THESE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivrée par *l'Université Toulouse III - Paul Sabatier*

Discipline ou spécialité : *Systèmes Embarqués*

-

Présentée et soutenue par *Dora Luz Almanza Ojeda*
Le 7 Janvier 2011

Titre : *Détection et suivi d'objets mobiles perçus depuis un capteur visuel embarqué*

-

JURY :

Président :
Philippe JOLY, Professeur des Universités, Université Toulouse III

Rapporteurs :
Fawzi NASHASHIBI, Chercheur HDR, Ecole des Mines de Paris
François CABESTAING, Professeur des Universités, Université Lille I

Examinateur
François BREMOND, Directeur de Recherche, INRIA Sophia Antipolis
Michel DEVY, Directeur de Recherche, LAAS-CNRS Toulouse
Ariane HERBULOT, Maître de Conférences, LAAS-CNRS, Université Toulouse III

-

Ecole doctorale : Ecole Doctorale Systèmes (EDSYS)
Unité de recherche : Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS)
Directeurs de thèse: *Michel DEVY, Ariane HERBULOT*

Table des matières

Table des figures

Chapitre 1

Perception des environnements dynamiques

Sommaire

1.1 Introduction

Considérons un robot mobile ou un véhicule autonome qui doit se déplacer dans un environnement extérieur, soit sur route, soit en milieu ouvert naturel ; il doit suivre une trajectoire définie par une courbe au sol, ou par une consigne définie dans les données sensorielles. Ce robot doit embarquer des capacités fonctionnelles et décisionnelles lui permettant de contrôler l'exécution de tous ses mouvements, d'estimer les mouvements réellement effectués, cela afin de rester sur la trajectoire prévue, et de détecter en temps réel que cette trajectoire est libre d'obstacles. Dans ce contexte nous nous sommes focalisée sur des environnements naturels dynamiques comme celui représenté dans la figure 1.1, figure générée en utilisant le simulateur MORSE [49].

Nous nous intéressons dans ce document, aux fonctions de perception embarquées sur un robot ou un véhicule, fonctions qui doivent analyser l'environnement, pour estimer d'une part l'état et les caractéristiques du lieu et des objets qu'il contient, et d'autre part l'état du robot, à savoir sa position, son orientation et sa vitesse. Les objets sont détectés à partir de données sensorielles acquises par des capteurs actifs et passifs ; dans un environnement naturel, le choix des capteurs doit privilégier les techniques les plus robustes et adaptées aux conditions météorologiques, d'illumination... quelques-uns de ces capteurs sont décrits dans le chapitre 2. Nous n'exploitons dans notre travail qu'une seule caméra, mais nos traitements pourraient être perfectionnés grâce à l'utilisation de technologies plus récentes, comme des lecteurs RFID si l'environnement est instrumenté, ou des capteurs plus rapides et de haute précision. Citons le télémètre laser Velodyne popularisé par le DARPA Grand Challenge, ou des nouveaux capteurs

actifs comme le Swiss Ranger ou la caméra RGBD Kinect, ou encore des dispositifs programmables dédiés, parmi d'autres ; la technologie progresse en permanence, et de nouveaux capteurs peuvent être intégrés sur notre véhicule autonome.

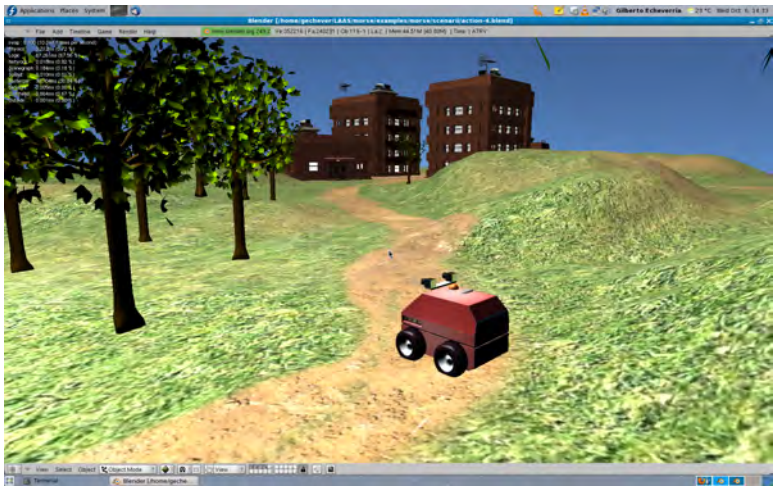


FIGURE 1.1 – Navigation d'un robot tout terrain dans un milieu extérieur naturel dynamique.

