Navigation visuelle pour les missions autonomes des petits drones

Cédric Le Barz

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE
École Doctorale Informatique, Télécommunications et Électronique (Paris)

Spécialité
Informatique

Présentée par
Cédric Le Barz

Pour obtenir le grade de
DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse

Navigation visuelle pour les missions autonomes des petits drones

Soutenue le 30 juin 2015, devant le jury composé de :

M. François Brémond	INRIA	Rapporteur
M. Matthieu Cord	Université Pierre et Marie Curie	Directeur de thèse
M. Marcin Detyniecki	Université Pierre et Marie Curie	Examinateur
M. Jean-Yves Dufour	Thales	Examinateur
M. David Filliat	ENSTA ParisTech	Rapporteur
M. Stéphane Herbin	ONERA	Encadrant
M. Nicolas Thome	Université Pierre et Marie Curie	Examinateur

La connaissance est une navigation dans un océan d'incertitudes à travers des archipels de certitudes.

Edgar Morin – Les sept savoirs nécessaires à l'éducation du futur

Table des matières

Liste des figures

Liste des tableaux

CHAPITRE 1

Introduction

Sommaire

1.1 Contexte

Les drones, c'est à dire les aéronefs sans pilote, connaissent aujourd'hui une croissance remarquable. Ils ont initialement été développés pour des applications militaires car ils offrent 2 avantages principaux : un faible coût de construction et de mise en œuvre et, ils permettent de réaliser des missions dangereuses sans mettre en jeu la vie des pilotes. Ils sont actuellement principalement utilisés pour des missions de collecte d'informations visuelles à des fins de surveillance, de détection et/ou de reconnaissance. Ils sont de plus en plus utilisés pour des applications civiles : surveillance d'infrastructures, suivis de chantiers, agriculture, prises de vue télévisuelle, etc. L'évolution des technologies a permis la mise au point de petits drones de quelques kilogrammes et d'envergure inférieure à un mètre (Fig. 1.1) aptes à évoluer dans des environnements denses, c'est à dire dans des environnements intérieurs ou urbains.

L'évolution dans ce type d'environnement nécessite un système de navigation fiable et robuste. Un système de navigation doit permettre à un drone, ou plus généralement à un robot, de se diriger dans un environnement afin qu'il puisse rejoindre une position donnée à partir de n'importe quelle autre position. La navigation nécessite la détermination de la position courante du

(a) AR-Drone Société Parrot. (b) X4 Société Fly'n'sense. (c) Phantom Société Dji.

FIGURE 1.1 – Différents petits drones professionnels et grand public.

mobile dans un repère absolu, le calcul de la trajectoire à suivre pour rejoindre la destination, spécifiée également dans ce repère absolu, et, le calcul des commandes permettant de suivre la trajectoire voulue. Le robot doit, de plus, être conscient de son environnement dynamique, afin d'éviter les différents obstacles mobiles ou immobiles. Le développement de l'ensemble de ces fonctionnalités, c'est à dire localisation, planification, guidage et pilotage apporteront l'autonomie requise pour un robot mobile autonome.

Se localiser de façon précise, fiable et durable est, comme l'a écrit Sebastian Thrun dans [**?**], une compétence clé ("*a core competency*") pour les robots mobiles. De nombreux capteurs peuvent être utilisés à cette fin. Les systèmes de navigation globale par satellite (*Global Navigation Satellite System - GNSS*) sont de nos jours très utilisés. Cependant, leur fonctionnement est altéré dans les environnements denses et leur précision est à la fois limitée et variable. Ceci est dû aux conditions de propagation des ondes radio. De plus, dans le cas de missions militaires, le brouillage des ondes radio est aisé. Des solutions alternatives basées sur la navigation à l'estime, c'est à dire qui consiste à déduire la position d'un mobile depuis sa dernière position connue, sont utilisées lorsque les solutions de type GNSS sont inopérantes. Il existe en effet des centrales de navigation basées sur des capteurs inertiels ayant de très bonnes performances. Cependant, ces capteurs restent très chers et/ou trop encombrants pour être embarqués sur de petits drones. Les capteurs inertiels bas-coût ne peuvent être utilisés seuls, car leurs mesures bruitées et biaisées engendrent une dérive lors du calcul de la trajectoire. Leur utilisation reste envisageable mais ils doivent être combinés de manière complémentaire avec d'autres types de capteurs.

Différentes approches sont envisageables et de nombreux capteurs peuvent être utilisés. Parmi ceux-ci, les caméras semblent être des capteurs idéaux de par la richesse des informations qu'elles collectent. Contrairement aux capteurs de type centrale inertielle qui ne fournissent que des informations proprioceptives, les images issues des caméras fournissent des informations extéroceptives sur l'environnement, comme par exemple la présence d'obstacles, et même dans certains cas la distance du robot aux obstacles. Contrairement

aux systèmes GNSS dont le fonctionnement est altéré dans les environnements denses, les caméras fournissent toujours une information exploitable dans les environnements urbains et intérieurs pourvu que la luminosité et le contraste soient suffisants. Le traitement des images, issues des caméras, nécessite néanmoins une interprétation et généralement des ressources de calcul importantes pour extraire les informations utiles.

1.2 Problématique

Les principales solutions de navigation visuelle reposent sur la construction, lors de l'étape de navigation, d'une carte au sein de laquelle se positionne le robot, ou sur la reconnaissance de caractéristiques dont les positions sont connues.

Dans le premier cas, les solutions proposées supposent très peu de connaissances *a priori* sur l'environnement. Elle sont particulièrement adaptées à l'exploration d'environnements inconnus. Elles permettent une navigation locale, c'est à dire une navigation dans un environnement dont la taille est limitée (par exemple quelques pièces ou une rue) et court-terme, car la trajectoire estimée est sujette à des dérives. Elles demeurent donc insuffisantes pour permettre la navigation long-terme d'un robot qui aurait à parcourir une trajectoire de plusieurs kilomètres. D'autre part, ce type de solutions ne fournit pas une position dans un repère connu *a priori*.

Dans le deuxième cas, les solutions proposées nécessitent au préalable la construction d'une base de caractéristiques géoréférencées. Lors de l'étape de navigation, il va s'agir de retrouver, parmi cette base de caractéristiques, celles actuellement observées par le robot, afin qu'il puisse se localiser. La difficulté principale est la mise en correspondance des caractéristiques acquises avec celles de la base. Pour faciliter cette recherche, plusieurs simplifications sont généralement réalisées. D'une part, lors de la réalisation de la base, on s'arrange pour que les conditions d'acquisition soient similaires à celles qui seront rencontrées lors de l'étape de navigation. D'autre part, la base est le plus souvent réalisée avec les mêmes capteurs que ceux utilisés lors de la phase de navigation.

Les solutions actuelles ne permettent donc pas à un robot de suivre de manière autonome et fiable une trajectoire de plusieurs kilomètres qui aurait été préalablement définie par un opérateur ou calculée par un algorithme de planification.

Les travaux réalisés dans cette thèse s'inscrivent dans cette deuxième famille de méthodes. Plus particulièrement, la problématique traitée est celle de la localisation absolue visuelle d'un robot autonome au sein d'un environnement urbain. L'objectif poursuivi est l'ego-localisation grâce aux images acquises par ce dernier afin de corriger des informations erronées ou bruitées, voire temporairement absentes, fournies par d'autres capteurs. La figure 1.2

illustre cette problématique : étant donné une ou plusieurs images acquises par le robot, il s'agit de retrouver ces images dans une base d'images géoréférencées, afin de se localiser et suivre la trajectoire spécifiée. La base d'images

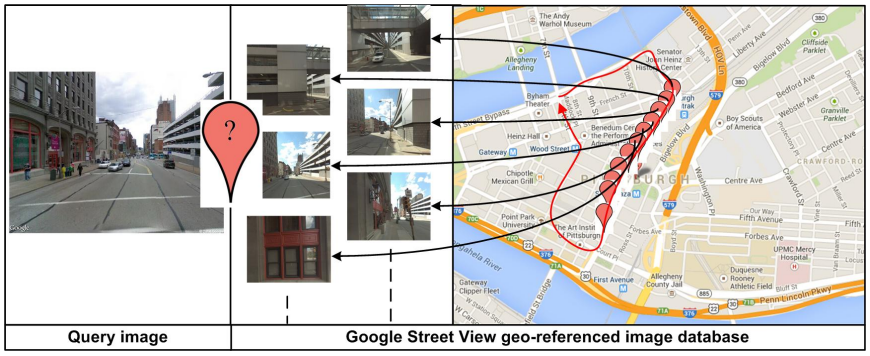


FIGURE 1.2 – Localisation visuelle d'un robot : notre problématique consiste à mettre en correspondance une image requête avec l'une des images géoréférencées afin de déterminer la position absolue du robot.

géoréférencées utilisée n'est pas réalisée spécifiquement pour le besoin de localisation visuelle. Compte-tenu de la difficulté à mettre en correspondance des images représentant la même scène, les données utilisées lors de nos expérimentations sont celles issues d'un robot terrestre (*Unmanned Ground Vehicle - UGV*). Il est à noter que cette fonctionnalité de localisation visuelle absolue permet de corriger les dérives inhérentes à l'emploi de solutions de localisation n'ayant recours à aucune information sur l'environnement extérieur.

Le scénario opérationnel typique se déroule en deux étapes. La première est une étape de préparation de mission lors de laquelle un opérateur saisit sur une carte la trajectoire que le robot doit suivre pour la réalisation de sa mission. Les images géoréférencées correspondant à ce que verrait le robot si ce dernier suivait cette trajectoire sont alors traitées de manière automatique pour en extraire des données qui seront utiles à la localisation. Il peut s'agir, par exemple, de caractéristiques visuelles saillantes ou de signatures visuelles décrivant les images. Cette première étape étant réalisée avant le déroulement de la mission, il n'y a donc aucune contrainte concernant les temps de traitement. Lors de la deuxième étape, c'est à dire lors de la navigation, les images acquises par le robot sont traitées et comparées avec les données précédemment calculées pour se localiser. La figure 1.3 illustre ces deux étapes distinctes.

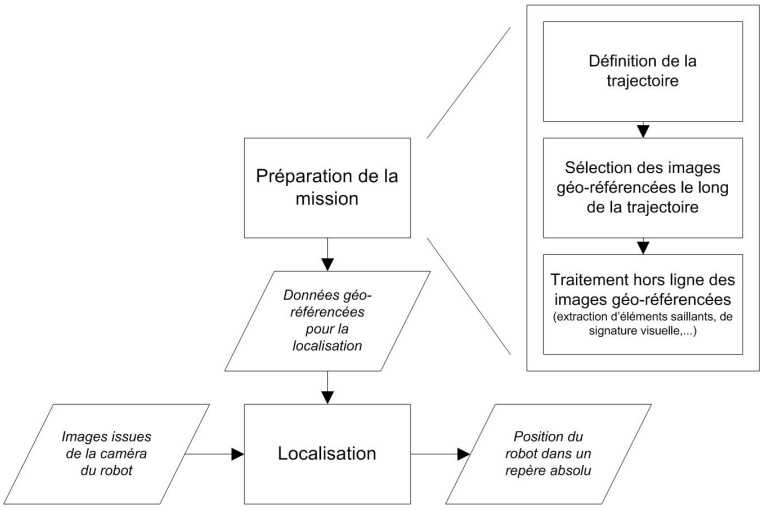


FIGURE 1.3 – Mission typique - Première étape : préparation de mission, i.e. définition de la trajectoire, et traitement des images géoréférencées pour en extraire des données ; Deuxième étape : localisation, par comparaison des images acquises par le robot avec les données extraites lors de la préparation de la mission.

1.3 Contributions

Dans ce cadre, nous avons étudié et développé une approche de localisation absolue et précise qui repose sur deux phases successives. La première a pour objectif de localiser approximativement le robot par mise en correspondance des images acquises par ce dernier avec les images géoréférencées et, la deuxième a pour objectif d'affiner cette position grâce à l'estimation de l'orientation relative entre ces deux images.

Plus précisément, nos contributions sont les suivantes :

— Mise en correspondance d'une séquence d'images avec celles d'une base d'images géoréférencées :
La mise en correspondance d'une unique image donnée avec une image représentant la même scène est une tâche difficile dans notre contexte. Pour résoudre notre problème, nous avons proposé de mettre en correspondance une séquence d'images afin d'exploiter les contraintes spatio-temporelles inhérentes au déplacement d'un robot. Cette solution a fait l'objet d'une publication [**?**] :

C. Le Barz, N. Thome, M. Cord, S. Herbin and M. Sanfourche, "Global Robot Ego-localization Combining Image Retrieval and HMM-based Filtering", in the Proceedings of the 6th workshop on Planning Perception and Navigation for Autonomous Navigation whithin International Conference on Intelligent Robots and Systems (IROS), september 2014.

— Amélioration de la mesure de similarité visuelle entre images :
Les performances obtenues dépendent principalement de la qualité de la mesure de similarité visuelle entre images. Pour améliorer cette mesure, nous avons proposé d'apprendre de manière supervisée une distance locale permettant de mesurer la similarité entre l'image requête et les images géoréférencées proches de la position estimée. Cette nouvelle mesure de similarité, dédiée à notre contexte, a fait l'objet d'une publication [**?**] :

C. Le Barz, N. Thome, M. Cord, S. Herbin and M. Sanfourche, "Exemplar Based Metric Learning for Robust Visual Localization", in the Proceedings of the International Conference on Image Processing (ICIP), september 2015.

Cette mesure de similarité a été intégrée dans notre solution de mise en correspondance de séquence d'images. Les résultats obtenus ont également fait l'objet d'une publication [**?**] :

C. Le Barz, N. Thome, M. Cord, S. Herbin and M. Sanfourche, "Absolute geolocalization thanks to Hidden Markov Model and exemplar-based metric learning", in the Proceedings of the 6th workshop on Computer Vision in Vehicle Technology within International Conference on Computer Vision and Pattern Recognition, June 2015.

— Affinage de la localisation par estimation de pose :
La navigation autonome nécessitant une localisation précise, un affinage de la position estimée a été étudié. Étant donné deux images représentant la même scène, il est possible d'estimer la position relative entre les deux caméras qui ont acquis ces images, par mise en correspondance de points. Compte-tenu de la variabilité d'apparence entre les deux images appariées, une étude a été réalisée afin de déterminer le nombre de points à mettre en correspondance, le taux de faux appariements acceptable et la précision des points appariés pour obtenir une certaine qualité sur l'estimation du positionnement relatif entre les deux caméras.

Ces trois contributions sont décrites dans trois chapitres distincts : notre contribution concernant l'affinage de la localisation par estimation de pose est décrite dans le chapitre 3, celle concernant la mise en correspondance d'une séquence d'images dans le chapitre 4, et enfin celle concernant l'amélioration de la similarité visuelle dans le chapitre 5. Préalablement à la description de nos contributions, notre travail est positionné par rapport à l'existant. Pour cela, les différentes stratégies de navigation et les différentes solutions de localisation visuelle sont présentées dans le chapitre 2.

CHAPITRE 2

Navigation autonome et localisation visuelle

Sommaire

Après une brève définition du mot "navigation", nous détaillons les différentes fonctionnalités requises pour développer un système de navigation autonome ainsi que les différentes stratégies de navigation. Les principales méthodes de localisation visuelle sont ensuite présentées. Nous distinguons les méthodes de navigation locale qui permettent à un robot de se déplacer dans un environnement inconnu grâce à une cartographie locale, des méthodes de navigation globale qui permettent à un robot de rejoindre une position préalablement spécifiée dans un repère absolu, ou identique au repère utilisé lors de la phase de préparation de mission. Un état de l'art de ces dernières méthodes est proposé. Enfin, l'architecture de notre système de traitement est présentée avec nos différentes contributions positionnées par rapport à cet état de l'art.

2.1 Navigation

2.1.1 Définition

Si l'on se réfère au Larousse, le verbe "Naviguer" signifie *"Faire suivre à un navire, un avion ou une automobile une route déterminée"*. Un sys-

tème de navigation doit donc permettre à un robot de se diriger dans un environnement afin de se rendre à un endroit précis préalablement défini, et ce, à partir de n'importe quelle autre position. Les méthodes de navigation sont liées à l'environnement sur lequel ou dans lequel le robot évolue. On distingue la navigation maritime et fluviale, la navigation sous-marine, la navigation terrestre, la navigation aérienne et la navigation spatiale. La fonctionnalité "Navigation" est indispensable pour un robot mobile, mais ce n'est pas la seule. Pour suivre le chemin lui permettant de réaliser sa mission, une "boucle" de navigation, guidage et pilotage est nécessaire. Les termes "Guider" et "Piloter" sont, quant à eux, définis ainsi dans le Larousse :

— Guider : *"Accompagner quelqu'un pour lui montrer le chemin"*.
— Piloter : *"Conduire un navire, un avion, ou une automobile"*.

Le chemin déterminé est généralement d'un niveau d'abstraction supérieure à ce que sait exécuter le robot. C'est pourquoi, il est nécessaire de convertir les directives en une suite de commandes exécutables directement. "Guider", c'est donc calculer les consignes de déplacement, à partir d'informations issues d'un système maître en prenant en compte les contraintes physiques du robot. Le guidage est souvent associé à une loi de guidage qui est une expression mathématique permettant d'élaborer les commandes à exécuter par le robot. Ce modèle permet de déterminer, en fonction de l'état estimé du système, les consignes optimales pour satisfaire certaines contraintes, généralement exprimées à l'aide d'un critère de performances à optimiser. Les commandes calculées doivent garantir la stabilité du robot. "Piloter", c'est la gestion des actionneurs en cohérence avec la consigne qui va permettre de suivre la trajectoire voulue. De nombreux aléas, tels que des imprécisions de vitesse, de position, ou de temps d'exécution, sont susceptibles de perturber chaque commande. L'ensemble des commandes doivent donc être mises à jour en permanence.

La manière dont fonctionne un robot autonome est généralement décrite par différents modèles qui sont composés de trois modules principaux que sont la perception, la planification et l'action. Leur agencement définit des architectures de contrôle différentes. On distingue trois grandes catégories : les contrôleurs hiérarchiques, les contrôleurs réactifs et les contrôleurs hybrides. Une description complète de ces architectures peut être trouvée dans [**?**]. La figure 2.1 représente un exemple haut niveau d'architecture de contrôle. La figure 2.2 est une illustration plus bas niveau, faisant apparaître les différentes fonctionnalités précédemment mentionnées, i.e. localisation, planification, guidage et pilotage. Les deux modules "Perception" et "Décision" constituent un système de navigation visuelle.

2.1.2 Fonctionnalités requises pour la navigation

Pour déterminer la trajectoire optimale, la navigation nécessite trois fonctionnalités distinctes :

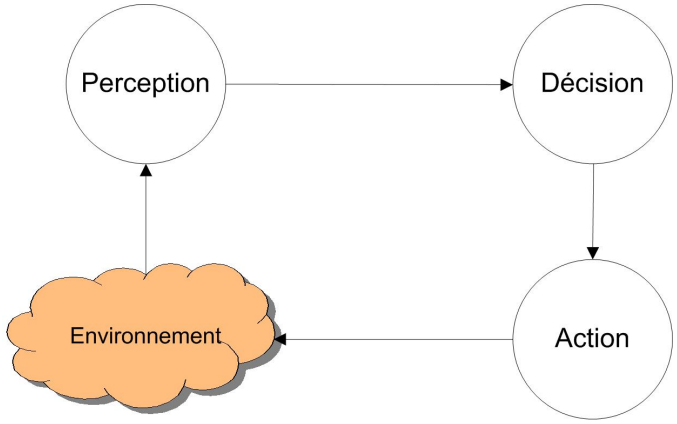


FIGURE 2.1 – Exemple d'architecture de contrôle d'un robot.

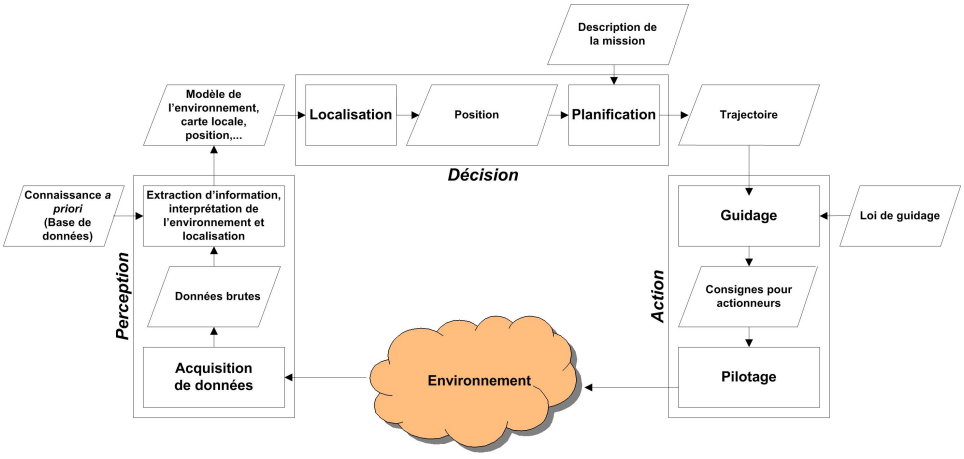


FIGURE 2.2 – Architecture de contrôle d'un robot pour la navigation autonome.

— la détermination de la position courante du robot (ses coordonnées) par rapport à un repère de référence. Il s'agit de la fonctionnalité de localisation. Le robot étant mobile, il peut occuper dans son environnement différentes positions à différents instants. Grâce aux mesures et aux observations faites par celui-ci lors de son déplacement, il doit déterminer sa position avec une précision maximale. Cette précision dépend de différents facteurs, comme par exemple le milieu et l'environnement dans lequel il évolue, ou encore de la mission attribuée. La localisation peut être réalisée de façon active ou passive, c'est à dire qu'il peut réaliser des actions spécifiques qui lui permettront de se localiser quitte à ralentir le déroulement de sa mission, ou bien affiner sa localisation lors de la réalisation de sa tâche.
— L'adaptation à son environnement : l'environnement étant le plus souvent dynamique, il est nécessaire de s'adapter aux modifications de celui-ci. Le robot peut se retrouver face à un autre robot qui l'empêche de se déplacer comme prévu. Les commandes doivent donc être modifiées en conséquence. Une alternative doit être déterminée. Les conditions ambiantes peuvent également avoir un impact sur la perception qu'a le robot de son environnement (conditions d'éclairage par exemple).
— La planification : il s'agit du calcul de la route à suivre pour rejoindre le point de coordonnées connues (*path-planning*). Ce calcul prend en compte la position du robot et la position des obstacles statiques et dynamiques connus avec une certaine précision, il faut déterminer le ou les chemin(s) possible(s) afin qu'il puisse atteindre son objectif. La planification des chemins doit tenir compte de la précision de la localisation précédemment estimée afin d'assurer une efficacité et une sécurité optimales.

La navigation inclut également le calcul de toute autre information relative au déplacement du robot, c'est à dire distance restant à parcourir, durée, vitesse de déplacement, heure d'arrivée estimée, etc.

Comme mentionné dans le chapitre 1, nous traitons, dans cette thèse, la problématique de localisation en utilisant principalement des informations visuelles.

2.1.3 Stratégies de navigation visuelle

Il existe de nombreuses méthodes de navigation, dont certaines sont inspirées du comportement des animaux. F. Bonin-Font *et al.* ont proposé [**?**] une classification en trois catégories. Les trois catégories proposées sont :

— les méthodes basées cartes, appelées *"Map-Based navigation"*. Dans ce cas, les cartes sont préalablement réalisées et créées pour le besoin de l'utilisateur. Les cartes utilisées peuvent être très différentes : il peut s'agir de modèles géométriques 3D plus ou moins riches, de

carte 2D plus ou moins détaillées, de graphes, etc. On distingue généralement les cartes topologiques et les cartes métriques [**?**] :

- une carte topologique représente les relations spatiales entre les différents lieux. Ces relations sont décrites sous la forme d'un graphe qui permet de calculer la trajectoire à suivre pour atteindre n'importe quel lieu présent dans ce graphe à partir d'un autre lieu présent dans ce graphe. Le robot ne peut pas se rendre à un lieu n'appartenant pas au graphe. Autrement dit, l'exploration d'une zone inconnue est impossible.
- dans le cas des cartes métriques, une information métrique est disponible entre les différents lieux. Ceci permet de calculer une trajectoire permettant d'atteindre la position voulue à partir de n'importe quelle autre position. Contrairement aux approches basées "cartes topologiques", les trajectoires planifiées permettent d'explorer des lieux inconnus, et ce, même si aucune trajectoire n'a été mémorisée sous la forme d'un lien.

— les méthodes basées cartes créées lors de la mission, appelées *"Map-Building-Based navigation"*. Les cartes sont réalisées lors du déplacement du robot. Ces systèmes utilisent des capteurs appropriés pour construire une carte de leur environnement (modèles géométriques ou carte topologique).

— les méthodes sans carte, appelées *"Mapless navigation"*. Ces systèmes n'utilisent aucune représentation explicite de l'environnement dans lequel le robot évolue, mais ils tentent de reconnaître des objets dans l'environnement ou de suivre des objets pour en déduire leur propre déplacement.

Ces différentes méthodes sont détaillées dans les sections 2.1.3.1, 2.1.3.2, et 2.1.3.3.

2.1.3.1 Navigation sans carte

Dans cette catégorie sont inclus tous les algorithmes qui n'utilisent aucune information *a priori* et qui ne construisent aucune représentation de leur environnement. Les déplacements des robots sont directement déduits des observations réalisées. Les méthodes les plus répandues qui font partie de cette catégorie sont :

- les méthodes basées reconnaissance d'objet. Les robots reçoivent une liste de commandes du type "Va à la porte" ou "Va devant le bureau qui est en face de toi". Pour cela, le robot identifie les éléments de son environnement (porte, bureau, fenêtre,...) et se dirige depuis la position courante vers l'objet préalablement spécifié. C'est une stratégie locale puisque l'objet à atteindre doit être visible.
- les méthodes basées guidage : Cette approche ressemble à la précédente approche. La position à atteindre n'est pas un objet mais un

point de l'espace qui est spécifié à l'aide des positions d'un ensemble de points, de motifs ou d'objets remarquables. Cette approche est également locale puisque l'ensemble considéré doit également être visible.

— les méthodes basées flot optique : la méthode la plus connue est celle développée par Santos-Victor [?] qui imite le comportement d'une abeille. La méthode consiste, pour le déplacement dans un couloir, à calculer le flot optique des images issues de deux caméras latérales, et à égaliser les deux flots optiques. Cette égalisation permet de maintenir le robot au centre du couloir.

— les méthodes basées apparence : L'idée consiste à stocker des images de l'environnement et à associer à chacune de ces images des commandes qui permettront au robot de se rendre à sa destination finale. La succession des différentes actions associées à chaque lieu reconnu permet de suivre un chemin qui permet d'atteindre la position souhaitée. Cette approche est considérée comme globale, puisque qu'elle permet d'atteindre un lieu sans aucune représentation de l'environnement de ce lieu. Par exemple, Gaussier *et al.* [?] ou Joulain *et al.* [?] ont entraîné un réseau de neurones pour faire la correspondance entre les différentes signatures visuelles (représentant différents lieux) et les actions.

2.1.3.2 Navigation à partir de cartes préalablement réalisées

Les solutions de navigation à partir de cartes (*"Map-Based navigation"*) ont recours à des cartes préalablement réalisées/construites créées pour le besoin de l'utilisateur. Ces modèles peuvent être plus ou moins détaillés : il peut s'agir d'un modèle 3D complet de l'environnement ou d'un simple graphe reliant un ensemble d'éléments de l'environnement. Le principe sous-jacent consiste à fournir au robot un ensemble de marques qu'il sera capable de repérer lors de sa navigation et qui lui permettra de se localiser. La problématique est de chercher et identifier visuellement ces marques. Une fois identifiées, le robot utilisera la carte fournie pour estimer sa position en mettant en correspondance la marque observée avec celle attendue présente dans la base de données. Un tel système visuel se décompose en quatre étapes :

1. Acquisition de l'information visuelle : une ou plusieurs caméras fournissent les images qui permettent de réaliser les traitements voulus.
2. Détection des marques ou des amers : en général, cela signifie extraire les coins et/ou les bords ou segmenter les images par une analyse des différences de gris, de couleurs ou de mouvements.
3. Mise en correspondance entre les observations et les prédictions : le système essaie d'identifier les amers observés avec ceux précédemment extraits et stockés en mémoire. De nombreuses méthodes sont proposées pour réaliser cette étape.

4. Calcul de la position : grâce aux correspondances, le système est capable d'estimer sa position.

La troisième étape est la plus difficile. Cette recherche peut être contrainte par une information *a priori* concernant les amers. Les différentes méthodes de navigation à partir de cartes sont des solutions de localisation globale ou absolue. Elles sont à différencier des méthodes de localisation incrémentale, pour lesquelles la position initiale du robot est connue. L'objectif de l'algorithme de vision est alors d'estimer le déplacement entre deux images successives pour estimer la nouvelle position à partir de la précédente. Ceci peut être réalisé, par exemple, par odométrie visuelle (section 2.2.1).

2.1.3.3 Navigation à partir de cartes réalisées lors de la mission

Avoir un modèle ou une carte avec les détails souhaités n'est pas toujours aisé. C'est pourquoi des solutions de navigation visuelle ont été proposées. Le robot au fur-et-à-mesure de l'exploration de son environnement construit sa propre carte. Il s'agit dans ce cas :

— de cartographier l'environnement dans lequel il évolue, c'est à dire de déterminer où sont les autres lieux par rapport à lui. Autrement dit, il s'agit de construire une carte exploitable en mémorisant des données acquises.
— de se localiser, c'est à dire de déterminer la position du robot au sein de la carte construite.

Cartographie et localisation sont interdépendants : utiliser une carte pour localiser le robot nécessite d'avoir une carte et, construire une carte nécessite d'estimer la position du robot par rapport à un morceau de carte construit précédemment. Dans le premier cas, une localisation précise est souhaitée. Dans le second cas, elle ne l'est pas. Les algorithmes de type SLAM pour *Simultaneous Localisation And Mapping*, présentés dans la section 2.2.2, appartiennent à cette catégorie.

2.1.4 Informations intrinsèques vs. extrinsèques

Pour naviguer, deux types d'information peuvent être utilisés [**?**] :

— les informations intrinsèques représentatives du mouvement du robot (vitesse, accélération, rotation des roues,etc.). Certaines de ces informations peuvent être fournies, par exemple, par une centrale inertielle. La position peut être estimée par intégration des accélérations. Cependant les erreurs commises sont accumulées et la précision de la position diminue donc avec le temps. Il n'est donc pas possible de faire confiance à ce type d'information sur de longues périodes.
— les informations extrinsèques : vision monoculaire ou stéréo, laser, sonars, etc. pour les robots. Il s'agit de toutes les observations faites par le robot et de sa perception de l'environnement. Les informations

extrinsèques peuvent permettre au robot de reconnaître directement le lieu où il se situe, ou de se localiser sur une carte métrique après interprétation de ces informations. Pour ce type d'informations, il n'y a pas de problème de dérive, elles sont stables dans le temps. Par contre, pour un capteur donné, la réponse du capteur peut être similaire pour deux endroits différents. Ce problème est appelé *perceptual aliasing* (Fig.2.3). D'autre part, une difficulté supplémentaire est liée aux changements dans la scène qui interviennent au cours du temps. Il peut s'agir par exemple des variations d'éclairage dus aux changements météo ou même saisonniers. Ce problème est appelé *perceptual variability* (Fig.2.4). Les systèmes développés doivent donc prendre en compte ces problèmes afin d'être le plus robuste possible. Les informations extrinsèques peuvent également permettre au robot de percevoir les obstacles fixes ou mobiles dans un environnement proche.

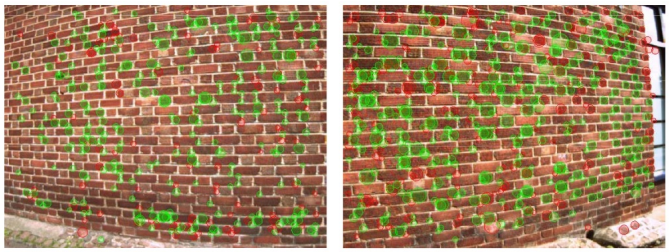


FIGURE 2.3 – Illustration du problème appelé *perceptual aliasing* : deux caméras photographiant deux scènes différentes donnent deux images similaires. Figure réalisée par M.Cummins and P. Newman.

FIGURE 2.4 – Illustration du problème appelé *perceptual variability* : deux caméras photographiant une scène à deux instants différents donnent deux images non similaires. Figure réalisée par M.Cummins and P. Newman.

Comme l'a affirmé Cox dans [**?**], il est nécessaire, afin de naviguer de façon fiable et robuste, de combiner ces deux types d'information. Les informations extrinsèques doivent être utilisées pour compenser la dérive des informations intrinsèques, et celles-ci doivent être utilisées pour éviter toute ambigüité avec les informations extrinsèques.

2.1.5 Stratégies de localisation

Le robot, pour se localiser, doit posséder deux aptitudes :

— il doit être capable de se localiser globalement, c'est à dire dans un repère absolu, sans savoir où il était précédemment. Cette fonctionnalité est indispensable si l'on souhaite que le robot soit capable de réaliser une mission long-terme. Ceci permet également de résoudre le problème du robot perdu (*"lost-robot problem"* ou *"the drop-off problem"*).

— il doit être capable de se localiser localement par rapport à son environnement proche afin d'éviter les obstacles statiques et dynamiques.

Il est à noter que ces deux aptitudes sont nécessaires, mais qu'elles n'exigent pas forcément la même précision de localisation. D'autre part, ces deux aptitudes ont des propriétés duales dans le cas de cartes métriques. La localisation globale est discrète et qualitative car, étant donné les informations extrinsèques, le robot doit identifier les éléments de perception qui lui permettront de choisir parmi plusieurs hypothèses de lieux distincts. La localisation locale est continue et quantitative car, étant donné les informations extrinsèques, la position du robot est mise à jour à l'aide de sa perception de l'environnement. Dans le cas de cartes topologiques ces deux aptitudes ont des propriétés similaires parce qu'elles permettent toutes les deux de déterminer le nœud qui explique au mieux la position du robot.

Dans [**?**], les différentes stratégies de localisation sont classifiées ainsi :

— Les stratégies qui permettent au robot de déduire directement sa localisation en utilisant uniquement les informations extrinsèques. Dans ce cas, les informations doivent être suffisantes pour permettre au robot de se localiser de manière globale. Ces stratégies lui permettent de se localiser dans des environnements sans *perceptual aliasing*.

— Les stratégies qui permettent au robot de suivre une seule hypothèse en utilisant à la fois les informations extrinsèques et intrinsèques. Si une erreur intervient dans l'estimation de la position, le robot est définitivement perdu. Ces stratégies permettent au robot de se localiser dans des environnements avec *perceptual aliasing*.

