrerait un récepteur GPS virtuel placé au même endroit que la caméra sur le cycab. Pour cela, on utilise (R_{RC}, T_{RC}).

A ce stade, on dispose de la trajectoire du même point du véhicule enregistré par les deux capteurs chacun dans leur repère propre. L'échantillonage des deux capteurs est différent (10 Hz pour le GPS, et 15 Hz pour la caméra), mais les données sont datées et une interpolation linéaire sur la trajectoire GPS est suffisante pour obtenir des données synchronisées. Le recalage entre les deux nuages de points est calculé par la méthode de Faugeras et Hébert [17]. La similitude ainsi obtenue est appliquée à l'ensemble de la reconstruction 3D (trajectoire de la caméra et nuage de points reconstruits). Dès lors, la reconstruction 3D et les positions provenant du GPS sont toutes exprimées dans le repère terrestre $\mathbf{R}_T$.

On peut noter que le recalage est simplement un changement de repère global sans déformation de la trajectoire ni de la structure 3D. Cette méthode a pu être appliquée à des reconstructions 3D de taille réduite : pas plus d'une centaine de mètres avec deux ou trois grands virages. Au delà, la reconstruction 3D produite par vision ne donne pas un résultat suffisamment fidèle à la réalité pour qu'un recalage global soit pertinent. Cela dit, la méthode de recalage est satisfaisante pour évaluer la qualité de la reconstruction 3D et de la localisation sur des séquences courtes.

5.4.3 Précision de la reconstruction

Le problème qu'on cherche à résoudre est celui de la localisation du robot. Notre but n'est pas d'obtenir la meilleure reconstruction 3D possible. En fait, nous pensons qu'une reconstruction 3D précise n'est pas absolument nécessaire

FIG. 5.7 – Quelques images de la séquence $CUST_1$

pour que le robot puisse naviguer de façon autonome, surtout si le robot reste sur la trajectoire ayant servi pour l'apprentissage. La carte construite doit être précise localement (sur une cinquantaine de mètres) mais une dérive à long terme est acceptable. Ce point sera précisé au moment d'aborder le cas des séquences en boucle.

Quoi qu'il en soit, il est tout de même intéressant d'examiner la précision des reconstructions obtenues et de s'assurer qu'elles sont précises localement. Les résultats présentés ici concernent uniquement la précision de reconstruction de la trajectoire de la caméra et non pas du nuage de points reconstruits.

Quatre séquences nommées $CUST_1$ à $CUST_4$ ont été enregistrées en conduisant manuellement le cycab sur une trajectoire de 80 mètres de long. Les quatre séquences ont été faites le même jour, à peu près sur la même trajectoire (avec un écart latéral ne dépassant pas 1 mètre). Pour ces séquences, la caméra utilisée avait un angle de vue réduit par rapport au matériel décrit au paragraphe 5.1.2 (environ 60°). Une reconstruction 3D a été calculée à partir de chacune des quatre séquences. Selon la séquence, l'algorithme de sélection des images clefs a donné entre 113 et 121 images clef. La reconstruction finale comporte de 14323 à 15689 points 3D. Quelques images extraites de la séquence $CUST_1$ sont visibles sur la figure 5.7. Les positions des images clef calculées pour cette séquence apparaissent en figure 5.8 en vue de dessus ainsi que la trajectoire enregistrée par le GPS centimétrique. L'erreur de reconstruction pour chacune des séquences est la distance moyenne entre la position du centre de chaque caméra et la position obtenue à partir de la mesure GPS. Pour ces quatre séquences, cette erreur vaut 25 *cm*, 40 *cm*, 34 *cm* et 24 *cm* pour une longueur totale de 80 *m* avec deux grands virages. Cette erreur est causée principalement par une dérive lente du processus de reconstruction.

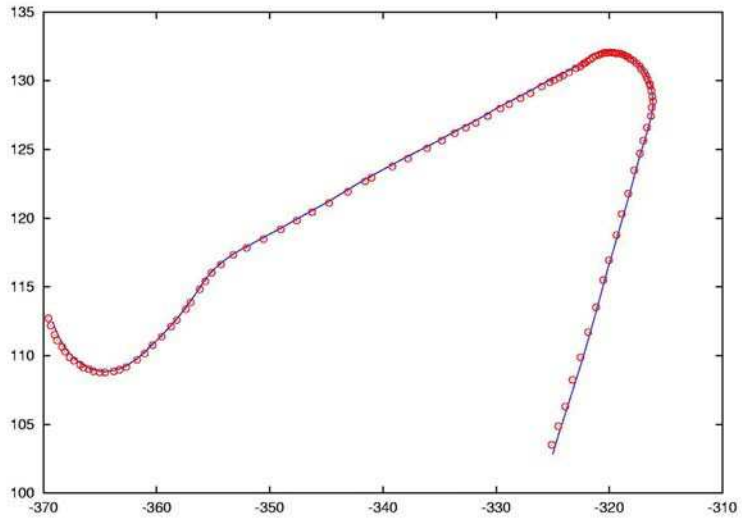

FIG. 5.8 – Position des images clef (cercles) superposés avec la trajectoire enregistrée par le GPS (ligne continue), unités en mètres.

5.4.4 Précision de la localisation

Position

La précision de localisation est calculée à partir des mêmes séquences que celles utilisées au paragraphe 5.4.3. Les quatre séquences ont été utilisées tour à tour comme séquence de référence. Douze séries de mesures ont été faites en calculant la position des images de la séquence $CUST_i$ en utilisant la séquence $CUST_j$ comme séquence de référence pour $i, j \in \{1,2,3,4\}$ et $i \neq j$.

L'erreur de localisation est définie différemment de l'erreur de reconstruction. Si on se contente de calculer l'écart entre la position donnée par l'algorithme de vision et celle donnée par le GPS, on calcule une erreur de localisation globale qui intègre à la fois les erreurs commises à la localisation mais aussi à la reconstruction. Et finalement, on retrouve approximativement la même erreur que celle de la reconstruction. Mais heureusement, dans un grand nombre d'applications, une position globale n'est pas nécessaire. C'est le cas en particulier si on veut faire naviguer un robot. Dans ce cas nous avons juste besoin de calculer l'écart latéral entre la position courante du robot et sa position lors de l'apprentissage de la trajectoire, ainsi que l'écart angulaire entre ces positions. Ce sont les deux données utilisées par la loi de commande. On peut définir l'erreur de localisation latérale comme l'erreur commise sur l'écart latéral à la trajectoire de référence car c'est

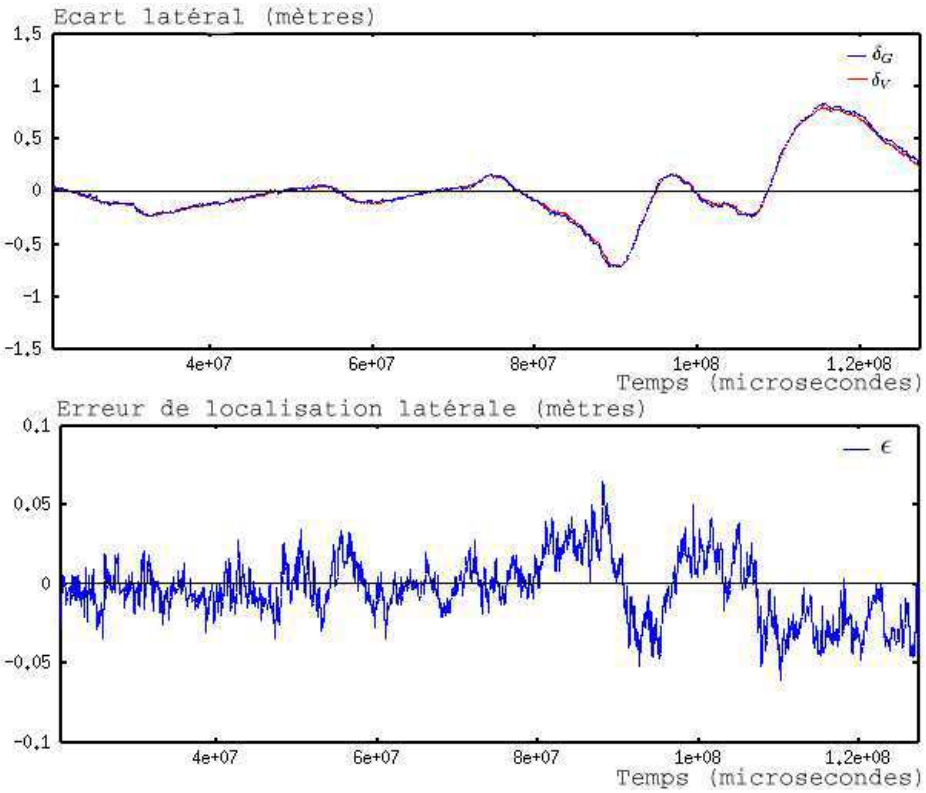

FIG. 5.9 – En haut : écart latéral à la trajectoire de référence mesuré avec le GPS y_g en bleu ou par vision y_v en rouge (les deux courbes sont quasiment confondues). En bas : erreur de localisation latérale $\varepsilon = y_v - y_g$

cette variable qui est utilisée ensuite par la loi de commande. L'écart latéral à la trajectoire de référence est calculé par vision pour obtenir y_v. Ce même écart est aussi calculé à partir des enregistrements GPS pour obtenir y_g. On note l'erreur de localisation latérale $\varepsilon = y_v - y_g$. De là, on peut calculer l'écart type de ε pour la totalité de la trajectoire : on appelle ceci l'erreur de localisation latérale moyenne.

Nous avons calculé l'erreur de localisation latérale moyenne pour chacune des 12 expériences : l'erreur varie de 1,4 *cm* à 2,2 *cm*, avec une moyenne de 1,9 *cm*. Il faut noter que l'erreur obtenue ici est du même ordre de grandeur que la précision donnée pour le récepteur GPS par le fabricant (écart type de 1 *cm*). Cette erreur intègre donc pour partie le bruit de mesure du récepteur GPS. La figure 5.9 montre l'écart latéral et l'erreur de localisation latérale calculés pour une des expériences dont l'erreur de localisation latérale moyenne était de 1,9 *cm*, l'erreur maximale atteint environ 6 *cm*. Les valeurs données ici sont le résultat

brut issu de l'algorithme de localisation, le filtre de Kalman fondé sur un modèle de mouvement du cycab n'a pas été utilisé.

Cas d'une séquence en boucle

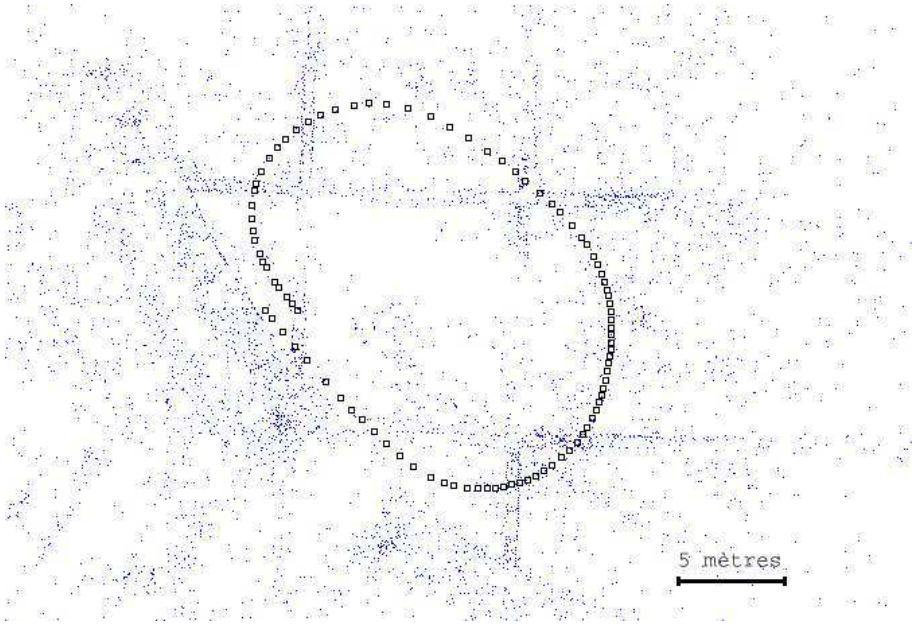

FIG. 5.10 – Reconstruction 3D d'une séquence en boucle

Le cas d'une séquence en boucle est très intéressant car c'est un bon moyen de visualiser l'influence de l'erreur de reconstruction sur la navigation du robot. Bien sûr, il serait possible de traiter le cas d'une boucle de façon spécifique en recherchant des points en correspondance entre la dernière et la première image d'une séquence. Mais ce n'est pas l'objectif de ce paragraphe. Nous voulons utiliser le cas d'une boucle pour montrer ce qui se passe en présence d'une erreur de reconstruction.