Néanmoins, le type de capteurs embarqués sur un robot mobile n'influence que peu l'organisation des traitements dans un système de détection et de suivi d'objets dynamiques. Le fait que les obstacles présents peuvent être mobiles ajoute des contraintes supplémentaires à la détection d'objets. Nous représentons en figure 1.2, les relations existantes entre plusieurs fonctions, regroupées dans cinq modules (dans les boîtes rectangulaires), et les données prises en entrée ou générées en sortie de ces modules (dans les formes elliptiques) :

— *Acquisition-Extraction de primitives :* analyse des données sensorielles courantes Z_t acquises à l'instant t, et extraction de primitives 2D ou 3D selon la nature des capteurs. Cette fonction d'extraction peut être active : elle est alors guidée par les modèles courants et intégrée avec les procédures de suivi et de détection.

— *Suivi-Estimation des Objets Mobiles :* parmi les primitives extraites dans les données courantes Z_t, il s'agit de rechercher le sous-ensemble M_t des éléments qui correspondent ou appartiennent aux objets et primitives déjà perçus lors des acquisitions précédentes, sur des composantes mobiles de l'environnement. Ce module reconnaît si une primitive nouvelle détectée aux instants précédents, est fixe ou mobile. Il génère donc un ensemble de primitives fixes nouvelles qui seront fusionnées à la carte des objets et primitives stationnaires de l'environnement. Parmi ces primitives nouvellement perçues, celles considérées comme mobiles seront intégrées dans M_t, la liste des primitives mobiles de l'environnement.

— *Détection des Objets Mobiles/Appariement des Primitives Fixes :* parmi les primitives extraites dans les données courantes Z_t, ce module recherche le sous-ensemble de primitives qui correspondent ou appartiennent aux objets et primitives déjà perçus lors des acquisitions précédentes, sur des composantes fixes de l'environnement. En échangeant des informations avec le module de *Suivi*, il en déduit une liste de primitives nouvelles qui seront analysées dans les itérations suivantes pour les classer comme Mobile ou Stationnaire.

— *Segmentation-Fusion-Identification pour la localisation simultanée (SLAM) :* à partir des listes de primitives fixes appariées et fixes nouvelles, ce module met à jour la carte des objets fixes. Cette carte sera exploitée pour estimer les déplacements ou pour localiser le robot qui porte les capteurs. Les étapes de Segmentation et de Fusion, puis d'Identification, exploitent des connaissances *a priori* sur la nature des objets fixes potentiellement présents dans l'environnement, nature caractérisée par des descripteurs. Elles visent à regrouper des primitives qui appartiennent à un même objet, et à faire émerger ce concept d'objet. Ceci permet de simplifier la carte et donc, de la rendre plus facile à gérer.

— *Segmentation-Fusion-Identification pour le suivi d'objets mobiles (MOT)* : même traitement que le module précédent, mais pour les composantes mobiles de la scène. L'appellation MOT (pour Mobile Object Tracking) évoque les approches de coopération avec le SLAM (Simultaneous Localization And Mapping) donc les approches du type SLAMMOT qui seront décrites au chapitre 2. Les primitives mobiles sont stockées dans des listes ; le regroupement en objets exploite également des connaissances *a priori*, mais aussi la cohérence des mouvements des primitives appartenant au même Objet.