— Les stratégies qui permettent au robot de suivre plusieurs hypothèses en utilisant à la fois les informations extrinsèques et intrinsèques. Un ensemble d'hypothèses de localisation est maintenu, ce qui en cas d'erreur, permet au robot de corriger l'erreur faite. Ces stratégies permettent au robot de se localiser dans des environnements avec *perceptual aliasing*.

2.1.6 Stratégies d'apprentissage de carte et de planification

Un système de navigation, en plus de sa fonctionnalité de localisation, doit posséder les fonctionnalités d'apprentissage de carte et de planification. Nous ne détaillons pas ici les différentes stratégies existantes car elles sortent de la problématique de cette thèse énoncée dans le chapitre 1. Le lecteur souhaitant un panorama des différentes méthodes existantes pourra se référer à [**?**]. Il est à noter que la planification doit prendre en compte une carte globale et une carte locale, représentant l'environnement proche du robot. La carte globale permet de déterminer la trajectoire permettant d'atteindre la position finale souhaitée. La carte locale permet d'éviter les obstacles statiques et dynamiques. La planification est un problème de trajectographie. Ces deux cartes sont complémentaires et indispensables pour le développement d'une solution de navigation autonome.

2.2 Solutions de localisation visuelle locale

Dans cette section, nous présentons brièvement les principaux algorithmes de localisation visuelle locale de la littérature sans détailler les nombreuses variantes qui ont été proposées.

2.2.1 Odométrie visuelle

L'odométrie visuelle consiste à estimer le propre déplacement du robot en utilisant une ou plusieurs caméras. L'objectif de l'odométrie visuelle est l'extraction du mouvement apparent, afin de calculer les différentes poses de la caméra, c'est à dire la trajectoire du robot. La trajectoire finale peut être affinée *a posteriori* avec un algorithme d'ajustement de faisceaux (cf. section 2.2.4). Pour fonctionner, l'odométrie visuelle nécessite que les images acquises successivement aient un recouvrement.

Remarquons que l'odométrie visuelle est un cas particulier des solutions qui consistent à déterminer la pose relative d'une caméra et la structure 3D d'un environnement à partir d'un jeu d'images, appelé *Structure From Motion* (SFM) dans la communauté de la vision. L'odométrie visuelle, contrairement à la SFM, consiste à estimer séquentiellement et en temps réel le déplacement 3D de la caméra. Le problème d'estimation de son propre déplacement a commencé au début des années 1980 [**?**]. Les premières études ont été réalisées pour les rovers dédiés à l'exploration de planètes autre que la terre et ont donc été soutenues par la NASA, notamment pour son programme d'exploration de la planète Mars. La solution proposée, encore utilisée de nos jours, repose sur 4 modules (Fig. 2.5) :

1. Un détecteur de points ;
2. Une mise en correspondance des points par corrélation entre deux images successives ;

3. Une suppression des mises en correspondance jugées comme étant fausses, généralement par filtrage de type RANSAC [**?**] ;
4. Le calcul de la transformation rigide à partir des points mis en correspondance.

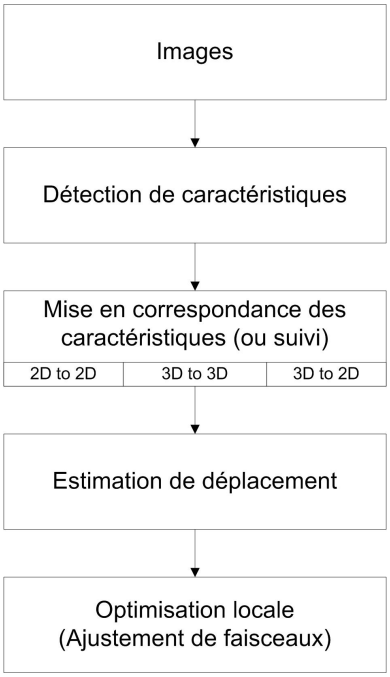


FIGURE 2.5 – Architecture des solutions d'odométrie visuelle.

La solution proposée par Moravec [**?**] est une solution stéréo (il s'agit d'une caméra qui se déplace sur un rail). Des solutions monoculaires ont été développées ultérieurement. L'inconvénient des solutions monoculaires est que le mouvement est estimé à un facteur d'échelle inconnu. L'échelle doit alors être estimée grâce à des mesures directes, comme par exemple la mesure de la taille d'un objet reconnu dans la scène, grâce à des contraintes de mouvement, ou encore grâce aux mesures effectuées par d'autres capteurs, comme par exemple la centrale inertielle. Deux axes de recherche ont été poursuivis au fil du temps : l'un monoculaire, l'autre stéréo. Pour une synthèse des principaux algorithmes proposés, le lecteur pourra se référer à [**?**].

L'association temporelle des points d'une image à l'autre est l'étape critique. De mauvaises associations peuvent engendrer des incohérences. Lors de mouvements brusques, d'occultations, de flou, le suivi des points échoue et l'estimation du déplacement est corrompue. Le fonctionnement de tels algorithmes étant basé sur un calcul incrémental du chemin de la caméra, les erreurs s'accumulent et une dérive entre le chemin réellement parcouru et le chemin estimé apparaît. Pour certaines applications, il est indispensable

de minimiser cette dérive. Cela peut être effectué par une optimisation sur les dernières poses estimées des caméras avec un algorithme d'ajustement de faisceaux (cf. section 2.2.4). Konolige en 2007 [**?**] a démontré que cela permettait pour un chemin de 10 km de réduire l'erreur sur la position finale d'un facteur 2 à 5. Évidemment, la dérive peut également être réduite par l'utilisation d'autres capteurs comme un GPS, un laser, ou même avec une centrale inertielle.

2.2.2 Simultaneously Localisation And Mapping

Un algorithme de type SLAM (*Simultaneaous Localization And Mapping*) permet, comme son nom l'indique, d'obtenir une carte cohérente de l'environnement dans lequel le robot évolue ainsi qu'une estimation de sa position au sein de cette carte. A partir de chaque image issue d'une caméra calibrée, l'algorithme détermine en temps réel la pose de la caméra, c'est à dire la localisation et l'orientation de la caméra dans un repère fixé lors de l'initialisation de l'algorithme, ainsi qu'une carte 3D de la scène représentée par un ensemble de points 3D appelés amers. La trajectoire de la plate-forme mobile et la localisation des amers sont estimées au fil de l'eau et ne nécessitent aucune information *a priori*. Deux approches sont à distinguer pour le SLAM visuel : les approches basées filtrage et les approches basées images clé. Une approche basée filtrage est par exemple la solution temps réel proposée par Davison [**?**], notée SLAM-EKF. Elle met en œuvre un filtre de Kalman pour prédire le mouvement de la caméra et la position des points 3D (avec une certaine incertitude). Pour cela, les points d'intérêts de l'image précédente sont appariés avec les points d'intérêts de l'image courante grâce à la prédiction réalisée par le filtre de Kalman. Après cet appariement, l'erreur de prédiction est corrigée. Cet algorithme fonctionne à la fréquence image. FastSLAM proposée par M. Montemerlo [**?**] est une autre solution basée filtrage. Elle utilise un filtre particulaire. Parmi les approches basées images clé, on peut citer par exemple celle de G.Klein *et al.* appelée PTAM. Les auteurs proposent une solution SLAM temps réel mettant en œuvre une optimisation globale et locale de type ajustement de faisceaux. Une comparaison de ces différentes approches est présentée dans [**?**].

En théorie, la carte construite devrait permettre de savoir si le robot est de nouveau à un endroit déjà visité mais dans la pratique les dérives accumulées lors de la construction de la carte rendent impossible cette fonctionnalité [**?**]. Ceci est dû notamment à un mauvais appariement des points d'intérêt. Pour corriger ce problème, une solution consiste à utiliser/injecter des connaissances supplémentaires sur la trajectoire estimée ou sur la géométrie de l'environnement. Pierre Lothe *et al.* [**?**] ont proposé par exemple d'estimer une déformation non-rigide de la reconstruction issue du SLAM de façon à la mettre en correspondance avec un modèle CAO approximatif. Cette solution est une méthode *a posteriori* pour des applications de carto-

graphie mais elle n'est pas adaptée pour un problème de navigation en temps réel. Une autre solution consiste à utiliser un algorithme dit de fermeture de boucle, comme l'ont proposé Strasdat *et al.* [?]. Cette catégorie d'algorithmes est présentée dans la section 2.2.3.

2.2.3 Fermeture de boucle

La fermeture de boucle (*Loop-Closure*) consiste à détecter lorsqu'un robot est déjà passé à un endroit. Un algorithme de fermeture de boucle est un algorithme de reconnaissance des lieux déjà visités qui permet de corriger les défauts de la carte créée par les algorithmes de type SLAM et ainsi de maintenir une carte spatialement cohérente.

B. Williams *et al.* ont proposé de classer les différents algorithmes de fermeture de boucle en trois catégories principales [?].

- les algorithmes de type "Image-to-Image" : ce type d'algorithme consiste à mettre en correspondance la dernière image acquise avec les images précédemment acquises. Citons ici l'algorithme FAB-MAP proposé par M. Cummins et P. Newman [?] et présenté ci-après.
- les algorithmes de type "Image-to-Map" : ce type d'algorithme consiste à mettre en correspondance les caractéristiques d'une carte avec celles de la dernière image acquise. B. Williams *et al.* [?] ont proposé une telle méthode.
- les algorithmes de type "Map-to-Map" : ce type d'algorithme consiste à mettre en correspondance les caractéristiques de deux cartes locales en prenant en compte leur apparence et leur position relative. Clemente *et al.* ont proposé dans [?] une telle solution.

Une comparaison de ces différentes approches peut également être trouvée dans [?].

Un algorithme de type fermeture de boucle ne repose donc pas sur l'existence préalable d'une base de caractéristiques. La base de caractéristiques est construite au fur et à mesure du déplacement du robot. Les instants d'acquisition sont proches et la même caméra est utilisée pour acquérir les images de la base et les images requête.

L'algorithme FAB-MAP M. Cummins et P. Newman ont proposé un algorithme de fermeture de boucle temps réel, basé apparence pour reconnaître un lieu, appelé FAB-MAP pour *Fast Appearence Based MAPping*) [?][?][?]. L'algorithme repose sur les occurrences des caractéristiques image pour détecter si les deux images représentent ou non la même scène. Une faible probabilité d'appartenance au même lieu est attribuée lorsque les caractéristiques sont identiques mais peu discriminantes. La méthode proposée s'inspire des sacs de mots visuels (*Bag of Visual Words - BoVW*) [?][?] et elle l'étend en apprenant le fait que des ensembles de caractéristiques sont susceptibles

d'apparaître ou disparaître ensemble. Autrement dit, une corrélation entre mots visuels est apprise. Cette corrélation entre mots visuels est due au fait qu'un ensemble de caractéristiques appartient à un objet donné. La solution proposée permet également de déterminer s'il s'agit d'un nouveau lieu et dans ce cas de l'incorporer dans la carte. La complexité de l'algorithme est linéaire avec le nombre de lieux dans la carte. Une solution, basée FAB-MAP, et incorporant des informations odométriques a été proposée dans [**?**].

2.2.4 Ajustement de faisceaux

L'algorithme d'ajustement de faisceaux (*Bundle Adjustment*) est un algorithme d'optimisation non-linéaire des moindres carrés, qui permet d'affiner des valeurs précédemment estimées. En vision, cet algorithme est utilisé pour affiner conjointement les amers (points 3D) et les paramètres de vue (pose de la caméra et/ou calibration). Les valeurs à optimiser doivent être préalablement évaluées. Pour cela, une fonction de coût doit être définie. Pour le SLAM, l'algorithme d'ajustement de faisceaux cherche à minimiser l'erreur de reprojection entre les points observés et les points prédits, ce qui s'exprime comme une somme de carrés de valeurs. Les mesures et les paramètres sont considérés comme étant distribués selon une gaussienne, et le problème est donc résolu par une optimisation non linéaire des moindres carrés (comme l'algorithme de Levenberg-Marquardt [**?**][**?**] ou par la méthode de Gauss-Newton). L'optimisation est globale sur N images clés. Mouragnon *et al.* ont proposé une solution temps réel de type SLAM par ajustements de faisceaux réalisés localement [**?**].

2.2.5 Limites des approches locales

Un algorithme de fermeture de boucle couplé à un algorithme de type SLAM, éventuellement couplé à un ajustement de faisceaux pour "robustifier" la solution, ne peut être vu comme une solution de localisation long-terme car des limites demeurent. Une telle solution proposée par la communauté robotique est la réponse au problème suivant : un robot placé dans un environnement inconnu peut-il construire une carte cohérente pour se déplacer ? Les limites d'une telle solution sont les suivantes :

- La carte construite peut devenir inexploitable de par sa taille. La taille de la carte construite est, en effet, de complexité croissante. Il faut donc construire des sous-cartes, ce qui est complexe à gérer.
- La localisation n'est pas absolue. Les orientations et positions de la caméra ainsi que la position des points 3D de la scène sont calculés dans un repère fixé arbitrairement. Il s'agit le plus souvent du repère caméra associé à la première position de la caméra.
- Les solutions de type fermeture de boucle visant à corriger les dérives et décrochages d'un algorithme de type SLAM sont inadaptées au scé-

nario que nous avons énoncé dans le chapitre 1. En effet, une solution de type fermeture de boucle suppose que le robot repasse dans des lieux déjà visités. Dans le cas de notre scénario, il est peu probable que cela se réalise, à moins que le robot ne soit perdu.

— La carte construite et la localisation sont obtenues à un facteur d'échelle près lors de l'emploi d'une caméra monoculaire. L'estimation de ce facteur d'échelle n'est pas aisée. Il peut être calculé à partir d'une mesure image directe (par exemple à partir de la taille d'un élément de la scène) ou à partir de mesures effectuées à l'aide d'autres capteurs (par exemple centrale inertielle ou capteur de profondeur). Il est à noter que l'emploi d'une caméra stéréo ne permet pas forcément d'estimer ce facteur d'échelle, car lorsque la distance entre les caméras et la scène est largement supérieure à la distance entre les deux caméras, alors une caméra stéréo est équivalente à une caméra monoculaire. Aucune information métrique fiable n'est donc disponible.

2.3 Solutions de localisation visuelle globale

Dans cette section, nous présentons un ensemble de systèmes de l'état de l'art ayant pour objectif la géolocalisation, à partir duquel nous identifierons les éléments méthodologiques importants à retenir pour notre application.

2.3.1 État de l'art

Hays *et al.* [?] ont proposé une solution, appelée *IM2GPS*, pour estimer la position d'une photographie à partir d'une seule image. L'objectif est de retrouver la scène représentée parmi une base d'images géoréférencées constituée de 6 millions d'images réparties sur l'ensemble de la terre, et représentant des lieux touristiques connus des 20 villes les plus peuplées. La solution repose sur la mise en correspondance de nombreux descripteurs (GIST [?], histogrammes de couleurs, histogrammes des longueurs et des angles des lignes droites, etc.). Les k images les plus similaires sont retrouvées à l'aide d'une distance de type χ^2 pour les histogrammes et d'une norme ℓ_1 pour les autres descripteurs. Chacune des k images vote pour un lieu. La zone géographique ayant obtenue le plus de votes détermine le position approximative dans le monde. Même si la solution proposée est simple, ils démontrent, via leur étude, qu'il est possible d'estimer la position géographique d'une image, sans autre information que l'image requête.

A. Hakeem *et al.* [?] ont proposé une solution pour estimer la trajectoire d'une caméra à partir d'un jeu d'images géoréférencées. Trois étapes sont requises. Tout d'abord les points d'intérêt et descripteurs SIFT des images acquises sont extraits et mis en correspondance avec ceux de la base d'images géoréférencées. Une matrice de similarité est ainsi obtenue. Pour

chaque image acquise, l'image ayant la similarité la plus élevée est sélectionnée et utilisée pour calculer la pose de la caméra par estimation de la matrice essentielle (les paramètres intrinsèques des deux caméras sont donc requis). La position relative étant obtenue à un facteur d'échelle près, une triangulation est réalisée entre l'image requête et deux images adjacentes de la base pour résoudre l'ambiguïté. Enfin, les positions aberrantes sont supprimées et une interpolation utilisant des b-spline est effectuée.

Schindler *et al.* [?] ont proposé un algorithme pour améliorer la précision de la fonctionnalité de recherche d'images pour de grandes bases d'images ($3 \cdot 10^4$ images) par optimisation d'un arbre et par sélection des mots visuels à inclure au sein de l'arbre. La solution proposée permet de se localiser au sein d'une ville, mais les ressources requises restent prohibitives.

D. Filliat [?] a proposé une solution de localisation interactive par apprentissage incrémental qui permet à un robot de se localiser dans les pièces d'un appartement. La solution proposée ne donne pas une localisation précise, mais elle permet de reconnaître la pièce dans laquelle le robot évolue pour des points de vue très différents. Un dictionnaire de mots visuels est créé via l'algorithme k-means avec un rayon constant. Les descripteurs utilisés sont des descripteurs de couleur, de texture et des gradients orientés locaux. A chaque mot du dictionnaire est attribué une position, c'est à dire une pièce. Pour chacun des descripteurs de l'image, le mot visuel le plus proche est déterminé. Ce mot visuel peut ne pas exister si la distance entre ce mot visuel et le descripteur dépasse un seuil. Chaque mot visuel vote pour une pièce.

Knopp *et al.* [?] proposent une approche constitué de deux étapes : 1) recherche des images les plus semblables à l'image requête avec une solution de type sac de mots visuels, 2) classement de ces images similaires à l'aide d'un filtrage géométrique de type RANSAC [?]. L'originalité de leur solution est lors de la construction du dictionnaire : les mots-visuels identifiés comme non-discriminants sont supprimés pour éviter les mots visuels ambigus. Il s'agit, par exemple, des mots visuels décrivant les arbres ou les marquages au sol des routes.

Sattler *et al.* [?] ont développé une méthode de géolocalisation basée sur des caractéristiques 2D et 3D obtenues par SFM (*Structure From Motion*). Pour chaque caractéristique 3D est associée une caractéristique 2D permettant de retrouver les caractéristiques 3D correspondants aux caractéristiques 2D des images requête. Les caractéristiques 3D ainsi retrouvées permettent d'estimer de façon robuste (i.e. à l'aide de l'algorithme RANSAC [?]) la pose de la caméra et d'en déduire la position du robot. Les auteurs ont ensuite proposé dans [?] une autre méthode basée à la fois sur la mise en correspondance de caractéristiques 2D-3D et 3D-2D. Cette double mise en correspondance permet de mettre à profit les avantages des deux méthodes. Ces méthodes nécessitent un modèle 3D de l'environnement. Plus la taille de l'environnement est important, plus le nombre de points 3D et de descripteurs associés

est important, ce qui rend le temps de mise en correspondance très long.

Zamir *et al.* [?] ont proposé une méthode pour annoter *a posteriori* les vidéos du web. L'objectif poursuivi est de localiser à chaque instant la caméra. Pour cela, les descripteurs SIFT de la base de caractéristiques sont indexés par un arbre. Pour chacun des descripteurs de l'image requête, le plus proche voisin est trouvé. Les votes les plus faibles sont supprimés et chaque vote considéré comme fiable vote pour une image géoréférencée. Tous les votes sont accumulés sur une carte 2D et sont ensuite filtrés par un filtre spatial gaussien. L'image géoréférencée avec le nombre de votes le plus élevé détermine le lieu. Dans [?], la méthode décrite dans [?] est améliorée en interprétant la carte de l'ensemble des votes comme une vraisemblance. Celle-ci est ensuite exploitée par un filtre temporel Bayésien pour estimer la trajectoire. Plus récemment, en partant du constat que les méthodes de géolocalisation étaient essentiellement basées sur les descripteurs locaux, les auteurs ont proposé une amélioration pour ajouter, lors de la phase de mise en correspondance d'images, un descripteur global de type GIST ou un histogramme de couleurs [?].

H. Badino et T. Kanade [?] ont proposé une solution de localisation d'un véhicule utilisant des caractéristiques visuelles 2D et 3D. Leur solution nécessite, dans une phase préliminaire, la construction d'une carte compacte, décrite sous la forme d'un graphe. Les nœuds incluent la position du véhicule et les caractéristiques 3D. La distance entre chaque nœud est constante. La carte est construite à l'aide d'un véhicule équipé d'un GPS, de deux caméras latérales et deux LIDAR latéraux. Lors de l'étape de mise en correspondance des caractéristiques visuelles 2D et 3D acquises avec celles de la base précédemment construite, un filtre Bayésien est mis en œuvre. Les mêmes capteurs sont utilisés durant l'étape de construction de la carte et durant l'étape de localisation.

M. Milford et G. Wyeth [?] ont proposé une solution de localisation par reconnaissance de séquences d'images cohérentes à l'aide de mesures de similarité locale entre les dernières images acquises et les images pouvant représenter la même scène. Les similarités sont basées sur le calcul de la somme des valeurs absolues (*Sum of Absolute Difference - SAD*) entre des condensés (*patches*) de l'image acquise avec les M condensés des images situées dans un voisinage de la dernière position connue. Ces condensés d'images sont les images réduites par sous-échantillonnage puis normalisées pour produire un condensé au contraste amélioré. L'objectif poursuivi est d'être robuste à d'importants changements visuels (jour/nuit, ensoleillé/nuageux,...). Leur solution suppose que la vitesse du véhicule soit constante. Dans [?], la solution est améliorée pour fournir une invariance à la vitesse du véhicule, mais elle reste inadaptée lors d'importants changements de point de vue.

Doersch *et al.* [?] ont, quant à eux, proposé une méthode pour identifier le jeu de caractéristiques visuelles, notamment les éléments architecturaux qui caractérisent une ville et qui permettent donc, pour une image donnée,

d'identifier la ville dans laquelle l'image a été prise.

Liu *et al.* [?] proposent une solution de localisation visuelle précise d'une photo de *smartphone* qui exploite la localisation approximative fournie par un GPS et des modèles 3D préalablement construits à partir d'images géo-référencées téléchargées sur Flick et Panoramio. La solution proposée inclut une méthode pour estimer également la position du téléphone par rapport à la scène observée (distance et direction). Leur solution est basée sur une solution de recherche d'images de type sacs de mots visuels suivie d'une vérification géométrique. Pour chacune des images géoréférencées pouvant potentiellement correspondre, le modèle 3D associé est utilisé. Les points 2D de l'image requête sont mis en correspondance avec les points 3D de chaque modèle. Ceci permet de déterminer la meilleure image géoréférencée et également de calculer la pose relative entre les deux images.

Dans [?], les auteurs proposent une solution de localisation basée sur le descripteur GIST [?][?] pour des images panoramiques. Leur solution est une solution de fermeture de boucle. Le panorama est constitué de quatre images couvrant 360°horizontalement et 127°verticalement. Le descripteur GIST du panorama est constitué des quatre descripteurs GIST calculés pour chacune des quatre images. La complexité est linéaire avec le nombre d'images dans la base. Pour de très grandes bases d'images, ce n'est pas acceptable, c'est pourquoi les auteurs proposent d'autres mesures de similarité moins complexes : l'une basée sur la quantification des descripteurs et l'autre basée sur les *k-d tree*.

Dans [?], les auteurs proposent une solution de localisation pour drones (ou *Unmanned Aerial Vehicle - UAV*). Leur solution concerne la mesure de la similarité visuelle. Pour être robuste à d'importants changements de point de vue, ils génèrent des vues artificielles à partir des images de la base (images *image Google Streetview*) et des images acquises. La similarité visuelle utilisée est basée sur le nombre de descripteurs mis en correspondance après une vérification géométrique. La mise en correspondance est réalisée à l'aide des *k-d tree*.

Dans [?], les auteurs ont considéré le problème de reconnaissance de lieux comme une tâche de classification. Un classifieur pour chaque image de la base est appris en utilisant les approches de type exemplar-SVM [?]. La principale contribution du papier est la méthode de calibration de tous les classifieurs appris en utilisant principalement des exemples négatifs afin de pouvoir comparer les scores de tous les classifieurs.

Dans [?], Fang *et al.* propose une solution appelée GIANT dont le principe consiste à créer par apprentissage un dictionnaire d'éléments qui sont caractéristiques d'une ville et discriminants. Le dictionnaire est réalisé par apprentissage à l'aide d'un SVM latent [?]. La variable latente binaire indique si la région de l'image contribue ou pas à la reconnaissance d'un lieu. La solution proposée permet de déterminer dans quelle ville a été prise la photographie.

La solution proposée par M.A. Brubaker *et al.* [**?**] est basée sur l'appariement des mesures odométriques visuelles avec une carte routière 2D. La carte routière est représentée sous la forme d'un graphe orienté et une approche probabiliste est définie afin de naviguer au sein de ce graphe.

Très récemment, enfin, en partant du constat que les caractéristiques locales ne fonctionnent bien que lorsque la scène est observée avec des conditions similaires à celles présentes lors de la réalisation de la base d'images, C. McManus *et al.* ont proposé d'apprendre un jeu de détecteurs spécifiques à chaque lieu, afin d'identifier les structures qui le caractérisent.

2.3.2 Synthèse

Pour se localiser visuellement de nombreuses approches ont donc déjà été proposées. La plupart des solutions reposent sur la mise en correspondance de caractéristiques acquises avec des caractéristiques géolocalisées. Les solutions visuelles de localisation comportent généralement trois modules :

1. un algorithme de détection et de description de caractéristiques visuelles,
2. un algorithme permettant de mettre en correspondance les caractéristiques visuelles extraites de l'image requête avec les caractéristiques géoréférencées. Cette étape est la plus difficile à cause du *perceptual aliasing* et de la *perceptual variability*.
3. un algorithme de filtrage géométrique, spatial ou temporel dont l'objectif est de résoudre les cas ambigus, c'est à dire les cas ou plusieurs images sont similaires à l'image requête.

Les approches proposées différent par :

— le système d'acquisition utilisé. Généralement, il s'agit d'une unique caméra mais des capteurs supplémentaires peuvent être utilisés. Dans [**?**] par exemple, les auteurs utilisent deux caméras latérales et deux LIDAR.

— le type et la quantité d'information que l'on accepte d'utiliser. Les informations à disposition et collectées peuvent être plus ou moins riches (de type 2D uniquement ([**?**][**?**][**?**]), ou mixtes 2D et 3D ([**?**][**?**][**?**]). Dans [**?**] par exemple, les auteurs exploitent les informations visuelles ainsi qu'une carte routière 2D. Elles peuvent être également plus ou moins nombreuses : par exemple dans [**?**], les auteurs utilisent les images du service web *Google Streetview* et ils ne disposent que d'une seule image par lieu. Par contre dans [**?**][**?**] par exemple, les auteurs ont recours au service web *Flickr* et exploitent plusieurs images représentant la même scène pour généralement calculer des caractéristiques 3D.

— le type de détecteur et descripteur utilisé. Généralement, il s'agit de détecteurs de points et de descripteurs locaux (SIFT ou SURF) mais il peut s'agir également de descripteurs globaux [?]. Les descripteurs peuvent également être combinés comme dans [?] où les descripteurs locaux sont combinés avec un descripteur global de type GIST [?][?]. Les détecteurs utilisés peuvent également être appris comme par exemple dans [?].
— la technique de filtrage utilisée. Le filtrage peut être géométrique : ce type de filtrage supprime les mises en correspondance de points erronées par l'estimation d'une transformation géométrique entre les deux images considérées comme similaires. Ce type de filtrage s'appuie généralement sur l'algorithme RANSAC [?]. Le filtrage peut être temporel : dans ce cas une séquence d'images successives est mise en correspondance au lieu d'une unique image. [?][?] ont utilisé par exemple un filtre Bayésien. Le filtrage peut être spatial : chaque caractéristique de l'image requête vote pour une position sur une carte. L'endroit ayant le plus grand nombre de votes est la position estimée, comme dans [?].

Certaines des approches précédemment présentées apprennent, à l'aide d'algorithmes issus de la communauté du *machine learning*, les spécificités de chaque lieu comme par exemple [?][?][?][?][?]). Dans [?] par exemple, les auteurs ont recours à un SVM latent.

Enfin, il est également possible de différencier l'ensemble de ces méthodes selon un critère opérationnel : la précision de localisation attendue. Nous distinguons trois classes : localisation à l'échelle du monde ([?][?][?]), localisation à l'échelle d'une ville ([?][?]), ou localisation à l'échelle d'une rue ([?][?][?][?]).

2.4 Notre approche

Comme énoncé dans le chapitre 1, la problématique traitée par cette thèse est celle de la localisation visuelle précise à des fins de navigation autonome au sein d'un environnement urbain. Ce besoin de localisation a pour objectif de permettre à un robot de parcourir un trajet, préalablement spécifié, de plusieurs kilomètres.

Compte-tenu des problèmes afférents aux algorithmes de localisation locale (mentionnés dans la section 2.2.5) et de notre objectif de navigation long-terme, la solution proposée doit permettre une localisation absolue. Un repère absolu est un repère fixe, connu de tous, et, en ce qui nous concerne, le repère dans lequel la mission est spécifiée. Il est souhaitable que la solution proposée utilise le moins d'information possible et que ces informations soient facilement disponibles. Ce problème peut être résolu par la mise en correspondance de la dernière image acquise par le robot avec l'une des

FIGURE 2.6 – Exemple d'images représentant la même scène et illustrant la difficulté à retrouver parmi une base d'images géoréférencées (à droite) l'image représentant la même scène que l'image requête (à gauche). (a) : Difficulté liée à la présence de nouveaux objets dans la scène - (b) : Difficulté liée aux ombres portées - (c) : Difficulté liée au point de vue et aux ombres portées- (d) : Difficulté liée aux variations d'illumination.

images géoréférencées. Il s'agit alors d'un problème de mise en correspondance d'images représentant les mêmes lieux, ou autrement dit, de reconnaissance de lieux. Dans notre contexte, la recherche d'images s'avère difficile de part l'extrême variabilité des conditions d'acquisition entre les deux types d'images (Fig.2.6) : systèmes d'acquisition différents (focales différentes, résolutions différentes,...), scènes dynamiques, acquisition des images à des instants différents (ombres portées, influence des conditions météorologiques...). La plate-forme visée étant des petits drones, nous nous plaçons dans le cas monoculaire.

Face à ces difficultés, nous proposons dans cette thèse une solution qui repose sur deux principes qui ont été récemment explorés dans l'état de l'art. Le premier principe, que l'on a retenu, consiste à proposer une solution basée sur l'appariement d'une séquence d'images, comme par exemple dans [**?**][**?**][**?**]. De telles solutions permettent de rendre plus robuste l'estimation de la position réalisée en intégrant des contraintes spatio-temporelles. Le second principe consiste à déterminer les spécificités visuelles, par exemple les éléments discriminants, d'un lieu donné. D'autre part, nous supposons que la position approximative du robot et que l'incertitude associée sont connues. Comme évoqué dans le chapitre 1, même si un système de type GNSS n'est pas fiable (signaux radio temporairement masqués) et peu précis (particulièrement en milieu urbain), il fournit une information qu'il convient d'exploiter. Nous ne traitons donc pas dans cette thèse l'aspect "large échelle" qui consiste à retrouver une image parmi plusieurs milliers ou millions d'images. L'indexation pour des recherches rapides ne fait donc pas partie de nos contraintes. Cette approche permet de restreindre les images géoréférencées utilisées pour l'appariement de la séquence d'observations avec les images géoréférencées. Une fois la position approximative du robot estimée à partir de l'appariement d'images, nous proposons d'affiner la position grâce à l'estimation de la transformation rigide (rotation et translation) entre les deux images représentant la même scène. La solution que nous proposons est illustrée par la figure 2.7.

Dans le détail, nous proposons, en ce qui nous concerne, pour la mise en correspondance de la séquence d'observations, de mettre en oeuvre une chaine de Markov à états cachés. Elle permet de prendre en compte de manière flexible à la fois des informations visuelles et des informations odométriques. Ces mesures odométriques peuvent être fournies, par exemple, par une centrale inertielle dont les incertitudes de mesures sont connues. En ce qui concerne les mesures de similarités visuelles, nous proposons une solution basée *metric learning*, appelée Exabal pour *EXAmplar BAsed metric Learning*. Notre solution apprend une distance visuelle permettant de discriminer une image donnée de ses images voisines, grâce, notamment à la sélection de mots visuels discriminants. L'état de l'art réalisé démontre que le choix des mots visuels à considérer est important [**?**][**?**][**?**][**?**][**?**][**?**].

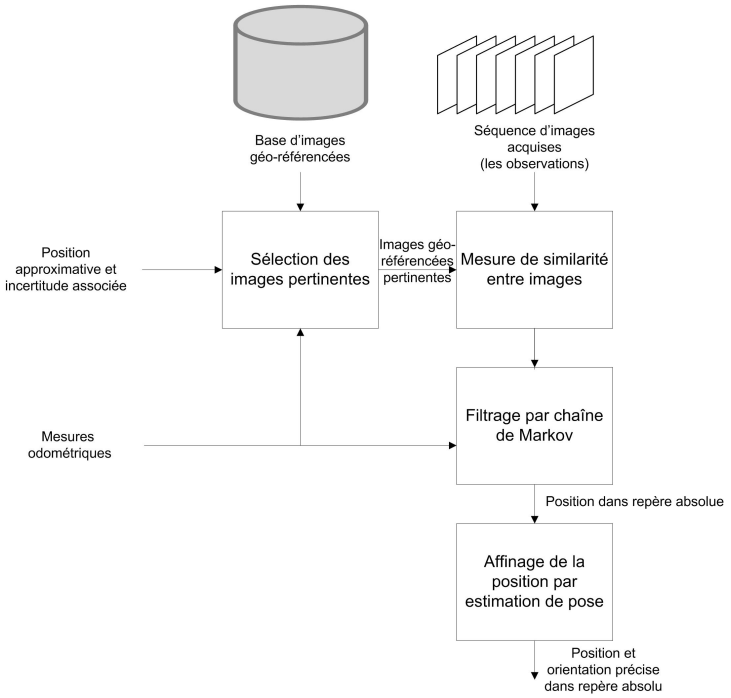


FIGURE 2.7 – Schéma de principe du système proposé.

Notre solution appartient aux solutions basées carte et basée apparence. La carte est de type métrique puisque les distances entre chacune des images de la base sont connues. La stratégie que nous proposons met en œuvre à la fois des informations extrinsèques et intrinsèques.

2.5 Bases d'images et protocoles d'évaluation

Pour nos travaux, nous avons besoin d'une base constituée d'images géo-référencées acquise dans un milieu urbain et d'un ensemble d'images requête. Pour chacune des images requête, il existe, parmi la base d'images géoréférencées, une image représentant la même scène mais ayant été acquise à des instants différents pour garantir des changements visuels importants.