Un exemple de trajectoire en boucle est visible sur la figure 5.10 ainsi que quelques images extraites de la vidéo (figure 5.11). Les positions de la première et de la dernière caméra devraient être confondues. Mais à cause d'une dérive, la reconstruction obtenue ne se referme pas vraiment (l'écart mesure environ 1.5 m pour une longueur totale de 51 m). Dans ce cas, certains points 3D physiquement identiques sont reconstruits à deux positions différentes : une position lorsque le point est vu au début de la séquence et une autre fois lorsqu'il est vu à la fin de

FIG. 5.11 – Quelques images extraites de la séquence en boucle

la séquence. Malgré tout, il a été possible de faire naviguer un robot sur cette boucle pendant plusieurs tours sans le moindre à-coup visible dans sa trajectoire. La figure 5.12 permet de mieux comprendre pourquoi. La loi de commande utilisée [82] nécessite en entrée l'écart latéral par rapport à la trajectoire de référence (y_v) ainsi que l'écart angulaire par rapport à cette trajectoire. Or la localisation est faite par rapport au nuage de points visibles à un moment donné par le robot (les points qui sont cohérents par rapport à une seule image clef). La localisation absolue n'est pas utilisée pour le calcul des écarts latéral et angulaire. C'est la localisation dans un repère local lié à la position de l'image clef utilisée qui importe. Sur la figure 5.12, ce repère est représenté pour le début de la boucle ($I_1, \vec{T}_1, \vec{N}_1$) et pour la fin de la boucle ($I_N, \vec{T}_N, \vec{N}_N$), les vecteurs $\vec{T}$ et $\vec{N}$ étant respectivement la tangente et la normale à la trajectoire de référence. Lorsqu'on passe de la fin d'une boucle au début d'une nouvelle, cela correspond à un changement de ce repère local qui concerne la trajectoire de référence, la position courante et le nuage de points visibles, et dans ce cas l'écart latéral à la trajectoire de référence y_v ne présente pas de discontinuité et l'écart angulaire non plus. Ceci est vrai à condition que le nuage de points soit reconstruit de manière identique au début et à la fin de la boucle, ce qui est quasiment le cas en pratique. Cette propriété est très intéressante parce que cela permet d'envisager des parcours de plusieurs kilomètres de long sans devoir faire une reconstruction 3D très coûteuse en temps de calcul et en mémoire vive. Il suffit d'enchaîner des morceaux de trajectoire d'une centaine de mètres. Cela devrait permettre de réaliser des parcours complexes en chaînant à la demande des morceaux de trajectoire déjà enregistrés. En stockant séparément

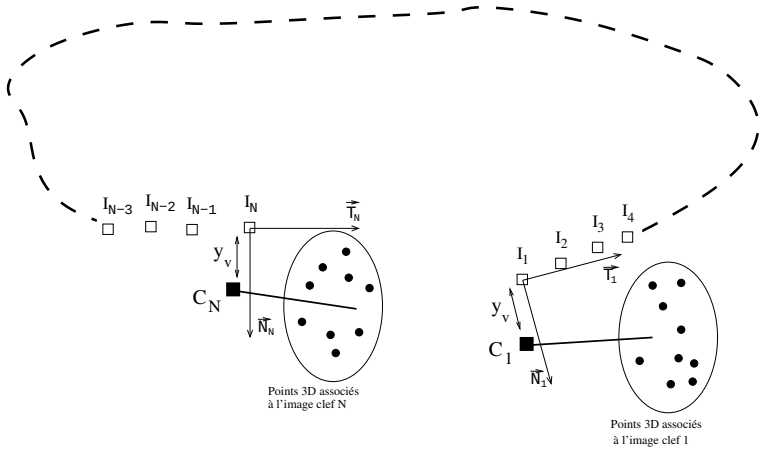

FIG. 5.12 – Localisation par rapport au début ou à la fin d'une boucle. I_i représente la position de l'image clef i. C_1 et C_N désignent les positions de caméra obtenues en se localisant respectivement par rapport à la première et à la dernière image clef.

chaque rue d'une ville, on pourrait par exemple demander au véhicule de suivre la rue de la République puis suivre un bout de parcours correspondant à un carrefour et d'enchaîner sur l'avenue de la Libération.

On peut légitimement se demander s'il est nécessaire de chercher à fermer la boucle, c'est-à-dire d'avoir un algorithme de reconstruction capable de reconnaître une position déjà atteinte au cours de la trajectoire. C'est un problème qu'il est important de traiter lorsque la mission du robot est l'exploration d'une zone. Mais pour l'application que nous visons (faire naviguer un robot sur une trajectoire préalablement apprise), l'expérience nous a montré que cela n'est pas nécessaire. Fermer la boucle pourrait apporter quelques avantages comme rajouter des contraintes supplémentaires dans l'ajustement de faisceaux ou éviter de reconstruire en double certains points. Cela pourrait être envisagé à condition de gérer les incertitudes tout au long de la reconstruction mais il n'est pas certain qu'un algorithme automatique puisse retrouver de façon efficace une position déjà atteinte si la boucle se referme après un parcours de plusieurs dizaines ou centaines de mètres.

Orientation

Pour mesurer la précision de l'orientation fournie par l'algorithme de localisation, nous avons placé la caméra sur une plate-forme rotative qui peut être orientée

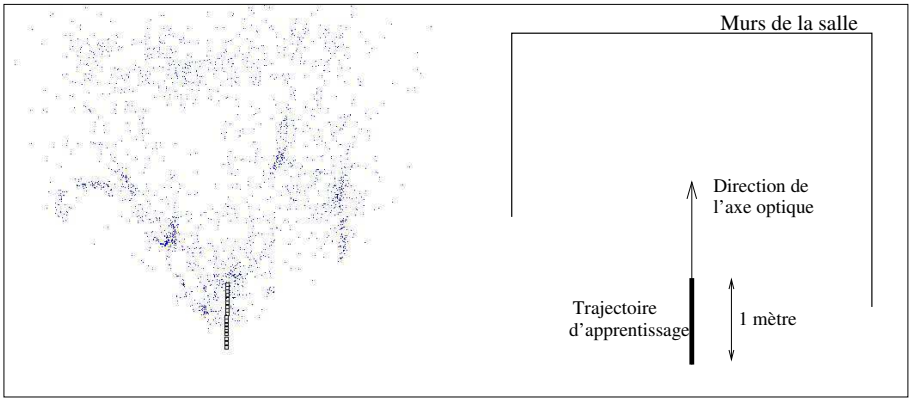

FIG. 5.13 – Reconstruction 3D utilisée pour évaluer la précision de l'orientation

précisément avec une graduation au degré près (l'erreur sur la lecture de la graduation est de l'ordre de $\pm 0.1^\circ$). La séquence d'apprentissage était une séquence composée de 18 images clef enregistrées sur une trajectoire en ligne droite de 1 *m* de long orientée dans le même sens que l'axe optique de la caméra (ce qui correspond à l'orientation 0°). La reconstruction de la trajectoire ainsi que le nuage de points associé est visible en vue de dessus sur la figure 5.13. L'image présentée en figure 5.14 permet de mieux se rendre compte du champ vu par la caméra. L'objectif utilisé donne un champ de 130° environ dans la largeur de l'image.

Dans la phase de localisation, pour chaque orientation de la caméra de $\alpha_0 = -94^\circ$ à $\alpha_0 = +94^\circ$ par incrément de 2°, nous avons noté l'angle α_0 mesuré sur la graduation de la plate-forme et l'angle α fourni par l'algorithme de localisation. Au delà de 95°, la zone d'image commune entre l'image courante et l'image de référence devient très petite et on n'est plus certain de trouver un nombre de points en correspondance suffisant pour continuer à se localiser. L'erreur de localisation calculée à partir de ces données apparaît sur la figure 5.15. Des images capturées pour des orientations variant de -90° à $+90^\circ$ apparaissent en figure 5.16.

L'algorithme de localisation est capable de fournir une orientation avec une précision de l'ordre du dixième de degré même si on s'écarte de 90° d'un côté ou de l'autre par rapport à l'axe d'apprentissage. Pour quantifier plus précisément l'erreur, il faudrait disposer d'un moyen de mesure plus précis de l'orientation de la plate-forme. La lecture de la graduation ne donne pas une précision meilleure que l'algorithme de localisation.

FIG. 5.14 – Une image prise dans l'axe de la trajectoire

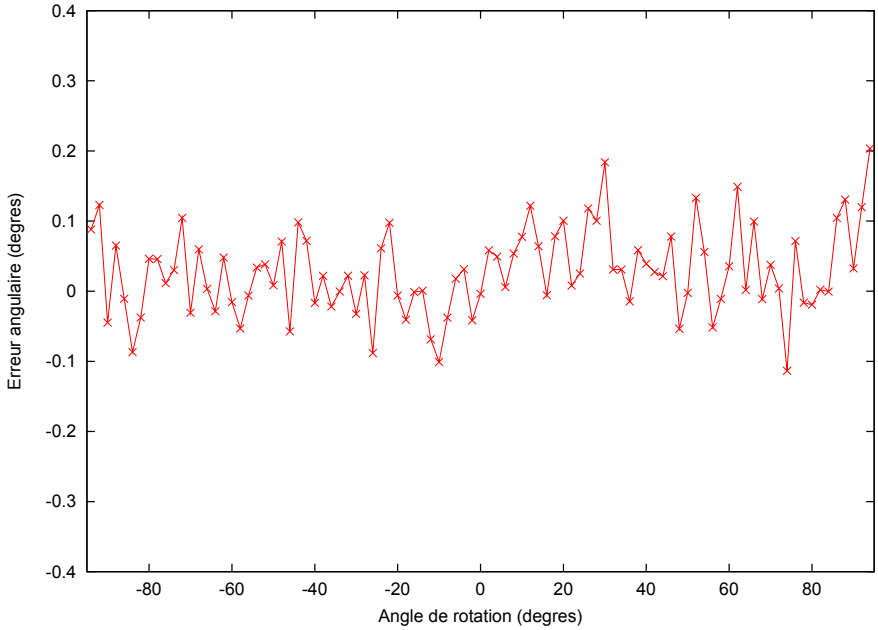

FIG. 5.15 – Erreur angulaire $|\alpha_0| - |\alpha|$ en fonction de α_0

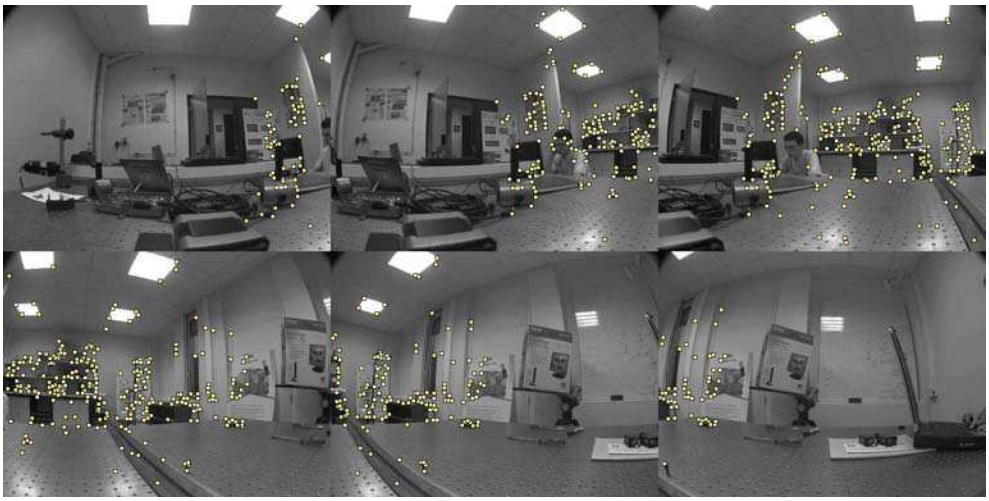

FIG. 5.16 – Images prises pour différentes valeurs de α_0 avec les points d'intérêt correctement appariés en surimpression. De gauche à droite et de haut en bas : $-90°$, $-60°$, $-30°$, $30°$, $60°$ et $90°$.