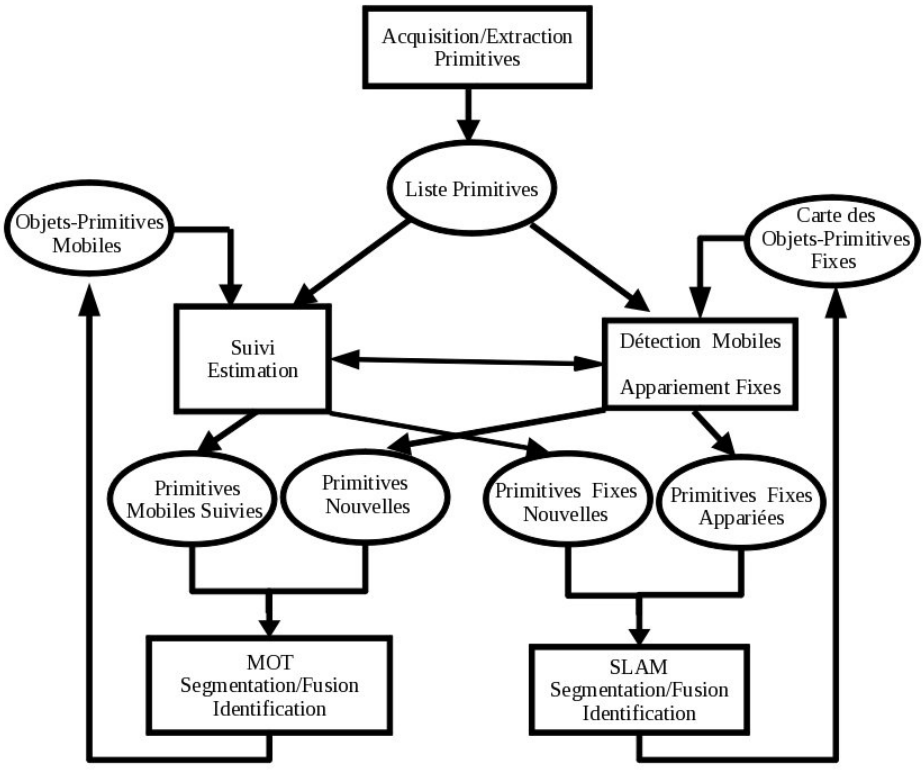


FIGURE 1.2 – Organisation des traitements dans un système de détection d'Obstacles Mobiles depuis des capteurs embarqués sur un robot mobile.

Nous avons mis en place une stratégie qui vise à développer quatre fonctions pour les intégrer dans le système temps réel embarqué sur un robot ou un véhicule robotisé ; ces fonctions s'exécuteront pendant les mouvements du véhicule, et devront utiliser des capteurs qui fournissent une quantité suffisante d'in-

formations avec un faible coût. Nous avons exclu de nos recherches, l'étude de la tâche d'identification des objets car cela nécessite d'exploiter des connaissances *a priori* sur l'environnement et sur les objets mobiles susceptibles de s'y trouver. Le fait de ne faire aucune hypothèse sur l'environnement permet à notre approche de rester générique ; elle pourra ensuite être adaptée à divers types d'environnements. C'est dans ce contexte que nous situerons nos travaux et nous présentons dans ce document une approche pour aider à la navigation depuis un système visuel, en exploitant les capteurs proprioceptifs du véhicule (odométrie, centrale inertielle).

1.2 Description du problème

La détection et le suivi d'objets mobiles dans un environnement dynamique est un sujet essentiel dans le contexte de la navigation autonome. Nous avons traité ce sujet dans le contexte de la navigation d'un robot mobile dans des endroits naturels inconnus *a priori* en utilisant des capteurs embarqués : une caméra, l'odométrie et/ou une centrale inertielle. On étudie le cas où la caméra est en mouvement, et l'environnement observé est aussi dynamique. Il est alors primordial de pouvoir distinguer le mouvement propre (ou ego-motion) de la caméra ou du robot qui la porte, de celui des obstacles.

L'approche proposée doit s'exécuter en ligne, pendant les mouvements du robot, ce qui implique une contrainte forte sur le temps de calcul des algorithmes. Pour atteindre ce but, nous proposons une méthode éparse de détection de points dans le but d'extraire uniquement les points les plus représentatifs dans l'image. En plus, nous proposons une stratégie active pour la sélection de nouveaux points le long de la séquence ; le choix des nouveaux points dépend du comportement des points dynamiques et statiques déjà perçus. Une autre partie de cette thèse concerne la détection, la caractérisation et la validation de groupes de points dynamiques en tant que régions qui correspondent à différents objets mobiles. La représentation de ces objets étiquetés comme dynamiques, exploite plusieurs indices de l'objet afin de rendre plus stable leur suivi tant qu'ils sont présents dans le champ de vision de la caméra ; ainsi, un objet dynamique détecté comme un groupe de points d'intérêt, est ensuite représenté comme une région de l'image où d'autres propriétés (apparence, couleur, texture, histogrammes des gradients parmi d'autres) vont garantir la mise en correspondance de l'objet et la performance du suivi.