Lors de nos études, nous utilisons une base principale construite à partir d'une base publique appelée "Pittsburgh" [**?**] et fournie à des fins de recherche par la société *Google*. Cette base a été acquise par un véhicule se déplaçant dans la ville de Pittsburgh aux États-Unis. Tous les 1 mètre, une image géoréférencée est fournie avec la position GPS. La trajectoire correspondante mesure 11 km (Fig.2.8). Pour nos expérimentations, les images ont été redimensionnées de telle façon à ce que leur définition soit égale à la définition des images obtenues via le service web *Google Streetview*, c'est à dire 640x480 pixels. Afin de garantir le fait qu'il existe pour chaque image re-

FIGURE 2.8 – Position des images de la base d'images géoréférencées "Pittsburgh".

quête, une image dans la base d'images géoréférencées représentant la même scène, nous avons imposé, lors de la construction de notre base d'image géoréférencées, un léger recouvrement entre deux images successives. Nous avons échantillonné les images de ce corpus pour ne retenir qu'une image tous les 5 mètres. La dernière partie de la séquence d'images n'a pas été retenue car l'environnement dans lequel évolue le véhicule n'est plus de type urbain. Les images restantes, au nombre de 2524, constituent la base d'images géoréférencées. Les images requête, c'est à dire les images acquises par la caméra sur le robot, sont obtenues grâce au service internet *Google Streetview*. L'API *Google Streetview* [**?**] permet de télécharger les images au format jpeg en spécifiant la position et l'orientation de la caméra, ainsi que le champ de la caméra. Nous avons téléchargé une image tous les 15 mètres environ. Les paramètres utilisés lors de la requête HTTP sont les suivants : une définition de 640x480, un champ de vision de 100° et une inclinaison de la caméra de 5° par rapport à l'horizontale. 846 images requête ont été téléchargées. Les positions GPS spécifiées lors du téléchargement des images requête via l'API sont les positions GPS associées aux images du corpus "Pittsburgh". Certaines de ces images sont présentées dans l'annexe 6.2. Elles sont représentatives des difficultés qu'un robot peut rencontrer lors de la navigation. La vérité terrain construite, dans une première étape, à l'aide des informations GPS disponibles, a été, dans une deuxième étape, corrigée manuellement. Cette correction est nécessaire car les positions GPS fournies avec le corpus "Pittsburgh" sont imprécises et, d'autre part, la base d'images *Google Streetview* ne dispose, en moyenne, que d'une image tous les 12 mètres. Le service *Google Streetview* quand on requête une image à une position don-

née retourne l'image la plus proche. Cette base, ayant été réalisée à partir d'une base publique, présente l'avantage de pouvoir être ré-utilisée par l'ensemble de la communauté scientifique pour réaliser des comparaisons entre les différentes méthodes proposées.

Pour ne pas utiliser uniquement des images *Google*, nous avons également construit une base à l'aide d'une caméra Gopro installée sur un vélo. Cette base ayant été réalisée dans la ville de Limours, nous la nommons "Limours". Les images ont été acquises à la fréquence de 25 images par seconde puis échantillonnées aléatoirement. Les distorsions ont été corrigées et les images ont été redimensionnées pour que leur définition soit égale à 640x480 pixels. Pour chacune des images conservées, l'image *Google Streetview* représentant la même image a été sélectionnée manuellement parmi l'ensemble des images de la trajectoire téléchargées automatiquement. Pour cela une application web a été réalisée. L'interface permet de spécifier les points de passage du robot (du vélo) (Figure 2.9) et de télécharger automatiquement toutes les images *Google Streetview* le long de la trajectoire définie par ces points de passage. Cette base a principalement été utilisée lors de tests préliminaires. Certaines de ces images sont présentées dans l'annexe 6.2.

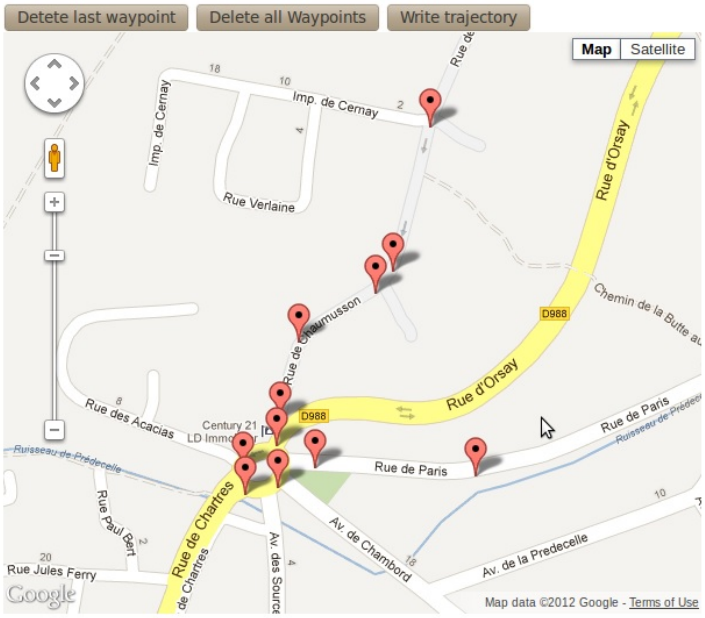


FIGURE 2.9 – Interface de l'application web permettant de spécifier une trajectoire via la définition de points de passage.

Les deux bases d'images sont constituées d'images acquises à des instants différents : d'importants changements pour la même scène peuvent donc être présents (cf. annexes 6.2 et 6.2). Les focales des deux caméras sont différentes, les conditions d'illumination sont différentes, les champs de vue et les

points de vue sont également différents.

Les performances obtenues sont évaluées à l'aide de deux métriques :

— l'erreur moyenne de localisation e exprimée en mètres : cette métrique permet d'évaluer la fonctionnalité de géolocalisation. Elle est définie par :

$$e = \frac{\sum_{k=1}^{k=n} |\hat{S}(k) - S^*(k)|}{n} \tag{2.1}$$

où $\hat{S}$ est la position estimée et S^* est la position vraie.

— la précision : cette métrique permet d'évaluer la fonctionnalité de recherche d'images en mesurant la capacité de la solution à ne faire aucune erreur, c'est à dire à trouver l'image pertinente représentant la même scène que l'image requête. Pour nos bases de données, pour chaque image requête, une seul image est pertinente, et le système ne retournera qu'une seule image, la précision est donc égale au rappel et au taux d'erreur.

2.6 Conclusion

Suite à notre état de l'art, nous avons retenu deux principes qui nous ont guidé lors de l'élaboration de notre solution de localisation visuelle. Les deux contributions majeures, qui constituent cette solution, sont présentées dans les chapitres 4 et 5. Elles permettent de retrouver, dans la base d'images géo-référencées, les images représentant la même scène que la séquence d'images requête. Préalablement, nous présentons dans le chapitre 3 une étude, réalisée pour notre contexte particulier, d'une solution d'affinage de la position absolue par estimation de la transformation rigide entre les deux images appariées.

CHAPITRE 3

Localisation précise par estimation de pose

Sommaire

Comme mentionné dans l'introduction de ce document (chapitre 1) la navigation autonome nécessite une localisation précise. Dans les chapitres 4 et 5, nous proposons une solution de recherche d'images permettant de retrouver parmi une base d'images géoréférencées l'image représentant la même scène que la dernière image acquise par le robot. Ceci permet d'obtenir une première estimation de la position du robot. La position obtenue peut être affinée. En effet, d'une part, les orientations des images de la base sont connues par rapport à un repère fixe, et, d'autre part, il est possible de calculer la direction et l'orientation relatives entre deux caméras grâce à la mise en correspondance de caractéristiques. Il s'agit d'un problème d'estimation de pose relative. Les tests préliminaires représentatifs de notre scénario ont montré que le nombre de points appariés entre deux images représentant la même scène sont peu nombreux, car d'importants changements visuels sont présents entre les deux images considérées. Compte-tenu de cette spécificité, nous proposons dans ce chapitre d'étudier une solution d'estimation de pose pour déterminer les contraintes sur les caractéristiques à mettre en correspondance.

Après une brève présentation des principales méthodes d'estimation de pose dans la section 3.1, nous présentons les tests préliminaires réalisés dans

la section 3.2. La solution d'estimation de pose 2D-2D de l'état de l'art que nous avons retenue est, ensuite, détaillée dans la section 3.3. Enfin, les simulations réalisées pour identifier et quantifier les contraintes sur les caractéristiques appariées sont décrites dans la section 3.4 et les résultats obtenus sont synthétisés dans la section 3.5.

3.1 Principales solutions d'estimation de pose

L'estimation de la pose relative (translation et orientation) entre deux images est un module fondamentale en odométrie visuelle. De nombreuses solutions ont été proposées. Cette estimation est possible par la mise en correspondance d'un ensemble de caractéristiques dans les deux images. Ces caractéristiques sont généralement des points mais il peut également s'agir de lignes. Les principales étapes d'un calcul de pose sont : (i) la détection de caractéristiques dans les deux images, (ii) la mise en correspondance de ces caractéristiques, et (iii) le calcul de pose. Dans la suite de ce chapitre, nous considérons que les caractéristiques visuelles utilisées sont des points.

3.1.1 Formulation du problème et notations

Nous supposons que le repère global, dont on connaît la position et l'orientation dans un repère absolu connu (repère dit géographique), est celui de la caméra 1. Le repère attaché à la caméra C_1 est noté $\mathcal{R}_1$. Le repère attaché à la caméra C_2 est, quant à lui, noté $\mathcal{R}_2$. La position et l'orientation relative de la caméra C_2 par rapport à la caméra C_1, est définie par une transformation rigide $\mathbf{S}_{2,1} \in \mathbb{R}^{4\times4}$ de la forme suivante :

$$\mathbf{S}_{2,1} = \begin{pmatrix} \mathbf{R}_{2,1} & \mathbf{t}_{2,1} \\ 0 & 1 \end{pmatrix} \tag{3.1}$$

où $\mathbf{R}_{2,1} \in SO(3)$ est la matrice de rotation et $\mathbf{t}_{2,1} \in \mathbb{R}^{3\times1}$ le vecteur de translation.

Le problème d'estimation de pose consiste à calculer $\mathbf{R}_{2,1}$ et $\mathbf{t}_{2,1}$ à partir d'un ensemble d'appariements de caractéristiques définis en géométrie image 2D et/ou 3D.

3.1.2 Mise en correspondance de points 2D-2D vs. 3D-2D vs. 3D-3D

Le calcul de pose peut être réalisé de trois manières différentes :

— par mise en correspondance de points 2D-2D : dans ce cas, les deux jeux de points sont spécifiés dans le repère image (2D par définition) ;

— par mise en correspondance de points 3D-2D : dans ce cas, les points 3D vus par la caméra 1 sont spécifiés dans le repère 3D attaché à la caméra 1 et leurs équivalents sont spécifiés dans le repère image 2D de la caméra 2 ;
— ou par mise en correspondance de points 3D-3D : dans ce cas, les points 3D sont spécifiés dans les repères des caméras 1 et 2.

Une revue de l'ensemble de ces approches est réalisée dans [**?**]. Selon Nister *et al.* [**?**], utiliser les correspondances 2D-2D ou 3D-2D donnent de meilleurs résultats que les correspondances 3D-3D. Ceci est dû au fait que les estimations des coordonnées 3D sont réalisées par triangulation. Les incertitudes sur la profondeur sont telles qu'elles ont un effet négatif sur l'estimation de la transformation entre les deux images. Dans le cas 3D-3D, les erreurs de position sont optimisées, alors que dans le cas 2D-2D ceux sont les erreurs de reprojection qui sont optimisées. Une solution basée mise en correspondance 3D-2D est plus rapide qu'une solution 2D-2D, car dans le premier cas seuls trois points doivent être appariés alors que dans le second cas au minimum 5 points doivent être appariés. Ceci peut avoir son importance lors de l'étape de rejet des faux appariements : le temps de calcul est en effet lié au nombre de points minimum requis.

Puisque nous sommes dans le cas monoculaire, nous avons choisi d'utiliser les correspondances 2D-2D.

3.1.3 Méthodes linéaires, itératives et robustes

Une synthèse des différentes méthodes d'estimation de pose a été réalisée par Zhang [**?**] et par Armangué et Salvi [**?**]. Armangué et Salvi ont étudié 19 méthodes pour calculer cette matrice. Ces méthodes peuvent être classées en méthodes linéaires, itératives ou robustes. Les études ont montré que les méthodes linéaires, et notamment celle introduite par Longuet-Higgins [**?**], ont comme avantage d'avoir un temps de calcul réduit mais qu'elles sont sensibles au bruit de localisation des points et aux faux appariements. Les méthodes itératives, comme par exemple celles utilisant la technique d'optimisation de Levenberg-Marquardt [**?**], ne sont, quant à elles, pas robustes aux faux appariements et le temps de calcul est plus élevé que les méthodes linéaires. Par contre, les méthodes robustes sont capables de donner un résultat précis avec des images bruitées, et peuvent aussi prendre en compte les faux appariements.

Compte-tenu de l'étude réalisée par Armangué et Salvi [**?**] et de notre scénario, nous avons retenu une méthode robuste.

Les cinq méthodes, robustes à la fois à la présence de faux appariements et à une mauvaise localisation des points appariés, sont : les *M-Estimators* [**?**],

les *Least-Median-Squares - LMedS* [**?**], *Random Sampling* notée dans [**?**] RANSAC [**?**]), MLESAC [**?**] and MAPSAC [**?**]. Les M-Estimateurs sont limités en ce qui concerne leur capacité à rejeter les faux appariements. Les solutions LMedS et RANSAC, quant à elle, sont similaires. Elles reposent toutes les deux sur la sélection aléatoire d'un jeu de points appariés utilisé pour l'estimation de la matrice fondamentale à l'aide d'une méthode linéaire. La différence entre ces deux méthodes est le critère utilisé pour déterminer la matrice fondamentale. Pour la solution LMeds, le critère est la minimisation de la distance médiane entre les points et les lignes épipolaires. Pour la solution RANSAC, le critère est la maximisation du nombre de points respectant la transformation estimée, ou autrement dit la minimisation du nombre de points rejetés. Le principal inconvénient de ces deux méthodes est leur manque de répétabilité. Ceci est dû au à la sélection aléatoire des points. Même si les LMeds donnent de meilleurs résultats en terme de précision, il semble que cette méthode soit moins robuste que RANSAC [**?**].

Nous avons retenu pour notre étude la méthode RANSAC [**?**]. Il s'agit de la méthode la plus utilisée à ce jour.

3.2 Tests préliminaires

L'estimation de la pose relative entre deux images peut être réalisée à l'aide de l'appariement de points. Nous avons réalisée des premiers tests à l'aide d'une chaîne de traitement standard de l'état de l'art :

1. Détection dans les deux images des points d'intérêt : utilisation du détecteur SIFT ;
2. Calcul des descripteurs SIFT pour chacun des points d'intérêt ;
3. Pour chaque descripteur SIFT recherche du descripteur SIFT le plus proche dans l'image représentant la même scène.
4. Suppression des points mis en correspondance pour lesquels le ratio de la distance entre le descripteur requête et le premier descripteur le plus proche et, la distance entre le descripteur requête et le second descripteur le plus proche est supérieur à 0.8 [**?**] [**?**], c'est à dire pour lesquels $\frac{\mathcal{D}(\mathbf{x}_q, NN(\mathbf{x}_q,1))}{\mathcal{D}(\mathbf{x}_q, NN(\mathbf{x}_q,2))} \geq 0.8$ où $NN(\mathbf{x}_q, i)$ est le i^{ieme} descripteur de la base le plus proche du descripteur requête $\mathbf{x}_q$;
5. Filtrage géométrique robuste pour ne conserver que les points qui respectent un modèle géométrique donné. Nous avons choisi d'estimer la matrice fondamentale grâce à l'algorithme 8 points [**?**] car comme nous le verrons dans la section 3.3 elle permet d'estimer la pose relative.

La figure 3.1 représente la distribution du nombre de points mis en correspondance pour les 846 images de la base "Pittsburgh" décrite dans le

chapitre 2.

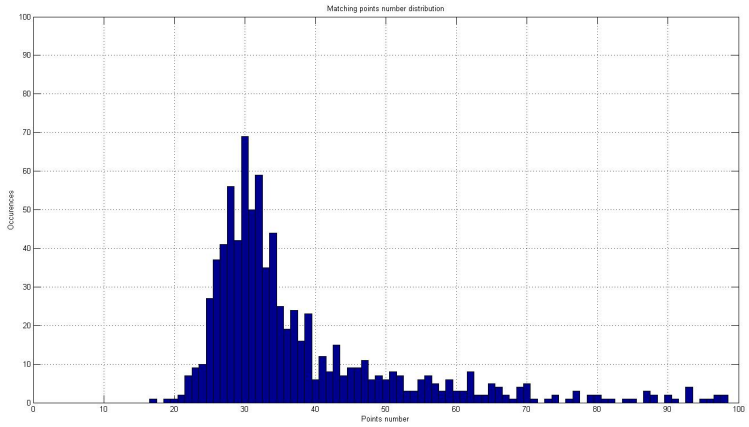


FIGURE 3.1 – Distribution du nombre de points appariés pour les images de la base "Pittsburgh".

Même en connaissant l'image de la base qui représente la même scène que celle de l'image requête, il est souvent difficile de mettre en correspondance de nombreux points : 71% des images de la base "Pittsburgh" ont entre 20 et 40 points appariés. Un nombre important de points appariés ne garantit pas des appariements corrects, comme l'illustre la figure 3.2. Même lorsque la majorité des appariements sont corrects, de faux appariements demeurent, comme le démontre la figure 3.3.

Ces premiers tests révèlent la difficulté, pour notre contexte, de la mise correspondance de points entre deux images représentant la même scène. Nous constatons que le nombre de points appariés est généralement inférieur à 40 et que parmi ces poins appariés il reste de faux appariements. C'est pourquoi nous proposons dans ce chapitre d'étudier par simulation l'influence du nombre de points appariés, l'influence du taux de faux appariements, ainsi que l'influence de la précision de localisation des points appariés sur l'estimation de la pose.

3.3 Description de la solution retenue

Les solutions d'estimation de pose sont basées sur l'estimation d'une matrice 3x3. Si les paramètres intrinsèques des caméras sont connus (focales, distorsions, et coordonnées du centre optique), alors il faut considérer les coordonnées image normalisées [?]. La matrice 3x3 est appelée dans ce cas la matrice essentielle [?]. Elle est notée **E**. Si les paramètres intrinsèques ne sont pas connus alors il faut considérer les coordonnées pixel des images. La matrice 3x3 est appelée dans ce cas la matrice fondamentale [?] [?], et elle

FIGURE 3.2 – Appariement de points : Exemples d'images difficiles pour lesquelles les points appariés sont pour la plupart faux.

est notée **F**. Connaissant les paramètres intrinsèques de deux caméras, il est possible de calculer la matrice essentielle à partir de la matrice fondamentale.

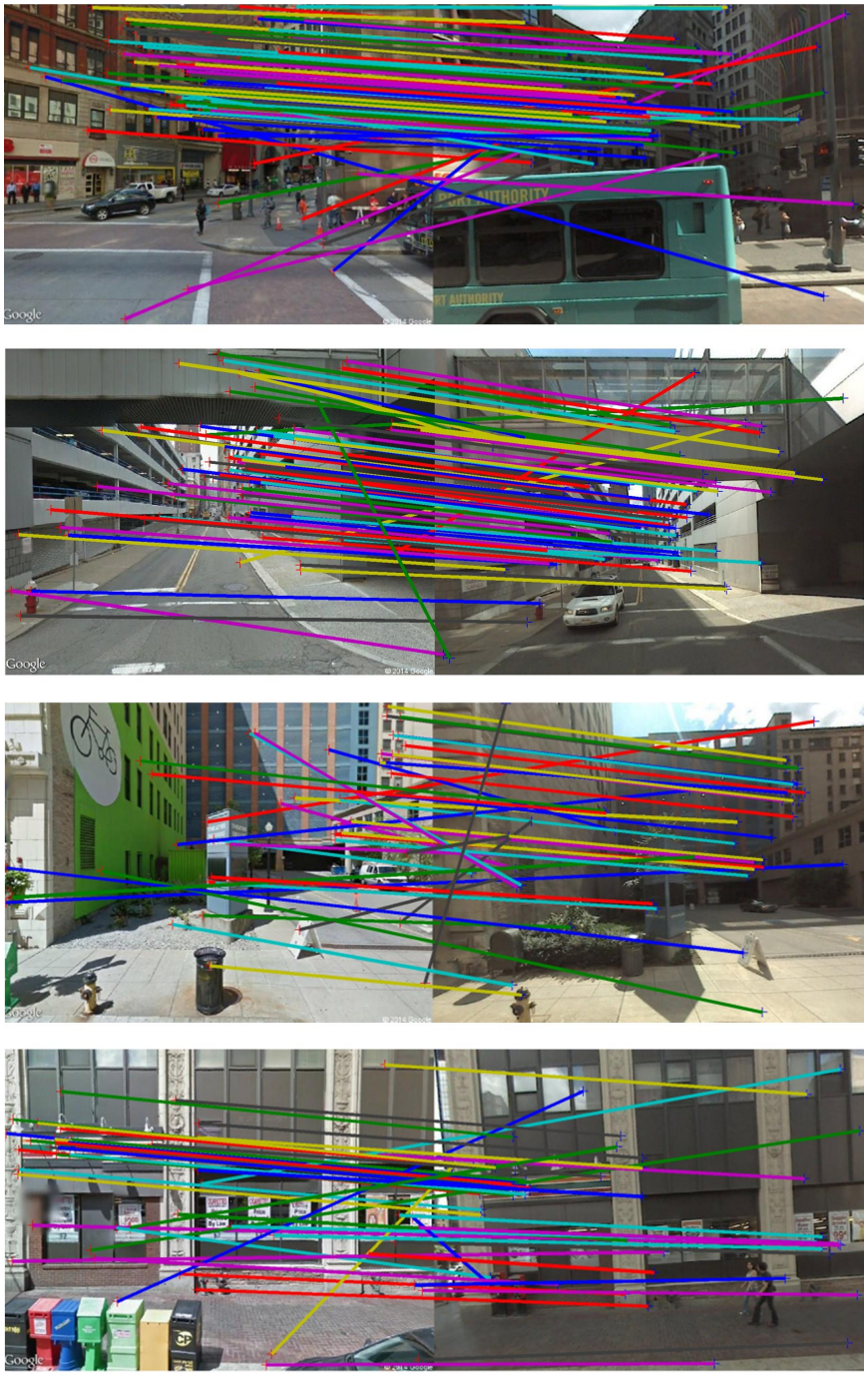


FIGURE 3.3 – Appariement de points : Exemples d'images pour lesquelles il demeure de faux points appariés même si de nombreux points appariés sont corrects.

3.3.1 Géométrie épipolaire

La géométrie épipolaire s'applique à tout système composé au moins de deux caméras, et ce, quelque soit le modèle utilisé pour les caméras (modèle du sténopé, modèle de projection orthographique ou modèle de projection perspective). Elle a été introduite par Longuet-Higgins [?]. Elle décrit la relation géométrique entre deux images prises sous deux angles de vue différents. Soient p_1 et p_2, les deux projections du point p de la scène sur les capteurs des caméras 1 et 2. Soit c_1 et c_2 les centres optiques correspondant aux cameras 1 et 2 respectivement. La contrainte géométrique est une contrainte de coplanarité entre les points p_1, p_2, c_1 et c_2. Cette contrainte est aussi appelée contrainte épipolaire. Elle réduit l'ensemble des points correspondant à un point d'une caméra à une droite dans l'autre caméra, par exemple la droite (d_2) pour le point p_1 (Fig. 3.4).

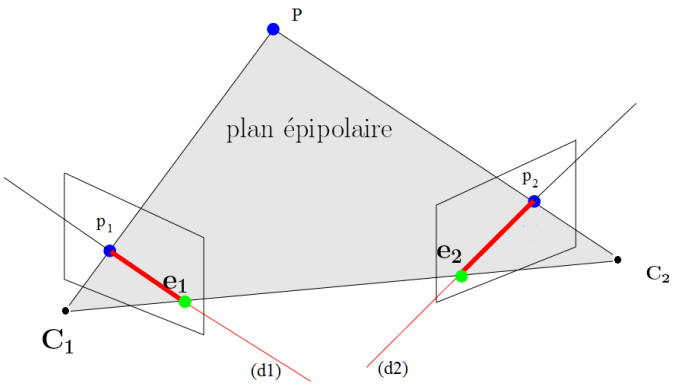


FIGURE 3.4 – Illustration de la contrainte épipolaire - Le point p se projette en p_1 dans la caméra 1 et en p_2 dans la caméra 2. c_1 et c_2 sont les centres optiques des caméras 1 et 2 respectivement. Les points p_1, p_2, c_1 et c_2 sont coplanaires. e_1 et e_2 sont les épipoles. (d_1) et (d_2) sont les droites épipolaires.

La contrainte épipolaire s'exprime mathématiquement, dans un repère donné, par :

$$\langle \overrightarrow{c_1p_1}, \overrightarrow{c_1c_2} \wedge \overrightarrow{c_2p_2} \rangle = 0 \tag{3.2}$$

On peut remarquer que le point p_1 de l'image 1 représente la projection de tous les points de l'espace appartenant à la droite (p_1p). Les points de cette droite, lorsqu'ils sont projetés sur le capteur de la caméra 2, sont contraints de se trouver sur la droite (d_2) appelée droite épipolaire. L'ensemble des droites épipolaires de l'image 2 se coupent en un même point qui est la projection de c_1 dans le plan image de la caméra 2. Ce point particulier noté e_1 s'appelle l'épipole. De la même façon, la projection de c_2 dans le plan image de la caméra 1 est appelée épipole e_1. On appelle plan épipolaire le plan défini par un point p de l'espace et les deux centres optiques c_1 et c_2.

Ce plan coupe les deux plans images en deux droites : les droites épipolaires. Tous les plans épipolaires passent par la droite (c_1c_2) et donc par les épipoles e_1 et e_2. Cette relation entre les deux images 1 et 2 constitue une contrainte : étant donné un point p_1 de l'image 1, son correspondant p_2 dans l'image 2 se trouve obligatoirement sur la droite épipolaire correspondante.

3.3.2 Estimation de la matrice essentielle

La géométrie épipolaire permet de calculer la transformation rigide $\mathbf{S}_{2,1}$, c'est à dire la rotation $\mathbf{R}_{2,1}$ et la direction de la translation $\mathbf{t}_{2,1}$ entre les deux images représentant la même scène, via le calcul de la matrice essentielle. Cette matrice dérive de la contrainte épipolaire. Nous avons vu dans la section 3.3.1 que la contrainte épipolaire s'exprime mathématiquement, dans un repère donné, par la relation suivante :

$$\langle \overrightarrow{c_1p_1}, \overrightarrow{c_1c_2} \wedge \overrightarrow{c_2p_2} \rangle = 0 \tag{3.3}$$

Les coordonnées du vecteur $\overrightarrow{c_1p_1}$ dans le repère $\mathcal{R}_1$ sont notées $\mathbf{p}_1$, les coordonnées $\overrightarrow{c_1c_2}$ dans le repère $\mathcal{R}_1$ sont par définition $\mathbf{t}_{2,1} = [t_x, t_y, t_z]^T$, les coordonnées du vecteur $\overrightarrow{c_2p_2}$ dans le repère $\mathcal{R}_2$ sont notées $\mathbf{p}_2$. Les coordonnées des points $\mathbf{p}_1$ et $\mathbf{p}_2$ sont des coordonnées homogènes normalisées de la forme $\mathbf{p}_i = [u_i, v_i, 1]^t$ avec u_i et v_i les coordonnées du point dans le repère image $\mathcal{R}_i$ associé à la caméra C_i. La contrainte géométrique (Eq.3.3) s'écrit donc algébriquement, de manière équivalente, dans le repère $\mathcal{R}_1$, ainsi :

$$\mathbf{p}_1^T \cdot \mathbf{t}_{2,1} \times \mathbf{R}_{2,1}\mathbf{p}_2 = 0 \tag{3.4}$$

En posant

$$\mathbf{T}_{2,1} = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_z & t_x & 0 \end{pmatrix} \tag{3.5}$$

on a

$$\mathbf{t}_{2,1} \times \mathbf{R}_{2,1}\mathbf{p}_2 = \mathbf{T}_{2,1}\mathbf{R}_{2,1}\mathbf{p}_2 \tag{3.6}$$

et on obtient donc :

$$\mathbf{p}_1^T \cdot \mathbf{E} \cdot \mathbf{p}_2 = 0 \tag{3.7}$$

où $\mathbf{E} = \mathbf{T}_{2,1}\mathbf{R}_{2,1}$ est la matrice essentielle ($\mathbf{E} \in \mathbb{R}^{3\times3}$).

La matrice essentielle $\mathbf{E}$ contient donc la translation et la rotation entre les deux caméras, mais à un facteur d'échelle près inconnu puisque l'équation (3.7) reste valide à un facteur multiplicatif près. La matrice essentielle a été introduite par Longuet-Higgins [**?**]. D'autre part, remarquons que si la translation est nulle, la rotation ne peut être calculée. De même si les points p_1 et p_2 sont coplanaires alors, la rotation ne peut être calculée. Dans ce cas, il est néanmoins possible de calculer la rotation et la translation grâce à la mise en correspondance de 4 points via le calcul de l'homographie. Dans ce cas, une description de la méthode à utiliser peut être trouvée dans [**?**].

3.3.3 Propriétés de la matrice essentielle

Les propriétés de la matrice essentielle $\mathbf{E}$ sont les suivantes :

- $\mathbf{E} \in \mathbb{R}^{3\times3}$
- $\mathbf{E}$ est une matrice singulière de rang 2 puisque $\mathbf{R}_{2,1}$ est une matrice inversible car $\mathbf{R}_{2,1} \in SO(3)$ et, puisque $\mathbf{T}_{2,1}$ est une matrice singulière de rang 2 car il s'agit d'une matrice antisymétrique non nulle de taille 3x3. Donc le déterminant de $\mathbf{E}$ est nul : $det(\mathbf{E}) = 0$.
- Une matrice singulière possède un noyau, c'est à dire qu'il existe des vecteurs non-nuls pour lesquels le produit matrice-vecteur est nul. Pour la matrice essentielle, il s'agit des deux épipoles e_1 et e_2.
- $\mathbf{E}$ est définie à un facteur multiplicatif près, puisque quelque soit ce facteur l'équation (3.7) sera vérifiée.
- la matrice essentielle n'a que 5 degrés de liberté puisque $\mathbf{R}_{2,1}$ et $\mathbf{T}_{2,1}$ ont chacun 3 degrés de liberté et on ne peut déterminer la matrice essentielle qu'à un facteur multiplicatif près. Le nombre d'éléments indépendants est donc de 5.

Il est possible de montrer qu'une matrice 3x3 est une matrice essentielle si et seulement si deux de ses valeurs singulières sont égales et si la troisième valeur est nulle. Une démonstration peut être trouvée dans [**?**].

3.3.4 Calcul de la matrice essentielle

La matrice essentielle peut être calculée à partir de jeux de points appariés. Nister *et al.* ont démontré [**?**] qu'il suffisait au minimum de 5 points appariés pour estimer $\mathbf{E}$. D'autres solutions existent avec la mise en correspondance de 8 points [**?**] ou de 7 points [**?**]. Les algorithmes 7 points et 8 points sont dégénérés quand les points sont coplanaires. Ce n'est pas le cas pour l'algorithme 5 points proposé par Nister. Néanmoins, l'algorithme 8 points est souvent préféré car la résolution est simple (méthode linéaire) et la sélection de la bonne solution parmi l'ensemble des solutions générées est beaucoup plus simple. La solution à 5 points génèrent en général 11 solutions. Il faut ensuite sélectionner la bonne, soit à l'aide d'une troisième image comme l'a proposé Nister dans [**?**] soit via une série de trois filtres successifs [**?**] (les points 3D doivent se trouver devant la caméra, les points 3D ne doivent pas être trop près du plan focal, et enfin la solution à retenir est celle qui a agrégé le plus grand nombre de points). D'autre part, les méthodes à 7 ou 8 points sont valides aussi bien pour les caméras calibrées que pour les caméras non calibrées, alors que la méthode 5 points suppose que les caméras soient calibrées.

Chacune des 8 paires de points appariés fournie la contrainte suivante :

$$\begin{aligned} [u_1u_2 \; u_2 \; v_1 \; u_2 \; u_1v_2 \; v_1v_2 \; v_2 \; u_1 \; v_1 \; 1]\mathbf{E}' = 0 \\ \text{avec } E' = [e_{11} \; e_{12} \; e_{13} \; e_{21} \; e_{22} \; e_{23} \; e_{31} \; e_{32} \; e_{33}]^T \end{aligned} \tag{3.8}$$

Les huit contraintes obtenues peuvent être ré-écrites par le système linéaire suivant $\mathbf{AE'} = 0$ avec $\mathbf{E'} = [e_{11}, e_{12}, \ldots, e_{32}, e_{33}]$. La résolution de ce système linéaire permet de calculer $\mathbf{E'}$ et donc $\mathbf{E}$. A cause du bruit sur les coordonnées des points appariés, il n'y a en général pas de solution exacte pour l'équation. D'autre part, le fait d'avoir plus de 8 points conduit à un système sur-déterminé. La solution consiste à utiliser une méthode de résolution par moindres carrés, c'est à dire à calculer $\min_{E'} \|\mathbf{AE'}\|^2$ sous la contrainte $\|E'\| = 1$ (pour ne pas obtenir la solution triviale $\mathbf{E'} = 0$). Soit $\tilde{\mathbf{E}}$ la matrice obtenue. Il faut s'assurer que la matrice obtenue soit de rang 2. Pour obtenir une matrice essentielle respectant cette contrainte, il faut la projeter sur l'espace des matrices essentielles valides. Pour cela, $\tilde{\mathbf{E}}$ est décomposée à l'aide d'une décomposition SVD (*Single Value Decomposition*) : $\tilde{\mathbf{E}} = \mathbf{UDV}^T$ avec $\mathbf{D} = diag(r, s, t)$, $r \geq s \geq t$ et $r = s$. La matrice essentielle finale est alors $\mathbf{E} = \mathbf{U}\ diag(r, s, 0)\ \mathbf{V}^T$. Elle est définie à un facteur d'échelle près.

De plus, Hartley a montré [**?**] qu'une normalisation préalable des données permettait d'améliorer l'estimation de la matrice essentielle $\mathbf{E}$ car cette dernière est ainsi mieux conditionnée. Il propose l'algorithme dit des "huit points normalisés". La normalisation à appliquer sur les données consiste à déplacer l'origine des coordonnées vers le centre de gravité des points image appariés, et à normaliser les coordonnées de manière à ce que leur ordre de grandeur soit égal à 1. Dans la pratique, il impose que la norme moyenne des vecteurs associés aux points soit égale à $\sqrt{2}$. Ces deux opérations reviennent à multiplier les points des images 1 et 2 par une matrice 3x3. Avec cette normalisation, l'estimation de la matrice $\mathbf{E}$ est significativement améliorée [**?**].