5.4.5 Navigation autonome du cycab

Pour compléter les mesures de précision effectuées, nous avons aussi voulu évaluer les performances globales du système en mesurant avec quelle précision le cycab suit la trajectoire de référence. Pour cela, la pose de la caméra est calculée par l'algorithme de localisation par vision et la loi de commande présentée au paragraphe 5.3.2 est utilisée pour contrôler le robot. En même temps le récepteur GPS est utilisé pour enregistrer la position du robot avec une précision centimétrique.

La trajectoire d'apprentissage mesurait 127 mètres de long et comportait des sections de ligne droite et des virages serrés. La vitesse du véhicule était fixée à 2 km/h et constante sur tout le parcours. La reconstruction 3D obtenue à partir de la séquence de référence est visible sur la figure 5.18 en vue de dessus. Cette séquence comporte 182 images clef et 16295 points 3D ont été reconstruits.

FIG. 5.17 – Trois images prises pendant la navigation autonome du cycab (en bas) et les images clef correspondantes (en haut). Les points d'intérêt utilisés pour le calcul de pose sont en surimpression.

La séquence vidéo de référence a été enregistrée par temps couvert. Deux expériences de navigation autonome ont été réalisées quelques jours plus tard, toujours par temps couvert. Puis deux nouvelles expériences ont été faites par temps ensoleillé. Pour ces expériences, le soleil était bas dans le ciel et il était parfois dans le champ de la caméra. La figure 5.17 montre quelques images enregistrées pendant la dernière expérience de navigation autonome ainsi que les images clef correspondantes et les points d'intérêt appariés en surimpression. La

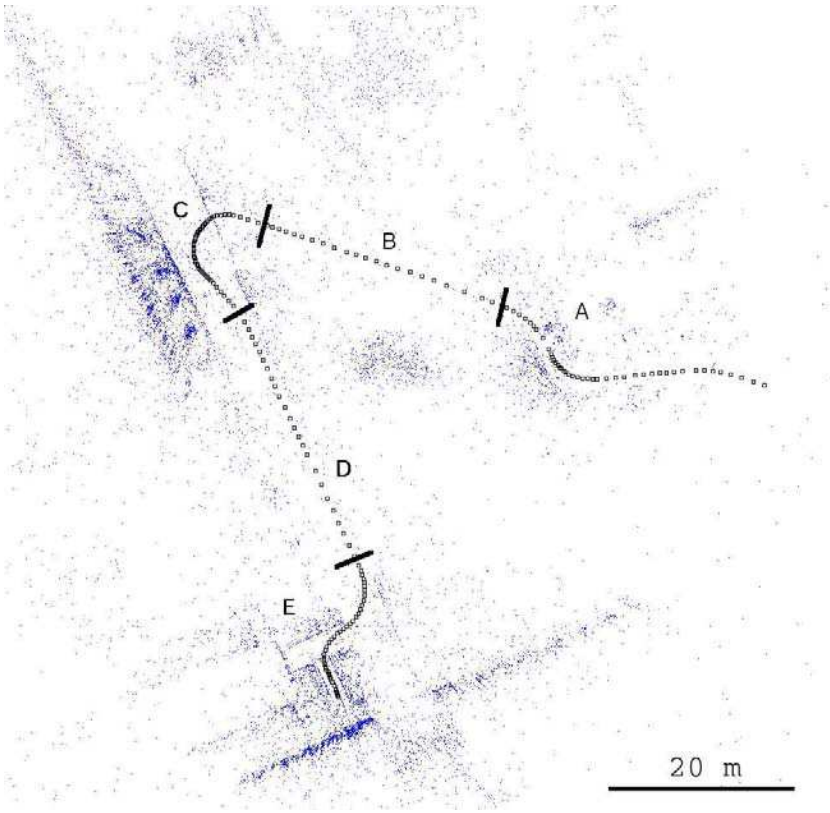

FIG. 5.18 – Reconstruction 3D calculée à partir de la séquence de référence (en vue de dessus). Les lettres indiquent les différentes parties de la trajectoire.

dernière image souligne l'intérêt d'utiliser une caméra équipée d'un objectif à très grand angle et des points d'intérêt répartis sur toute la surface de l'image. Lorsque le soleil est dans le champ de la caméra, le capteur est saturé ce qui rend inutilisable une partie de l'image. Il est tout de même possible d'utiliser cette image pour calculer la pose de la caméra, car le bâtiment à gauche est toujours visible.

Pour comparer la précision de la localisation par vision et la localisation par GPS centimétrique dans une application de navigation autonome, une cinquième expérience a été réalisée. Pour cette dernière expérience, c'est la position indiquée par le GPS (et non pas la position calculée par vision) qui est utilisée pour calculer le vecteur d'état du robot qui est utilisé dans la loi de commande. L'écart latéral du robot par rapport à la trajectoire de référence pour trois de ces expériences apparaît sur la figure 5.19. Que le véhicule soit guidé par vision ou par GPS, c'est à chaque fois l'écart latéral mesuré avec le GPS qui est représenté. Les lettres permettent de retrouver chaque portion de la trajectoire en se référant à la

figure 5.18. Les courbes correspondant aux deux autres expériences (une obtenue par temps nuageux et une par temps ensoleillé) sont quasiment identiques aux courbes représentées.

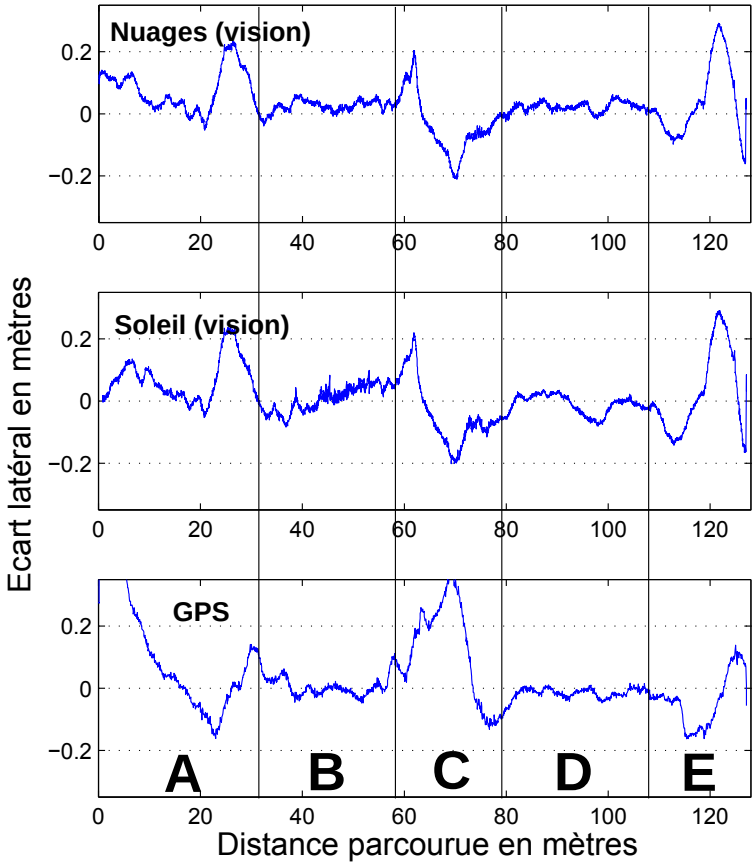

FIG. 5.19 – Ecart latéral à la trajectoire de référence mesuré avec le GPS.

On peut voir que les performances en guidage sont approximativement les mêmes quelque soit le capteur utilisé. De plus la trajectoire suivie est très semblable d'une expérience à l'autre lorsque le véhicule est guidé par vision. On peut également noter que des conditions d'éclairage difficiles ne provoquent pas une dégradation significative des performances. Ceci doit être nuancé par le fait que le cycab se comporte comme un filtre et qu'un bruit dans le résultat de la localisation ne se traduit pas forcément par une erreur de position importante du cycab. La robustesse aux changements de conditions lumineuses sera reprise au paragraphe 5.5. Les performances en ligne droite et dans les courbes sont analysées séparément dans les tableaux 5.2 et 5.3. Le tableau 5.2 donne la valeur moyenne

	Soleil 1	Soleil 2	Nuages 1	Nuages 2	GPS
B	3.5 cm	4.8 cm	3.4 cm	2.8 cm	2.7 cm
D	2.4 cm	1.9 cm	1.8 cm	2.3 cm	1.8 cm

TAB. 5.2 – Moyenne de l'écart latéral en ligne droite mesuré avec le GPS.

	Soleil 1	Soleil 2	Nuages 1	Nuages 2	GPS
C max	22.0cm	26.8cm	20.1cm	20.4cm	37.9cm
C min	-20.2cm	-25.4cm	-22.2cm	-21.1cm	-14.3cm
E max	29.1cm	35.4cm	30.0cm	29.2cm	13.9cm
E min	-16.5cm	-19.7cm	-16.5cm	-16.1cm	-16.3cm

TAB. 5.3 – Maximum et minimum de l'écart latéral dans les courbes mesuré avec le GPS.

de l'écart latéral ($|y|$) dans les lignes droites (parties B et D). Dans la situation la plus favorable (temps nuageux), la localisation par vision donne les mêmes performances que la localisation par GPS différentiel. Dans le cas le plus défavorable (temps ensoleillé), les performances sont légèrement dégradées, mais restent satisfaisantes. Le tableau 5.3 donne les valeurs extrêmes atteintes par y dans les virages (parties C et E). Une fois de plus, les performances de guidage sont similaires.

5.4.6 Navigation sur une trajectoire différente de la trajectoire d'apprentissage

Expérience réalisée

D'un point de vue pratique, il peut être intéressant de s'écarter de la trajectoire qui a été apprise. Par exemple, si un obstacle est présent sur la trajectoire, il faut pouvoir le contourner. Cela est possible car on dispose d'un modèle 3D de l'environnement. En utilisant les coordonnées 3D des points observés, il est possible de calculer la position du robot même si celui-ci n'est pas exactement sur la trajectoire apprise. L'expérience décrite ici a pour but de déterminer jusqu'à quel point il est possible de s'éloigner de la trajectoire d'apprentissage.

Dans un premier temps, une séquence vidéo a été enregistrée sur une trajectoire de 70 mètres de long. Une reconstruction 3D a été calculée à partir de cette séquence vidéo. La figure 5.20 montre quelques images extraites de la vidéo d'apprentissage. La reconstruction compte 102 images clef et 9579 points 3D. Nous

FIG. 5.20 – Images extraites de la vidéo d'apprentissage

avons ensuite défini une nouvelle trajectoire (appelée trajectoire cible), légèrement différente de la trajectoire d'apprentissage. La consigne donnée au robot était de suivre la trajectoire cible en utilisant pour se localiser la trajectoire et la vidéo d'apprentissage. La navigation autonome a été faite deux semaines après l'apprentissage. La figure 5.21 montre la reconstruction 3D de la séquence d'apprentissage et la trajectoire cible définie à la souris dans un éditeur graphique. La trajectoire cible a été définie pour correspondre à ce que le robot devrait faire pour éviter deux obstacles en faisant d'abord un écart vers la droite, puis un peu plus loin en s'écartant sur la gauche. L'écart latéral maximum par rapport à la trajectoire d'apprentissage est de 3 m. L'écart angulaire atteint au maximum 20°. Un véhicule a été garé sur la trajectoire d'apprentissage pour simuler l'occultation provoquée par l'obstacle (voir figure 5.22). D'autres occultations ont été provoquées par des piétons qui passaient à ce moment là.