1.2.1 Objectifs

Pour répondre au problème de la détection et du suivi d'objets dans des environnements dynamiques, nous nous sommes fixée 4 objectifs principaux :

— Développer et valider une méthode fondée sur la vision mono-caméra permettant de détecter et de suivre des objets mobiles rigides ou non rigides dans un environnement dynamique ouvert.

— Aucune connaissance *a priori* sur l'environnement de navigation ou les types d'objets mobiles ne sera requise.

— Les indices visuels obtenus à partir des images acquises depuis un robot tout terrain pourront être fusionnés avec les mesures acquises par un capteur proprioceptif (odométrie, centrale inertielle) afin de compléter l'information obtenue par la caméra, notamment pour compenser le mouvement propre de la caméra.

— L'algorithme doit atteindre une performance en temps réel de façon à minimiser le temps de latence lorsqu'un objet détecté peut être un obstacle (son mouvement peut l'amener à croiser la trajectoire du robot), et à pouvoir mettre en œuvre une coopération avec d'autres tâches sur le robot (SLAM, asservissement...).

1.2.2 Motivations

Dans ces travaux, nous privilégions l'utilisation d'un système mono-caméra car la tendance actuelle est orientée vers des capteurs bas coût, moins encombrants et surtout faciles à intégrer sur des véhicules destinés au grand public. D'un autre côté, n'exploiter qu'un seul capteur permet d'éviter le problème de synchronisation comme dans le cas de systèmes plus sophistiqués combinant vision et télémétrie laser.

Mais, nous devons assurer l'efficacité de la fonction critique de détection des objets mobiles, potentiellement obstacles pour les prochains mouvements du robot, et cela uniquement à partir de l'information visuelle. Ceci implique un fort compromis dans le développement des algorithmes ; en effet, plusieurs approches fondées sur la vision mono-caméra, offrent de bonnes performances pour la navigation dans un monde statique [29], [90], [24] ; ici, nous cherchons à enrichir la tâche de navigation en détectant de nouveaux points statiques de l'environnent et en garantissant le suivi des objets mobiles présents dans la scène.

En ce qui concerne la détection des objets dynamiques, nous ne favorisons pas de formes spécifiques. Pour représenter un tel objet, nous cherchons à identifier un contour initial de forme quelconque, typiquement la silhouette de cet objet dans la vue courante. Depuis longtemps la théorie des contours actifs a été utilisée dans le but de trouver la silhouette réelle d'un objet, même partiellement observé. Nous allons utiliser les contours actifs de façon similaire à [17] et [58] dans le but d'extraire, puis de suivre un objet le long de la séquence des images, cela sans chercher à détailler sa silhouette. On profite de l'information déjà connue sur les objets, à savoir l'apparence et les déplacements des points d'intérêt détectés sur leur surface, pour caractériser le modèle et rendre plus robuste le suivi des objets, qu'ils soient rigides (typiquement d'autres véhicules) ou non-rigides (typiquement des piétons).

Dans la mesure où notre approche est capable d'obtenir en ligne, des informations sur l'environnement, nous chercherons à intégrer ces données avec celles obtenues par d'autres fonctions exploitées par le système de navigation, fonctions qui traitent de la localisation simultanée ou de l'aide à la conduite.