3.3.5 Extraction de R et T de la matrice essentielle

Une fois $\mathbf{E}$ calculée, la rotation et la translation entre les deux images peuvent être extraites par une décomposition en valeurs singulières de $\mathbf{E}$. Une démonstration détaillée de la détermination de la rotation et de la translation à partir de la matrice essentielle peut être trouvée dans [**?**] . Les quatre solutions sont :

$$\mathbf{T}_{2,1} = \pm\mathbf{U}\,[0, 0, 1]^T \tag{3.9}$$

$$\mathbf{R}_{2,1} = \mathbf{U}(\pm\mathbf{W}^T)\mathbf{V}^T \tag{3.10}$$

où

$$\mathbf{W}^T = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

et où, $\mathbf{V}$ et $\mathbf{U}$ sont issues de la décomposition en valeurs singulières de la matrice $\mathbf{E}$.

$$\mathbf{E} = \mathbf{U}\ diag(1, 1, 0)\ \mathbf{V}^T \tag{3.11}$$

Il existe deux solutions pour la direction de la translation (Eq. 3.9), et deux solutions pour la rotation (Eq. 3.10). Il existe donc quatre solutions mais il est possible de déterminer la solution correcte par triangulation d'un point [?]. La matrice essentielle, étant définie à un facteur d'échelle près, la translation est, elle aussi, connue à un facteur d'échelle près ou autrement dit, seule la direction de la translation est connue.

Après avoir sélectionné la bonne solution par triangulation et en choisissant le point qui se trouve devant les deux caméras, une optimisation non-linéaire des paramètres translation et rotation peut être effectuée en utilisant les valeurs calculées comme valeurs initiales. La fonction à minimiser est l'erreur de reprojection définie dans [?].

3.4 Simulations

Afin de permettre une localisation précise, l'estimation de $\mathbf{R}_{2,1}$ et $\mathbf{t}_{2,1}$ (direction uniquement), via l'estimation de la matrice essentielle, est possible. Il est nécessaire, pour cela, de déterminer une liste de points appariés entre l'image géoréférencée et l'image acquise par le robot. Les méthodes traditionnelles consistent à extraire des points d'intérêt dans les deux images puis à mettre en correspondance les descripteurs de chacun de ces points. La difficulté majeure est la mise en correspondance de ces points. Quelque soit la méthode retenue, des erreurs sont inévitables : il peut s'agir d'erreurs dans l'appariement des points ou d'erreurs sur la localisation 2D des points appariés.

Remarquons que la qualité de l'estimation de la matrice essentielle peut être calculée grâce au calcul des distances des points aux épipoles. Ce critère est parfois directement utilisé pour estimer la matrice essentielle.

3.4.1 Objectifs

Les tests préliminaires ont montré que le nombre de points appariés était faible : entre 20 et 40 d'après nos tests. Parmi ces points, de faux appariements demeurent après filtrage géométrique. Ceci est dû au fait que les deux images sont acquises par des caméras différentes dans des conditions très différentes. Notre scénario est spécifique par rapport aux applications d'odométrie visuelle, où de nombreux points sont suivis d'image en image. C'est pourquoi une étude concernant le nombre de points à apparier pour estimer $\mathbf{R}_{2,1}$ et $\mathbf{t}_{2,1}$ de façon fiable est nécessaire. D'autre part, il est également requis de quantifier le taux d'erreurs d'appariement acceptable, ainsi que la précision requise sur la localisation du jeu de points 2D détectés et mis en correspondance. Tels sont les objectifs de cette section.

3.4.2 Principe

Pour répondre à l'ensemble des questions, précédemment mentionnées, nous avons décidé de mettre en place une chaîne de simulation. Pour cela, un ensemble de points est généré de façon aléatoire dans le plan image de la caméra 1. Cet ensemble de points 2D est 'projeté' en 3D : la distance de chaque point à la caméra est fixé aléatoirement dans une plage fixée. L'ensemble des points 3D obtenus est reprojeté dans la caméra 2. La rotation et la translation qui permettent de passer de la caméra 1 à la caméra 2 sont connues. Seuls sont conservés les points qui sont visibles par la caméra 1 et la caméra 2. La figure 3.5 représente une scène générée.

Pour étudier l'influence de la qualité de l'appariement entre les points de la caméra 1 et ceux de la caméra 2, l'ensemble des points de la caméra 2 peut être bruité de deux manières :

— par ajout d'un bruit blanc gaussien sur les coordonnées des points 2D, représentatif de l'imperfection de la localisation des points appariés. L'écart type, σ, du bruit permet de contrôler cette précision de localisation.

— par la création de faux appariements (*outliers*). Parmi l'ensemble des points appariés, une proportion fixée r sera volontairement erronée.

Connaissant $\mathbf{R}_{1,2}$ et $\mathbf{t}_{1,2}$, il est possible de calculer l'erreur commise lors de l'estimation $\hat{\mathbf{R}}_{1,2}$ et $\hat{\mathbf{t}}_{1,2}$ avec les n points appariés bruitées et/ou erronés.

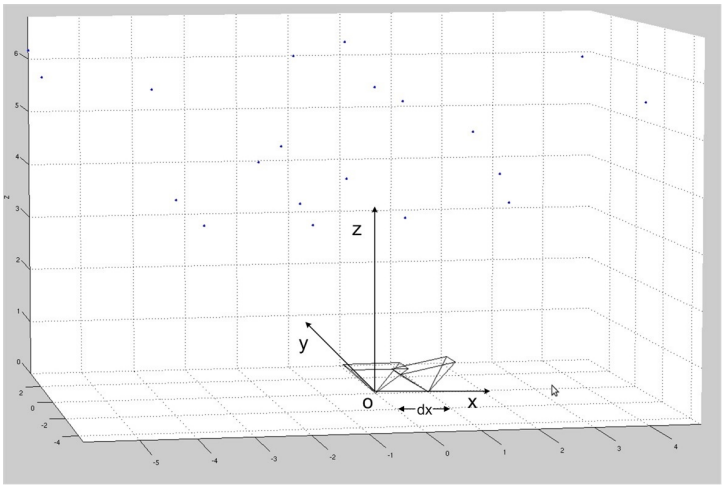


FIGURE 3.5 – Exemple d'une scène simulée : Position des caméras 1 et 2, et ensemble des points 3D observés.

3.4.3 Paramètres

Soit un repère orthonormé direct (ox,oy,oz) d'origine o. Le centre optique de la caméra 1 est l'origine o de ce repère. La caméra 1 et la caméra 2

sont séparés de d_x mètres selon l'axe (ox), ou autrement dit les coordonnées du centre optique de la caméra 2 sont [d_x,0,0] (Fig. 3.5). L'ensemble des points 3D sont compris à une distance comprise entre d_{min} et d_{max} mètres de la caméra 1 (Fig. 3.5). La caméra 2 est tournée selon l'axe (oy) d'un angle de R_y. Pour chaque simulation, 10000 réalisations sont effectuées. Pour chaque réalisation, les coordonnées des points 3D, les valeurs des bruits sur les positions des points 2D dans les plans image et les faux points appariés sont modifiés.

Pour les simulations, nous avons défini des paramètres représentatifs du scénario opérationnel défini dans le chapitre 1 :

- La définition des images des caméras 1 et 2 est de 640x480 pixels.
- $2m \leq d_{min} \leq 5m$ et $5m \leq d_{max} \leq 10m$.
- En ce qui concerne, les paramètres intrinsèques des deux caméras. Nous avons supposé qu'ils étaient identiques pour les deux caméras et que :
  - Les focales des caméras 1 et 2 sont égales à $f_x = 300$ pixels et $f_y = 300$ pixels. Ces valeurs ont été estimées à l'aide d'images *Google Streetview*.
  - Les centres optiques sont $u0 = 320$ et $v0 = 240$.
  - L'orthogonalité entre les lignes et les colonnes du capteur est supposée parfaite.
  - L'objectif est parfait : aucune distorsion n'est appliquée.
- $5° \leq R_y \leq 20°$

Lors des simulations réalisées, nous avons fait varier la variance (σ) du bruit blanc sur les points 2D des plans image de deux caméras, le taux d'*outliers* r exprimé en % et le nombre de points appariés n.

3.4.4 Résultats

Nous présentons dans cette section les principaux résultats de nos simulations. Nous reportons les résultats d'un cas réaliste et représentatif de notre contexte. Les paramètres utilisés pour ce cas sont les suivants : $d_{min} = 3m$, $d_{max} = 7m$, $[t_x, t_y, t_z] = [1, 0, 0]$ et $[R_x, R_y, R_z] = [0°, +11°, 0°]$.

Influence de l'erreur de localisation des points

La figure 3.6 donne les erreurs moyennes commises pour σ variant de 0 à 5, pour $n = 30$ points appariés et un taux de faux appariements égal à $r = 0\%$.

Nous constatons que quelque soit la valeur de $\sigma \in [0, 5]$, les erreurs moyennes commises sur les trois rotations restent inférieures à 1°. Par contre, l'erreur angulaire sur la direction du vecteur translation croit significativement avec σ. Pour $\sigma = 2$ par exemple, l'erreur moyenne sur la direction est de 4°. La direction estimée est de plus incertaine : l'écart type est supérieur à 3°. La figure 3.7 représentant la distribution des erreurs commises sur la direction pour $\sigma = 2$ confirme cette analyse : des erreurs de 10° à 15° sont

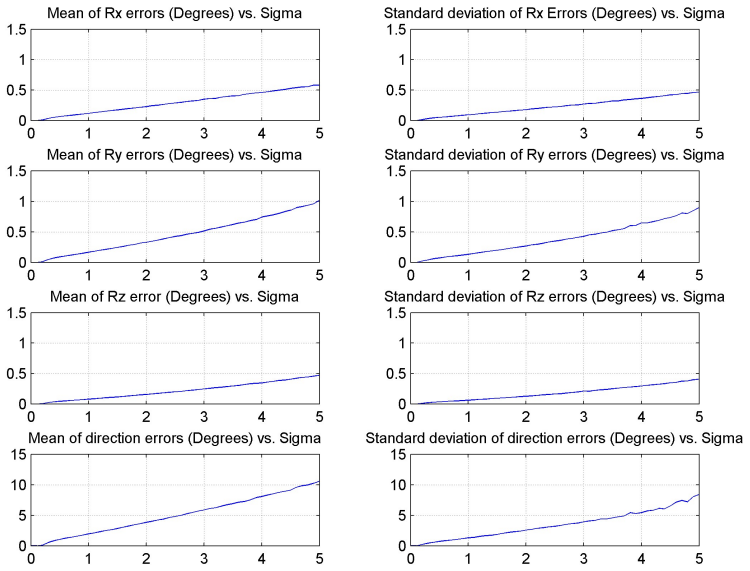


FIGURE 3.6 – Erreurs commises pour σ variant de 0 à 5, $n = 30$ et $r = 0\%$.

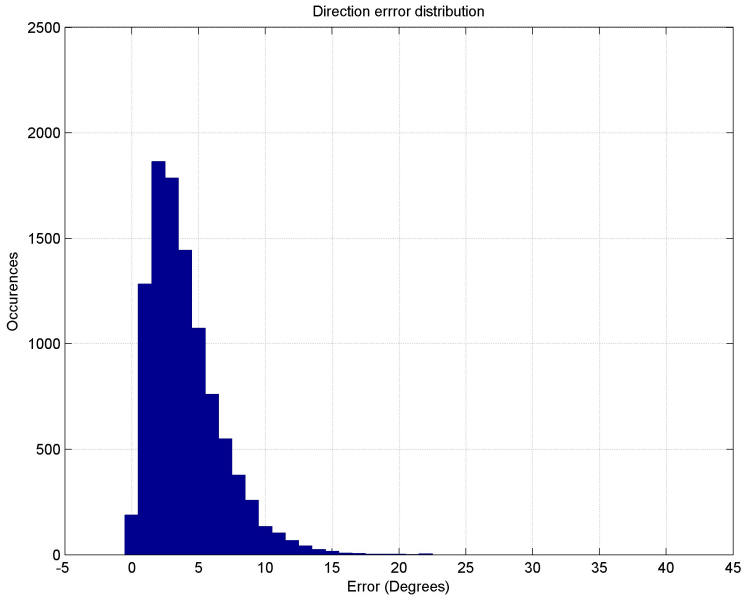


FIGURE 3.7 – Distribution des erreurs commises sur la direction pour $\sigma = 2$ (en degrés).

possibles. Si l'on souhaite une bonne estimation de la direction de la translation, c'est à dire inférieure à 2°, les points appariés doivent être localisés avec précision : $\sigma = 1$ est un seuil suffisant, ce qui correspond à environ 2 pixels de précision.

Influence du nombre de points appariés

L'influence du nombre de points appariés est étudiée pour $\sigma = 2$ et pour un taux de faux appariements de $r = 0\%$. Les nombres de points appariés varient de $n = 10$ à $n = 50$ points. La figure 3.8 montre les résultats obtenus.

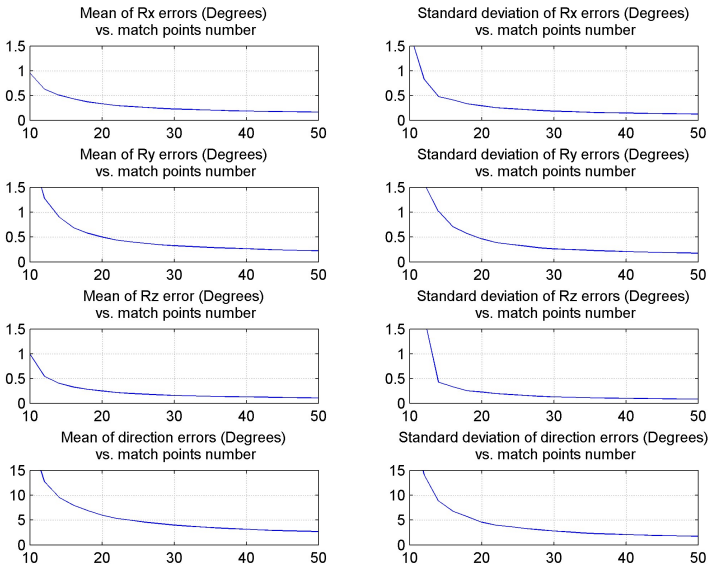


FIGURE 3.8 – Erreurs commises pour $\sigma = 2$, $r = 0\%$ et n variant de 10 à 50.

Le nombre de points appariés influent sur l'ensemble des paramètres. Au delà de $n = 15$ points appariés (sans erreur), les erreurs moyennes sur l'estimation des rotations sont inférieures à 1° avec au maximum un écart de type de 1°. L'estimation de la direction de la translation reste l'élément le plus sensible : pour $n = 30$ points appariés, l'erreur moyenne est 4° pour un écart type de 3° ; pour $n = 40$ points, elle de de 3 ° avec un écart type de 2°. Au delà de 40 points, l'amélioration sur l'estimation n'est plus significative.

Influence du taux de faux appariements

L'influence du taux de faux appariements a été étudiée pour $\sigma = 2$ et pour 30 points appariés. Les taux d'erreurs varient de $r = 0\%$ à $r = 30\%$ (Fig. 3.9). La direction du vecteur translation est de nouveau le paramètre le plus sensible :

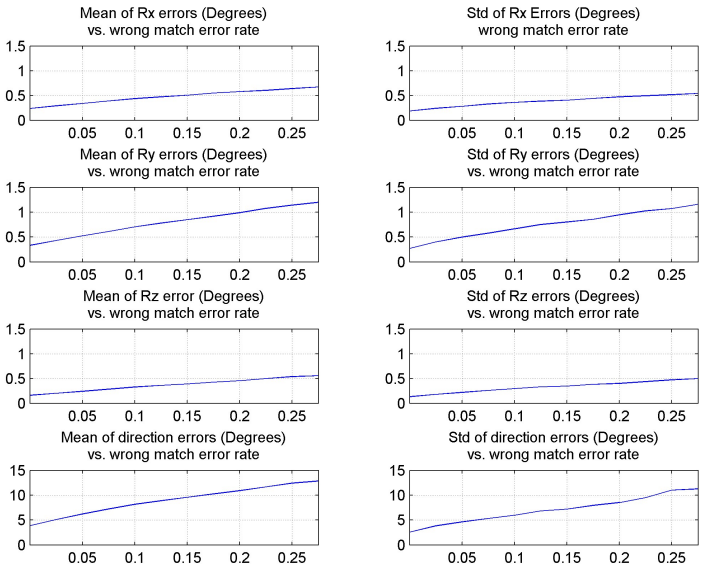


FIGURE 3.9 – Erreurs commises pour $\sigma = 2$, $n = 30$ et r variant de 10% à 30%.

Si le taux d'erreur passe de $r = 0\%$ à $r = 10\%$ l'erreur moyenne augmente de 4° à 8° avec un écart type respectivement de 3° à 6°.

3.4.5 Bilan des simulations

Ces simulations ont montré, que l'estimation de la direction de la translation était le paramètre le plus sensible aux défauts d'appariements. Ce paramètre est, malheureusement, le paramètre primordial pour affiner notre position. Les différentes simulations réalisées permettent d'affirmer que l'estimation de la direction de la translation de façon précise ($< 10°$) n'est pas possible. Pour un contexte donné (distances caméra/scène minimales et maximales données, focales des caméras fixées, etc.) et pour des défauts d'appariement donnés par exemple $\sigma = 2$, $n = 30$ et $r = 10\%$, alors l'erreur moyenne commise sur la direction est de 8° avec un écart type 6°. Ce type d'approche permet néanmoins d'affiner la localisation du robot mais il faut être conscient des incertitudes. Pour les quantifier, des simulations peuvent être réalisées. Pour être exploitables, elles doivent l'être pour un matériel donné, c'est à dire des caméras données focales, centres optiques, etc.) et, pour un scénario donné (par exemple, déplacement d'un robot dans un environnement urbain).

3.5 Conclusion

Dans notre contexte, l'appariement de points est compliqué car d'importants changements visuels sont généralement présents, ce qui rend le problème d'estimation de pose singulier.

Pour bien analyser l'impact des différents paramètres représentant les principaux défauts pouvant exister (peu de points appariés, faux appariements, et précision des points 2D appariés faible), nous avons réalisé des simulations dans un contexte entièrement contrôlé. Il ressort de notre étude que ces défauts ont un impact significatif sur l'estimation de la direction du vecteur translation. L'estimation de la pose permet certes d'affiner la position du robot, mais la direction ne pouvant être estimée de façon précise, il est nécessaire de réaliser au préalable, pour un scénario et un matériel donnés des simulations pour caractériser les incertitudes.

Il est à noter que l'estimation de la matrice essentielle est calculée à un facteur d'échelle près. La norme du vecteur translation reste donc inconnue. Pour affiner encore la position, une solution doit également être proposée pour estimer ce facteur d'échelle.

Dans ce chapitre, nous avons fait l'hypothèse que, pour chaque image acquise, nous disposions de l'image géoréférencée représentant la même scène. Dans les chapitres 4 et 5, nous traitons la problématique de reconnaissance de lieux afin de trouver pour chaque image acquise, l'image géoréférencée correspondante.

CHAPITRE 4

Localisation absolue par exploitation d'informations visuelles et odométriques

Sommaire

L'analyse réalisée dans le chapitre 2 nous a permis de montrer que la navigation autonome nécessite de se localiser dans un repère absolu. Ce problème peut être vu comme un problème de reconnaissance de lieu qui peut être résolu par une recherche d'images au sein d'une base d'images géoréférencées (*Image retrieval*). Face aux difficultés d'appariement d'images, l'état de l'art réalisé a révélé que les méthodes proposées, pouvant correspondre à notre scénario, cherchent à améliorer les solutions de recherche d'images existantes, en exploitant les contraintes spécifiques à la navigation d'un robot.

L'objectif de ce chapitre est de présenter notre solution de localisation absolue. Cette solution repose sur un algorithme de recherche d'images de l'état de l'art et elle exploite l'aspect séquentiel des images acquises et les informations issues d'un système odométrique (centrale inertielle ou algorithme d'odométrie visuelle).

Dans un premier temps, nous exposons le principe de notre solution (section 4.1) avant de décrire les solutions de recherche d'images de l'état de l'art (section 4.2). Nous détaillons ensuite notre solution (section 4.3). Enfin, nous évaluons la méthode proposée par rapport aux méthodes de recherche

d'images de l'état de l'art (section 4.4).

4.1 Principe de la méthode proposée

L'état de l'art réalisé montre que les algorithmes classiques de recherche d'image ne sont pas assez sélectifs pour notre application car, d'une part, en milieu urbain les mêmes caractéristiques visuelles tendent à être partagées par plusieurs lieux proches géographiquement (phénomène appelé *perceptual aliasing*) et, d'autre part, les scènes imagées à deux instant différents présentent des changements visuels (phénomène appelé *perceptual variability*) Ceci a pour conséquence de produire de faux appariements images. Pour résoudre ce problème, la communauté robotique propose, notamment, l'emploi d'un filtre temporel : au lieu de rechercher une image dans la base d'images, c'est une suite d'images acquises à des instants régulièrement espacés ou non qui est recherchée ([**?**][**?**][**?**][**?**][**?**]).

De telles solutions ont déjà montré leur utilité pour des tâches de reconnaissance de lieux. Nous proposons d'exploiter la connaissance d'une position approximative et de fusionner les informations visuelles avec les informations fournies par un système odométrique (centrale inertielle ou odométrie visuelle) afin de réduire le taux de faux appariements. La connaissance, même très approximative, de la position du robot permet de limiter le nombre d'images de la base d'images géoréférencées à considérer lors de la recherche de l'image correspondant à l'image requête. Cette connaissance approximative peut être estimée, par exemple, à l'aide d'une position qui aurait été fournie par un GPS quelques minutes auparavant et, des mesures odométriques réalisées depuis la perte du signal GPS. D'autre part, les mesures odométriques, incertaines elles aussi, permettent d'imposer des contraintes sur la séquence d'images géoréférencées pouvant correspondre à la séquence d'images requête. Autrement dit, elles permettent d'exploiter la contrainte spatio-temporelle inhérente au déplacement d'un robot mobile. La figure 4.1 illustre ce principe.

Nous proposons d'utiliser, comme filtre Bayésien, une chaîne de Markov à états cachés [**?**] (*Hidden Markov Model - HMM*) car il permet de prendre en compte à la fois l'incertitude sur la position estimée, les mesures de similarité visuelle, et les mesures odométriques réalisées entre chaque couple d'images acquises successives. Dans une chaîne de Markov à états cachés, la séquence d'états est l'élément caché. Les observations, par définition, ne sont pas cachées. A chaque observation correspond un état. Dans notre cas, les états cachés sont les positions successives du robot en déplacement. L'idée est d'estimer, grâce à cette chaîne, les positions successives, c'est à dire la trajectoire du robot, expliquant au mieux les dernières observations effectuées tout en prenant en compte les informations odométriques.

Pour mettre en œuvre une chaîne de Markov à états cachés de nom-

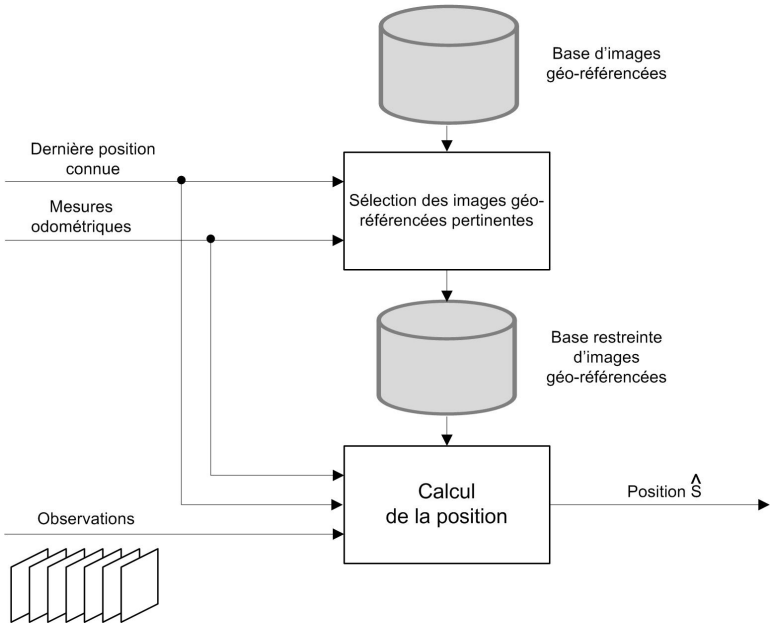


FIGURE 4.1 – Principe général de la solution de localisation absolue par exploitation d'informations visuelles et odométriques (1/2).

breux paramètres doivent être définis. L'instanciation de la chaîne de Markov que nous proposons est détaillée dans la section 4.3. En particulier, compte-tenu des informations à notre disposition, il va s'agir de déterminer les états à considérer, et de prendre en compte l'incertitude sur la localisation supposée (calculée à partir de la dernière position connue et des mesures odométriques), tout en exploitant les informations visuelles et odométriques (Fig.4.1).

4.2 Solutions pour la recherche d'images

La problématique de recherche d'images est un sujet de recherche à part entière. De nombreuses méthodes ont été proposées. La majorité d'entre elles reposent sur l'utilisation de descripteurs locaux invariants, dans une certaine limite, aux translations, aux rotations et aux changements d'échelle. Parmi les principales méthodes de la littérature, deux grandes classes peuvent être distinguées : les méthodes basées votes (section 4.2.2) et les méthodes basées dictionnaires (section 4.2.3).

4.2.1 Formulation du problème et notations

Le problème consiste à trouver pour une image requête donnée, notée I_q, l'image similaire, notée I^*, parmi la base d'images $\mathcal{B} = \{I_k\}$. Mathématique-

ment, ce problème s'exprime ainsi :

$$I^* = \arg\max_k S(I_q, I_k) \tag{4.1}$$

où $S(I_q, I_k)$ est une mesure de similarité entre les images I_q et I_k.

La définition d'un score de similarité $S(I_q, I_k)$ entre deux images est la principale difficulté. Le calcul de la similarité nécessite pour chacune des images considérées, l'extraction préalable d'un ensemble de descripteurs locaux qui représente chacune des images. Soit $\mathcal{D}_q = \{\mathbf{d}_i^q\}$ l'ensemble des descripteurs locaux de l'image I_q et $\mathcal{D} = \{\mathbf{d}_i^k\}$ l'ensemble des descripteurs locaux de toutes les images I_k de $\mathcal{B}$.

Pour les approches basées votes, ces descripteurs locaux sont utilisés tels quels. Pour les approches basées dictionnaires, il sont utilisés pour calculer une signature compacte et représentative de chaque image. Soient $\mathbf{x}_q = \phi(I_q)$ et $\mathbf{x}_k = \phi(I_k)$ les signatures des images I_q et I_k. Dans ce cas, $S(I_q, I_k)$ est définie par $S(\phi(I_q), \phi(I_k) = S(\mathbf{x}_q, \mathbf{x}_k)$. Le choix d'une fonction ϕ appropriée est une autre difficulté.

4.2.2 Approches basées votes

Le principe général consiste à mettre en correspondance chacun des descripteurs de l'image requête appartenant à $\mathcal{D}_q = \{\mathbf{d}_i^q\}$ avec le ou les descripteurs les plus proches des images I_k. L'image de la base ayant le plus de descripteurs mis en correspondance est l'image la plus ressemblante. Deux approches sont à distinguer.

4.2.2.1 Comparaison par paires d'images

La première méthode consiste à mettre en correspondance chacun des descripteurs de l'image I_q avec l'ensemble des descripteurs de l'image I_k. Cette mise en correspondance est réalisée pour chacune des images de la base. Une fois les descripteurs mis en correspondance, un filtrage géométrique est appliqué, pour ne conserver que les points conformes à une transformation géométrique donnée (par exemple une homographie). Ce processus est illustré par les figures 4.2 et 4.3. La figure 4.2 représente les points mis en correspondance pour deux images représentant la même scène, avant filtrage géométrique Fig. 4.2 (a) et après filtrage géométrique Fig. 4.2 (b). La figure 4.3 représente les points mis en correspondance pour deux images ne représentant pas la même scène, avant filtrage géométrique Fig. 4.3 (a) et après filtrage géométrique Fig. 4.3 (b).

Le principal problème d'une telle méthode est le temps de calcul, puisque qu'il est linéaire avec le nombre de descripteurs moyen par image multiplié par le nombre d'images dans la base. Il devient rapidement prohibitif lorsque le nombre d'images de la base augmente et lorsque la dimension du vecteur augmente.

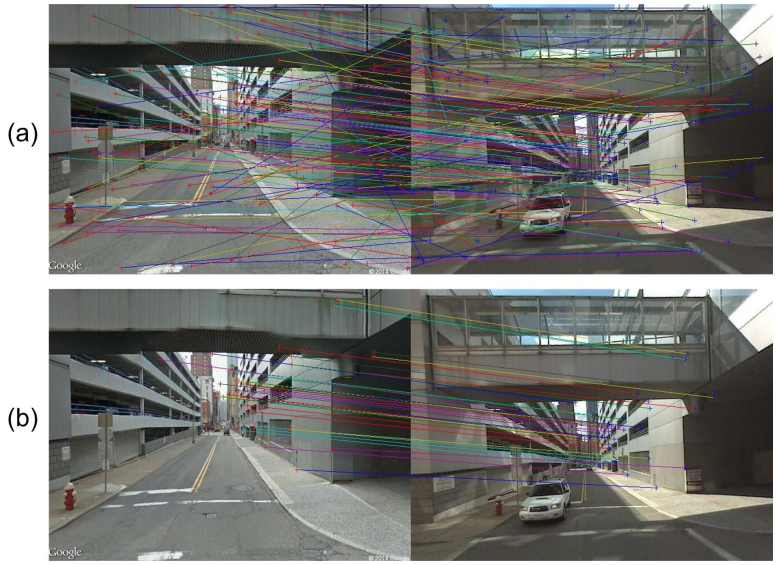


FIGURE 4.2 – Mise en correspondance de descripteurs locaux entre deux images représentant la même scène avant (a) et après (b) filtrage géométrique.

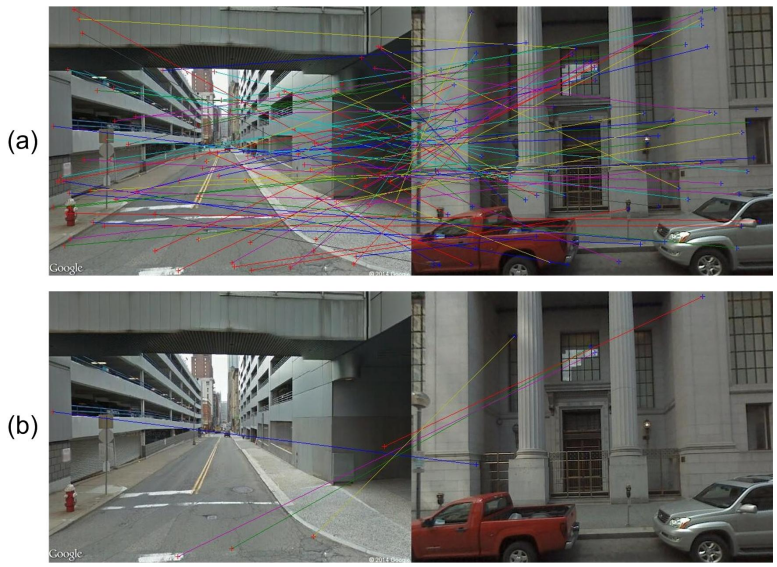


FIGURE 4.3 – Mise en correspondance de descripteurs locaux entre deux images ne représentant pas la même scène avant (a) et après (b) filtrage géométrique.

4.2.2.2 Vote kNN

Le principe est de rechercher pour chacun des descripteurs de l'image requête appartenant à $\mathcal{D}_q = \{\mathbf{d}_i^q\}$ les k descripteurs les plus proches, autrement dit, les k plus proches voisins (ou k *Nearest Neighboors - kNN*) parmi $\mathcal{D}$. Chacun de ces k plus proches voisins vote pour l'image de $\mathcal{D}$ à laquelle il appartient. Les images ayant le plus grand nombre de votes sont supposées être des images similaires à l'image requête. Cette approche a été proposée par David Lowe [**?**]. Ce type de méthode repose donc directement sur les descripteurs de l'image (*Bag of Features - BoF*). Un raffinement consiste à ne voter que lorsque la distance entre le premier plus proche descripteur et deuxième plus proche descripteur est suffisamment grande. Ceci permet de ne voter que lorsque le descripteur est suffisamment discriminant.

Le temps de calcul pour une recherche de type *kNN* exacte est également linéaire avec le nombre de descripteurs moyen par image multiplié par M le nombre d'images dans la base mais, par rapport à la méthode précédente, le nombre de recherches de descripteurs est divisé par M. Ce temps de calcul dépend également de la dimension du descripteur. Il existe de nombreux algorithmes de recherche des plus proches voisins. Une description et une comparaison de ces différentes méthodes peut être trouvée dans [**?**]. Ces méthodes peuvent être exactes comme par exemple la méthode *k-d tree* décrite dans [**?**] ou approximées comme celle décrite dans [**?**]. Les méthodes approximées permettent d'obtenir un bon compromis entre précision et temps de calcul.

4.2.3 Approches basées dictionnaire

Les méthodes les plus populaires sont les méthodes de type sacs de mots visuels (ou *Bag Of Visual Words - BOVW* [**?**]), développées au cours de la dernière décennie. Ces approches consistent à calculer pour l'ensemble des images de la base et pour chaque image requête une signature vectorielle via l'utilisation d'un dictionnaire (ou *codebook*). Lors de la phase de recherche, la signature visuelle de l'image requête sera comparée avec les signatures visuelles des images de la base (Fig.4.4). Les scores de similarité $S(I_q, I_k)$ sont alors égaux à $S(\mathbf{x}_q, \mathbf{x}_k)$, où $\mathbf{x}_q \in \mathbb{R}^p$ et $\mathbf{x}_k \in \mathbb{R}^p$ sont respectivement les signatures visuelles des images I_q et I_k.