La trajectoire cible a simplement été définie à la souris en déformant la trajectoire d'apprentissage. La définition d'une nouvelle trajectoire ne demande pas de calcul important et pourrait être fait en ligne si nécessaire. Par exemple, nous pourrions utiliser un capteur (radar, sonar ou télémètre) pour détecter les obstacles devant le robot. Dans le cas où un obstacle est détecté et localisé par ce capteur dans le repère robot, il serait possible de redéfinir une nouvelle trajectoire dans le repère terrestre sans stopper le véhicule.

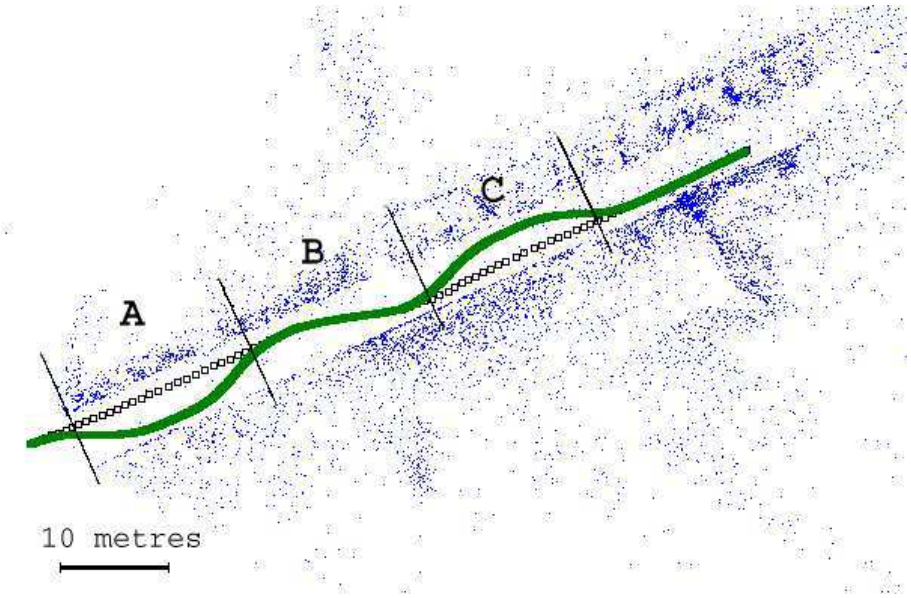

FIG. 5.21 – Reconstruction 3D de la trajectoire d'apprentissage (carrés noirs) et nouvelle définition de la trajectoire à suivre (ligne continue épaisse) en vue de dessus.

Précision de la localisation

Nous allons examiner la trajectoire suivie par le robot pendant la navigation autonome. Dans la suite on l'appellera trajectoire résultat. On dispose de l'enregistrement GPS correspondant à la trajectoire d'apprentissage ainsi qu'à la trajectoire résultat. A peu de choses près, le robot a suivi la trajectoire cible. On peut le voir sur la figure 5.23. La courbe noire correspond à l'écart latéral de la trajectoire cible par rapport à la trajectoire d'apprentissage. La courbe grise correspond à l'écart latéral entre la trajectoire résultat et la trajectoire d'apprentissage. La trajectoire résultat dévie légèrement de la trajectoire cible dans les virages qui ont été redéfinis. Cela s'explique par le fait que la trajectoire cible a été définie à la souris sans prendre en compte la contrainte sur l'angle de braquage maximum admissible pour le robot.

L'écart latéral à la trajectoire d'apprentissage calculé par l'algorithme de localisation par vision apparaît sur la figure 5.24. Cette courbe est un résultat brut sans aucun filtrage. Le résultat de la localisation est plus bruité lorsque le robot s'écarte de la trajectoire d'apprentissage, mais ce résultat permet toujours de commander le robot. Simultanément, la vérité terrain a été enregistrée grâce au GPS ce qui permet de mesurer l'erreur commise par l'algorithme de localisation par vision.

FIG. 5.22 – Points d'intérêt correctement appariés entre l'image de référence et l'image courante. A gauche : quand le robot est proche de la trajectoire d'apprentissage. A droite : quand l'écart latéral par rapport à la trajectoire d'apprentissage est maximal (3 *m*). L'image clef est en haut.

La figure 5.25 montre l'erreur commise sur l'écart latéral. Quand le robot est sur la trajectoire d'apprentissage, l'erreur est inférieure à 2 *cm* (précision identique au GPS). Lorsque le robot est plus loin de la trajectoire d'apprentissage, cette erreur augmente et dépasse 10 *cm*. Cela s'explique par le nombre de points correctement appariés entre l'image courante et l'image de référence. La figure 5.22 montre les points appariés pour deux positions du robot. On peut constater que l'algorithme apparie moins de points lorsque le robot est loin de la trajectoire d'apprentissage. On peut également le voir sur la figure 5.26 qui montre le nombre de points utilisés dans le calcul de pose pour chaque image de la phase de navigation autonome. Lorsque le robot est sur la trajectoire d'apprentissage, plus de 200 points sont appariés. Ce nombre descend à 100 lorsque le robot est à 3 *m* sur le côté. On peut aussi remarquer que les points manquants sont en majorité des points proches de la caméra et ce sont ces points qui sont les plus intéressants pour calculer la position du robot. Le tableau 5.4 montre comment la précision de la localisation est reliée au nombre de points appariés. Le graphique de la figure 5.25 a été divisé en

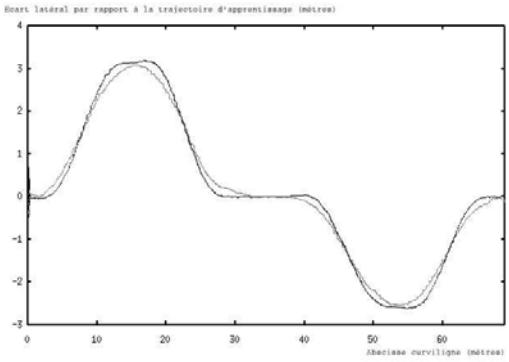

FIG. 5.23 – Ecart latéral de la trajectoire cible par rapport à la trajectoire d'apprentissage (noir), et écart latéral mesuré par le GPS pendant la phase de navigation autonome (gris)

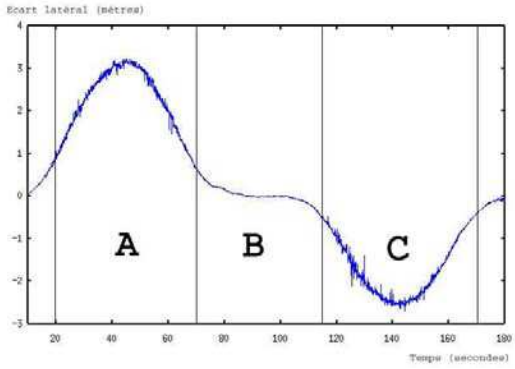

FIG. 5.24 – Ecart latéral calculé par vision

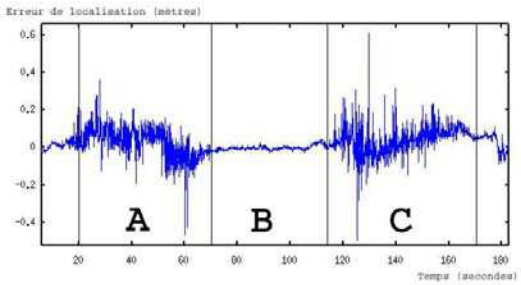

FIG. 5.25 – Erreur de localisation latérale de l'algorithme de vision

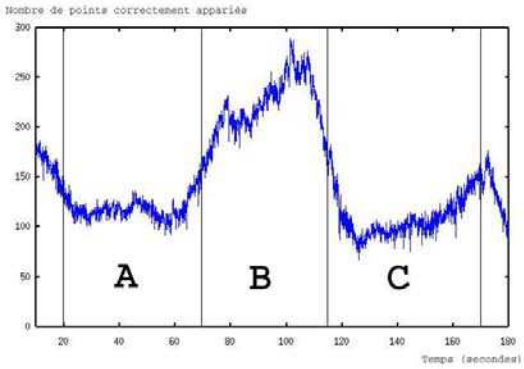

FIG. 5.26 – Nombre de points d'intérêt utilisés dans le calcul de pose

trois parties. Les zones A et C correspondent aux parties dans lesquelles le robot est éloigné de la trajectoire d'apprentissage. La zone B correspond à une partie où le robot est sur la trajectoire d'apprentissage. Pour chaque zone, on a calculé le nombre de points appariés (et retenus lors du calcul de pose) ainsi que l'erreur moyenne commise sur l'écart latéral calculé par vision. Le calcul de l'erreur de localisation sur la fin de la trajectoire n'a pas été possible car sur cette partie du trajet la précision de la localisation par GPS a fortement diminué à cause d'un masquage de certains satellites par des bâtiments proches.

Zone	A	B	C
Limites temporelles	20 s - 70 s	70 s - 115 s	115 s - 170 s
Nombre moyen de points appariés	119	220	110
Erreur de localisation	8 cm	1 cm	8 cm

TAB. 5.4 – Nombre de points appariés et erreur de localisation

Le robot peut s'écarter de la trajectoire d'apprentissage jusqu'à une certaine limite qui dépend du nombre de points qui peuvent être appariés entre l'image courante et l'image de référence la plus proche. Des points sont perdus car ils sont observés depuis un point de vue différent de celui sous lequel ils ont été appris. Dans ce cas, le score de corrélation obtenu est trop faible pour retenir l'appariement. C'est ce qui se produit quand l'écart latéral est important. Cela pourrait être compensé en utilisant une méthode d'appariement plus robuste aux changements de points de vue. Lorsque c'est l'écart angulaire qui augmente, des points sont perdus car ils sortent du champ de la caméra et la seule solution envisageable est d'utiliser une caméra avec un champ de vue plus grand ou une caméra omnidirectionnelle. L'autre solution qui pourrait être utilisée pour pouvoir s'écarter de la trajectoire apprise consiste à fusionner plusieurs séquences vidéo d'apprentissage enregistrées sur des trajectoires légèrement différentes. On pourrait par exemple prendre trois vidéos d'apprentissage : une sur le côté gauche de la route, une sur la droite et une au milieu.

Validation expérimentale du calcul des ellipsoïdes de confiance

Le but de cette étude est de vérifier que la taille des ellipsoïdes de confiance calculés varie de la même façon que l'erreur de localisation mesurée avec le GPS. Les données expérimentales enregistrées en faisant naviguer le robot loin de la trajectoire d'apprentissage nous donnent l'opportunité de faire cette vérification.

Dans les expériences réalisées précédemment, les ellipsoïdes étaient trop petits par rapport à la précision du GPS pour obtenir des résultats significatifs.

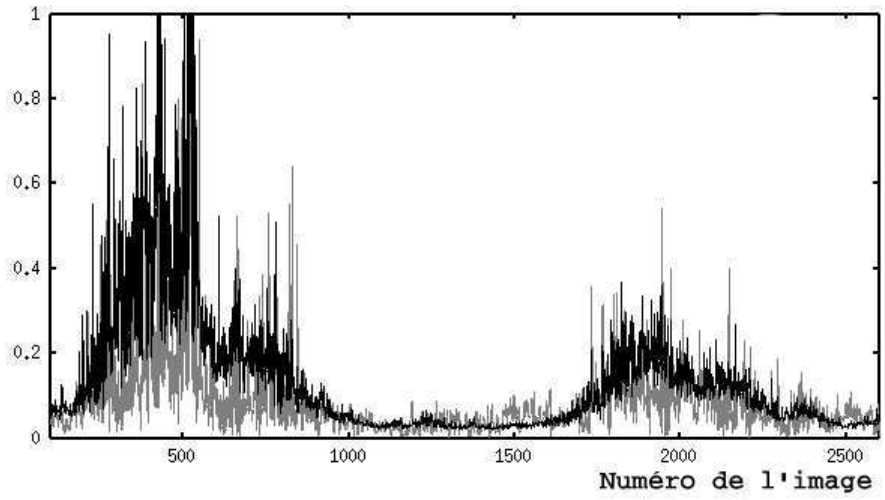

FIG. 5.27 – Comparaison entre la longueur du grand axe de l'ellipsoïde à 90% calculé par l'algorithme de localisation par vision (noir) et erreur de localisation mesurée grâce au GPS (gris). L'échelle sur l'axe des ordonnées est le mètre.