1.3 Approche développée

Notre travail se place dans le contexte de la navigation visuelle en environnement ouvert et dynamique. Dans ce cadre, nous nous sommes focalisée sur la détection et le suivi d'objets mobiles en utilisant des images acquises par une caméra. Afin d'atteindre cet objectif, nous avons retenu la technique du flot optique épars pour initialiser la phase de détection d'objets mobiles avec des points caractéristiques. Ces points sont extraits en analysant les gradients dans l'image, puis suivis par l'algorithme KLT (Kanade-Lucas-Tracker [68] et Shi et Tomasi [86]) pendant un petit nombre fixe d'images N que nous appelons *temps de pistage*. Une piste (ou tracklet) est faite de N mesures qui seront accumulées pour un même point, à raison d'une mesure par image. Donc le *temps de pistage* est directement lié à la fréquence d'acquisition des images. Ainsi, si nous accumulons les mesures des points pendant 5 images successives sur une séquence acquise à 25 images par seconde, alors notre *temps de pistage* réel sera de 200 milli-

secondes. En plus, dans une configuration purement séquentielle, le *temps de pistage* correspond aussi à la période pour l'extraction de nouveaux points d'intérêt sur les images. Nous avons réglé ce paramètre fondamental pour notre approche, typiquement entre 4 et 6 images successives.

Ensuite, nous analysons le comportement des points à partir des pistes, donc de leurs positions et vitesses fournies par le flot optique pendant le *temps de pistage*. Une analyse spatio-temporelle, basée sur la théorie *a contrario*, permet de regrouper les points issus de composantes statiques de la scène, et les points acquis sur chaque objet mobile suivi sur les N images. Un objet mobile est donc détecté comme un "cluster" de points caractéristiques, cluster dont l'enveloppe définit une région de l'image.

Dans les images suivantes, chaque région détectée dans la phase initiale, est directement suivie en utilisant un Filtre de Kalman pour la prédiction. Dans une première version, les points de chaque "cluster" continuait à être suivi par l'algorithme KLT, mais en exploitant une même prédiction associée au centre de gravité du "cluster". De l'analyse des résultats de cette procédure de suivi pour des objets rigides ou non rigides, nous avons trouvé plus approprié de modéliser un objet mobile comme une région irrégulière, délimitée par un contour actif, initialisé en calculant l'enveloppe du "cluster" initial, puis de caractériser l'objet en exploitant les valeurs d'intensité à l'intérieur de cette région. Un algorithme classique de "snake" a été évalué pour suivre ensuite ces régions [17].

Afin de détecter de nouveaux objets mobiles ou d'élargir les silhouettes des objets déjà détectés jusqu'à leurs limites réelles, une nouvelle sélection des points caractéristiques est réalisée au début de chaque *temps de pistage*. Cette sélection se fait de manière active en exploitant une carte probabiliste de mouvement 2D, plaquée sur l'image. Cette carte donne en chaque zone de l'image, la probabilité d'y détecter un objet mobile. Nous proposons une expression mathématique des contraintes d'initialisation pour instancier et mettre à jour cette probabilité en chaque zone de la carte, en fonction des mouvements apparents des objets déjà détectés, de connaissance *a priori* (par exemple, les objets rentrent dans le champ visuel par les bords de l'image ou par la ligne d'horizon), le but étant de minimiser le nombre de non détections.

La procédure détection-suivi-clustering est répétée avec une période égale au *temps de pistage*. De nouveaux points caractéristiques seront extraits dans les zones où il est le plus probable de trouver un nouveau point mobile, zones indiquées par la carte probabiliste de mouvement alignée sur l'image. A chaque itération, des traitements supplémentaires permettent de limiter le nombre de fausses alarmes ou de non détections :

1. D'abord, le mouvement de la caméra est compensé pour éviter de détecter de faux points mobiles sur le sol à proximité du robot.

2. Ensuite, une procédure de fusion région-"cluster" est appliquée afin d'éviter d'associer plusieurs régions à un même objet mobile.

Tous les concepts proposés dans nos travaux, ont été validés sur des séquences d'images dynamiques réelles acquises d'abord depuis une caméra fixe au mur, ensuite obtenues depuis une caméra embarquée sur un robot qui effectue un déplacement en milieu extérieur ou intérieur.

1.3.1 Contributions de la thèse

Nos travaux ont donc permis de développer une méthode de détection et suivi d'objets mobiles, fondée sur des algorithmes d'analyse d'images mono-caméra. Étant donné le caractère général de notre approche, deux applications peuvent être visées :

— La surveillance d'un environnement intérieur ou extérieur depuis une caméra fixe pour y détecter et suivre des objets dynamiques.