Le calcul des signatures visuelles des images consiste à assigner à chaque descripteur $\mathbf{d}_i^k$ d'une image donnée le mot le plus proche $\mathbf{w}_j$ appartenant au dictionnaire. Autrement dit, chaque descripteur est quantifié en l'un des mots de ce dictionnaire. Un histogramme contenant le nombre d'occurrences de chaque mot est ensuite construit. Cet histogramme, ou sacs de mots visuels, constitue la signature visuelle de l'image à partir de laquelle il a été construit. Il est le plus souvent normalisé via la norme euclidienne. Le plus souvent le nombre d'occurrences est pondéré pour prendre en compte la fré-

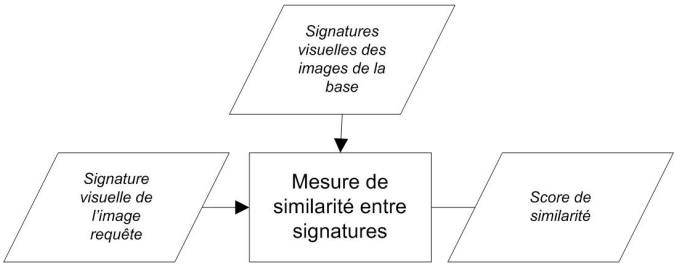


FIGURE 4.4 – Principe de calcul de la mesure de similarité pour les approches de recherche d'images basées dictionnaire.

quence d'apparition des mots et la rareté des mots. Cette amélioration est appelée *term frequency, inverse document frequency - tf-idf* [?] [?]. Le score de similarité utilisé entre deux signatures visuelles est le plus souvent la distance euclidienne car elle équivaut à calculer un produit scalaire entre les deux signatures : $S(\mathbf{x}_q, \mathbf{x}_k) = \langle \mathbf{x}_q, \mathbf{x}_k \rangle$. Ceci a pour principal avantage de pouvoir être calculé rapidement.

Les différentes étapes des approches basées dictionnaire sont donc les suivantes : (i) extraction de descripteurs visuels bas niveau, par exemple des descripteurs SURF, SIFT, ou autre, (ii) codage (ou *coding*) pour réduire l'espace de description, et (iii) *pooling* qui agrège les informations codées afin de construire une signature vectorielle (Fig.4.5). De nombreuses amé-

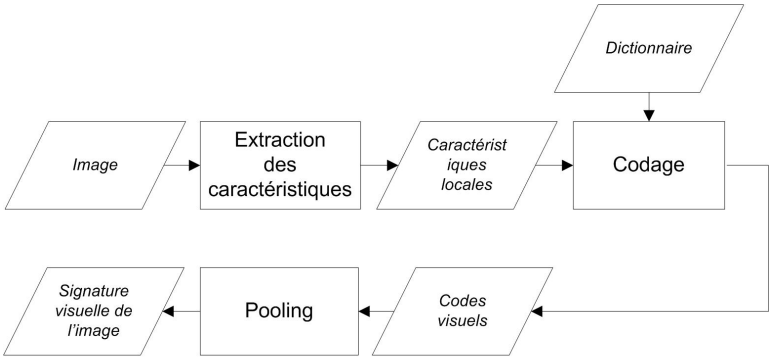


FIGURE 4.5 – Chaîne de traitement pour le calcul d'un sac de mots visuels.

liorations ont été proposées pour chacune des différentes étapes : calcul du dictionnaire, codage, *pooling* et mesure de similarité. Les principales sont brièvement décrites ci-après.

4.2.3.1 Construction du dictionnaire

La méthode la plus populaire consiste à partitionner l'espace des caractéristiques visuelles d'une base d'images d'apprentissage en p cellules re-

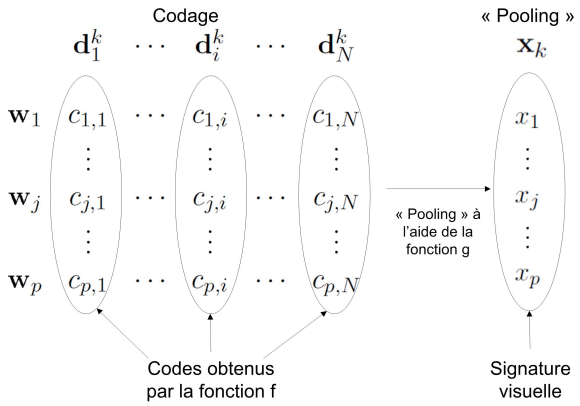


FIGURE 4.6 – Construction d'un sac de mots visuels : codage et *pooling*.

présentées par p centroïdes $\mathcal{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_p\}$. Ce regroupement est généralement réalisé à l'aide de l'algorithme k-means. Le nombre de mots du dictionnaire p est un paramètre. D'autres approches ont été proposées pour apprendre de manière supervisée un dictionnaire discriminant [**?**] [**?**].

4.2.3.2 Codage

La méthode de base consiste à déterminer pour chaque descripteur $\mathbf{d}_i^k$ de l'image I_k le mot du dictionnaire le plus proche. Soit $\mathbf{w}_j$ ce mot. Dans ce cas, $c_{l,i}$ est égal à 1 pour $l = j$ et est égal à 0 pour $l \in \{1, ..., p\}$ et $l \neq j$. Pour une image donnée I_k, chacun des descripteurs $\mathbf{d}_i^k$ est donc codé à l'aide d'un dictionnaire $\mathcal{W}$ et à l'aide d'une fonction f : $f(\mathbf{d}_i^k, \mathcal{W}) = \mathbf{u}_i = [c_{1,i}, c_{2,i}, ..., c_{p,i}] \in \{0, 1\}^p$ (Fig.4.5). Cette méthode est appelée *hard coding* par opposition au *soft coding* pour laquelle la fonction f est définie dans ce cas par : $f(\mathbf{d}_i^k, \mathcal{W}) = \mathbf{u}_i = [c_{1,i}, c_{2,i}, ..., c_{p,i}] \in \mathbb{R}^p$ ([**?**] ou [**?**]). Le *soft coding* consiste à atténuer les erreurs de codage réalisées par une quantification. Pour cela, chaque descripteur vote pour les descripteurs les plus proches en pondérant ce vote par la distance entre descripteurs. $c_{j,i}$ est donc une valeur réelle dans l'intervalle $[0, 1]$ qui dépend de la proximité du descripteur $\mathbf{d}_i^k$ avec le mot du dictionnaire $\mathbf{w}_j$.

4.2.3.3 *Pooling*

L'étape de *pooling* consiste à agréger les codes pour calculer une signature visuelle compacte $\mathbf{x}^k = x_1, x_2, ..., x_p$. Ceci est effectué à l'aide d'une fonction g : $g(f(\mathbf{d}_i^k, \mathcal{W}))$ (Fig.4.5). Parmi les méthodes existantes, citons les plus populaires : le *pooling* de type "somme" (*sum pooling*) et le *pooling* de type "maximum" (*max pooling*). Le pooling de type "somme" consiste à sommer chacune des composantes des mots code : $x_i = \sum_{j=1}^{j=N} c_{i,j}$. Le pooling de type "maximum" consiste à conserver la valeur maximale de chacune des

composantes des mots code : $x_i = max_{j \in \{1,\ldots,N\}} c_{i,j}$.

Ces méthodes de base ne prennent en compte que les statistiques du premier ordre de la distribution des descripteurs. D'autres approches ont donc été proposées pour décrire comment les descripteurs se répartissent par rapport à chaque mot du dictionnaire. Ceci est réalisé grâce à un modèle paramétrique génératif. Dans [**?**] par exemple, Jégou *et al.* propose une représentation appelée *Vector of Locally Aggregated Descriptors - VLAD*. Comme pour les approches basées dictionnaires, un dictionnaire $\mathcal{W}$ de p mots est tout d'abord appris $\mathcal{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_p\}$. A chaque descripteur $\mathbf{d}_i^k$ est associé le mot du dictionnaire le plus proche $\mathbf{w}_j$. Pour chaque mot du dictionnaire $\mathbf{w}_j$, les différences $\mathbf{d}_i^k - \mathbf{w}_j$ sont alors accumulées. Ceci permet de caractériser la distribution des descripteurs autour du mot assigné. $\mathbf{v}_j = \sum_{i \in V(\mathbf{w}_j)} \mathbf{d}_i^k - \mathbf{w}_j$ où $V(\mathbf{w}_j)$ sont les descripteurs appartenant au voisinage de $\mathbf{w}_j$. Le VLAD est la concaténation des vecteurs $\mathbf{v}_j$ de dimension d. La dimension est donc de $p \cdot d$. Dans [**?**], Perronnin *et al.* proposent une autre solution où les mots visuels sont modélisés par un mélange de gaussiennes (ou *Gaussian Mixture Model - GMM*). Les matrices de covariance sont supposées être diagonales pour chacune des g gaussiennes. La dimension de la signature est, dans ce cas, égale à $(2g + 1) \cdot d$. Cette signature est appelée vecteur de Fischer. Picard and Gosselin [**?**] ont généralisé cette approche à l'aide d'une description statistique à des ordres plus élevés appelée Vector of Locally Aggregated Tensors (VLAT). Dans la pratique, les calculs deviennent complexes et l'estimation des statistiques d'ordre élevé difficile. Plus récemment, Avila *et al.* [**?**] ont proposé une solution appelée *BossaNova*, qui consiste à calculer, pour chaque mot du dictionnaire, la distribution des distances avec les descripteurs assignés à ce mot. Cette distribution est exprimée sous la forme d'un histogramme qui est concaténé avec le code *soft*.

L'inconvénient majeur d'un sac de mots, tel qu'il a été présenté jusqu'à présent, est qu'il ne contient aucune information spatiale, ou, autrement dit, qu'aucune information sur la répartition des mots visuels au sein de l'image n'est conservée. Pour résoudre ce problème, S. Lazebnik *et al.* ont proposé en 2006, lors de l'étape de *pooling*, de calculer un sac de mots pour différentes régions de l'image et à différentes échelles et de concaténer l'ensemble des sacs de mots obtenus dans un unique vecteur. Leur méthode est communément appelée *Spatial Pyramidal Matching* (SPM). La figure 4.7 illustre le mécanisme. La méthode consiste à construire une pyramide spatiale de l'image. Si l'on se réfère à la figure 4.7, par exemple, pour le premier niveau de la pyramide, toute l'image est considérée pour le calcul d'un sac de mots ; pour le deuxième niveau un sac de mots est calculé pour chacune des 4 régions spécifiée ; pour le troisième niveau un sac de mots est calculé pour chacune des 16 régions spécifiée. Les 21 sacs de mots obtenus sont alors concaténés un

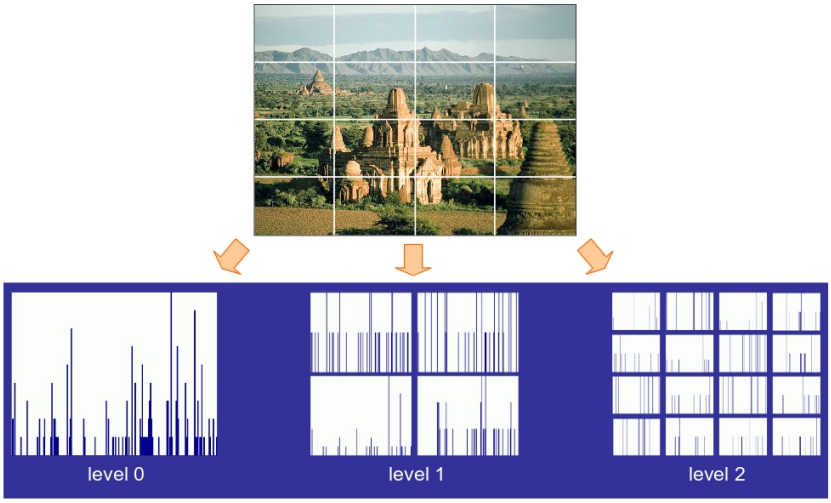


FIGURE 4.7 – Principe du calcul d'un sac de mots avec information spatiale. Figure réalisée par S. Lazebnik.

sein d'un vecteur unique $\mathbf{x}$. Différentes configurations de pyramide spatiale peuvent être spécifiées. Celle de la figure 4.7 est de type (1x1,2x2,4x4).

4.2.3.4 Mesure de similarité

La mesure de similarité permettant de comparer deux signatures visuelles repose en général sur la distance euclidienne : $S(\mathbf{x}_q, \mathbf{x}_k) = 1 - \mathcal{D}_{L2}(\mathbf{x}_q, \mathbf{x}_k)$ avec $\mathcal{D}^2_{L2}(\mathbf{x}_q, \mathbf{x}_k) = \langle \mathbf{x}_q, \mathbf{x}_k \rangle$.

4.2.4 Discussion

Les méthodes basées dictionnaire visent à construire une signature image compacte et représentative afin de pouvoir retrouver l'image la plus similaire à l'image requête. Il est à noter que la majorité de ces méthodes ont été mises au point afin de retrouver une image parmi de très grandes bases d'images (de plusieurs milliers à plusieurs millions d'images). C'est pourquoi, les méthodes proposées cherchent, d'une part, à créer une signature compacte et, d'autre part, à employer une fonction de similarité peu complexe à calculer, généralement le produit scalaire. Pour de telles bases, les contraintes de complexité et d'empreintes mémoire sont importantes. Pour notre problématique, le fait de connaître la position approximative du robot implique que l'on recherche l'image représentant la scène observée par le robot parmi une base d'images géoréférencées dont le nombre est limité (quelques dizaines d'images). La taille de la base d'images à considérer dépend directement de l'incertitude de localisation du robot à un instant donné. Les contraintes de temps de traitement sont donc moins importantes pour notre application,

mais les contraintes mémoire demeurent. L'ensemble des descripteurs requis pour une mission de plus de 10 km doivent pouvoir être stockés sur une plateforme de type mini-drone. D'autre part, les représentations basées dictionnaire entraînent une perte d'information importante lors de l'étape de fusion de l'ensemble des vecteurs en un vecteur unique. C'est pourquoi, nous avons décidé, dans un premier temps, d'utiliser une méthode basée votes kNN pour notre mesure de similarité visuelle, afin d'obtenir la meilleure précision. Par la suite, nous chercherons à l'optimiser pour avoir une solution moins coûteuse en mémoire et éventuellement plus rapide.

4.3 Méthode proposée

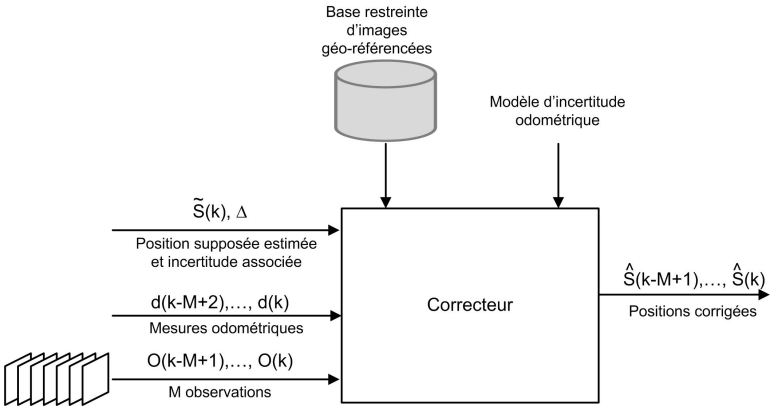


FIGURE 4.8 – Principe général de la solution de localisation absolue par exploitation d'informations visuelles et odométriques (2/2).

Lorsque le signal GPS est absent, toutes les s secondes, après avoir parcouru une certaine distance (distance approximative mesurée par le système odométrique), ou bien encore à la demande selon un critère quelconque, le système proposé permet de corriger les positions estimées du robot à partir des informations odométriques en les combinant avec des informations visuelles. Nous disposons pour cela des M dernières observations $\mathbf{O}_k = \{O_{k-M+1}, \dots O_k\}$ et des mesures de déplacement entre chaque paire d'observations successives $\{d(k-M+2), ..., d(k)\}$. Ces dernières mesures permettent de calculer une estimation des positions correspondant à chaque observation $\tilde{\mathbf{S}}_m = \{\tilde{S}_{m-M+1}, \dots \tilde{S}_m\}$. Ces positions estimées sont connues avec une incertitude dépendant de la qualité du système odométrique utilisé. Ces estimations doivent donc être corrigées. La position corrigée est notée $\hat{S}(k)$. L'incertitude sur la localisation estimée est supposée connue. On suppose, de plus, disposer d'un modèle d'incertitude concernant les mesures odométriques. La figure 4.8 illustre ce principe. Pour cela, nous proposons de mettre en œuvre une chaîne de Markov à états cachés [**?**]. Il est à noter

que ces mêmes informations odométriques avec la dernière position connue du robot permettent préalablement de restreindre le nombre d'images géo-référencées à considérer (Fig.4.1).

4.3.1 Chaîne de Markov à états cachés

Une chaîne de Markov est un processus stochastique à temps discret et à espaces d'états discrets dont l'horizon de dépendance temporelle est de taille fixe. Généralement, cet horizon est limité au dernier instant, et on parle alors de chaîne de Markov à l'ordre 1. Elle vérifie la propriété suivante :

$$P(q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, ..., q_1 = S_l) = P(q_t = S_j | q_{t-1} = S_i) \quad (4.2)$$

Les chaînes de Markov à états cachés permettent de résoudre 3 problèmes :

- Estimation de la vraisemblance que la séquence observée $\mathbf{O}_k = \{O_{k-M+1}, \dots O_k\}$ ait été produite par un modèle λ donné. La résolution de ce problème s'effectue via l'algorithme *forward/backward* qui permet de calculer $P(\mathbf{O}|\lambda)$;
- Estimation de la meilleure séquence des états cachés $\mathbf{S}_m = \{S_{m-M+1}, \dots, S_m\}$ étant donné la séquence observée $\mathbf{O}_k = \{O_{k-M+1}, \dots O_k\}$ et un modèle λ donné. Ce problème, c'est à dire le calcul de $\text{Arg}\max_{\mathbf{S}} P(\mathbf{S}|\mathbf{O}, \lambda)$, peut se résoudre par une méthode de programmation dynamique : l'algorithme de Viterbi [**?**] ;
- Estimation des paramètres du modèle qui maximisent la probabilité des observations $P(\lambda|\mathbf{O})$. La résolution de ce problème s'effectue via l'algorithme de Baum-Welch [**?**] qui permet de résoudre $\text{Arg}\max_{\lambda} P(\lambda|\mathbf{O})$. L'algorithme Baum-Welch est un cas particulier de l'algorithme *Expectation-Maximization - EM*.

L'utilisation d'une chaîne de Markov à états cachés nécessite la définition d'un modèle λ approprié. L'état du système à l'instant t est considérée comme une variable aléatoire q_t dont les valeurs discrètes possibles sont parmi un ensemble fini de N états. Le modèle $\lambda = \{N, M, A, B, \Pi\}$ complet d'une chaîne est caractérisé par :

- Un ensemble de N états : $\mathbf{S}_m = \{S_{m-N+1}, \dots, S_m\}$, que l'on notera également, pour simplifier, $\mathbf{S} = \{S_1, \dots S_N\}$;
- Une matrice des probabilités de transition $\mathbf{A}$ entre chaque état : $a_{ij} = P[q_{t+1} = S_j | q_t = S_i]$, $1 \leq i, j \leq N$;
- Un vecteur $\mathbf{\Pi}$ des probabilités des états initiaux : $\Pi = \{\pi_j\}_{j=1}^{j=N}$ où $\pi_j = P[q_1 = S_j]$
- Un ensemble de M observations successives $\mathbf{O}_k = \{O_{k-M+1}, \dots O_k\}$, que l'on notera également, pour simplifier, $\mathbf{O} = \{O_1, \dots O_M\}$;
- Une matrice $\mathbf{B}$ composée des N densités de probabilités des observations sachant l'état : $b_j(k) = P[i_t = O_k | q_t = S_j]$, $1 \leq j \leq N$ et

$1 \leq k \leq M$.

Dans notre cas, nous nous intéressons à l'estimation de la meilleure séquence d'états cachés $\mathbf{S}$, c'est dire les positions du robot, étant donné la séquence observée $\mathbf{O}$, c'est à dire une séquence d'images. Étant donné le modèle de la chaîne de Markov à états cachés, il est possible de maximiser $P(\mathbf{S}|\mathbf{O}, \lambda)$ via l'équation 4.3 et donc d'obtenir la trajectoire qui explique le mieux les observations. Le premier terme de cette équation réfère aux similarités visuelles entre les observations et les images de la base. Il est modélisé par une matrice $\mathbf{B}$. Le second terme réfère à la dynamique du robot. Il est modélisé par la matrice $\mathbf{A}$ (Eq. 4.3).

$$\begin{aligned} \hat{\mathbf{S}} &= \arg\max_{\mathbf{S}} P(\mathbf{O}|\mathbf{S}, \lambda) \cdot P(\mathbf{S}, \lambda) \\ &= \arg\max_{\mathbf{S}} \left(\prod_{k=1}^{k=M} P(O_k|\mathbf{S}, \lambda) \right) \cdot \left(\pi_1 \cdot \prod_{k=2}^{k=M} a_{k-1,k} \right) \ pour\ M > 1 \end{aligned} \tag{4.3}$$

Nous allons expliciter dans la section suivante (section 4.3.2) la définition d'un modèle $\lambda = (N, M, \mathbf{A}, \mathbf{B}, \mathbf{Pi})$.

4.3.2 Modélisation du problème

La définition du modèle $\lambda = (N, M, \mathbf{A}, \mathbf{B}, \mathbf{Pi})$ a pour objectif d'estimer la meilleure séquence d'états, étant donné une séquence d'observations. Dans notre cas, à chaque état correspond une image géoréférencée dans la base d'images, et les observations sont les M dernières images acquises par le robot. L'ensemble des paramètres du modèle sont explicités dans la section 4.3.3, 4.3.4, 4.3.5 et 4.3.6. Nous montrons qu'ils dépendent du nombre d'observations passées considérées M, de l'incertitude sur la position supposée U, et des incertitudes odométriques $\Delta(d_k)$ pour un déplacement d_k.

4.3.3 Détermination des états

Il est tout d'abord nécessaire de déterminer les valeurs des états possibles Ces différentes valeurs sont les différents lieux, c'est à dire les différentes images de la base, pouvant correspondre aux M observations réalisées.

Si l'on considère une unique observation ($M = 1$), les différentes images de la base que l'on peut lui associer peuvent être déterminées à partir de la position supposée du robot et de l'incertitude associée à cette position U. La position supposée permet de déterminer l'image de la base I_k qui devrait correspondre si l'estimation était correcte (Fig. 4.9). L'incertitude U, quant à elle, permet de définir les images de la base à considérer situées avant et après l'image I_k : $n = \lceil U/d' \rceil$ images sont situées avant l'image I_k, et

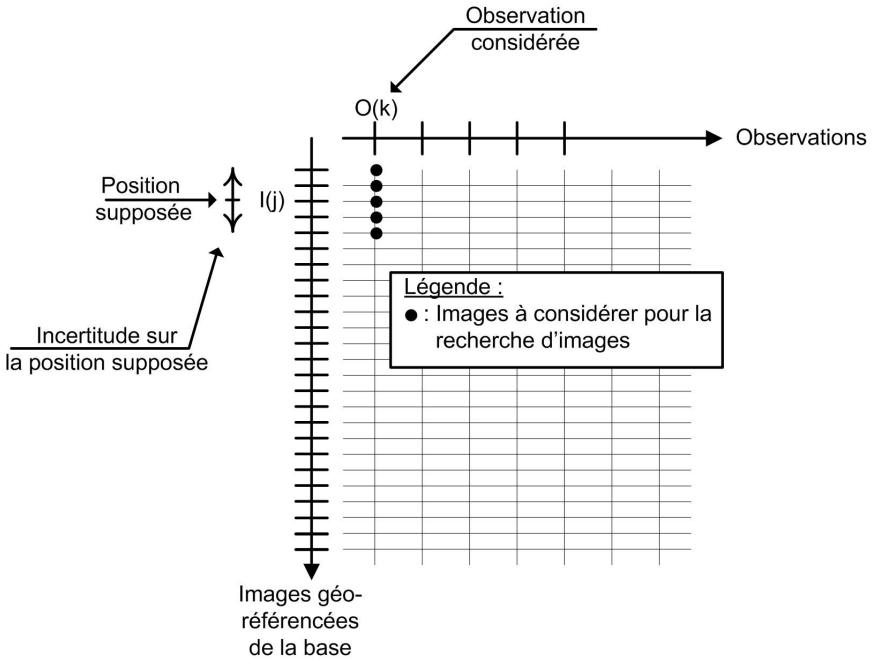


FIGURE 4.9 – Images de la base à considérer étant donné la position estimée et l'incertitude de localisation U pour une seule observation.

$n = \lceil U/d' \rceil$ images sont situées après l'image I_k. La figure 4.9 a été réalisée avec les hypothèses suivantes :

— l'incertitude U est de 10 mètres ;
— dans la base d'image géoréférencées, nous disposons d'une image tous les 5 mètres.

Dans le cas où l'on dispose de M observations ($M > 1$), les images de la base qui peuvent lui être associées dépendent de la position supposée du robot, de l'incertitude associée U et des incertitudes odométriques $\Delta(d_k)$ liées aux déplacement d_k entre deux observations successives O_k et O_{k-1} (Fig. 4.10). Ces données permettent de déterminer quelles sont les images de la base qui peuvent décrire les mêmes scènes que les images requête considérées. La figure 4.10 a été réalisée avec les hypothèses suivantes :

— l'incertitude U est de 10 mètres ;
— dans la base d'image géoréférencées, nous disposons d'une image tous les 5 mètres ;
— une observation est réalisée approximativement tous les 15 mètres ;
— l'incertitude d'odométrie pour un déplacement de 15 mètres est de 10 mètres.

Ainsi, pour une position supposée à laquelle correspond l'image I_j (c'est à dire l'état $\tilde{S}_j$), pour une incertitude sue la position estimée de U mètres

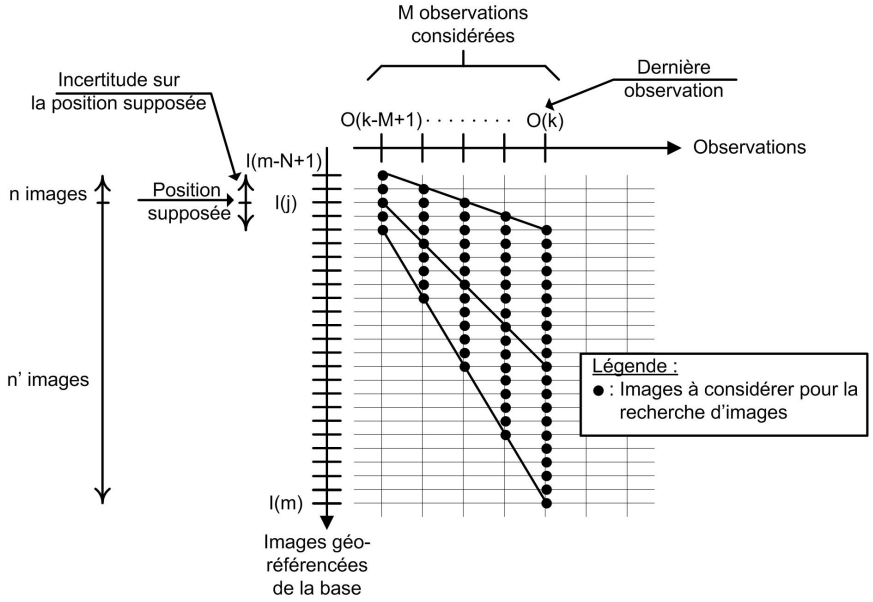


FIGURE 4.10 – Images de la base à considérer étant donné la position estimée, l'incertitude de localisation, et les incertitudes odométriques entre chaque chaque observation/déplacement.

et pour $M-1$ déplacements d'environ $d_k = d$ mètres, le nombre d'images à considérer pour les M observations est égal à $n' = \lceil ((M-1)(d+\Delta)+U)/d' \rceil$ et le nombre d'images à considérer après cette image est égal à $n = \lceil U/d' \rceil$ où d' est la distance entre deux images de la base d'images géoréférencées. Les incertitudes odométriques pour les $(M-1)$ déplacements de d mètres ont été ici supposées constantes et égales à $\Delta(d_k) = \Delta(d) = \Delta$ mètres. Au total, $N = n' + 1 + n$ images, ou états, sont à considérer pour la recherche d'images.

4.3.4 Probabilités des états initiaux

Le vecteur des probabilités des états initiaux Π, c'est à dire le vecteur donnant la probabilité que la position courante soit égale à chacun des N états considérés, peut être déterminé grâce à la position supposée $\tilde{S}_j$ et à son incertitude U exprimée en mètres. Par définition, $\Pi = \{\pi_j\}_{j=1}^{j=N}$ où $\pi_j = P[q_1 = S_j]$. Nous supposerons cette incertitude de localisation uniforme, ce qui donne :

$$\begin{aligned} \pi_j &= P[q_1 = S_j] = \frac{1}{1+2n},\ 1 \leq j \leq (2n+1) \\ \pi_j &= P[q_1 = S_j] = 0,\ (2n+2) \leq j \leq N \end{aligned} \tag{4.4}$$

4.3.5 Matrice des probabilités de transition

La matrice des probabilités de transition $\mathbf{A}$ entre l'ensemble des états considérés permet de modéliser la connaissance *a priori* des performances du système odométrique utilisé. $\mathbf{A}$ est définie par $\mathbf{A} = \{a_{ij}\}$ avec $a_{ij} = P[q_{t+1} = S_j | q_t = S_i]$, $1 \leq i, j \leq N$. Ces probabilités peuvent être décrites par un modèle qui permet d'estimer la probabilité d'avoir parcouru, pour une distance mesurée de d_1 mètres, une distance de d_2 mètres. Ce modèle est lié au capteur odométrique utilisé. Pour simplifier nos explications, nous supposons ci-après que les images requête sont acquises approximativement tous les d mètres (d constant) et qu'un déplacement de d mètres engendre une incertitude de Δ mètres ($\Delta(d_k) = \Delta(d) = \Delta$). Dans ce cas, la matrice $\mathbf{A}$ est égale à :

$$a_{ij} = \frac{1}{\lceil 2\Delta/d' \rceil} rect_{\lceil \Delta/d' \rceil}(j - i - (\lceil d/d' \rceil)) \tag{4.5}$$

où d' est la distance entre deux images de la base d'images géoréférencées et où la fonction $rect_\alpha(x)$ est la fonction porte définie par $rect_\alpha(x) = 1$ si $|x| \leq \alpha$ et $rect_\alpha(x) = 0$ sinon. La figure 4.11 est une illustration de la matrice $\mathbf{A}$.

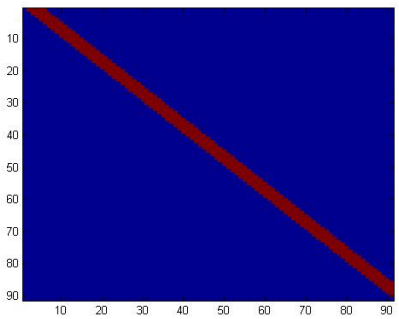


FIGURE 4.11 – Exemple de matrice des probabilités de transition entre chaque état pour $U = 100m$, $d = 15m$, $d' = 5m$ et $\Delta = 10m$ (86 états).

Il est à noter que d et Δ ne sont pas forcément constants entre deux observations. Si tel n'est pas le cas, la définition de la matrice $\mathbf{A}$ doit être adaptée en conséquence. d est fournie par le capteur odométrique et Δ par le modèle d'erreur associé au capteur.

4.3.6 Matrice des distributions de probabilité des observations

La matrice $\mathbf{B}$, matrice des distributions des probabilités des observations pour chaque état, est calculée à partir de la similarité visuelle entre les M

observations et les N images considérées de la base correspondant aux N états possibles.

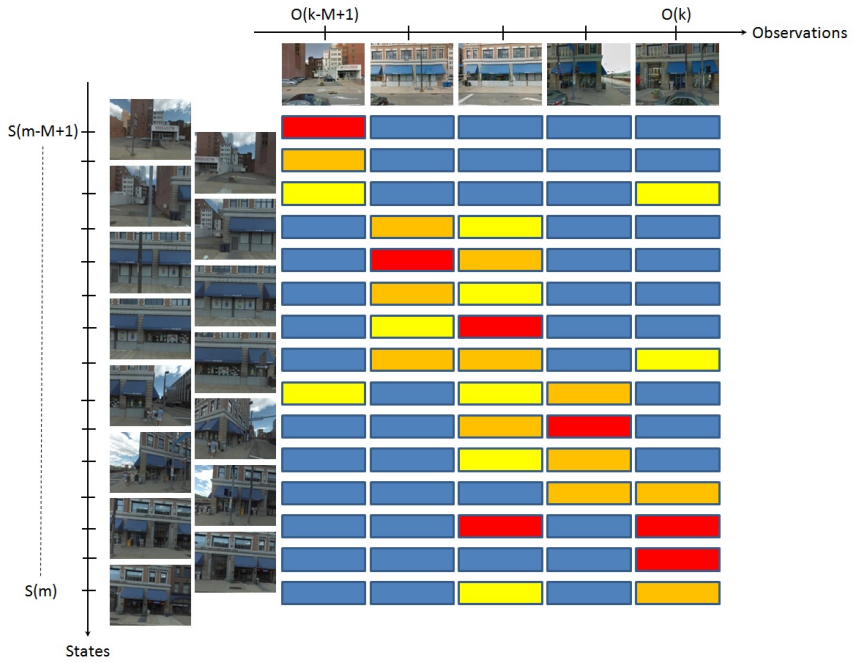


FIGURE 4.12 – Matrice des distributions de probabilité des observations pour chaque état (Matrice $\mathbf{B}$). La couleur indique le degré de similarité : plus la couleur est rouge, plus la similarité est élevée.

Nous avons choisi, dans un premier temps, une mesure de similarité visuelle basée sur une solution de recherche d'images de l'état de l'art basée votes (cf. section 4.2.2). Pendant la phase de navigation, les descripteurs SIFT [?] de tous les points d'intérêts extraits de l'image acquise sont calculés de la même façon que lors de la phase de préparation de mission. Chaque descripteur extrait vote pour les k descripteurs les plus proches parmi les images de la base considérées comme pertinentes, c'est à dire les images de la base les plus proches de la position estimée et prenant en compte les incertitudes de localisation et d'odométrie comme expliqué dans la section 4.3.3. Les points d'intérêt détectés dans une image étant souvent bruités, une étape de filtrage est appliquée pour ne conserver que les votes fiables :

— seuls les votes des points d'intérêt dont le ratio de la distance du descripteur requête et le premier plus proche descripteur et, la distance du descripteur requête et le second descripteur le plus proche est inférieur à 0.8 sont conservés [?] [?], c'est à dire pour lesquels $\frac{\mathcal{D}(\mathbf{x}_q, NN(\mathbf{x}_q,1))}{\mathcal{D}(\mathbf{x}_q, NN(\mathbf{x}_q,2))} \leq 0.8$ où $NN(\mathbf{x}_q, i)$ est le i^{ieme} descripteur de la base le plus proche du descripteur requête $\mathbf{x}_q$;

— un filtrage géométrique est appliqué pour ne conserver que les appariements de points satisfaisant un modèle de transformation géométrique. Le filtrage appliqué est de type RANSAC [?] par estimation de la matrice fondamentale avec l'algorithme 8 points [?].

Ainsi nous obtenons pour chaque observation O_k, le nombre de points mis en correspondance avec les points d'intérêt des images considérées de la base I_j. Cette valeur constitue notre mesure de similarité notée $S(O_k, I_j)$.