Pour chaque image prise pendant la phase de navigation autonome, nous avons calculé l'ellipsoïde de confiance à 90% comme expliqué au paragraphe 4.5. L'erreur de localisation a été calculée en utilisant les mesures GPS de façon légèrement différente par rapport au calcul fait à la section 5.4.4. Au paragraphe 5.4.4, les mesures provenant des deux capteurs étaient d'abord ramenées au même repère et la position du point milieu de l'essieu arrière du cycab était calculée. Puis l'erreur de localisation était mesurée selon la normale à la trajectoire. Ici, la première opération pour calculer la position du cycab dans le même repère est inchangée, mais l'erreur de localisation est la distance entre la position calculée par vision et la position donnée par le GPS. L'erreur inclut une composante d'erreur selon la normale à la trajectoire et une composante selon la tangente. La figure 5.27 montre l'erreur de localisation ε comparée à la longueur a du demi grand axe de l'ellipsoide de confiance à 90% calculé avec la méthode 2 du paragraphe 4.5.3, page 74. La longueur du demi grand axe de l'ellipsoïde calculé avec la méthode 2 est du même ordre que l'erreur de localisation et les deux quantités varient de la même façon.

On peut comparer les trois méthodes possibles pour calculer l'incertitude de localisation (la méthode 1 n'est pas considérée ici car inapplicable en temps réel).

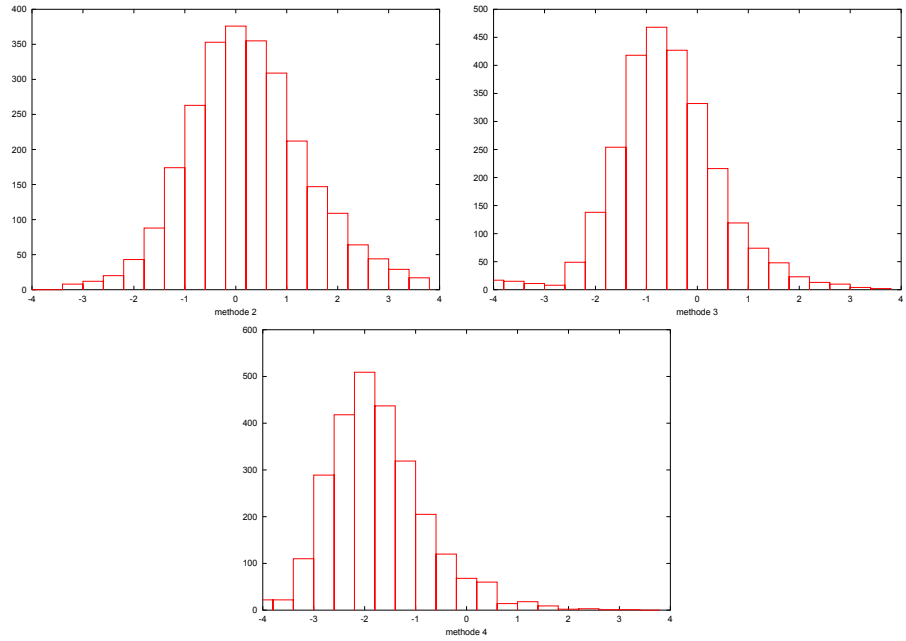

FIG. 5.28 – Histogramme de $log_2(\frac{a}{\varepsilon})$ pour les méthodes 2,3 et 4.

La figure 5.28 montre pour chaque méthode l'histogramme de $log_2(\frac{a}{\varepsilon})$ pour chaque image de la séquence vidéo (avec a la longueur du demi grand axe de l'ellipsoïde à 90% et ε l'erreur de localisation calculée à partir de la mesure GPS). Si on fait l'hypothèse de l'incertitude nulle sur les caméras reconstruites (méthode 3), la longueur du demi grand axe est à peu près la moitié de l'erreur mesurée avec le GPS. Si on néglige complètement l'incertitude sur les points 3D reconstruits (méthode 4), on sous-estime la taille des ellipsoïdes d'un facteur quatre.

5.5 Robustesse

On peut facilement définir et quantifier la précision d'une méthode de localisation. Il n'est pas non plus très difficile de mesurer cette précision lorsqu'on peut enregistrer la vérité terrain. Il est par contre plus difficile de quantifier et de rendre compte précisément de la robustesse d'un algorithme. De nombreuses causes peuvent affecter la localisation : occultations, changements de conditions d'éclairage, saturation du capteur, modifications dans la scène entre phase d'apprentissage et phase de localisation, ... Toutes les perturbations qui viennent d'être énumérées sont souvent présentes à des degrés divers et leur combinaison peut

éventuellement faire échouer la localisation. On peut définir quelques procédures de test pour évaluer la robustesse à certaines perturbations. Nous l'avons fait par exemple dans le cas des occultations. Mais c'est surtout en faisant de nombreuses expérimentations qu'il est possible de se faire une idée des conditions dans lesquelles le procédé va fonctionner correctement ou pas. Les exemples présentés dans ce paragraphe ont pour but de montrer que l'algorithme de localisation a été utilisé avec succès dans des lieux et des conditions d'éclairage variés. Des situations où des difficultés dans la localisation ont été constatées seront aussi évoquées afin d'en préciser les raisons et d'envisager des solutions pour améliorer les algorithmes développés.

5.5.1 Expérimentations en intérieur

Nous avons réalisé deux séries d'expérimentation en intérieur pour évaluer la robustesse de l'algorithme aux occultations et aux modifications dans la scène. Chaque fois, la procédure est la même. Une reconstruction 3D est calculée à partir d'une séquence d'apprentissage enregistrée sur une trajectoire de 1 *m* de long sur une ligne droite dirigée selon l'axe optique de la caméra (comme dans la mesure de la précision de l'orientation au paragraphe 5.4.4, page 98). Puis nous avons mesuré l'erreur de position de la caméra en ajoutant de plus en plus de perturbations (occultations, déplacement d'objets, changement de conditions d'éclairage). Pour cela la caméra était montée sur un rail gradué permettant de mesurer sa position avec une précision de l'ordre du millimètre.

Robustesse aux occultations

Pour cette expérience, la perturbation consistait à occulter des parties de la scène en faisant venir des personnes une par une devant la caméra. La procédure a été faite deux fois. La première fois la caméra était placée sur la trajectoire d'apprentissage. La deuxième fois, la caméra était placée à 1 *m* sur le côté de la trajectoire d'apprentissage. Pour chacune de ces deux expériences, les images et les points appariés sont visibles sur les figures 5.29 et 5.30. L'erreur de position obtenue pour chacune de ces images est donnée dans le tableau 5.5.

Si la caméra reste sur la trajectoire d'apprentissage, la précision de la localisation ne décroît pas, même en cas d'occultation importante. A une distance d'un mètre de la trajectoire d'apprentissage, une occultation modérée (trois ou quatre personnes) permet de garder une précision de localisation acceptable. Au delà, la précision décroît de façon importante. Cela s'explique par l'absence de points utilisables au premier plan lorsque la caméra est loin de l'axe d'apprentissage. De

Nombre de personnes	0	1	2	3	4	5	6
Erreur de position (mm) sur la trajectoire d'apprentissage	2	1	1	1	1	1	2
Erreur de position (mm) à 1 *m* de la trajectoire d'apprentissage	8	11	4	11	20	44	132

TAB. 5.5 – Erreur de localisation pour une occultation par un nombre croissant de personnes

plus, les points reconstruits ont un ellipsoïde de confiance allongé selon l'axe de déplacement de la caméra au moment de l'apprentissage. Lorsqu'on s'éloigne de cet axe, l'incertitude résultante sur la pose calculée est plus importante et il faut donc un plus grand nombre de points pour conserver le même niveau de précision.

Robustesse aux changements dans la scène

La deuxième expérience reprend le même principe que la première, mais cette fois-ci les perturbations sont plus variées : on commence par déplacer des objets dans la scène, puis on modifie les conditions d'éclairage et on ajoute des occultations. Les images des figures 5.31 et 5.32 permettent de se rendre compte des modifications qui ont été faites. Là encore, la procédure a été répétée deux fois en plaçant la caméra d'abord sur la trajectoire d'apprentissage puis à 1 *m* de celle-ci. On a procédé à 8 étapes de modification de la scène. L'erreur de localisation obtenue à chaque étape est donnée dans le tableau 5.6.

Etape de modification	1	2	3	4	5	6	7	8
Erreur de position (mm) sur la trajectoire d'apprentissage	1	1	2	0	2	5	2	5
erreur de position (mm) à 1 *m* de la trajectoire d'apprentissage	29	16	18	24	51	100	21	183

TAB. 5.6 – Erreur de localisation liée à la modification de la scène

Les résultats observés ici sont similaires à ceux de l'expérience précédente : lorsque la caméra est sur la trajectoire d'apprentissage, l'algorithme de localisation tolère des perturbations importantes. Quelques points correctement appariés suffisent à localiser la caméra de façon précise. Lorsque la caméra est loin de la trajectoire d'apprentissage, l'incertitude sur la position des points 3D se fait plus

FIG. 5.29 – Evaluation de la robustesse aux occultations (caméra sur la trajectoire d'apprentissage). L'image de référence est en haut à gauche.

FIG. 5.30 – Evaluation de la robustesse aux occultations (caméra à 1 *m* à côté de la trajectoire d'apprentissage). L'image de référence est en haut à gauche.

sentir dans le calcul de la localisation, si bien qu'il faut plus de points appariés pour conserver de la précision.

Globalement, on peut tirer certains enseignements de ces expériences. Lorsque la caméra reste sur la trajectoire d'apprentissage, la localisation reste précise même en présence d'occultations importantes. Déplacer certains éléments dans la scène ne perturbe pas plus l'algorithme que le simple fait de les masquer. En effet, l'appariement entre points 2D de l'image courante et points 3D de la carte repose sur des contraintes photométriques (calcul de corrélation) et géométriques (élimination des points non cohérents avec la pose calculée). Avec ces deux contraintes, il n'est pas possible de continuer à utiliser dans la localisation des points se trouvant sur un objet qui a été déplacé. Si la caméra est plus loin de la trajectoire d'apprentissage, alors la robustesse aux diverses perturbations est moins grande car le simple fait de s'écarter des images clefs rend l'appariement de points plus difficile et les points utilisés introduisent dans le calcul de pose une incertitude plus grande que lorsqu'ils sont observés dans l'axe.

5.5.2 Expérimentations en situation réelle

Lieux et conditions d'expérimentations

Nous avons eu la chance de pouvoir effectuer un grand nombre d'expérimentations dans des lieux variés et dans des conditions proches de celles que rencontrerait un système de transport urbain automatisé. Les algorithmes présentés dans ce travail ont été utilisés pour faire naviguer le cycab dans des zones de centre ville, au milieu du public et par des conditions météorologiques changeantes. C'est la répétition de telles expérimentations qui permet au final d'avoir une idée juste de la robustesse du système.