— La détection et le suivi d'objets mobiles par un robot depuis une caméra embarquée lors de sa navigation dans un milieu inconnu a priori.

Nous avons étudié et mis en œuvre la méthode de groupement *a contrario* en allant au delà d'une analyse sur une séquence courte comme cela a été fait à l'origine par les auteurs de cette approche. Notre méthode est exécutée dans un boucle à une fréquence fixe ; à chaque fois les nouveaux objets détectés pourront être fusionnés aux objets précédents. Cette stratégie de détection est mise en œuvre sur des environnements a priori inconnus parce que la méthode ne fait aucune hypothèse sur l'environnement. Ceci nous a permis de réaliser des expériences dans différents contextes de navigation, soit en intérieur ou en extérieur, et en plus, de garantir que plusieurs objets pourront être détectés et suivis simultanément.

Plusieurs contributions présentées dans ce manuscrit ont été publiées dans des conférences internationales :

— Dora Luz Almanza-Ojeda, Michel Devy, Ariane Herbulot. "Visual-Based Detection and Tracking of Dynamic Obstacles from a Mobile robot". In Proceedings of the 7th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2010), Juin 2010, Funchal (Madeira, Portugal), pp. 98-105 [2].

— Dora Luz Almanza-Ojeda, Michel Devy, Ariane Herbulot. "Incremental Detection and Tracking of Moving Objects by Optical Flow and a Contrario Method". In Proceedings of the International Conference in Computer Vision Theory and Application (VISAPP 2010), Mai 2010, Angers (France), pp. 480-483 [1].

— Mario-Alberto Ibarra-Manzano, Dora-Luz Almanza-Ojeda, Michel Devy, Jean-Louis Boizard, and Jean-Yves Fourniols. "Stereo vision algorithm implementation in FPGA using census transform for effective resource optimization". In Proceedings of the 12th EUROMICRO Conference on Digital System Design : Architecture, Methods and Tools (DSD 2009), IEEE Computer Society, August 2009, Patras (Grèce), pp. 799-805 [52].

— Dora Luz Almanza-Ojeda, Michel Devy, Ariane Herbulot. "Active method for mobile object detection from an embedded camera, based on a contrario Clustering". Accepted for publication "Lecture Notes in Electrical Engineering" (LNEE) published by Springer Verlag

— Michel Devy, Dora Luz Almanza-Ojeda, Ariane Herbulot. "Mobile Object Detection from an Embedded camera : how to minimize the latency time ?", IFAC World Congress, Milan (Italie), Août 2011.

1.4 Organisation du manuscrit

Ce document est structuré en cinq chapitres.

Le chapitre 2 propose une étude bibliographique du problème de la détection et du suivi des obstacles dans le contexte de la navigation en robotique mobile. Ce chapitre présentera différents types de capteurs actifs et passifs et quelques solutions adaptées à chacun. Nous étudierons notamment les algorithmes de détection d'obstacles par vision monoculaire et stéréo. Ensuite une étude bibliographique sur différents travaux liés à l'estimation et suivi d'objets sera présentée. Cette étude se focalise sur différents modèles de représentation des objets et sur l'estimation de l'état de ces modèles pour la navigation d'un véhicule intelligent en interaction avec l'environnement. Puis nous étudierons les approches SLAMMOT pour la navigation et le suivi d'objets car notre approche devra à terme coopérer avec un module SLAM. Dans la dernière partie nous présenterons le diagramme global de notre approche.

Le chapitre 3 décrit en détail notre méthode de détection d'obstacles dynamiques dans une séquence d'images. Tout d'abord, ce chapitre introduira la méthode utilisée pour la sélection des points d'intérêt. La notion de *temps de pistage* représente le temps pendant lequel on accumule les positions de ces points dans les images successives. Ensuite nous présenterons la méthode de groupement ou "clustering" ; l'intérêt du *temps de pistage* réside dans le fait que notre méthode pourra mieux détecter les vrais mouvements indépendants liés aux objets dynamiques, sans avoir à connaitre combien de trajectoires différentes il y a dans la scène. Ensuite nous expliquerons comment est construite et mise à jour la carte probabiliste de mouvement, carte alignée sur l'image, utilisée pour sélectionner les points de manière active. Finalement, nous présenterons les règles utilisées pour la fusion des objets dynamiques détectés à des instants différents.