La matrice $\mathbf{B} = \{b_j(k)\}$, où $b_j(k) = P[i_t = O_k | q_t = S_j]$, $1 \leq n \leq N$ et $1 \leq j \leq M$ est la matrice des probabilités d'observer O_k quand la position du robot est S_j. Nous proposons de calculer cette probabilité à l'aide de l'équation suivante :

$$b_j(k) = P[i_t = O_k | q_t = S_j] = \frac{\alpha}{1 + exp(-a \cdot (S(O_k, I_j) + b))} \tag{4.6}$$

où a and b sont deux constantes, $S(O_k, I_j)$ est la mesure de similarité visuelle précédemment définie et α est une constante de normalisation pour garantir $\sum_{j=1}^{j=N} b_j(k) = 1$.

La figure 4.12 est une illustration de la matrice $\mathbf{B}$. Chaque élément de cette matrice indique la similarité entre une observation et une image de la base. La valeur de chaque élément est représentée par une couleur indiquant le degré de similarité entre les deux images : plus la couleur est rouge, plus la similarité est élevée. La figure 4.13 schématise la solution proposée, et un résumé du processus de correction est présenté par l'algorithme 1.

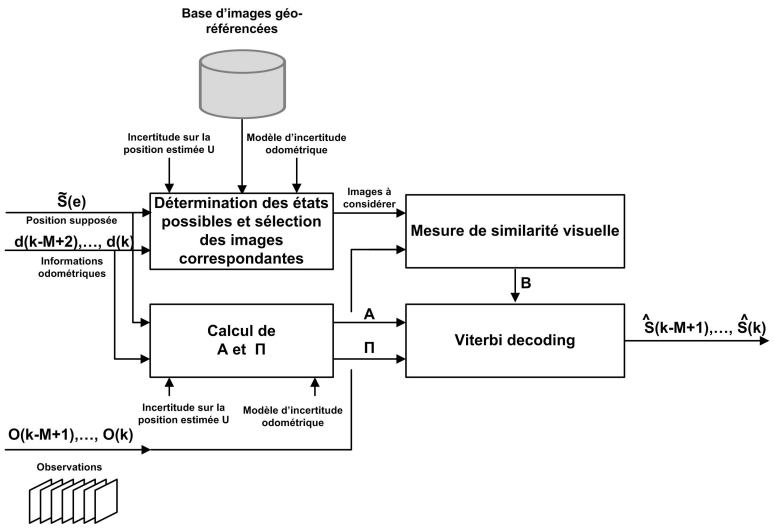


FIGURE 4.13 – Principe de la correction des positions estimées par l'utilisation d'une chaîne de Markov à états cachés.

Algorithme 1 : Algorithme de localisation absolue combinant mesures de similarité visuelles et informations odométriques.

Input : Position estimée $\tilde{S}_k$ et incertitude de localisation associée U, M dernières observations $\mathbf{O} = O_{k-M+1}, ..., O_k$, $M-1$ mesures odométriques $\{d(k-M+2), \vdots, d(k)\}$ avec les $M-1$ incertitudes correspondantes $\mathbf{\Delta} = \Delta_{k-M+2}, ..., \Delta k$, Descripteurs de la base d'images géoréférencées

Output : M positions corrigées du robot $\hat{\mathbf{S}} = \hat{S}_{k-M+1}, ..., \hat{S}_k$.

1 Calcul de $\mathbf{A}$ and Π à partir de $\tilde{\mathbf{S}}$, U et $\mathbf{\Delta}$ comme expliqué dans les sections 4.3.4 et 4.3.5;
2 Sélection des images pertinentes de la base d'images géoréférencées à partir de $\tilde{S}_k$, U et $\mathbf{\Delta}$ (Fig. 4.10);
3 Calcul des similarités entre les M dernières observations $\mathbf{O}$ et les images sélectionnées de la base comme expliqué dans la section 4.3.6;
4 Calcul de $\mathbf{B}$ à partir des similarités (Eq. 4.6);
5 Décodage de Viterbi pour résoudre l'équation 4.3 afin d'estimer la trajectoire $\hat{\mathbf{S}}$ et en particulier le dernier état, c'est à dire la position actuelle du robot $\hat{S}_k$;

4.4 Expérimentations

Les différentes expérimentations qui sont effectuées ont pour objectif de déterminer :

- l'apport des informations odométriques pour des solutions de recherche d'images de l'état de l'art. Nous évaluons cet apport en activant ou désactivant l'utilisation du filtrage HMM ;
- l'influence du nombre d'observations M utilisées pour l'estimation et de l'incertitude sur la position initiale supposée U ;
- l'influence de la mesure de similarité visuelle utilisée. Nous avons utilisé deux mesures de similarité différentes : une méthode basée votes décrite dans la section 4.2.2.2 notée (IR-Vote-kNN), et une méthode basée sac de mots visuels dont le principe est décrit dans la section 4.2.3, notée (IR-BOW-L2). Pour cette dernière méthode, la similarité entre deux sac de mots visuels est mesurée à l'aide de la distance euclidienne.

Les évaluations consistent à calculer l'erreur de localisation commise lors de la recherche d'une observation parmi une base restreinte d'images géoréférencées. Comme mentionné dans le chapitre 3, les métriques utilisées pour cela sont l'erreur moyenne de localisation et la précision (égale au taux d'erreur dans notre cas).

4.4.1 Paramètres

Dans cette section, les paramètres utilisés lors des expérimentations sont brièvement rappelés. Le robot acquiert une image tous les $d = 15$ mètres. La base d'image géoréférencées contient une image tous les $d' = 5$ mètres. La distante entre deux observations étant environ égale à d mètres, l'incertitude d'odométrie est supposée constante et vaut $\Delta(d_k) = \Delta(d) = \Delta(15) = 10m$. Par conséquent, la matrice A et le vecteur Π peuvent être définis respectivement à l'aide des équations 4.5 et 4.4. Pour le calcul de la matrice $\mathbf{B}$ à l'aide de l'équation 4.6, le coefficient a a été initialisé à $1/4$ et le coefficient b à -20. Ainsi la similarité est supérieure à 0.5 lorsque le nombre de points appariés est supérieur à 20. Ce seuil provient des tests préliminaires qui ont montré qu'au delà de 20 points appariés l'image retrouvée était, dans la majorité des cas, la bonne image. Pour 10 points appariés, la similarité est à 0.1. Pour 40 points appariés, elle est quasiment égale à 1. L'incertitude sur la position initiale supposée U et le nombre d'observations M sont deux paramètres dont l'influence est étudiée.

4.4.2 Résultats

4.4.2.1 Apport des informations odométriques séquentielles

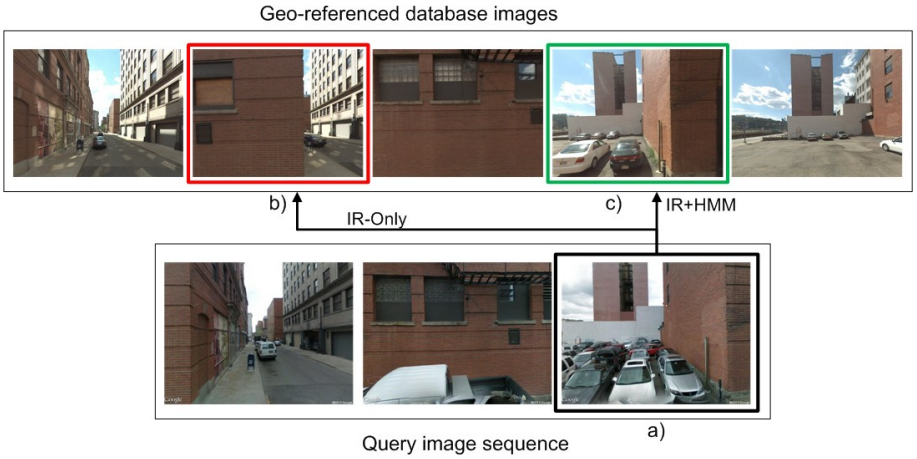


FIGURE 4.14 – Apport des contraintes spatio-temporelles - Images du haut : Images successives géoréférencées considérées pour la recherche d'images - Images du bas : Séquence d'observations, images séquentielles acquises par le robot - (a) Dernière image requête acquise par le robot, (b) Image retournée par un algorithme de recherche d'images basé votes uniquement et (c) par un algorithme de recherche d'images combiné avec une chaîne de Markov à états cachés.

Nous avons tout d'abord comparé notre méthode (IR-Vote-kNN-HMM),

à deux algorithmes de recherche d'images de l'état de l'art : l'un basé votes (IR-kNN-Vote) et l'un basé dictionnaire (IR-BOW-L2). Les paramètres utilisés pour notre méthode sont les suivants : $U = 50m$ et $M = 15$. La méthode basée votes est celle décrite dans le chapitre 4.2.2.2. La méthode basée dictionnaire est un sac de mots visuels (BOVW) avec informations spatiales (SPM) (cf. section 4.2.3.3). Les sacs de mots visuels sont construits à l'aide de descripteurs SIFTs extraits de manière dense à l'aide de [**?**] : le pas entre chaque descripteur est de 4 pixels et chaque descripteur est extrait à 4 échelles différentes 1, 1.5, 2, et 2.5. Le codage des BOVW est de type *hard assignment* et le *pooling* de type *sum pooling*. Le vecteur BOVW est normalisé ℓ_2. La configuration pour la pyramide spatiale est (1x1,2x2). La taille du dictionnaire utilisé, calculée sur l'ensemble de la base d'images géoréférencées, est de 5000 mots visuels (taille standard couramment utilisée).

Les résultats obtenus sont reportés dans le tableau 4.1. Les métriques utilisées sont l'erreur de localisation moyenne exprimée en mètres et la précision exprimée en % (cf. section 2.5). L'exploitation des mesures odométriques permet de réduire l'erreur de localisation : elle passe de $10.5m$ à $5.7m$. La précision, quant à elle, augmente de 46.8% à 50.2%. Alors que la gain sur l'erreur de localisation est d'environ 50%, le gain en précision n'est que de 7%. Ceci signifie que notre solution permet de corriger les mises en correspondance aberrantes d'un point de vue spatio-temporel. Par contre, le gain sur la fonctionnalité de recherche d'images est moins significatif. La figure 4.14 illustre ceci : considérer une séquence d'images avec les informations odométriques entre ces images permet de corriger les correspondances d'images ambigües.

D'autre part, même si les précisions pour les méthodes IR-BOW-L2 et IR-kNN-Vote sont équivalentes, les erreurs moyennes de localisation sont très différentes : l'erreur de localisation moyenne pour la méthode IR-BOW-L2 est de $35.7m$ alors qu'elle est de $10.5m$ pour la méthode IR-kNN-Vote. Rappelons que l'utilisation d'une méthode basée votes est envisageable puisque nous recherchons une image parmi quelques dizaines d'images : les ressources nécessaires (charge processeur, et mémoire occupée) restent donc acceptables. Notons enfin, que la métrique "précision" ne peut être utilisée seule pour notre application. Cette métrique prend en compte les erreurs commises pour la recherche d'images mais elle ne quantifie pas l'erreur commise sur la position. Elle doit être impérativement combinée avec l'erreur moyenne de localisation.

4.4.2.2 Influence du nombre d'observations et de l'incertitude de localisation initiale

Nous étudions maintenant la sensibilité de notre solution basée HMM par rapport au nombre d'observations M prises en compte, et à l'incertitude sur la position initiale estimée U. Ceci est intéressant d'un point de

	IR-BOW-L2	IR-Vote-kNN	IR-Vote-kNN-HMM
Erreur moyenne	35.7m	10.5m	5.7m
Ecart type	67.5m	14.7m	7.8m
Précision	45.2%	46.8%	50.2%

TABLE 4.1 – Erreur moyenne de localisation, écart type, et précision pour une solution de recherche d'image basée votes (IR-Vote-kNN), pour une solution basée dictionnaire (IR-BOW-L2) et pour notre solution (IR-Vote-kNN-HMM). Paramètres : $d = 15m$, $\Delta = 10m$, $U = 50m$, $M = 10$.

vue opérationnel : M a un impact sur les ressources mémoire et processeur, U caractérise la capteur odométrique. La figure 4.15 montre l'influence de

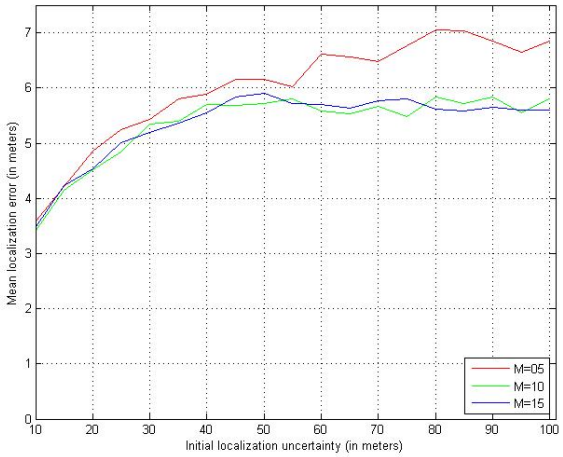


FIGURE 4.15 – Erreur de localisation moyenne en fonction de l'incertitude de localisation initiale U.

l'erreur de localisation initiale U sur l'erreur moyenne de localisation commise pour différentes valeurs de M. Si l'on souhaite une erreur moyenne de localisation après correction la plus faible possible pour une erreur de localisation initiale de $100m$, alors il faut au minimum considérer 10 observations. La figure 4.16, quant à elle, montre l'influence du nombre d'observations considérées pour différentes incertitudes de localisation U. Plus l'incertitude de localisation initiale est élevée, plus le nombre d'observations à prendre en compte doit être élevé si l'on souhaite garantir une erreur de localisation moyenne en dessous d'un certain seuil. On peut constater par exemple que 8 observations suffisent si l'on souhaite garantir une erreur de localisation moyenne inférieure à $6m$.

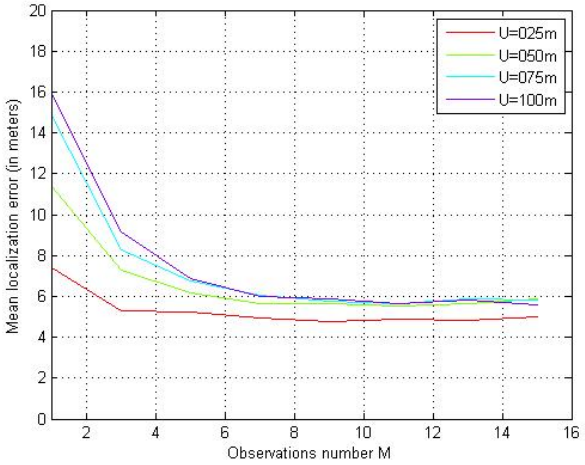


FIGURE 4.16 – Erreur de localisation moyenne en fonction du nombre d'observations M. Quand l'incertitude de localisation augmente, M doit être plus élevé pour garantir une erreur de localisation moyenne inférieure à un seuil donné.

4.4.2.3 Analyse comportementale

Les résultats présentés précédemment sont des moyennes. Cette section fournit des éléments d'analyse plus précis sur le comportement de notre solution.

La figure 4.17 représente l'erreur commise en mètres pour la solution proposée et pour chacune des 846 images requête, indexées par leur ordre d'observation. La figure 4.18 représente la distribution des erreurs commises. Nous pouvons constater à l'aide de la figure 4.17 que les erreurs importantes commises sont réparties régulièrement le long de la trajectoire suivie par le robot, et qu'elles sont relativement peu corrélées entre elles spatialement.

A l'aide de la figure 4.17, nous avons identifié les configurations pour lesquelles notre solution commet une erreur de localisation importante. C'est le cas par exemple des images requête n°114, n°304, n°660 et n°689 etc.

La figure 4.19 représente, tout d'abord, un cas pour lequel notre solution permet de réduire significativement l'erreur commise. Sans l'utilisation de la chaîne de Markov a états cachés, l'image la plus similaire à l'image requête n°17 est l'image correspondant à l'état 48 repéré par l'étoile jaune alors que l'état optimal est le 40 repéré par l'étoile rouge, soit une erreur de $40m$ (car deux images géoréférencées successives sont séparées d'une distance de $5m$). Avec notre solution, l'état retournée est le 38 soit une erreur de localisation de $10m$.

Les cas qui posent problème et qui conduisent à une erreur de localisa-

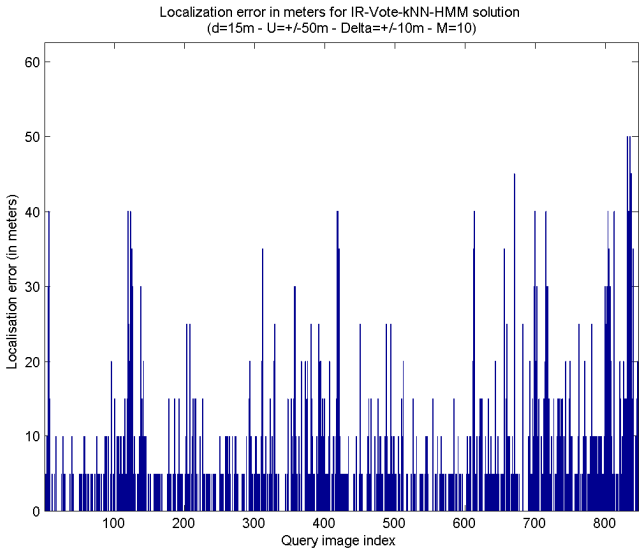


FIGURE 4.17 – Erreurs commises en mètres pour les 846 images requête.

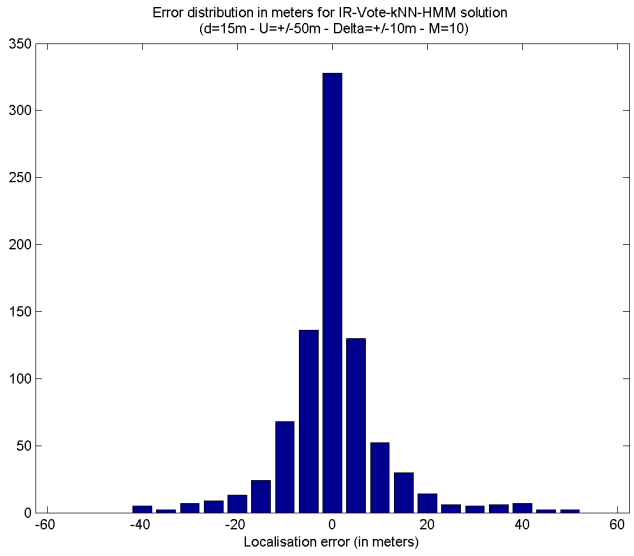


FIGURE 4.18 – Distribution des erreurs commises.

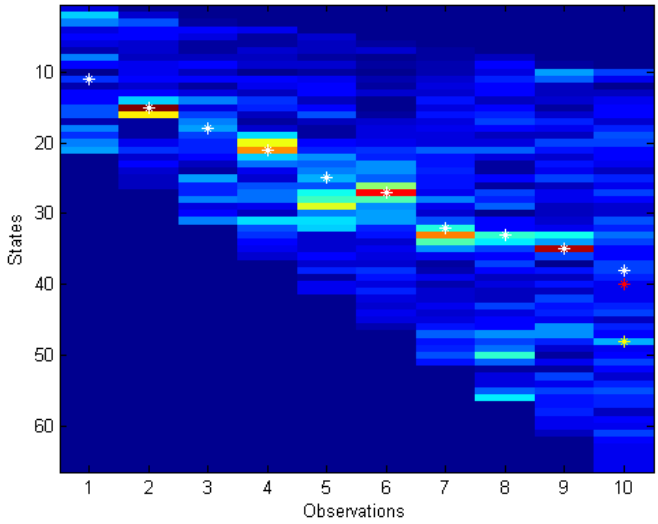


FIGURE 4.19 – Matrice **B** et décodage de Viterbi associé représenté par les étoiles blanches. L'étoile jaune représente l'état retourné sans l'utilisation du HMM, l'étoile rouge représente le vrai état. Image requête n°17.

tion importante sont les cas pour lesquels plusieurs observations consécutives ont des similarités non discriminantes avec les images géoréférencées. Les figures 4.20 et 4.21 sont deux exemples qui illustrent ce problème. Dans ces deux cas, l'erreur commise est de $40m$. Lorsque les mesures de similarités sont ambiguës, l'information apportée par les mesures odométriques doit permettre une estimation correcte. En ce qui nous concerne, nous avons supposé que la distribution des mesures pour un déplacement donné était uniforme sur un certain intervalle. L'information "apportée" concerne donc les transitions possibles entre deux états successifs. La distribution spécifiée ne privilégie aucune transition. Un meilleur modèle (de type gaussien ou autre) permettrait probablement d'améliorer l'estimation réalisée. Ce modèle est dépendant du capteur odométrique utilisé.

La figure 4.22 est un autre cas illustrant ce problème. L'erreur commise est de $30m$. Si l'on décide d'attendre pour prendre la décision, c'est à dire si l'on acquiert de nouvelles observations, alors l'erreur sur la position peut significativement se réduire. Pour cet exemple si l'on considère 3 observations de plus ($M = 13$), alors l'erreur de position n'est plus que de $5m$, comme l'illustre la figure 4.23. Notons, cependant, que l'estimation de la position pour l'observation 10 reste inchangée.

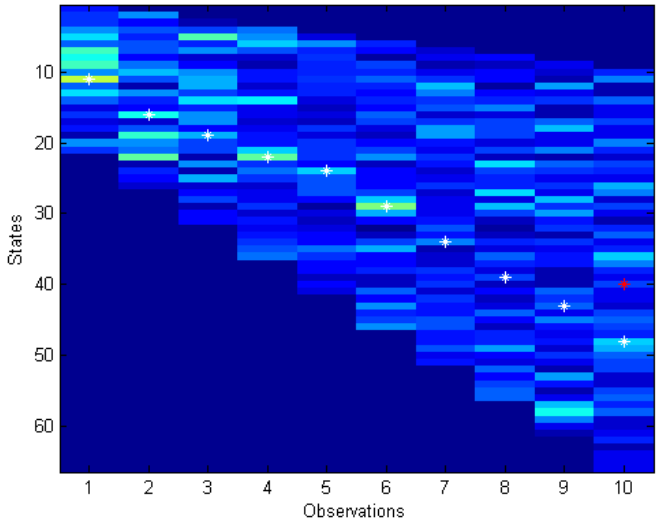


FIGURE 4.20 – Matrice **B** et décodage de Viterbi associé représenté par les étoiles blanches. L'étoile rouge représente le vrai état. Image requête n°417.

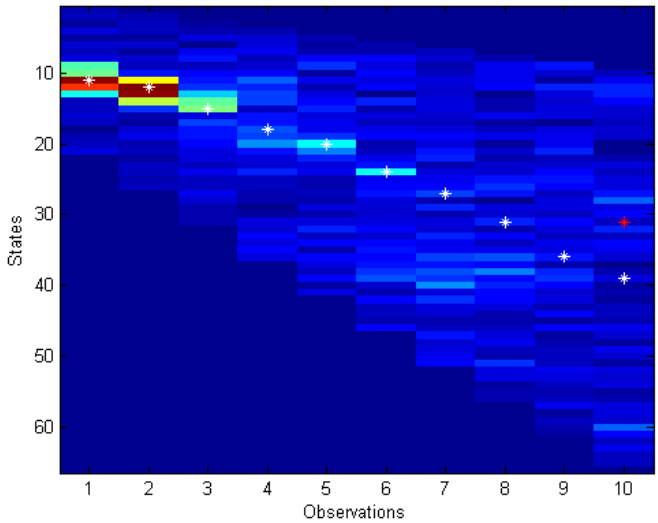


FIGURE 4.21 – Matrice **B** et décodage de Viterbi associé représenté par les étoiles blanches. L'étoile rouge représente le vrai état. Image requête n°612.

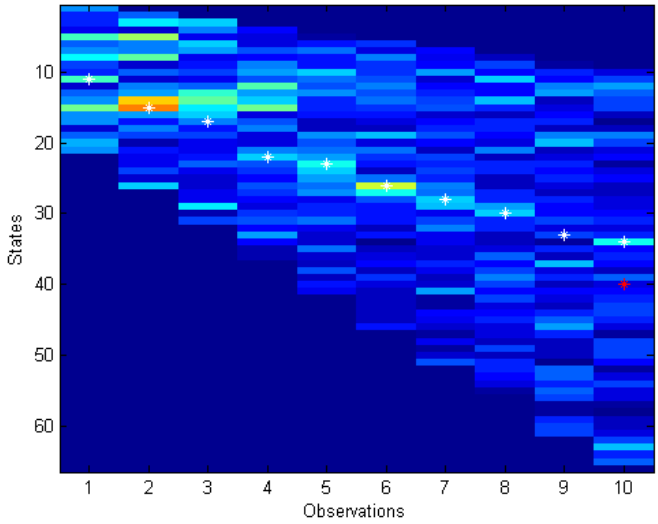


FIGURE 4.22 – Matrice **B** et décodage de Viterbi associé représenté par les étoiles blanches. L'étoile rouge représente le vrai état. Image requête n°688. 10 observations sont considérées.

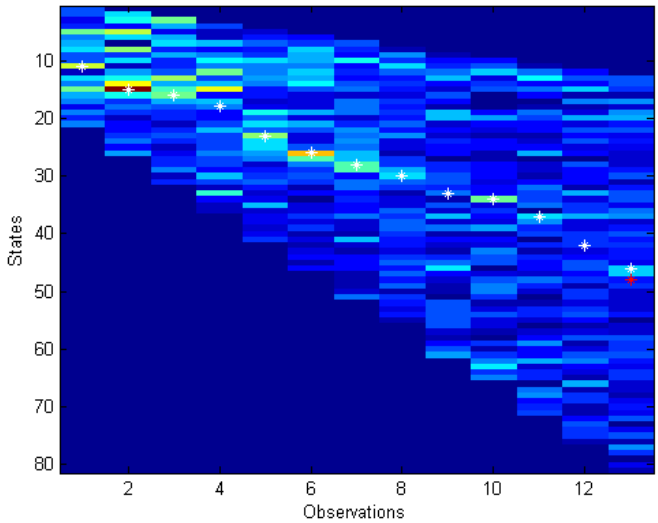


FIGURE 4.23 – Matrice **B** et décodage de Viterbi associé représenté par les étoiles blanches. L'étoile rouge représente le vrai état. Image requête n°688. 13 observations sont considérées.

4.5 Conclusion

Dans ce chapitre, nous avons présenté une méthode de localisation absolue reposant sur une séquence d'images et des mesures odométriques, combinées à l'aide d'une chaîne de Markov à états cachés. Le modèle de Markov permet, étant donné une séquence d'observations, d'estimer la meilleure séquence d'états et donc la trajectoire et la position actuelle du robot. L'emploi d'un modèle HMM nécessite la définition de paramètres que nous avons explicité en fonction de caractéristiques opérationnelles notamment la position initiale approximative et les incertitudes sur les mesures odométriques. La définition d'un modèle de Markov à un instant t permet de complètement ré-estimer la dernière partie de la trajectoire du robot correspondant aux observations visuelles considérées. Ainsi, les erreurs passées éventuelles peuvent être corrigées et une navigation long-terme devient possible.

Les expérimentations menées nous ont permis de valider la stratégie proposée : considérer les contraintes spatio-temporelles permet de réduire significativement l'erreur moyenne de localisation en supprimant les appariements d'images aberrants (d'un point de vue spatio-temporel). Nous avons quantifié l'apport de notre solution à l'aide de la base d'images "Pittsburgh". Nous avons également déterminé le nombre d'observations à considérer, pour une incertitude de localisation initiale donnée, afin de garantir une erreur de localisation moyenne inférieure à un seuil, et montrer qu'au delà d'un certain nombre d'observations le gain apporté n'est plus significatif.

Même si la solution proposée permet de réduire les erreurs commises, des erreurs importantes demeurent lorsque les similarités entre les images géoréférencées et les dernières images acquises sont ambiguës. Si l'on pouvait détecter ces cas ambigus, plusieurs approches pourraient être projetées. Il serait par exemple envisageable de retarder la prise de décision en réalisant de nouvelles acquisitions. Une autre solution pourrait consister à agir sur la matrice $\mathbf{A}$ pour augmenter notre confiance dans les mesures odométriques. Notons enfin, que les bases d'images que nous avons utilisées ont été construites de telle façon que, pour chaque image requête, une image géoréférencée représentant la même scène existe. Si tel n'est pas le cas d'autres solutions doivent être proposées : un état qui aurait le statut "indéterminé" pourrait, par exemple, être considéré pour la chaîne de Markov à états cachés que nous avons proposée.

Les performances de la solution proposée dépendent essentiellement de la qualité de la matrice $\mathbf{A}$, c'est à dire des mesures odométriques, et de la qualité de la matrice $\mathbf{B}$, c'est à dire des mesures de similarité visuelle. Nous avons décidé, pour la suite de nos travaux, de construire une mesure de similarité plus robuste, compatible des contraintes mémoire imposées par une plateforme de type petit drone. Cette solution est présentée dans le chapitre 5.

CHAPITRE 5

Amélioration de la similarité visuelle par apprentissage

Sommaire

Dans le chapitre 4, nous avons proposé une solution de localisation absolue basée sur l'exploitation de la contrainte spatio-temporelle inhérente à l'acquisition séquentielle d'images. La solution proposée combine les mesures de similarité visuelle, calculées à l'aide d'une solution de recherche d'images de l'état de l'art, avec des mesures odométriques au sein d'une chaîne de Markov à états cachés (*Hidden Markov Model - HMM*) où les états cachés sont les lieux géographiques. La résolution s'appuie sur un décodage de Viterbi, dont l'efficacité dépend de la qualité de la matrice de transition **A** (qui inclut les mesures odométriques et les incertitudes associées) et de la qualité de la matrice **B** (qui mesure les similarités entre les observations du véhicule et les images de la base). Dans ce chapitre, nous proposons une solution visant à améliorer les mesures de similarité visuelle.

Nous présentons, tout d'abord, le principe de notre solution (section 5.1) avant de décrire les solutions d'apprentissage de l'état de l'art (section 5.2). Nous détaillons ensuite notre solution d'apprentissage dédiée à la géolocalisation (section 5.3) et, enfin, nous évaluons la méthode proposée et intégrée au sein de la chaîne de Markov précédemment décrite (section 5.4).

5.1 Contexte et solution proposée

Le choix d'une mesure de similarité visuelle est un élément clé pour les solutions de recherche d'images. Les méthodes traditionnelles utilisent pour retrouver l'image la plus similaire à une image requête, la distance euclidienne ou la distance du χ^2 entre les signatures visuelles des images considérées. D'autre part, l'état de l'art réalisé dans la section 2.3.2 a démontré l'émergence de nouvelles solutions, issues de la communauté du *machine learning*, qui reposent sur l'apprentissage des spécificités de chaque lieu [**?**][**?**][**?**][**?**][**?**].

Une solution qui semble attractive est donc l'utilisation des solutions d'apprentissage de métriques (*metric learning*). Elles ont déjà montré leur utilité pour des tâches de classification d'images, de recherche d'images ou de reconnaissance de visages. Pour améliorer notre fonctionnalité de localisation absolue au niveau d'une rue, nous proposons une solution, appelée Exabal (pour *EXAmplar BAsed Learning*), dont le principe consiste à apprendre une distance locale pour chacune des images de la base. Nous souhaitons que la distance $\mathcal{D}(\mathbf{x}_q, \mathbf{x}_k)$ entre la signature visuelle $\mathbf{x}_q$ de l'image requête I_q et la signature visuelle $\mathbf{x}_k$ de l'image I_k soit inférieure aux distances $\mathcal{D}(\mathbf{x}_q, \mathbf{x}_{k'})$ entre la signature visuelle $\mathbf{x}_q$ de l'image requête avec les signatures visuelles des images voisines $\mathbf{x}_{k'}$ pour $k' \neq k$ (Fig. 5.1).

FIGURE 5.1 – Illustration des contraintes images imposées lors de l'apprentissage.

Pour atteindre cet objectif, plusieurs défis se posent :

— Ne disposant pas de l'image requête, il faut pouvoir générer des contraintes qui soient équivalentes de la contrainte $\mathcal{D}(\mathbf{x}_q, \mathbf{x}_k) < \mathcal{D}(\mathbf{x}_q, \mathbf{x}_{k'})$ pour $k' \neq k$ en exploitant uniquement les informations visuelles à notre disposition, c'est à dire la base d'images géoréférencées ;

— Pour que l'apprentissage soit efficace, nous devons utiliser des contraintes suffisamment informatives ;

— Pour éviter le sur-apprentissage, il est nécessaire d'avoir de nombreuses contraintes. Or, nous ne disposons localement que de très

peu d'images.

Les solutions que nous proposons pour résoudre ces différents problèmes sont détaillées dans la section 5.3.

5.2 Solutions d'apprentissage supervisé de distances

5.2.1 Formulation générale du problème

L'objectif d'un algorithme d'apprentissage supervisé de distances est d'adapter une distance donnée (par exemple, la distance de Mahalanobis) pour un problème donné (par exemple la recherche d'images) en utilisant un jeu de contraintes défini au moyen d'exemples d'apprentissage. Plus précisément, un algorithme d'apprentissage de distances a pour objectif de trouver les meilleurs paramètres, afin de respecter au mieux le jeu de contraintes spécifiées.

Ce problème se résout par l'optimisation d'une fonction qui a la forme générale suivante :

$$\min_{\mathbf{M}} [l(\mathbf{M}, \mathcal{E}) + \lambda \cdot r(\mathbf{M})] \tag{5.1}$$

où $\mathbf{M}$ est la matrice des paramètres à apprendre, $\mathcal{E}$ un ensemble de contraintes, $l(\mathbf{M}, \mathcal{E})$ est une fonction de coût (*loss function*), $r(\mathbf{M})$ est une fonction de régularisation des paramètres de $\mathbf{M}$ et λ est le paramètre de régularisation. Nous détaillons ces différents éléments, respectivement, dans les sections 5.2.2, 5.2.3, 5.2.4 et 5.2.5.

5.2.2 Paramétrisation

La distance de Mahalanobis est la distance la plus populaire utilisée dans les algorithmes d'apprentissage. Elle a fait l'objet de nombreuses recherches dans la communauté de la vision et de nombreuses solutions ont été proposées à l'aide de cette distance, pour la reconnaissance de visage par exemple [?][?]. Nous nous plaçons dans la lignée des travaux menés pour l'apprentissage de métriques comme par exemple [?][?][?][?][?].

La distance de Mahalanobis, entre deux vecteurs $\mathbf{x}_i$ et $\mathbf{x}_j$ est définie par :

$$\mathcal{D}_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt[2]{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j)} \tag{5.2}$$

où $\mathbf{M} \in \mathbb{R}^{d \times d}$ est une matrice symétrique définie semi-positive. Rappelons qu'une matrice $\mathbf{M} \in \mathbb{R}^{d \times d}$ est définie semi-positive si toutes ses valeurs propres sont positives ou nulles. L'ensemble de ces matrices est noté $\mathbb{S}_+^d$.