La figure 5.33 montre une image courante obtenue durant une de ces expérimentations et l'image clef correspondante dans la séquence d'apprentissage. On peut voir de nombreuses modifications dans l'image. Certains éléments sont enlevés (le fourgon garé devant le bâtiment), d'autres sont ajoutés (un autre fourgon à gauche, ainsi que des piétons). Les conditions d'éclairage sont modifiées ce qui entraîne plusieurs conséquences sur l'image. On note la présence d'ombres sur l'image courante qui forment de nouveaux contours qui n'étaient pas présents dans l'image de référence. Une partie de l'image est surexposée sur la droite de l'image courante. On peut également remarquer que les contrastes sont inversés au sol : les jointures étaient plus sombres que les dalles au moment de l'apprentissage à cause de l'humidité, elles sont devenues plus claires par temps sec. Seulement deux jours se sont écoulés entre les deux prises de vue. Des modifications

FIG. 5.31 – Evaluation de la robustesse aux changements dans la scène (caméra sur la trajectoire d'apprentissage)

FIG. 5.32 – Evaluation de la robustesse aux changements dans la scène (caméra à 1 *m* à côté de la trajectoire d'apprentissage)

FIG. 5.33 – Diverses modifications pouvant intervenir entre phase d'apprentissage et phase de navigation autonome du robot. L'image de référence est en haut, l'image courante en bas.

de cet ordre sont représentatives des conditions rencontrées habituellement. Avec ces images, le robot a pu fonctionner tout à fait convenablement.

D'autres exemples de situations durant lesquelles nous avons fait naviguer le cycab sont présentés sur les figures 5.34, 5.35, 5.36. Sur chaque figure, la carte construite d'après la séquence vidéo d'apprentissage apparaît en haut. La ligne du bas montre quelques images prises pendant la navigation autonome du cycab, et la ligne du milieu permet de voir l'image clef utilisée pour la localisation. Les différences visibles entre image clef et image courante sont dues principalement aux changements de conditions d'éclairage ou de conditions météorologiques comme la présence de neige sur les images de référence de la figure 5.35.

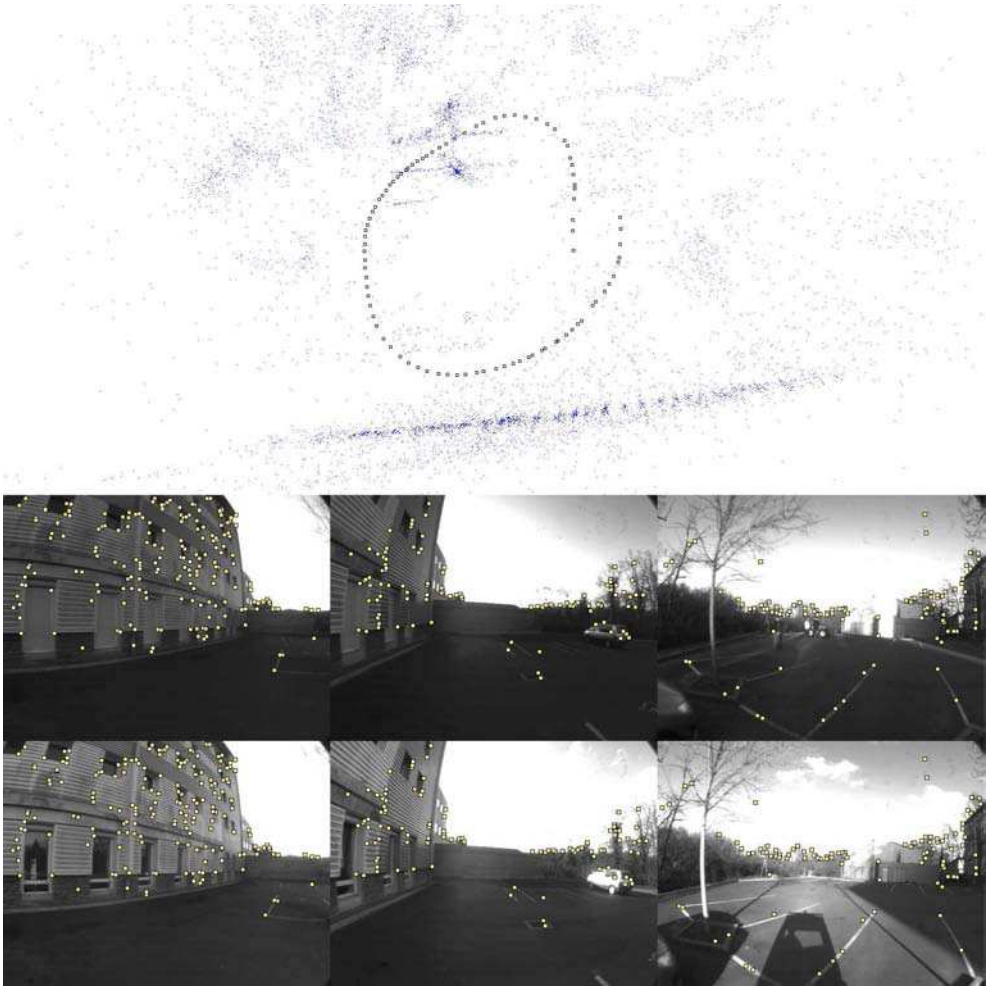

FIG. 5.34 – Navigation autonome du cycab sur le parking de l'ISTIA à Angers.

Difficultés rencontrées

Le paragraphe 5.5.2 doit être complété par une discussion sur les situations qui peuvent provoquer une difficulté ou une impossibilité de se localiser. Le plus souvent, c'est la combinaison de plusieurs perturbations qui provoque une baisse de précision dans la localisation. On peut tout de même citer deux principales causes.

La première difficulté est liée aux conditions de luminosité. Lorsque la position du soleil change, les ombres se déplacent et cela peut perturber l'appariement des points. En effet le calcul de corrélation utilisé (le ZNCC) est robuste aux changements affines positifs de luminosité, mais la modification de l'éclairage peut ne

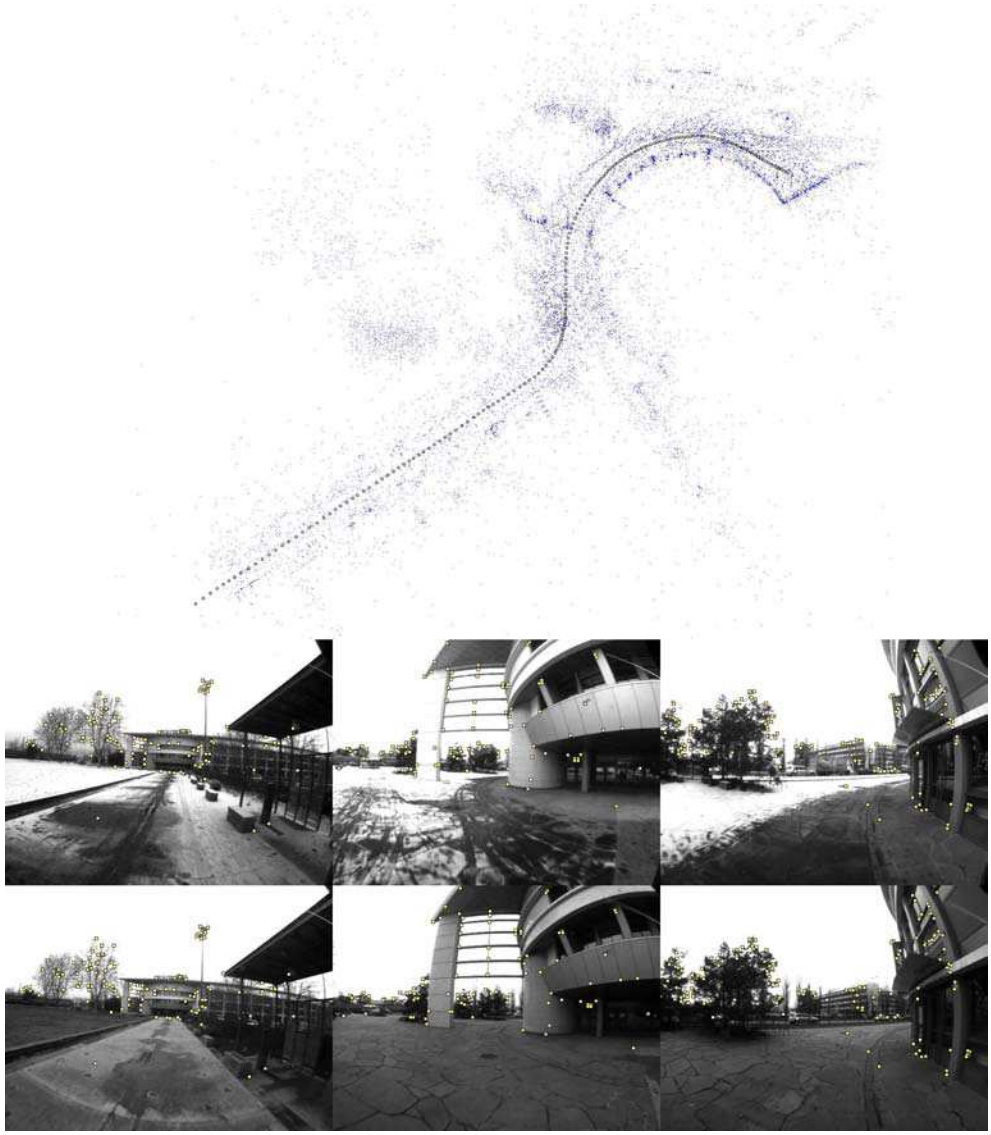

FIG. 5.35 – Navigation autonome du cycab sur le parvis de Polydome à Clermont-Ferrand.

pas être homogène sur toute l'imagette considérée autour du point d'intérêt. C'est ce qui se produit fréquemment lorsque un point est détecté sur le bord d'un objet. Le fond peut être plus lumineux que l'objet lors de l'apprentissage et moins lumineux ensuite. De plus, une partie significative des difficultés rencontrées lors des changement de conditions lumineuses peut être attribuée au matériel utilisé. En effet, notre caméra, comme toutes les caméras conventionnelles, a une dynamique

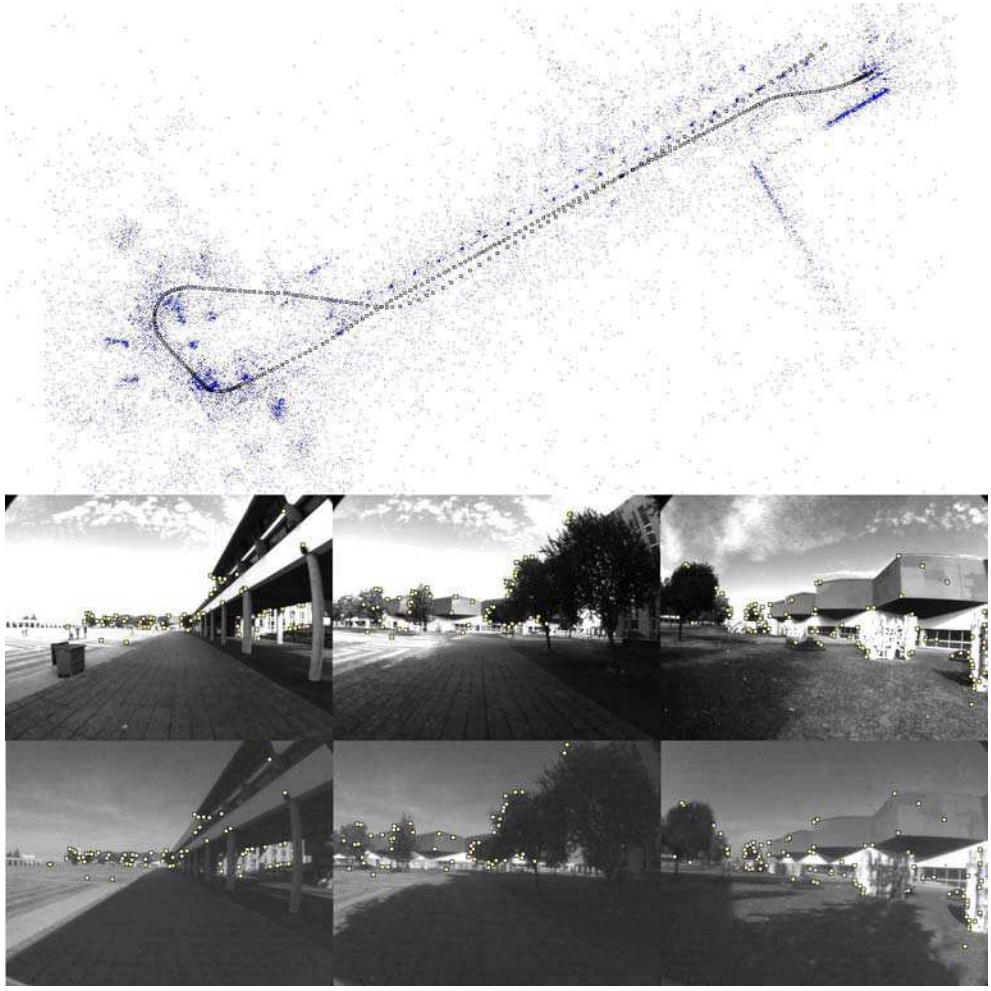

FIG. 5.36 – Navigation autonome du cycab devant la bibliothèque universitaire des Cézeaux à Aubière.