Le chapitre 4 présente le suivi dynamique des objets détectés dans les images précédentes. Nous analyserons tout d'abord les deux modèles exploitables pour suivre les objets : un modèle fondé sur les points d'intérêt, directement issu de la détection, et un deuxième modèle fondé sur les contours actifs. Pour les deux modèles nous décrirons l'initialisation, la méthode de mise en correspondance sur l'image suivante et la mise à jour. Des résultats en utilisant les deux modèles seront présentés. Ensuite, le chapitre présentera en détail la méthode de filtrage choisie et en particulier une analyse de performance illustrée sur le traitement d'une séquence d'images. Enfin, nous présenterons les résultats de la méthode de suivi dans un contexte multi-cibles ; elle permet de suivre en permanence un nombre quelconque d'objets en gérant automatiquement la fusion entre des objets détectés à des temps de pistage différents, mais qui correspondent à la même entité physique.

Le chapitre 5 concerne l'intégration de notre approche dans le système de navigation d'un robot mobile. Après avoir défini le modèle de compensation de mouvement, nous verrons dans ce chapitre qu'il est possible de tester notre approche sur des séquences réelles avec une vitesse de navigation plus importante du robot. Ensuite, nous proposerons une stratégie pour fusionner notre approche de détection-groupement-suivi d'objets mobiles avec une approche SLAM de localisation et construction de cartes simultanées. Nous intégrons dans cette carte, uniquement les points détectés comme statiques par notre méthode, en tant qu'amers de localisation, ce qui permet de diminuer l'incertitude du SLAM. La dernière partie de ce chapitre est dédiée aux expérimentations et à l'évaluation des algorithmes présentés précédemment le long du manuscrit. Nous présenterons tout d'abord l'architecture matérielle du robot *Dala*. Nous décrirons également les solutions logicielles que nous avons développées pour mener les expérimentations. Ensuite, nous décrirons séparément les différents modules du processus de détection d'obstacles. Enfin, pour évaluer l'ensemble, nous présenterons les résultats obtenus sur une séquence

acquise durant l'exécution d'un déplacement à vitesse modérée.

Finalement nous terminerons ce document avec les conclusions et les perspectives pour des futurs travaux.

Chapitre 2

Contexte : développement des véhicules autonomes

Sommaire

2.1 Introduction

Les systèmes de navigation autonome, actuellement, privilégient des capteurs fournissant des informations sur les caractéristiques de l'environnement, comme des systèmes mono ou multi-caméras, les télémètres laser, les radars, des sonars, entre autres modalités sensorielles. Chaque capteur a des avantages et des inconvénients propres ; dans le contexte du développement de robots de service destinés au grand public, la tendance actuelle est de privilégier les méthodes de perception fondées sur des capteurs compacts (donc faciles à intégrer dans la structure mécanique du robot), sans danger pour l'utilisateur et bas coût. Ce sont les capteurs visuels qui satisfont le mieux ces contraintes, mais la vision n'est pas

toujours la solution la plus performante. Analysons par exemple l'environnement naturel que parcourt notre robot *Mana* dans la figure 2.1. Dans ce cadre, la probabilité de rencontrer des objets dynamiques, est faible. Donc il n'y aurait pas besoin d'utiliser des capteurs qui travaillent à haute fréquence pour les identifier car ce n'est que sporadiquement que seront trouvés des objets dynamiques ; en plus on n'attend pas que ces objets soient très rapides. Par contre, la sélection du meilleur capteur pour ce scénario est très liée à la contrainte de sol non plan et aux possibles glissements et vibrations du véhicule à cause du sol irrégulier et des trous.

FIGURE 2.1 – *Mana*, véhicule robotique tout terrain.

Par ailleurs, plusieurs situations ont été analysées dans la littérature, ce qui a donné lieu à des méthodes différentes pour détecter des objets :

— Le capteur peut être fixe, et la scène observée être en mouvement. C'est le contexte simplifié des applications de vidéo-surveillance : le mouvement observé est le fruit uniquement de l'objet que l'on détecte, car le dispositif de vision n'a aucun mouvement propre, ou un mouvement simple que l'on peut facilement compenser dans le cas des caméras Pan-Tilt-Zoom (PTZ).