Une telle paramétrisation nécessite d'avoir une représentation vectorielle de chacune des images considérées. Dans la suite du chapitre, nous considérons que ces signatures vectorielles sont des sacs de mots visuels (BOVW).

Une matrice symétrique définie semi-positive peut s'exprimer de la façon suivante : $\mathbf{M} = \mathbf{L}^T\mathbf{L}$ avec $\mathbf{L} \in \mathbb{R}^{e \times d}$ et $e \geq rang(\mathbf{M})$. A l'aide de cette propriété, la distance de Mahalanobis peut donc être ré-écrite :

$$\begin{aligned} \mathcal{D}_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) =& (\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{M}(\mathbf{x}_i - \mathbf{x}_j) \\ =& (\mathbf{x}_i - \mathbf{x}_j)^T\mathbf{L}^T\mathbf{L}(\mathbf{x}_i - \mathbf{x}_j) \\ =& (\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_j)^T(\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_j) \\ =& \|\mathbf{L}(\mathbf{x}_i - \mathbf{x}_j)\|^2 \\ =& \|\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_j\|^2 \end{aligned} \tag{5.3}$$

Calculer la distance de Mahalanobis $\mathcal{D}_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)$ équivaut donc à calculer la distance euclidienne entre les deux vecteurs $\mathbf{x}_i$ et $\mathbf{x}_j$ de $\mathbb{R}^d$ sur lesquels une transformation linéaire $\mathbf{L}$ aurait préalablement été appliquée. Cette équivalence implique que deux approches différentes peuvent être retenues pour apprendre une distance : soit l'optimisation est réalisée sur $\mathbf{L}$, soit elle est réalisée sur $\mathbf{M}$ avec la contrainte que $\mathbf{M}$ doit être une matrice symétrique semi-définie positive. Cependant il faut noter que, contrairement à l'optimisation sur $\mathbf{M}$, l'optimisation sur $\mathbf{L}$ rend le problème non-convexe : la convergence vers un minimum global n'est donc plus garantie.

La paramétrisation telle qu'elle a été précédemment présentée équivaut à appliquer une transformation linéaire et globale sur les données d'entrée. Remarquons que ce modèle a été enrichi pour apprendre des transformations locales et non-linéaires. Des transformations locales ont par exemple été proposées dans [?][?]. Les transformations non linéaires ont été proposées par exemple dans [?][?][?] en exploitant des fonctions noyaux sur la distance de Mahalanobis. Dans [?], les auteurs utilisent l'apprentissage de métriques non-linéaires pour la vérification de visages. La combinaison de ces deux types de paramétrisation a également été explorée dans [?].

5.2.3 Contraintes

Un algorithme d'apprentissage supervisé nécessite des données pour lesquelles on dispose d'annotations. Il peut s'agir d'une étiquette, c'est à dire la classe à laquelle appartient chacune de ces données ou bien de contraintes entre ces données. La plupart des méthodes apprennent une distance à partir de contraintes spécifiées à l'aide de paires d'exemples, de triplets d'exemples, ou même de quadruplets d'exemples.

Les contraintes spécifiées à l'aide de paires d'exemples sont définies par un ensemble d'exemples similaires noté $\mathcal{S}^+ = \{(\mathbf{x}_i, \mathbf{x}_j)\}$ pour lequel les distances $\mathcal{D}(\mathbf{x}_i, \mathbf{x}_j)$ doivent être minimisées et par un ensemble d'exemples non similaires noté $\mathcal{S}^- = \{(\mathbf{x}_i, \mathbf{x}_j)\}$ pour lequel les distances $\mathcal{D}(\mathbf{x}_i, \mathbf{x}_j)$ doivent être maximisées. Autrement dit, il s'agit de contraintes basées sur des critères de similarité/non similarité, qui, mathématiquement, s'expriment ainsi :

$$\begin{cases} \mathcal{D}_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \leq l \text{ pour } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}^+ \\ \mathcal{D}_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq u \text{ pour } (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}^- \end{cases}$$

Les contraintes spécifiées à l'aide de triplets d'exemples sont, quant à elles, définies par un ensemble $\mathcal{R} = \{(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)\}$ pour lequel les distances entre $\mathbf{x}_i$ et $\mathbf{x}_j$ doivent être plus faibles que les distances entre $\mathbf{x}_i$ et $\mathbf{x}_k$. Autrement dit, il s'agit de contraintes entre distances, qui, mathématiquement s'expriment par : $\mathcal{D}_M(\mathbf{x}_i, \mathbf{x}_j) \leq \mathcal{D}_M(\mathbf{x}_i, \mathbf{x}_k) - m$ où m est une marge, généralement égale à 1.

Récemment, les méthodes reposant sur la spécification de triplets ont été généralisées [?]. Elles reposent sur la spécification de quadruplets d'exemples et elles permettent de spécifier des contraintes relatives de la forme suivante : $\mathcal{D}_M(\mathbf{x}_i, \mathbf{x}_j) \leq \mathcal{D}_M(\mathbf{x}_k, \mathbf{x}_l) - m$ avec $\mathbf{x}_i \neq \mathbf{x}_j \neq \mathbf{x}_k \neq \mathbf{x}_l$. Elles ont montré leur intérêt dans certains contextes applicatifs [?].

5.2.4 Fonction de coût

La fonction de coût a pour objectif de pénaliser les contraintes qui sont violées. En général $l(\mathbf{M}, \mathcal{E})$ s'exprime de cette façon : $\sum_{i=1}^{i=p} c_i(\mathbf{M})$ où c_i est l'ensemble des p contraintes. La fonction naturelle pour cette fonction de coût est la fonction de coût $0/1$ (*zero-one loss function*) pour laquelle $c_i(\mathbf{M}) = 0$ lorsque la contrainte est respectée et $c_i(\mathbf{M}) = 1$ lorsque la contrainte est violée. Cependant il est difficile de réaliser l'optimisation de l'équation 5.1 avec cette fonction. C'est pourquoi la fonction de coût $0/1$ est remplacée par une fonction de substitution (*surrogate function*), qui est généralement convexe et qui a de bonnes propriétés de convergence. De nombreuses fonctions de substitution ont été proposées : la fonction *hinge loss* [?], la fonction *square hinge loss* [?], la fonction *Huber loss* [?], la fonction *exponantial loss* [?] et la fonction *logistic loss* [?] [?]. Le choix de la fonction de coût dépend de l'application visée. Dans le cas de la classification, par exemple, Rosasco *et al.* [?] ont démontré que la fonction de substitution *hinge loss* a une vitesse de convergence meilleure que la fonction *logistic loss*.

5.2.5 Fonction de régularisation

La fonction de régularisation $r(\mathbf{M})$ a pour objectif d'inclure une information *a priori* sur les paramètres à apprendre. Dans notre contexte, le nombre de paramètres à apprendre croît de manière quadratique avec la dimension des données d'entrée. Il est donc nécessaire de choisir une fonction de régularisation appropriée pour éviter le sur-apprentissage. Il peut s'agir, par exemple, de :

- la norme de Frobenius $\|\mathbf{M}\|_{\mathcal{F}}^2$, facile à optimiser [?] ;
- la trace de la matrice $\mathbf{M}$ notée $tr(\mathbf{M})$, c'est à dire la somme des valeurs propres. Cette fonction promeut des matrices $\mathbf{M}$ avec un faible rang ;
- la régularisation appelée Fantope proposée par M.Law *et al.* [?] qui

permet un contrôle explicite du rang de la matrice $\mathbf{M}$. Le contrôle du rang de la matrice $\mathbf{M}$ est important car il permet de limiter le sur-apprentissage et de mieux exploiter les corrélations entre les exemples ;

— la divergence LogDet [?], valable uniquement pour les matrices semi-définies positives. Cette régularisation simplifie les calculs ;

— la somme des distances entre les paires de vecteurs similaires $\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}^+} \mathcal{D}^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j)$ [?] [?].

5.2.6 Principales approches

Les différentes solutions d'apprentissage de distances de Mahalanobis existantes diffèrent de par leur capacité à converger globalement, le type de contraintes, la fonction de régularisation et la fonction de coût. Le tableau 5.1 classe les principales méthodes existantes à l'aide de ces critères.

Méthode	Convergence	Fonction de régularisation	Contrôle du rang	Contraintes
MMC [?]	Globale	$\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}^+} \mathcal{D}^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j)$	Non	Paires
Schultz and Joachims [?]	Global	Norme de Frobenius	Non	Triplets
NCA [?]	Local	Optimisation sur $\mathbf{L}$	Non	Pour kNN
LMNN [?]	Global	$\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}^+} \mathcal{D}^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j)$	Non	Triplets pour kNN
ITML [?]	Global	Divergence LogDet	Non	Paires
LDML [?]	Global	Aucune	Non	Paires
PCCA [?]	Local	Optimisation sur $\mathbf{L}$	Oui	Paires
Fantope [?]	Global	Fantope	Oui	Paires, Triplets & Quadruplets

TABLE 5.1 – Principales solutions d'apprentissage de distances de Mahalanobis.

Pour une liste exhaustive des algorithmes d'apprentissage de distances existants, il convient de se référer à [?] et à [?].

L'algorithme le plus populaire est l'algorithme LMNN *Large Margin Nearest Neighboor (LMNN)* proposé par K. Weinberger *et al.* [?]. Cet algorithme est adapté pour la classification kNN. Le principe repose sur le fait que pour chaque exemple d'entrée ses k plus proches voisins doivent appartenir à la même classe, alors que les exemples n'appartenant pas à la classe de l'exemple considéré doivent être distants d'une certaine marge. La figure 5.2 illustre le principe de la méthode. Pour un vecteur d'entrée donné, les k plus proches voisins appartenant à la même classe, sont sélectionnés avant l'apprentissage

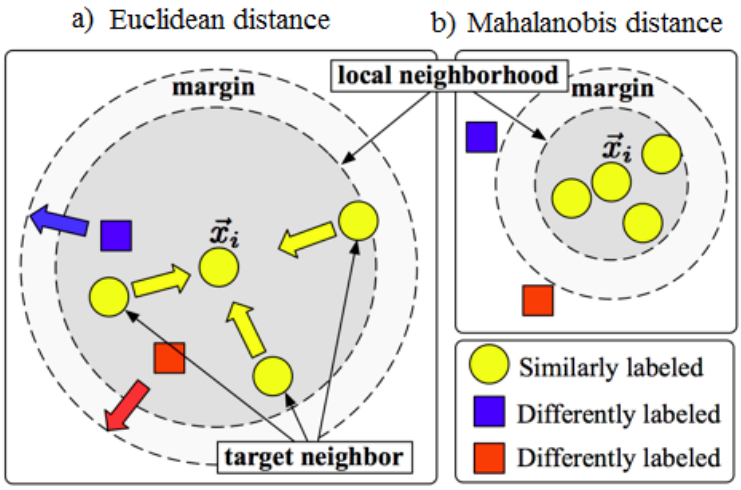


FIGURE 5.2 – Principe de l'algorithme *Large Margin Nearest Neighboor*. Figure réalisée par K.Weinberger.

à l'aide de la norme euclidienne dans l'espace d'entrée (Fig. 5.2-a). Ces k plus proches voisins sont appelés voisins cibles (*target neighbors*). Ils doivent, à l'issue de l'apprentissage de la distance de Mahalanobis, devenir les k plus proches voisins du vecteur d'entrée (Fig. 5.2-b). Au même moment, toujours pour ce même vecteur d'entrée, les k plus proches voisins n'appartenant pas à la même classe que ce vecteur d'entrée sont également sélectionnés (Fig. 5.2-a). Ces plus proches voisins sont appelés imposteurs. Le nombre d'imposteurs à l'issue de l'apprentissage doit être minimisé (Fig. 5.2-b). Autrement dit, pour chaque vecteur d'entrée, ses voisins cibles doivent être rapprochés alors que ses imposteurs doivent être éloignés. Le fait de ne considérer que k voisins cibles et imposteurs rend possible l'apprentissage, qui serait sinon trop complexe.

5.3 Méthode proposée

Nous détaillons dans cette section la méthode que nous avons retenue et nos contributions.

5.3.1 Génération de contraintes

Les contraintes que nous souhaitons générer s'expriment sous la forme de triplets. Pour rappel, nous souhaitons imposer que la distance entre la signature visuelle $\mathbf{x}_q$ de l'image requête I_q et la signature visuelle $\mathbf{x}_k$ de l'image I_k soit inférieure aux distances entre la signature visuelle $\mathbf{x}_q$ de

l'image requête avec les signatures visuelles des images voisines $\mathbf{x}_{k'}$ pour $k' \neq k$ (Fig. 5.1). Le premier défi est de trouver des images qui soient représentatives des images requête inconnues à ce stade et que nous devrons localiser. Pour atteindre cet objectif, nous proposons de générer des images en appliquant des transformations géométriques et photométriques sur les images géoréférencées afin de simuler de potentielles images requête. La figure 5.3 illustre notre schéma de génération d'images simulées et des signatures correspondantes. Mathématiquement, nous notons $\mathcal{T}^{(i)}$ l'ensemble

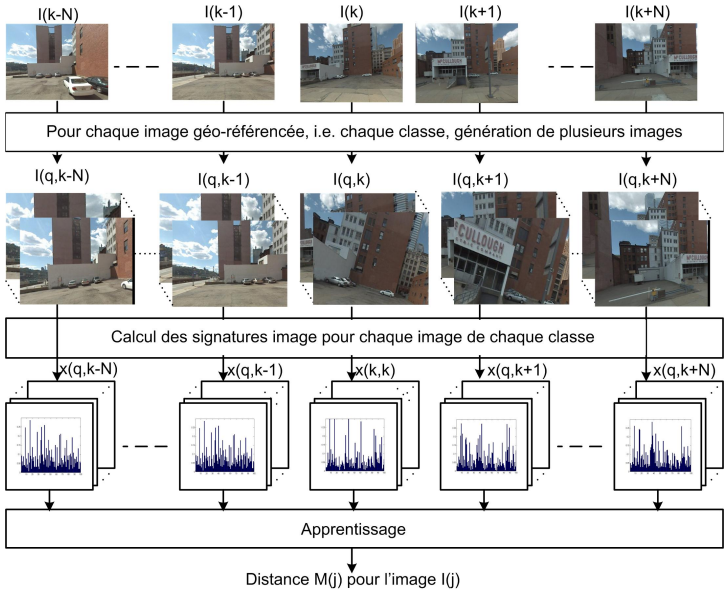


FIGURE 5.3 – Principe de la génération de données pour l'apprentissage.

des transformations image possibles. Pour $(T_m^{(i)}, T_n^{(i)}) \in \mathcal{T}^{(i)} \times \mathcal{T}^{(i)}$, nous obtenons les images transformées $T_m^{(i)}(I_k)$ et $T_n^{(i)}(I_{k'})$ pour I_k et $I_{k'}$ respectivement. Nous notons $T_m(\mathbf{x}_k)$ et $T_n(\mathbf{x}_{k'})$ la signature vectorielle de $T_m^{(i)}(I_k)$ et $T_n^{(i)}(I_{k'})$ respectivement. Nous notons $\mathcal{D}_{\mathbf{M}_k}$ la distance de Mahalanobis paramétrée par la matrice $\mathbf{M}_k$ et valide pour l'image I_k. Avec ces notations, les contraintes pour chaque paire $(\mathbf{x}_k, \mathbf{x}_{k'})$ entre les distances précédemment rappelées s'expriment mathématiquement par :

$$\mathcal{D}_{\mathbf{M}_k}(\mathbf{x}_k, T_n(\mathbf{x}_{k'})) \geq \mathcal{D}_{\mathbf{M}_k}(\mathbf{x}_k, T_m(\mathbf{x}_k)) + 1 \ \forall n, \forall m \tag{5.4}$$

Les contraintes de l'équation (5.4) permettent de promouvoir les matrices $\mathbf{M}_k$ qui vont permettre de discriminer les images I_k des images voisines $I_{k'}$, tout en prenant en compte de potentielles transformations. La marge que l'on impose est ici fixée à 1. Une propriété intéressante de notre solution est la capacité de générer un grand nombre de contraintes en utilisant différentes transformations $T_m^{(i)}$ et $T_n^{(i)}$, rendant l'optimisation de $\mathbf{M}_k$ (potentiellement avec un nombre élevé de paramètres) robuste au sur-apprentissage.

Le nombre de paramètres à apprendre est fonction de la dimension de la matrice $\mathbf{M}_k$ qui dépend directement de la dimension de la signature visuelle $\mathbf{x}_k$.

5.3.2 Apprentissage d'invariances

Compte-tenu de notre scénario, il peut exister d'importants changements visuels entre deux images représentant la même scène. Ces différences sont principalement dues à des différences de point de vue et à des différences d'échelle. C'est pourquoi nous proposons d'appliquer comme transformations géométriques des rotations selon les trois axes et des zooms (rognages et redimensionnement). On espère que les images ainsi simulées soient représentatives de potentielles images requête. D'autres transformations géométriques peuvent être appliquées, par exemple des transformations radiométriques, et facilement intégrées au sein de notre solution d'apprentissage. La figure 5.4 représente certaines de ces transformations pour une image donnée de la base.

5.3.3 Formulation du problème

L'objectif visé est la minimisation du nombre de contraintes non-respectées définies par l'équation (5.4). Nous introduisons pour cela une fonction de substitution (cf. section 5.2.4) de type *hinge loss* ℓ_d pour pénaliser chaque contrainte violée :

$$\ell_d(\mathbf{x}_k, T_m(\mathbf{x}_k), T_n(\mathbf{x}_{k'})) = \max\left[0, 1 - (\mathcal{D}_{\mathbf{M}_k}(\mathbf{x}_k, T_n(\mathbf{x}_{k'})) - \mathcal{D}_{\mathbf{M}_k}(\mathbf{x}_k, T_m(\mathbf{x}_k)))\right] \tag{5.5}$$

Nous introduisons également la fonction ℓ_s dont l'objectif est de minimiser la distance entre chaque image I_k avec ses versions transformées $T_m^{(i)}(I_k))$, c'est à dire avec ses images similaires : $\ell_s(\mathbf{x}_k, T_m(\mathbf{x}_k)) = \mathcal{D}_{\mathbf{M}_k}(\mathbf{x}_k, T_m(\mathbf{x}_k))$.

La fonction finale à optimiser $\mathcal{P}(\mathbf{M}_k)$ est une combinaison des fonctions

FIGURE 5.4 – Exemple de transformations géométriques appliquées sur une image géoréférencée donnée.

ℓ_s and ℓ_d, valables pour l'ensemble des contraintes :

$$\begin{aligned}
\mathcal{P}(\mathbf{M}_k) &= (1-\mu) \sum_{T_m \in \mathcal{T}} \ell_s(\mathbf{x}_k, T_m(\mathbf{x}_k)) \\
&\quad + \mu \sum_{\substack{k' \neq k, \\ (T_n, T_m) \in \mathcal{T} \times \mathcal{T}}} \ell_d(\mathbf{x}_k, T_m(\mathbf{x}_k), T_n(\mathbf{x}_{k'})) \\
\mathcal{P}(\mathbf{M}_k) &= (1-\mu) \sum_{T_m \in \mathcal{T}} \mathcal{D}_{\mathbf{M}_k}(\mathbf{x}_k, T_m(\mathbf{x}_k)) \\
&\quad + \mu \sum_{\substack{k' \neq k, \\ (T_n, T_m) \in \mathcal{T} \times \mathcal{T}}} \\
&\quad max\left[0, 1 - (\mathcal{D}_{\mathbf{M}_k}(\mathbf{x}_k, T_n(\mathbf{x}_{k'}) - D_{\mathbf{M}_k}(\mathbf{x}_k, T_m(\mathbf{x}_k))\right] \\
\mathcal{P}(\mathbf{M}_k) &= (1-\mu) \sum_{T_m \in \mathcal{T}} (\mathbf{x}_k - T_m(\mathbf{x}_k))^T \mathbf{M}_k (\mathbf{x}_k - T_m(\mathbf{x}_k)) \\
&\quad + \mu \sum_{\substack{k' \neq k, \\ (T_n, T_m) \in \mathcal{T} \times \mathcal{T}}} max[0, 1 - \\
&\quad (\mathbf{x}_k - T_n(\mathbf{x}_{k'}))^T \mathbf{M}_k (\mathbf{x}_k - T_n(\mathbf{x}_{k'})) + (\mathbf{x}_k - T_m(\mathbf{x}_k))^T \mathbf{M}_k (\mathbf{x}_k - T_m(\mathbf{x}_k))]
\end{aligned} \tag{5.6}$$

où μ est le paramètre de régularisation. La fonction de coût exprimée par l'équation (5.6) est convexe en $\mathbf{M}_k$.

5.3.4 Optimisation

L'équation (5.6) étant convexe en $\mathbf{M}_k$, il est possible de la résoudre à l'aide d'un algorithme de descente de gradient avec une projection à chaque itération de la matrice $\mathbf{M}_k$ sur le cône des matrices semi-définies positives. Ceci peut se résoudre à l'aide d'un *solver* générique. Développer un *solver* optimisé est une lourde tâche. Nous avons décidé de nous appuyer sur l'algorithme LMNN proposé par K.Weinberger car il inclut le *solver* voulu et car son implémentation a été optimisée d'un point vue algorithmique et processeur. Le code est disponible [**?**] et a démontré son efficacité.

L'algorithme LMNN, tel qu'il a été proposé par Weinberger [**?**], permet également d'optimiser des contraintes entre distances exprimées sous la forme d'un ensemble de triplets $\mathcal{R}$. Il est donc adapté à notre problème. La fonction de coût (Equation 5.7) qu'il optimise, est composée de deux termes. Le premier terme $\varepsilon_{Pull}(M)$ pénalise les distances élevées entre chaque vecteur $\mathbf{x}_i$ et ses voisins cibles. Autrement dit, il a pour objectif de rapprocher pour chaque vecteur ses k plus proches voisins. Le second terme $\varepsilon_{Push}(M)$ pénalise les distances faibles entre chaque vecteur et tous les autres vecteurs ne partageant pas la même étiquette. Autrement dit, il a pour objectif d'éloigner pour chaque vecteur les vecteurs ayant une étiquette différente. Ces deux termes ont donc des effets opposés.

$$\mathcal{P}_{\mathbf{M}} = (1-\mu)\varepsilon_{Pull}(M) + \mu\varepsilon_{Push}(M) \tag{5.7}$$

$\mu \in [0,1]$ est un paramètre qui est typiquement défini par validation croisée. K. Weinberber a démontré que son influence était faible sur les performances obtenues [**?**]. En pratique $\mu = 0.5$ est la valeur utilisée.

5.3.4.1 Détails d'implémentation

Pour pouvoir résoudre l'équation 5.6, un algorithme optimisé de descente de gradient est requis. Nous avons décidé d'exploiter l'algorithme LMNN. Pour cela, des adaptations doivent être réalisées.

Instanciation bi-classe vs. multi-classe
L'algorithme LMNN prend en entrée des vecteurs avec une étiquette indiquant leur classe d'appartenance. Pour notre problème, on distingue deux classes : la classe des images similaires à l'image de la base I_k considérée et la classe des images voisines, non similaires à l'image I_k et à ses versions transformées. Si nous instancions l'algorithme LMNN en ne considérant que ces deux classes, cela signifie que l'algorithme va essayer de rapprocher la signature visuelle de deux images voisines différentes, ce qui n'a pas de sens

et peut rendre la matrice apprise inadéquate. C'est pourquoi le LMNN doit être instancié en multi-classe : si nous considérons les $2n$ images voisines (les n images voisines précédant l'image considérée et les n images suivant l'image considérée), alors $2n + 1$ classes sont à considérer.

Informativité des contraintes
Si l'on ne prend pas garde aux exemples générés, les taux de classification avant apprentissage peuvent être déjà élevés, ce qui signifie que, parmi l'ensemble des contraintes générées, très peu de contraintes sont "actives" (c'est à dire que les contraintes spécifiées sont déjà respectées avant même de réaliser l'apprentissage). Rappelons que l'algorithme LMNN, comme nous l'avons mentionné, considère pour chaque vecteur d'entrée ses k plus proches voisins sélectionnés dans l'espace d'entrée à l'aide d'une distance euclidienne. Si nous générons des images similaires aux images requête potentielles, à l'aide de transformations "régulières", par exemple des rotations de -10° à +10° par pas de 1°, alors pour chaque vecteur d'entrée donné $\mathbf{x}_k$, les vecteurs $T_m(\mathbf{x}_k)$, pour lesquels les rotations sont faibles, sont proches du vecteur $\mathbf{x}_k$. Les contraintes basées sur les k plus proches voisins sont donc vérifiées d'emblée. Une solution consisterait à augmenter la valeur de k afin de sélectionner les vecteurs éloignés, mais, dans ce cas, la complexité de l'optimisation rend la résolution impossible. Nous avons donc proposé de ne pas générer des transformations "régulières", mais au contraire uniquement de grandes transformations, par exemple uniquement des rotations de -10°,0° et +10° en espérant que la distance apprise reste valide pour des rotations comprises entre -10° et +10°. Nous l'avons vérifié lors de nos expérimentations.

5.3.5 Mesure des similarités visuelles

Dans cette section, nous précisons comment les différentes distances apprises doivent être utilisées dans notre contexte et la façon dont nous les avons intégrées au sein de la solution décrite dans le chapitre 4.

Comparaison des distances
Pour une image requête donnée I_q, les distances entre cette image et plusieurs images I_j de la base sont calculées. Il n'y a aucune garantie que les différentes distances $D_{\mathbf{M}_j}(\mathbf{x}_q, \mathbf{x}_j)$ soient comparables, puisque chaque optimisation a été réalisée de manière indépendante. Pour résoudre ce problème, chaque matrice $\mathbf{M}_j$ est normalisée de façon à ce que la norme de Frobenius soit égale à 1. D'autres méthodes de normalisation auraient pu être employées, comme celles proposées dans [**?**], mais cette méthode a montré de bonnes performances, lors de nos expérimentations.

Mesure des similarités visuelles
Une fois la matrice $\mathbf{M}_j$ apprise pour chaque image I_j de la base, il de-

vient possible pour une observation donnée O_k de calculer $b_j(k)$ via l'équation (5.8). $b_j(k)$ permet de calculer la matrice $\mathbf{B}$ qui est un des paramètres de la chaîne de Markov proposée dans le chapitre 4. $\mathbf{x}_k$ est la signature visuelle de l'observation O_k et $\mathbf{x}_j$ sont les signatures visuelles des images géoréférencées considérées pour la géolocalisation.

$$\begin{aligned} b_j(k) = & \alpha \cdot exp(-a \cdot \mathcal{D}_{\mathbf{M}_j}(\mathbf{x}_j, \mathbf{x}_k)) \\ & \alpha \cdot exp(-a \cdot (\mathbf{x}_j - \mathbf{x}_k)^T \mathbf{M}_j (\mathbf{x}_j - \mathbf{x}_k)) \end{aligned} \tag{5.8}$$

5.4 Expérimentations

Nous évaluons dans cette section la capacité de chacune des distances apprises à retrouver l'image représentant la même scène que l'image requête. Nous l'évaluons également au sein de la solution proposée dans le chapitre 4. La base d'images et les métriques utilisées sont identiques à celles utilisées dans le chapitre 4 et décrites dans la section 2.5.

5.4.1 Paramètres

Chaque image I_k est décrite par une signature vectorielle $\mathbf{x}_k$ de type BOVW [?] incluant une information spatiale [?]. Les BOVW sont construits à l'aide de descripteurs SIFTs extraits de manière dense à l'aide de [?] : le pas entre chaque descripteur est de 4 pixels et chaque descripteur est extrait à 4 échelles différentes 1, 1.5, 2, et 2.5. Le codage des BOVWs est de type *hard assignment* et le *poolling* de type *sum pooling*. Le vecteur BOVW est normalisé ℓ_2. Différentes configurations pour la pyramide spatiale ont été testées (1x1), (1x1,2x2) et (1x1,2x2,1x3). Pour un dictionnaire de p mots visuels, les dimensions des vecteurs sont donc de $p \times 1$, $p \times 4$ ou $p \times 8$ et le nombre de paramètres de chaque matrice $\mathbf{M}_k$ à apprendre est donc de p^2, $16 \times p^2$, ou $64 \times p^2$. Il augmente de manière quadratique avec la taille du dictionnaire. Pour limiter la complexité de l'optimisation, la taille du dictionnaire a été fixée à 100 mots visuels.

Lors de nos expérimentations (5.4), les transformations appliquées étaient de type rotations (pour simuler les différences de point de vue) et zoom (pour simuler les différences de focales entre images requête et images de la base). Nous avons généré pour chaque image de la base 2475 (pour des rotations variant de 0°, -20°et +20°selon les trois axes et des rognages horizontaux et verticaux variant de 0 à 50 pixels).

En ce qui concerne l'instanciation du LMNN, la valeur de μ dans l'équation 5.7 a été fixée de 0.5 comme le suggère K. Weinberger [?]. La valeur de k a été fixée à 9. Lors des apprentissages, nous avons considéré 10 images voisines (les 5 images précédent l'image géoréférencée considérée et les 5 images suivant l'image géoréférencée considérée), ce qui équivaut à 11 classes.

Avec ces paramètres, l'algorithme LMNN converge rapidement, l'optimisation est rapide, et le nombre de contraintes générées est raisonnable. Chaque optimisation dure environ 1 heure sur un cœur d'un processeur cadencé à 2.8GHz.

5.4.2 Évaluation

Nous comparons, tout d'abord, notre solution, notée (Exabal), avec deux solutions de l'état de l'art. La première met en œuvre une mesure de similarité basée sur la norme Euclidienne entre les BOVW de l'image requête et des images de la base. Cette solution est notée (L2). La deuxième solution, notée (kNN), est basée sur le nombre de descripteurs mis en correspondance entre l'image requête et chacune des images de la base après filtrage géométrique de type RANSAC [**?**]. Ensuite, nous comparons notre solution basée HMM décrite dans les sections 4.3.1 et 4.3.2 intégrant les différentes mesures de similarités citées précédemment. Ces différentes solutions sont notées (Exabal+HMM), (L2+HMM) et (Vote+HMM).

Nous reportons et analysons les résultats obtenus dans la section 5.4.2.1, avec et sans utilisation du filtre HMM donc, et pour différentes configurations de la pyramide spatiale. Nous procédons ensuite à une analyse qualitative des résultats obtenus dans la section 5.4.2.2.

5.4.2.1 Résultats

Les performances obtenues, c'est à dire l'erreur moyenne de localisation, la précision de la recherche d'images et le temps pour traiter une image requête en utilisant les données et les paramètres précédemment décrits, sont reportées dans le tableau 5.2.

Les résultats démontrent l'intérêt de l'apprentissage de distances locales pour la géolocalisation. Sans filtre HMM, les performances obtenues par notre solution sont meilleures qu'une solution standard basée sur la distance Euclidienne entre BOVW. Dans ce cas, notre solution améliore les performances de 7% à 8% en fonction de la configuration spatiale utilisée. Que ce soit avec au sans HMM, elles sont comparables à la méthode basée kNN. Par contre, notre solution a l'avantage d'être compatible avec une plate-forme de type petits drones. La solution proposée est simple et rapide : pour chaque image requête, seulement un sac de mots visuels est à calculer et N distances de Mahalanobis entre ce sac de mots visuels et N sacs de mots des images de la base considérées. Les ressources mémoire requises sont également compatibles avec une telle plate-forme : pour une mission de $11km$ et 2524 images géoréférencées, 18 gigaoctets doivent être stockés (cf. table 5.3). Pour une solution kNN, il faudrait stocker l'ensemble des descripteurs de chacune des images de la base ce qui nécessiterait beaucoup plus de mémoire.

La solution complète intégrant la mesure de similarité au sein d'une

Méthode	Configuration spatiale des BOVW	Erreur de localisation moyenne	Précision	Temps de traitement d'une requête
L2 + HMM	1x1 1x1, 2x2 1x1, 2x2, 1x3	8.0m 5.3m 4.9m	42% 44% 46%	7.2s
Exabal + HMM	1x1 1x1, 2x2 **1x1, 2x2, 1x3**	4.5m 4.1m **3.9m**	52% 53% **54%**	8.6s
Vote + HMM		4.1m	50%	55,6s
L2	1x1 1x1, 2x2 1x1, 2x2, 3x1	17.4m 14.6m 12.9m	35% 38% 40%	6.8s
Exabal	1x1 1x1, 2x2 1x1, 2x2, 1x3	12.0m 11.3m 10.1m	42% 46% 48%	8.2s
Vote		11.8m	50%	55,2s

TABLE 5.2 – Erreur de localisation moyenne, précision et temps pour traiter un image requête avec et sans filtrage HMM pour notre solution et pour des solutions de l'état de l'art.

Dimension signature vectorielle	Taille d'un BOVW (en MO)	Taille de la matrice **M** (en MO)
100	2	2019
500	10	5048
800	16	12922

TABLE 5.3 – Taille en méga-octets des données à stocker pour 2524 images géoréférencées.

chaîne de Markov à états cachés confirment la supériorité de notre solution comparée à la méthode (BOW+HMM) : elle permet un gain significatif de 8% (de 46% à 54%). Dans le même temps, l'erreur de localisation moyenne est réduite de 4.9m à 3.9m.

5.4.2.2 Analyse qualitative

Dans cette section, nous analysons les raisons expliquant l'amélioration des performances obtenues par (Exabal). Nous illustrons, tout d'abord, le fait que la métrique apprise permet de retrouver la bonne image si la signature de l'image requête est proche de celle de l'une des signatures générées artificiellement. Nous démontrons ensuite que l'amélioration des performances est due à la sélection de caractéristiques visuelles discriminantes, à la sélection des zones discriminantes de l'image, et à l'apprentissage d'une invariance aux transformations géométriques et photométriques.

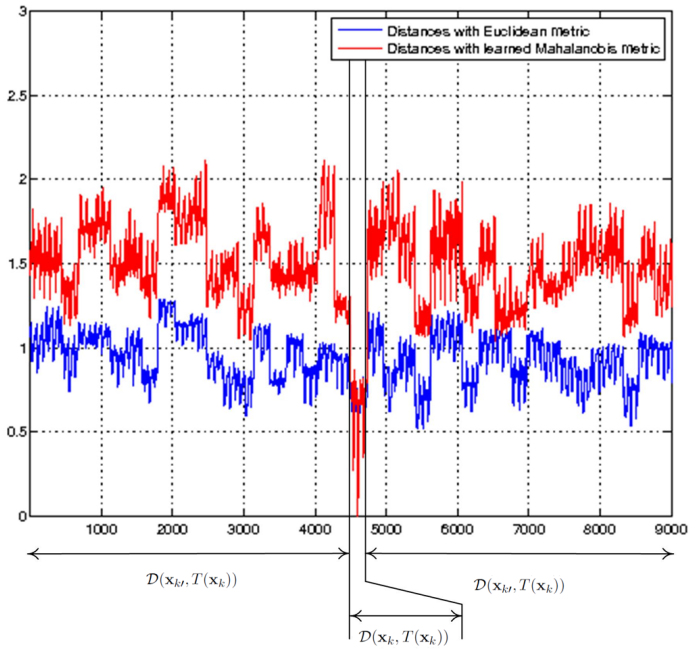


FIGURE 5.5 – Distance Euclidienne distance de Mahalanobis apprise entre de potentielles signatures requête $T_m(\mathbf{x}_k)$ et $\mathbf{x}_k$ et $\mathbf{x}_{k'}$, c'est à dire les signatures des images voisines $I_{k'}$ $(k' \neq k)$.