limitée, nettement inférieure à celle de l'œil humain par exemple. Cela se traduit par des problèmes de saturation lorsqu'on travaille en extérieur. L'écart de luminosité entre les parties de la scène au soleil et celles à l'ombre est trop important pour que la caméra enregistre toute la scène sans saturation. C'est ce qu'illustre la figure 5.37. Ce problème de saturation peut se manifester aussi par la disparition de certains objets dans l'image comme on peut le voir sur la figure 5.38. Sur l'image de gauche, par temps couvert, le ciel n'est pas saturé et les branches de l'arbre sont bien visibles. Un autre jour, avec un ciel plus lumineux, le ciel est saturé et le bout des branches disparaît. Ce phénomène perturbe la détection et

l'appariement des points dans les branches de l'arbre. Il est bien difficile d'apporter une solution logicielle à ce problème. On peut simplement espérer pouvoir utiliser des caméras disposant d'une dynamique plus étendue.

FIG. 5.37 – Manque de dynamique de la caméra : une partie de l'image est sous-exposée alors qu'une autre partie est surexposée.

La deuxième source de difficulté rencontrée provient des modifications qui interviennent dans la scène entre la phase d'apprentissage et la phase de localisation. Certains éléments de la scène sont enlevés ou ajoutés : des voitures en stationnement sont déplacées, la végétation change d'aspect, la neige peut masquer le sol, ... L'algorithme actuel est robuste à un certain nombre de changements, mais s'ils sont trop nombreux la localisation peut devenir impossible. Ainsi une séquence d'apprentissage est utilisable pendant quelques jours à quelques semaines selon la saison et le lieu. Deux solutions sont envisageables pour résoudre ce problème. La première idée consiste à remettre à jour la carte et les amers visuels chaque fois que le robot passe sur une trajectoire déjà enregistrée. Les nouveaux amers observés pourraient être ajoutés à la carte alors que les amers qui n'ont pas été observés depuis longtemps seraient enlevés. La deuxième idée est basée sur l'observation suivante. En milieu urbain, une partie des éléments est fixe à long terme (bâtiments, bords de route, ...), une partie est fixe à court terme (voitures en stationnement, affiches de publicité, végétation) et une partie est mobile (piétons, véhicules en circulation,...). La localisation peut alors se faire à partir de deux sources d'informations. Les éléments fixes à long terme peuvent être utilisés dans un appariement entre image courante et image d'apprentissage. Cette information

FIG. 5.38 – Changement d'apparence d'un arbre entre une image bien exposée et une image légèrement surexposée à cause d'un ciel trop lumineux.

de localisation est complétée par un calcul du mouvement propre fondé sur l'utilisation des éléments fixes à moyen terme (en tout cas supposés statiques pour la durée de la phase de navigation du robot). Connaître le mouvement propre de la caméra devrait permettre de mieux localiser le robot même si l'appariement avec la séquence de référence repose sur un petit nombre de points. Les deux solutions sont bien sûr utilisables simultanément.

Chapitre 6

Conclusion et perspectives

Principales contributions

Le travail présenté dans cette thèse a donné lieu à la mise au point d'un système complet de localisation pour un robot mobile fondé sur la vision monoculaire. Le robot est d'abord piloté par un opérateur humain sur une trajectoire d'apprentissage. Le traitement hors ligne de la vidéo enregistrée permet d'obtenir une modélisation tridimensionnelle de l'environnement du robot. Lors de la phase de localisation, une image fournie par la caméra est mise en correspondance avec le modèle 3D de la scène pour calculer en temps réel la pose du robot et un ellipsoïde de confiance associé.

Une fois l'étape de cartographie effectuée, il devient possible d'utiliser la caméra comme un capteur donnant la pose du robot. Cette information est suffisamment précise pour être utilisée directement pour la commande du robot à la place d'un récepteur GPS différentiel par exemple. De nombreuses expérimentations de navigation autonome ont été réalisées pour vérifier la robustesse de la localisation dans des lieux différents et par des conditions météorologiques variées. La précision de la localisation obtenue par vision a été évaluée en réalisant une comparaison avec la vérité terrain obtenue en extérieur avec un récepteur GPS différentiel.

Perspectives

Les résultats obtenus suffisent à faire fonctionner un démonstrateur comme le cycab, mais de nombreuses améliorations doivent être apportées au système avant de pouvoir envisager une utilisation à plus grande échelle. Bien sûr, être capable de localiser le véhicule n'est pas suffisant pour le faire naviguer de façon sûre en milieu urbain. Il faut pouvoir planifier les déplacements du véhicule, détecter les

obstacles, ... Mais même en se restreignant à la tâche de localisation, certaines améliorations apparaissent nécessaires.

La principale contrainte de la localisation par vision est la nécessité d'avoir un modèle de l'environnement à jour. A l'heure actuelle, la seule façon de maintenir la carte à jour est de relancer toute la procédure d'apprentissage à partir d'une nouvelle séquence vidéo. Idéalement, après un parcours autonome sur un trajet donné, le robot devrait être capable de mettre à jour la carte en y incorporant les nouveaux amers observés. Pour cela, on pourrait enregistrer les points d'intérêt détectés pendant la phase de navigation ainsi que la trajectoire de la caméra. Ensuite, une étape de traitement hors ligne permettrait d'optimiser les poses calculées et de trianguler de nouveaux points. Au bout de plusieurs passages sur la même trajectoire, il deviendrait possible d'identifier les primitives les plus stables. Les primitives qui n'ont pas été observées depuis longtemps seraient enlevées de la carte et remplacées par de nouvelles primitives.

La réduction du temps de calcul de la reconstruction 3D est aussi une perspective intéressante qui pourrait être utilisée pour la mise à jour rapide de la carte. Des travaux en cours au laboratoire ont d'ores et déjà montré que l'ajustement de faisceaux hiérarchique pouvait être remplacé par une méthode incrémentale nettement plus efficace.

La robustesse est aussi un point qu'il est possible d'améliorer. Le nombre d'expérimentations réalisées au cours de cette thèse est nécessairement limité. Le système de localisation a fonctionné quelques dizaines d'heures tout au plus. Passer à une application nécessitant l'emploi du système plusieurs heures par jour conduirait sans aucun doute à l'identification de nouveaux problèmes. Même si le système est robuste à un certain nombre de perturbations, les conditions de lumière en extérieur sont souvent difficiles. Les difficultés liées à la saturation du capteur ne peuvent pas être traitées de façon logicielle. Une caméra avec une plus grande dynamique ainsi qu'un réglage automatique du diaphragme serait d'un grand secours. Du point de vue des algorithmes, des progrès peuvent probablement être réalisés en améliorant la méthode d'appariement des points d'intérêt pour mieux tenir compte des changements de la direction d'éclairage.

Enfin nous nous sommes attachés à concevoir un système de localisation indépendant du robot sur lequel il sera utilisé. Cette généricité est un atout car elle permet d'envisager un grand nombre d'applications possibles y compris des utilisations où la caméra est tenue à la main. Mais pour obtenir les meilleures performances, il est utile de mieux intégrer la brique de localisation, les capteurs proprioceptifs et la loi de commande. Nous avons juste évoqué la possibilité d'utiliser un modèle de mouvement du véhicule. Ce travail pourrait être approfondi en prenant en compte notamment les informations provenant des capteurs proprioceptifs

du véhicule (vitesse de rotation des roues et angle de braquage).

Questions soulevées

La rédaction de ce rapport est aussi l'occasion de s'interroger de façon plus générale sur la conception d'un algorithme de vision temps réel. La question centrale à se poser est celle de l'utilisation la plus appropriée du temps de calcul.

Comme on l'a vu, l'algorithme de localisation est le résultat de l'association d'un certain nombre de briques élémentaires (détection de points d'intérêt, appariement, calcul de pose robuste, ...) pour lesquelles de nombreuses méthodes existent dans la littérature. Pour chacune de ces briques, un compromis doit être fait entre le temps de calcul et les qualités de l'algorithme. On peut comparer les qualités et le temps de calcul pour chaque brique de façon individuelle, mais ce qui est plus intéressant est l'impact sur les performances globales du système. Par exemple, on peut choisir comme nous l'avons fait d'utiliser un détecteur de points d'intérêt et une méthode d'appariement très simples et très rapides. Cette méthode produit un nombre d'appariements erronés important, ce qui impose de consacrer plus de temps au calcul de pose robuste. On aurait pu choisir une stratégie différente : consacrer plus de temps à l'appariement pour obtenir une proportion plus petite de faux appariements et passer ensuite moins de temps dans le calcul de pose. On peut se demander quelle solution est la plus performante. Mais la question la plus intéressante est la suivante : peut-on trouver une méthode pour choisir de manière optimale la combinaison d'algorithmes élémentaires à utiliser ?

On peut pousser la réflexion plus loin et se demander comment maximiser l'information de localisation récoltée pour un temps de calcul donné. Dans notre cas, apparier un point a un coût (en temps de calcul) et chaque appariement apporte une information partielle sur la pose de la caméra. La difficulté pour apparier un point et l'information qu'il apporte sont variables en fonction du point. Les point éloignés de la caméra donnent peu d'information sur la position. D'un autre côté, apparier ces points est plus facile car leur apparence dans l'image varie peu en fonction du point de vue. Pour l'instant, tous les points d'intérêt sont traités de manière identique. Mais il semble intéressant de consacrer plus de temps de calcul aux points du premier plan qui sont porteurs de plus d'information. Pour ces points un score de corrélation plus élaboré que le ZNCC serait mieux adapté. On pourrait concevoir un algorithme qui choisisse en ligne, pour chaque point, la méthode d'appariement en fonction de l'information de position que le point est susceptible d'apporter au calcul de pose.

Bibliographie

[1] A. Akhriev and C.-Y. Kim. Contour tracking without prior information. In *Proceedings of the VIIth Digital Image Computing : Techniques and Applications*, 2003.

[2] H. Araújo, R.J. Carceroni, and C.M. Brown. A fully projective formulation to improve the accuracy of Lowe's pose estimation algorithm. *Computer Vision and Image Understanding*, 70(2) :227–238, 1998.

[3] A. Argyros, K. Bekris, S. Orphanoudakis, and L. Kavraki. Robot homing by exploiting panoramic vision. *Journal of Autonomous Robots*, 19(1) :7–25, 2005.

[4] T. Bailey. Constrained initialisation for bearing-only slam. In *International Conference on Robotics and Automation*, 2003.

[5] P. Beardsley, P. Torr, and A. Zisserman. 3d model acquisition from extended image sequences. In *European Conference on Computer Vision*, pages 683–695, 1996.

[6] P.R. Beaudet. Rotationally invariant image operators. In *International Joint Conference on Pattern Recognition*, pages 579–583, 1978.

[7] S. Benhimane and E. Malis. Real-time image-based tracking of planes using second order minimization. In *International Conference on Intelligent Robots Systems*, 2004.