Distances apprises lors de l'apprentissage La figure 5.5 montre, pour une matrice $\mathbf{M}_k$ apprise, les distances entre tous les BOVWs similaires générés artificiellement $T_m(\mathbf{x}_k)$ (simulant de potentielles images requête) avec les BOVW $\mathbf{x}_k$ et $\mathbf{x}_{k'}$ de l'image géoréférencée I_k et des images géoréférencées

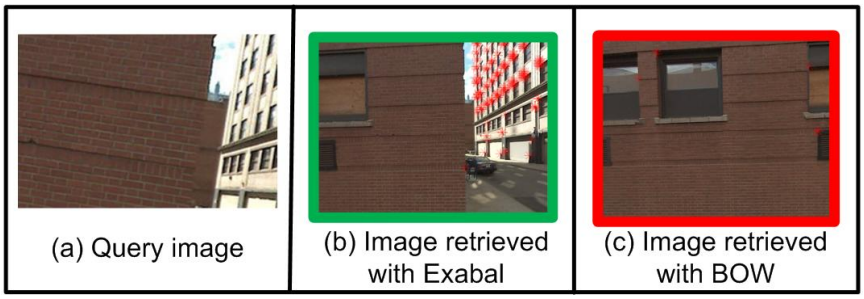


FIGURE 5.6 – (a) Image requête - (b) Image retournée par (Exabal) : La cadre vert indique que l'image retournée est la bonne. Le mot visuel "coin de fenêtre", a été appris comme étant discriminant, ce qui permet d'améliorer la tâche de recherche d'images. (c) Image retournée par la solution (BOW) : le cadre rouge indique qu'il s'agit de la mauvaise image.

voisines $I_{k\prime}$. La métrique apprise permet de discriminer toutes les images requête potentielles générées artificiellement, alors que cela n'était pas possible avec la norme Euclidienne. En effet, quelque soit l'image requête simulée $T_m(\mathbf{x_k})$, $\mathcal{D}_{\mathbf{M}_k}(T_m(\mathbf{x_k}), \mathbf{x_k})$ reste inférieure à $\mathcal{D}_{\mathbf{M}_k}(T_m(\mathbf{x_k}), \mathbf{x_{k'}})$. Cela signifie que si la signature de la requête réelle lors de l'étape de localisation est proche de l'une des images requête générées, alors la bonne image de la base pourra être retrouvée grâce à la distance apprise.

Sélection des caractéristiques visuelles discriminatives

Le premier avantage de notre méthode est qu'elle sélectionne les caractéristiques visuelles discriminatives, c'est à dire celles qui vont permettre de discriminer l'image de la base représentant la même scène que l'image requête des images voisines. Pour visualiser le mot le plus discriminant pour une image donnée de la base, nous avons calculé le vecteur propre $\mathbf{v}_1$ correspondant à la plus grande valeur propre λ_1 de $\mathbf{M}_k$. Ce vecteur représente l'importance de chaque mot visuel, puisque $\mathbf{M}^{'} = \lambda_1 \mathbf{v}_1 \mathbf{v}_1^T$ est la matrice de rang 1 la plus proche de $\mathbf{M}$ (au sens de la norme ℓ_2). Ainsi $D^2_{\mathbf{M}}(\mathbf{x}_j, \mathbf{x}_k) \approx \lambda_1 (\mathbf{v}_1^T(\mathbf{x}_j - \mathbf{x}_k))^2$ et par conséquent les composantes de $\mathbf{v}_1$ pondère l'importance de chaque mot visuel. La figure 5.6 montre le mot visuel ayant la composante la plus grande pour le vecteur $\mathbf{v}_1$. Ce mot visuel "coin de fenêtre" permet de retrouver la bonne image (b), et ce même si de nombreux mots visuels "briques" étaient communs entre l'image requête (a) et l'image voisine (c).

Sélection des zones discriminantes de l'image

Nous avons également vérifié que le poids associé à ce mot visuel était plus élevé pour le quart supérieur droit de l'image. Exabal permet d'apprendre, grâce à l'emploi de sac de mots avec information spatiale, l'endroit ou se

situe les mots visuels discriminants.

Apprentissage de l'invariance aux transformations photométriques et géométriques

Un autre avantage de notre méthode est qu'elle permet d'apprendre une invariance aux transformations géométriques et photométriques. Pour le démontrer, nous avons sélectionné aléatoirement 1000 images de la base sur lesquelles nous avons appliqué différentes rotations (de $-18°$ à $+18°$) autour des 3 axes et des rognages (*crops*) de 6 à 35 pixels sur les bords des images. Ainsi nous avons généré 10000 images requête $I_q = T^{(i)}(I_k)$ qui n'ont pas été utilisées lors de l'étape d'apprentissage. Le taux de classification moyen pour ces 10000 images requête simulées sont reportés dans le tableau 5.4 pour les solutions (BOVW) et (Exabal). Un taux de classification de plus de 99% lors de l'utilisation de la distance de Mahalanobis, alors qu'il n'est que de 90% avec l'emploi d'une distance euclidienne confirme notre affirmation concernant l'invariance aux transformations photométriques et géométriques.

Méthode	Taux de classification moyens
(BOVW)	90.2%
(Exabal)	99.1%

TABLE 5.4 – Taux de classification moyens pour des images test simulées.

5.5 Conclusion

Dans ce chapitre, nous avons proposé une solution apprenant des mesures de similarité visuelle locales, mettant en œuvre un algorithme d'apprentissage supervisé de distances de Mahalanobis. Ne disposant pas des images requête pour l'apprentissage, nous avons généré des images ressemblant aux images requête potentielles en appliquant des transformations géométriques sur les images de la base à notre disposition. Les contraintes d'apprentissage ont été formulées à l'aide de distances entre triplets d'images. Ces contraintes peuvent s'exprimer sous la forme d'une fonction de coût convexe, qui peut être résolue par une descente de gradient. Pour que la résolution soit efficace, nous avons proposé une solution d'optimisation à l'aide de l'algorithme LMNN que nous avons adapté pour notre besoin. Nous avons démontré que notre solution permet d'apprendre une invariance aux changements de point de vue et aux variations de facteurs d'échelle. Dans le cas où la signature visuelle est un sac de mots avec informations spatiales, nous avons également montré que l'apprentissage de la distance permet, pour une image géoréférencée donnée, de déterminer quels sont les attributs visuels, ou mots visuels, discriminants par rapport aux images proches géographiquement. L'évaluation de notre solution a montré qu'elle permettait d'améliorer la précision en

recherche d'images de 40% pour une solution de l'état de l'art basée sacs de mots visuels à 54%, tout en maintenant un temps de calcul quasi-identique. L'erreur de localisation moyenne est, dans le même temps, réduite de $12.9m$ à $3.9m$. La méthode d'apprentissage est générique : nous n'avons considéré que des transformations géométriques, mais, pour d'autres contextes applicatifs, d'autres types de transformations peuvent être considérées, comme par exemple des transformations radiométriques.

Les similarités visuelles apprises ont été incluses dans la chaîne de Markov à états cachés présentée dans le chapitre 4. Nous avons quantifié le gain apporté par la solution globale à l'aide du corpus "Pittsburgh".

Pour l'apprentissage de mesures de similarité visuelle, nous avons utilisé comme signature visuelle les sacs de mots. D'autres signatures peuvent être facilement mises en œuvre, et notamment les *deep features* dont l'efficacité a récemment été démontrée [**?**]. D'autre part, nous n'avons considéré, dans nos travaux, que le cas des transformations linéaires. Les transformations non linéaires ont été proposées par exemple dans [**?**][**?**][**?**] en exploitant des fonctions noyaux. D'autres solutions d'apprentissage pourraient également être testées, comme par exemple les *Structural SVM* [**?**]. L'idée sous-jacente consisterait à apprendre une mesure de similarité visuelle qui, pour une image requête donnée, ordonnerait les images de la base pouvant correspondre à cette image requête. Une variable latente pourrait également être introduite [**?**] pour apprendre explicitement les régions et les mots visuels discriminants en s'inspirant des travaux réalisés dans [**?**] par exemple. L'apprentissage explicite de motifs spatiaux non informatifs permettrait de s'affranchir des motifs parasites (*clutter*) présents dans les images, comme par exemple les voitures ou les nuages.

CHAPITRE 6

Conclusion générale et perspectives

Les travaux présentés dans ce manuscrit concernent l'ego-localisation d'un robot se déplaçant dans un environnement urbain. La localisation visuelle par reconnaissance de lieu est une problématique majeure pour la navigation autonome. Pour la résoudre, nous avons proposé une solution reposant sur l'appariement des dernières images acquises par le robot avec une base d'images géoréférencées provenant du service web *Google Streetview*. Notre solution permet d'améliorer les méthodes existantes de recherche d'images pour le contexte particulier de la navigation d'un robot.

6.1 Bilan

Le fil conducteur de notre démarche a consisté à exploiter les spécificités de notre problème de navigation afin d'améliorer les solutions de recherche d'images de l'état de l'art. Les spécificités, qui ont été en particulier exploitées, sont les contraintes spatio-temporelles inhérentes au déplacement d'un robot, la possibilité d'exploiter des informations non fiables et peu précises provenant d'autres capteurs, et la disponibilité d'une base d'images géoréférencées permettant de construire des mesures de similarités locales.

Après avoir construit des corpus d'images représentatifs de notre problématique, nous avons tout d'abord proposé une méthode hybride exploitant à la fois des informations visuelles fournies par une unique caméra et des informations odométriques pouvant être fournies, par exemple, par une centrale inertielle ou un algorithme d'odométrie visuelle. Notre méthode repose sur une chaîne de Markov à états cachés, où les états cachés sont les positions du robot pour lesquelles on dispose d'observations. Nous avons défini

explicitement les paramètres du modèle HMM en fonction de données opérationnelles notamment la position initiale approximative et les incertitudes sur les mesures odométriques. Nous avons également étudié l'influence de certains des paramètres du modèle HMM, et quantifié le gain engendré par l'adjonction de ce modèle sur un algorithme de recherche d'images standard. Cette solution et les résultats associés sont présentés dans [**?**].

Les performances obtenues dépendent principalement de la qualité de la mesure de similarité visuelle. C'est pourquoi, afin d'améliorer cette mesure, nous avons proposé de déterminer, pour chaque image géoréférencée, une mesure de similarité locale. Pour cela, nous apprenons de manière supervisée une distance de Mahalanobis qui permet de mesurer la similarité entre l'image requête et les images géoréférencées proches de la position supposée. Nous avons démontré que notre méthode permet d'apprendre une invariance aux transformations pouvant survenir lors de l'acquisition de l'image requête, en particulier une invariance aux changements de point de vue et aux variations d'échelles. L'évaluation de notre solution a montré qu'elle permettait d'améliorer la précision en recherche d'images. Cette solution de recherche d'images basée apprentissage a été décrite et évaluée dans [**?**]. L'évaluation de la solution complète intégrant les mesures de similarités apprises au sein d'une chaîne de Markov à états cachés a démontré l'intérêt de notre approche. La méthode et les résultats associés sont décrits dans [**?**].

Nous avons également étudié une solution pour affiner la position estimée grâce à un algorithme d'estimation de pose relative. L'étude réalisée a permis de quantifier l'impact des différentes erreurs commises lors de l'appariement de points. Cette étude a montré, notamment, que l'estimation de la direction du vecteur translation entre les deux caméras est instable dans notre contexte vis à vis des erreurs commises. Néanmoins, même si des incertitudes demeurent, il est possible de les quantifier à l'aide de simulations représentatives du scénario visé et du matériel utilisé (propriétés physiques des caméras).

6.2 Perspectives

A court terme, plusieurs améliorations semblent envisageables. Tout d'abord, que ce soit pour la fonctionnalité de recherche d'images ou pour la fonctionnalité d'estimation de pose, nous avons uniquement eu recours à des descripteurs locaux. Pour une localisation robuste et long-terme, ces descripteurs pourraient être combinés avec des descripteurs globaux, comme l'a proposé récemment A. Zamir dans [**?**], ou tout autre type de descripteurs. Dans le cas où le robot effectuerait les mêmes missions de manière régulière, un apprentissage continu pourrait être effectué. La base d'images utilisée pour l'apprentissage serait alors enrichie au fur-et-à-mesure des missions. Il serait également intéressant d'évaluer comment se comporte notre approche dans

des environnements autres, par exemple à l'intérieur de bâtiments.

A moyen terme, d'autres pistes pourraient être étudiées. L'interprétation sémantique de scènes en est une. Les solutions actuelles de géolocalisation visuelle reposent pour la plupart sur la mise en correspondance de descripteurs bas niveau (points SIFT, motifs, etc.) entre une ou plusieurs images requête et une base d'images de référence. Le développement de solutions de localisation qui prennent en compte une information plus haut niveau de type sémantique est un axe de recherche pour le futur. Ces informations peuvent être du texte, une segmentation des objets dans la scène, des informations sur l'architecture des bâtiments, etc. Une segmentation sémantique basique de la scène, c'est à dire avec très peu de classes, peut être une première étape. Elle permettrait de faciliter l'appariement de points. Une autre piste concerne l'apprentissage de détecteurs de motifs non locaux comme cela a été récemment proposé dans [**?**].

Enfin, comme nous l'avons mentionné dans le chapitre 2, une solution de localisation absolue est une des fonctionnalités requises pour le développement d'un système de navigation visuelle autonome. Ce n'est pas la seule : d'autres fonctionnalités sont indispensables. Notre solution doit être combinée avec une solution de navigation locale de type SLAM afin de permettre la navigation court-terme. Un algorithme de détection et de localisation des objets mobiles est également nécessaire afin que le robot puisse les éviter. Connaître leurs positions et leurs déplacements par rapport au robot est crucial. L'ensemble de ces informations sont indispensables pour planifier la trajectoire du robot de manière optimale. La figure 6.1 représente une architecture possible d'un système de navigation visuelle autonome.

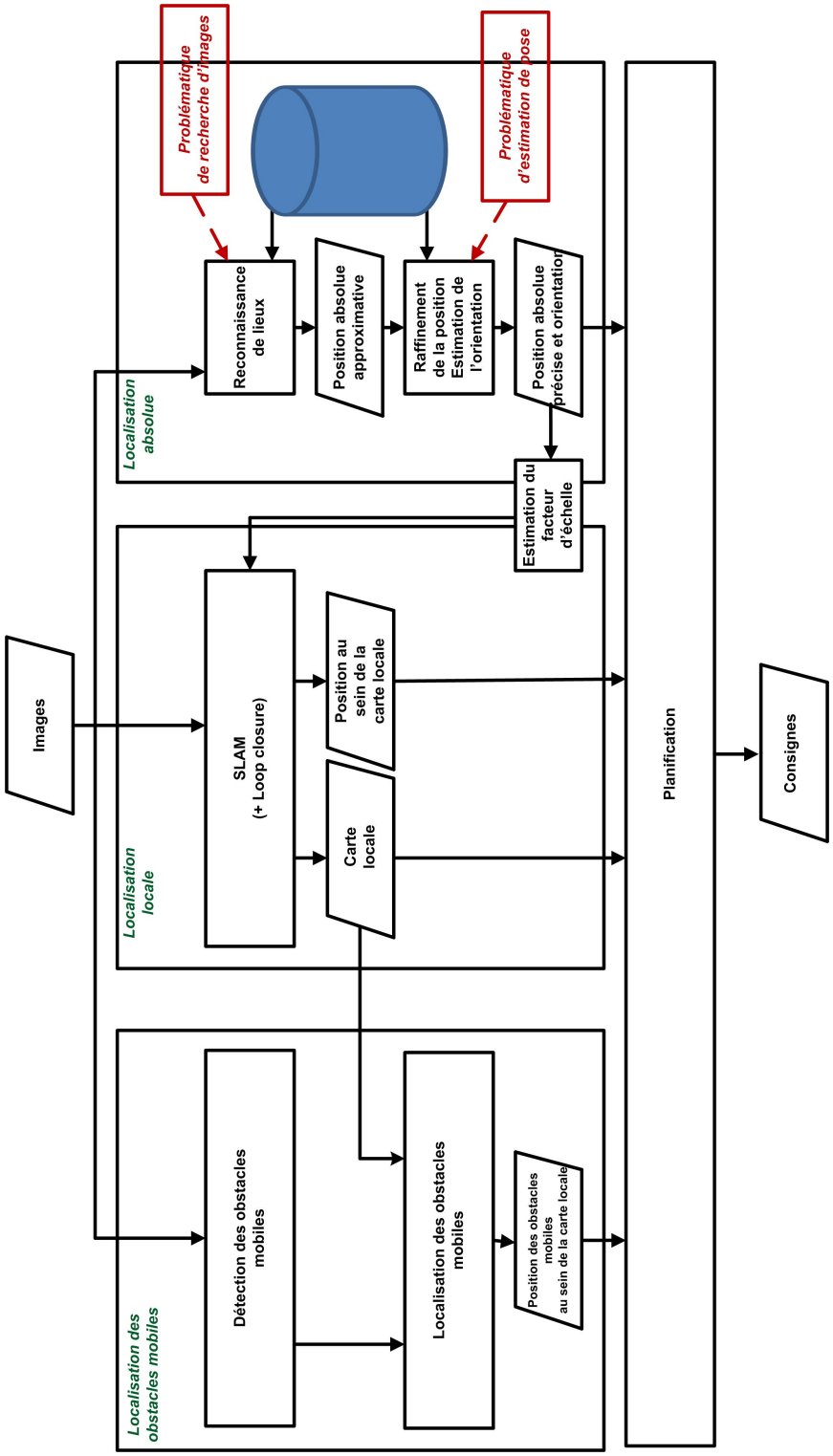
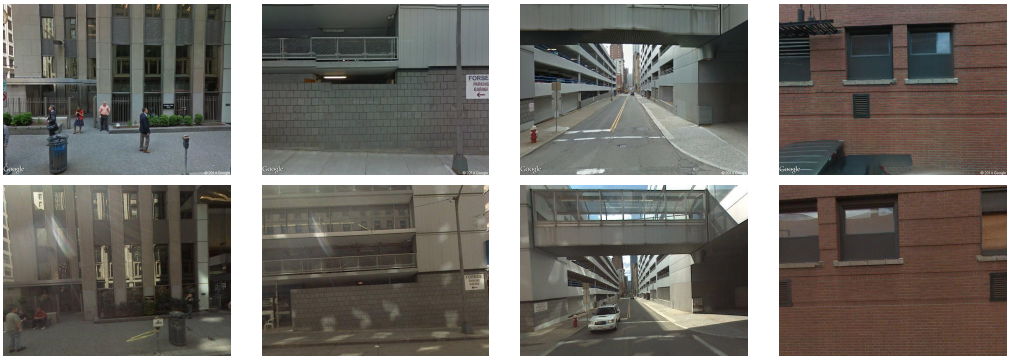
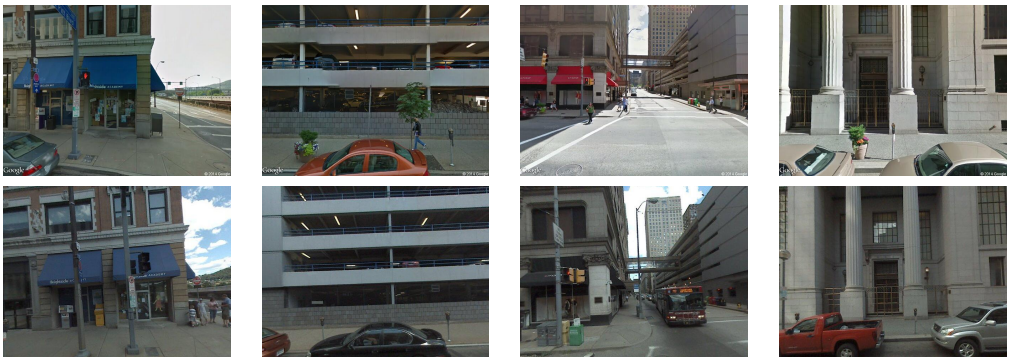
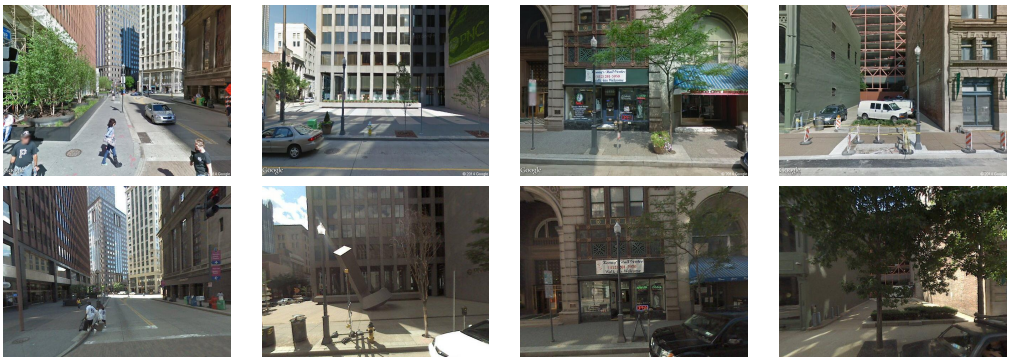


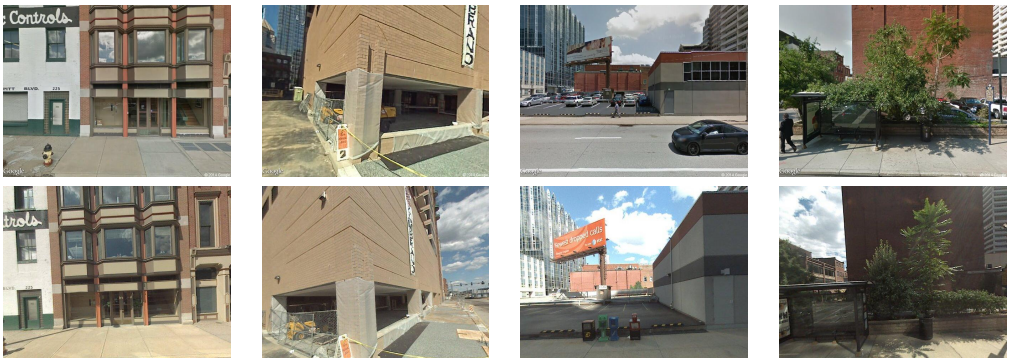
FIGURE 6.1 – Exemple d'architecture d'un système de navigation visuelle autonome.

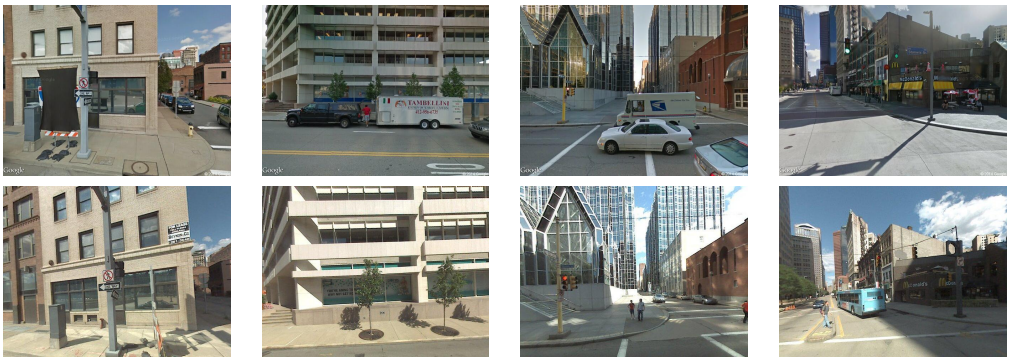
Annexe A

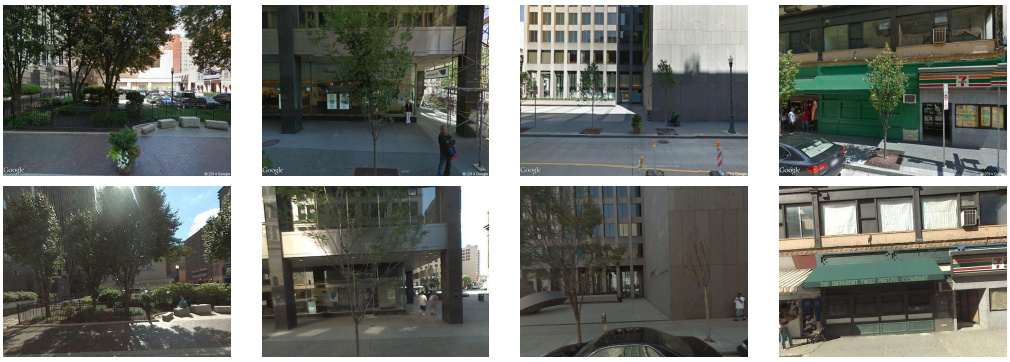


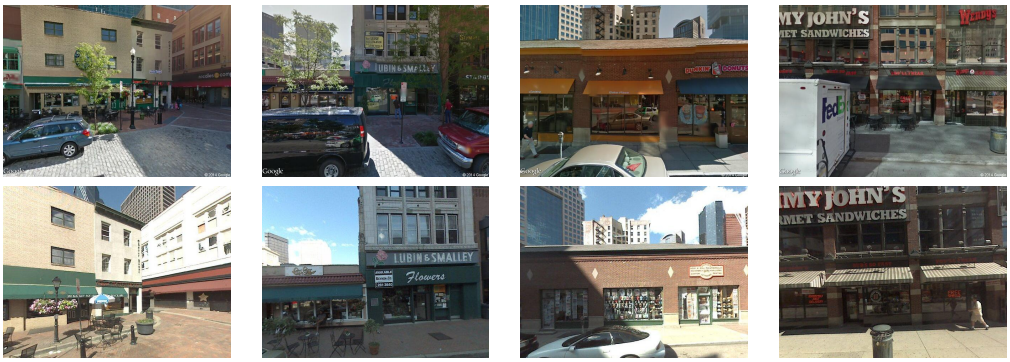


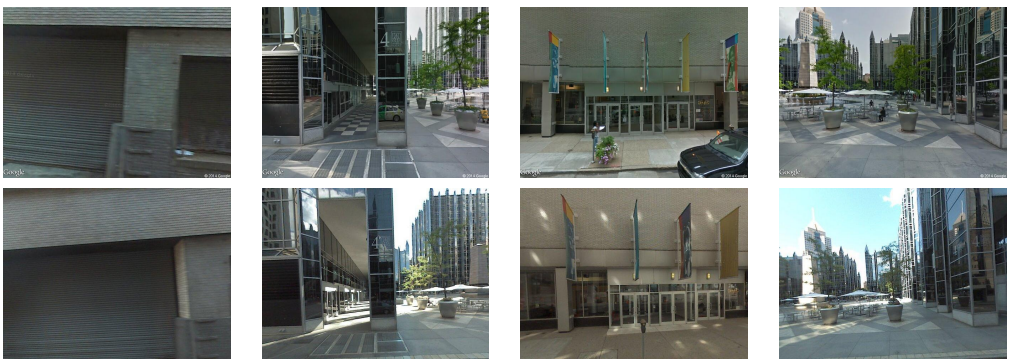


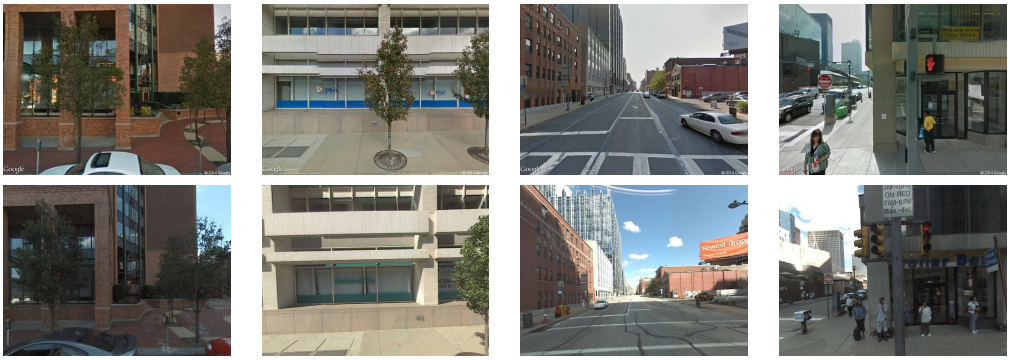

Controls

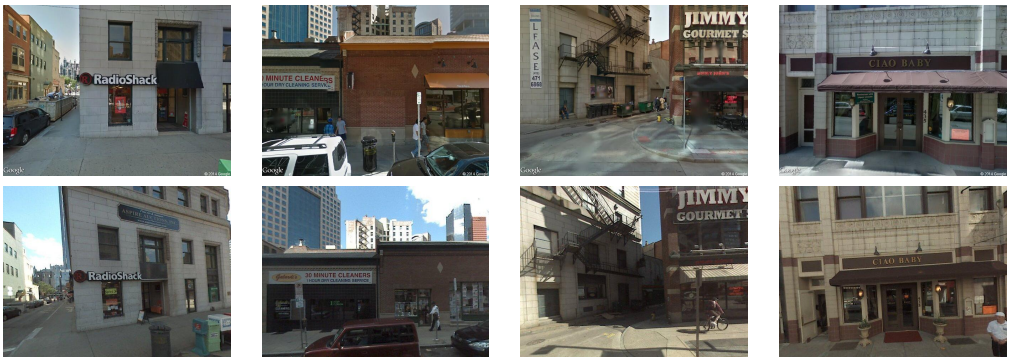


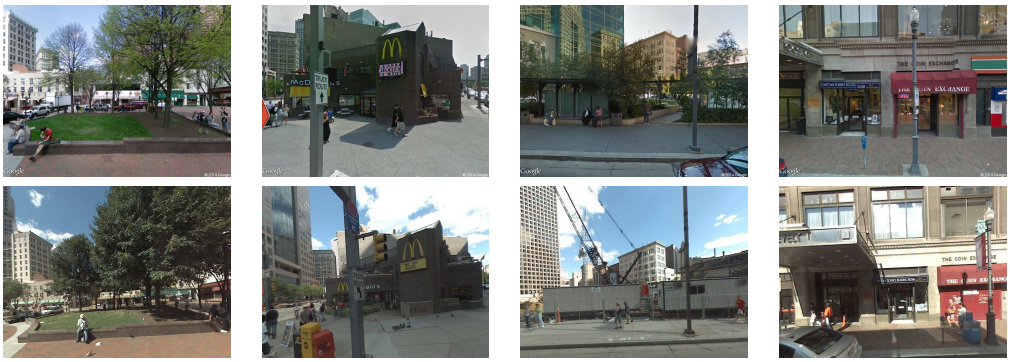



LUBIN & SMALLEY
Flowers

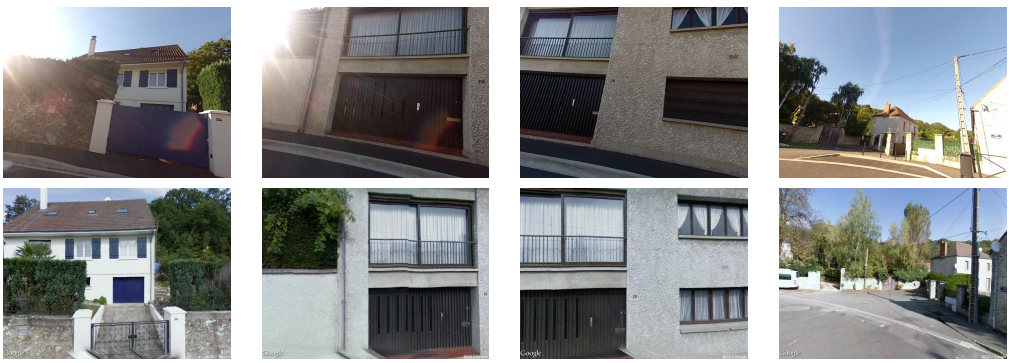





RadioShack
RadioShack
JIMMY
CIAO BABY
CIAO BABY



Annexe B



Résumé

Lors de dette dernière décennie, l'évolution des technologies a permis le développement de drones de taille et de poids réduit aptes à évoluer dans des environnements intérieurs ou urbains. Pour exécuter les missions qui leur sont attribuées, les drones doivent posséder un système de navigation robuste, comprenant, notamment, une fonctionnalité temps réel d'ego-localisation précise dans un repère absolu. Nous proposons de résoudre cette problématique par la mise en correspondance des dernières images acquises avec des images géoréférencées de type *Google Streetview*.

Dans l'hypothèse où il serait possible pour une image requête de retrouver l'image géo-référencée représentant la même scène, nous avons tout d'abord étudié une solution permettant d'affiner la localisation grâce à l'estimation de la pose relative entre ces deux images. Pour retrouver l'image de la base correspondant à une image requête, nous avons ensuite étudié et proposé une méthode hybride exploitant à la fois les informations visuelles et odométriques mettant en œuvre une chaîne de Markov à états cachés. Les performances obtenues, dépendant de la qualité de la mesure de similarité visuelle, nous avons enfin proposé une solution originale basée sur l'apprentissage supervisé de distances permettant de mesurer les similarités entre les images requête et les images géoréférencées proches de la position supposée.

Mots clés : Navigation autonome, localisation visuelle, vision géométrique, recherche d'images, apprentissage de distances, chaîne de Markov à états cachés.

Abstract

In this last decade, technology evolution has enabled the development of small and light UAV able to evolve in indoor and urban environments. In order to execute missions assigned to them, UAV must have a robust navigation system, including a precise ego-localization functionality within an absolute reference. We propose to solve this problem by mapping the latest images acquired with geo-referenced images, i.e. Google Streetview images.

In a first step, assuming that it is possible for a given query image to retrieve the geo-referenced image depicting the same scene, we study a solution, based on relative pose estimation between images, to refine the location. Then, to retrieve geo-referenced images corresponding to acquired images, we studied and proposed an hybrid method exploiting both visual and odometric information by defining an appropriate Hidden Markov Model (HMM), where states are geographical locations. The quality of achieved performances depending of visual similarities, we finally proposed an original solution based on a supervised metric learning solution. The solution measures similarities between the query images and geo-referenced images close to the putative position, thanks to distances learnt during a preliminary step.

Keywords : Autonomous navigation, visual localisation, geometrical vision, image retrieval, metric learning, Hidden Markov Model.