[8] G. Blanc, Y. Mezouar, and P. Martinet. Indoor navigation of a wheeled mobile robot along visual routes. In *IEEE International Conference on Robotics and Automation*, 2005.

[9] S. Bourgeois, S. Naudet-Colette, and M. Dhome. Recalage d'un modèle cao à partir de descripteurs locaux de contours. In *Congès francophone de Reconnaissance des Formes et d'Intelligence Artificielle*, 2006.

[10] O.T. Carmichael and M. Hébert. Object recognition by a cascade of edge probes. In *British Machine Vision Conference*, 2002.

[11] D. Cobzas, H. Zhang, and M. Jagersand. Image-based localization with depth-enhanced image map. In *International Conference on Robotics and Automation*, 2003.

[12] Youjing Cui and Shuzhi Sam Ge. Autonomous vehicle positioning with gps in urban canyon environments. *IEEE transactions on robotics and automation*, 19(1), February 2003.

[13] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the 9th International Conference on Computer Vision, Nice*, 2003.

[14] C. Canudas de Wit, B. Siciliano, and G. Bastin. *The Zodiac, theory of robot control*. Springer Verlag, 1996.

[15] T. Drummond and R. Cipolla. Real-time tracking of complex structures with on-line camera calibration. *Image and Vision Computing*, 20(5-6) :427–433, 2003.

[16] O. Faugeras. *Three-Dimensional Computer Vision - A Geometric Viewpoint*. MIT Press, 1993.

[17] O. Faugeras and M. Hebert. The representation, recognition, and locating of 3-d objects. *International Journal of Robotic Research*, 5(3) :27–52, 1986.

[18] O. Fischler and R. Bolles. Random sample consensus : a paradigm for model fitting with application to image analysis and automated cartography. *Communications of the Association for Computing Machinery*, 24 :381–395, 1981.

[19] A.W. Fitzgibbon and A. Zisserman. automatic camera recovery for closed or open image sequences. In *European Conference on Computer Vision*, pages 311–326, 1998.

[20] A. Georgiev and P.K. Allen. Localization methods for a mobile robot in urban environments. *IEEE Transactions on Robotics*, 20(5) :851–864, October 2004.

[21] T. Goedemé, T. Tuytelaars, L. Van Gool, G. Vanacker, and M. Nuttin. Feature based omnidirectional sparse visual following. In *International Conference on Intelligent Robots and Systems*, pages 1003–1008, 2005.

[22] N. Gracias and J. Santos-Victor. Underwater video mosaics as visual navigation maps. *Computer Vision and Image Understanding, special issue on underwater computer vision and pattern recognition*, 79(1) :66–91, 2000.

[23] J. A. Grunert. Das pothenotische problem in erweiterter gestalt nebst über seine anwendungen in der geodäsie. *Grunerts archiv für mathematik und physik*, (1) :238–248, 1841.

[24] R. Haralick, C. Lee, K. Ottenberg, and M. Nolle. Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13(3) :331–356, 1994.

[25] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, 1988.

[26] R. Hartley. In defense of the eight-point algorithm. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 19(6) :580–593, 1997.

[27] R. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2) :146–157, 1997.

[28] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2000.

[29] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5) :651–670, 1996.

[30] Intel. Ia-32 intel architecture optimization. *http ://developer.intel.com*.

[31] Intel. Ia-32 intel architecture software developer's manual volume 1 : Basic architecture. *http ://developer.intel.com*.

[32] Intel. Ia-32 intel architecture software developer's manual volume 2 : Instruction set reference. *http ://developer.intel.com*.

[33] S. Jeon and B. Kim. Monocular-based position determination for indoor navigation of mobile robots. In *IASTED International Conference*, 1999.

[34] I-K. Jung and S. Lacroix. A robust interest point matching algorithm. In *International Conference on Computer Vision*, 2001.

[35] F. Jurie and M. Dhome. Hyperplane approximation for template tracking. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 24(7) :996–1000, 2002.

[36] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D) :35–45, 1960.

[37] A. Kelly. Mobile robot localization from large scale appearance mosaics. *International Journal of Robotics Research*, 19(11) :1104–1125, 2000.

[38] K. Kidono, J. Miura, and Y. Shirai. Autonomous visual navigation of a mobile robot using a human-guided experience. *Robotics and Autonomous Systems*, 40(2-3) :124–1332, 2002.

[39] J. M. Lavest, M. Viala, and M. Dhome. Do we need an accurate calibration pattern to achieve a reliable camera calibration ? In *European Conference on Computer Vision*, pages 158–174, 1998.

[40] T. Lemaire, S. Lacroix, and J. Solá. A practical 3d bearing-only slam algorithm. In *International Conference on Intelligent Robots and Systems*, pages 2757–2762, 2005.

[41] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *Transactions on Pattern Analysis and Machine Intelligence*, 2006.

[42] M. Lhuillier and M. Perriollat. Uncertainty ellipsoids calculations for complex 3d reconstructions. In *International Conference on Robotics and Automation*, 2006.

[43] M. Lhuillier and L. Quan. A quasi-dense approach to surface reconstruction from uncalibrated images. *Transactions on Pattern Analysis and Machine Intelligence*, 27(3) :418–433, 2005.

[44] S. Li and S. Tsuji. Qualitative representation of scenes along a route. *Image and Vision Computing*, 17 :685–700, 1999.

[45] H. Ling and D. Jacobs. Deformation invariant image matching. In *International Conference on Computer Vision*, 2005.

[46] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293 :133–135, 1981.

[47] D. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, 1999.

[48] B.D. Lucas and T. Kanade. An iterative registration technique with an application to stereo vision. In *Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[49] Y. Matsumoto, K. Sakai, M. Inaba, and H. Inoue. View-based approach to robot navigation. In *International Conference on Intelligent Robots and Systems*, 2000.

[50] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision*, pages 128–142, 2002.

[51] K. Mikolajczyk, A. Zisserman, and C. Schmid. Shape recognition with edge-based features. In *British Machine Vision Conference*, 2003.

[52] N. Molton, A. J. Davison, and I. Reid. Locally planar patch features for real-time structure from motion. In *British Machine Vision Conference*, 2004.

[53] H. Moravec. Rover visual obstacle avoidance. In *International Joint Conference on Artificial Intellignece*, pages 785–790, 1981.

[54] H. Morita, M. Hild, J. Miura, and Y. Shirai. View based localization in outdoor environments based on support vector learning. In *International Conference on Intelligent Robots and Systems*, pages 3083–3088, 2005.

[55] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real time localization and 3d reconstruction. In *International Conference on Computer Vision and Pattern Recognition*, 2006.

[56] P. Moutarlier and R. Chatila. Stochastic multisensory data fusion for mobile robot location and environment modeling. In *5th International Symposium on Robotics Research*, 1989.

[57] P.M. Newman and J.J. Leonard. Consistent convergent constant time slam. In *International Joint Conference on Artificial Intelligence*, 2003.

[58] D. Nistér. Automatic dense reconstruction from uncalibrated video sequences. In *European Conference on Computer Vision*, pages 649–663, 2000.

[59] D. Nistér. Frame decimation for structure and motion. In *2nd workshop on Structure from Multiple Images of Large Environments, Springer Lecture Notes on Computer Science*, volume 2018, pages 17–34, 2001.

[60] D. Nistér. An efficient solution to the five-point relative pose problem. In *Conference on Computer Vision and Pattern Recognition*, pages 147–151, 2003.

[61] D. Nistér. An efficient solution to the five-point relative pose problem. *Transactions on Pattern Analysis and Machine Intelligence*, 26(6) :756–770, 2004.

[62] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *Conference on Computer Vision and Pattern Recognition*, pages 652–659, 2004.

[63] A. Ohya, Y. Miyazaki, and S. Yuta. Autonomous navigation of mobile robot based on teaching and playback using trinocular vision. In *IEEE Industrial Electronics Conference*, 2001.

[64] J. A. Pérez, J. A. Castellanos, J. M. M. Montiel, J. Neira, and J. D. Tardós. Continuous mobile robot localization : Vision vs. laser. In *International Conference on Robotics and Automation*, 1999.

[65] M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *International Conference on Computer Vision*, pages 90–95, 1998.

[66] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipies in C*. Cambridge University Press, 1988.

[67] A. Remazeilles, F. Chaumette, and P. Gros. Robot motion control from a visual memory. In *International Conference on Robotics and Automation*, volume 4, pages 4695–4700, 2004.

[68] E. Royer, J. Bom, M. Dhome, B. Thuilot, M. Lhuillier, and F. Marmoiton. Outdoor autonomous navigation using monocular vision. In *International Conference on Intelligent Robots and Systems*, pages 3395–3400, 2005.

[69] E. Royer, M. Lhuillier, M. Dhome, and T. Chateau. Towards an alternative gps sensor in dense urban environment from visual memory. In *Proceedings of the 15th British Machine Vision Conference*, pages 197–206, September 2004.

[70] E. Royer, M. Lhuillier, M. Dhome, and T. Chateau. Localization in urban environments : monocular vision compared to a differential gps sensor. In *International Conference on Computer Vision and Pattern Recognition, CVPR*, June 2005.

[71] E. Royer, M. Lhuillier, M. Dhome, and J-M. Lavest. Performance evaluation of a localization system relying on monocular vision and natural landmarks. In *Proceedings of the ISPRS Workshop BenCOS (Towards Benchmarking Automated Calibration, Orientation and Surface Reconstruction from Images)*, October 2005.

[72] E. Royer, M. Lhuillier, M. Dhome, and J.-M. Lavest. Localisation par vision monoculaire pour la navigation autonome. *Revue Traitement du Signal*, 23(1), 2006.

[73] E. Royer, M. Lhuillier, M. Dhome, and J.-M. Lavest. Localisation par vision monoculaire pour la navigation autonome : précision et stabilité de la méthode. In *RFIA 2006, 15e congrès francophone AFRIF-AFIA Reconnaissance des Formes et Intelligence Artificielle*, Janvier 2006.

[74] C. Samson. Control of chained systems. application to path following and time-varying point stabilization of mobile robots. *IEEE Transactions on Automatic Control*, 40(1) :64–77, January 1995.

[75] C. Schmid, R. Mohr, and Ch. Bauckage. Comparing and evaluating interest points. In *Proceedings of the 6th International Conference on Computer Vision*, 1998.

[76] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *International Journal of Robotic Research*, 21(8) :735–760, 2002.

[77] J. Shi and C. Tomasi. Good features to track. In *Conference on Computer Vision and Pattern Recognition*, 1994.

[78] R. Smith and P. Cheesman. On the representation of spatial uncertainty. *International Journal of Robotics Research*, 1987.

[79] C. Sturm. *Mémoire sur la résolution des équations numériques*. Académie royale des sciences de l'institut de France. Sciences mathématiques et physiques, Tome 6, 1835.

[80] P. Sturm. Critical motion sequences for monocular self-calibration and uncalibrated euclidean reconstruction. In *International Conference on Computer Vision and Pattern Recognition*, pages 1100–1105, 1997.

[81] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intellignece*, 2001.

[82] B. Thuilot, J. Bom, F. Marmoiton, and P. Martinet. Accurate automatic guidance of an urban vehicle relying on a kinematic gps sensor. In *Symposium on Intelligent Autonomous Vehicles IAV04*, 2004.

[83] P. Torr, A. Fitzgibbon, and A. Zisserman. The problem of degeneracy in structure and motion recovery from uncalibrated image sequences. *International Journal of Computer Vision*, 32(1) :27–44, 1999.

[84] P. Torr and D. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *International Journal of Computer Vision*, 24(3) :271–300, 1997.

[85] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – A modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms : Theory and Practice*, Lecture Notes in Computer Science, pages 298–375. Springer Verlag, 2000.

[86] T. Tuytelaars and L. Van Gool. Wide baseline stereo based on local, affinely invariant regions. In *British Machine Vision Conference*, pages 412–422, 2000.

[87] Z. Zhang, R. Deriche, O. Faugeras, and Q-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. In *Rapport de recherche INRIA N° 2273*, 1994.