— Le capteur peut être en mouvement, et la scène observée être statique. Cette hypothèse du monde statique (exploration planétaire par exemple) est aussi plus simple, car ce que l'on observe respecte une contrainte de conservation spatiale et temporelle. Cela permet de construire des représentations simples comme des Modèles Numériques de Terrain.

— Le cas le plus général de la détection d'obstacles est celui où la caméra est en mouvement, et l'environnement observé est aussi dynamique. Il est alors primordial de pouvoir distinguer le mouvement propre du dispositif de vision ou ego-motion, de celui des obstacles. De cette manière des méthodes issues des deux cas précédents (caméras fixes ou scène fixe) peuvent être prises en compte pour résoudre le problème complexe.

Dans ce chapitre nous allons présenter des méthodes ou algorithmes de détection et suivi des objets, méthodes qui exploitent les données acquises depuis un ou plusieurs capteurs embarqués sur un véhicule ou un robot. D'abord, la section 2.2 présente les solutions les plus utilisées, fondées sur les capteurs actifs. Puis la section 2.3 décrit quelques travaux dédiés à la détection d'obstacles par des algorithmes de vision.

A la suite de cette phase de détection, si l'objet est mobile, il faut analyser ses déplacements à partir de son suivi dans la séquence d'images acquises depuis la même caméra. Nous évoquerons dans la sec-

tion 2.4 les techniques pour le suivi d'objets, qui exploitent leurs propriétés d'apparence extraites et mises à jour d'une image à la suivante ; ces méthodes génèrent et exploitent une estimation du mouvement de l'objet suivi. Nous présenterons à la fin de cette section les méthodes bayésiennes exploitées pour réaliser ce suivi d'une part, et l'estimation du mouvement des objets suivis d'autre part. Ensuite, nous décrirons la technique du SLAMMOT proposée pour distinguer les objets stationnaires des objets dynamiques et pour estimer la position et la vitesse de ces derniers. Finalement, nous présenterons avec plus de détails dans la section 2.6, l'approche que nous proposons, approche appliquée pendant la navigation d'un robot pour la détection, la segmentation et le suivi d'objets.

2.2 Détection par des capteurs actifs

Un capteur est dit "actif" s'il produit une énergie envoyée sur la scène ; les obstacles sont détectés par l'énergie réfléchie par leur surface. Un capteur actif possède donc à la fois un émetteur et un récepteur. Nous décrivons rapidement les méthodes fondées sur la télémétrie laser et le radar. Ces deux capteurs actifs sont les plus utilisés sur des véhicules ou sur des robots d'extérieur, parce qu'ils sont robustes aux variations des conditions atmosphériques (pluie, neige) ou à des conditions dégradées d'illumination. Les méthodes visuelles connaissent beaucoup de défaillances dans ces situations. Sur les robots de service évoluant en milieu intérieur, les capteurs ultrasonores ou infrarouges sont également très souvent exploités pour détecter des obstacles proches pendant les mouvements ; ils sont intégrés sous la forme de ceintures de capteurs disposés tout autour du robot. Depuis quelques années, des proximètres ultra-sons sont également intégrés dans les pare-chocs des véhicules grand public, en tant que capteur de recul. Nous ne détaillerons pas ces méthodes, car généralement, les informations qu'ils fournissent peuvent être peu précises et très pauvres.

Cette section décrit rapidement les méthodes de détection d'objets qui utilisent le télémètre laser et le radar. Il décrit d'abord de façon séparée les stratégies utilisées pour chaque capteur et ensuite des approches qui utilisent les deux capteurs simultanément seront présentées. Une brève conclusion sera présentée à la fin de la section.

2.2.1 Détection par télémètre Laser

La télémétrie laser combine un système électronique pour la mesure ponctuelle de distance à partir d'un faisceau laser réfléchi par l'obstacle, et un mécanisme de déviation du faisceau émis et de balayage pour acquérir des coupes ou des images de distance dans une région d'intérêt quelconque. La lumière réfléchie par un obstacle quelconque se trouvant dans l'axe du tir revient vers le capteur et est perçue par un récepteur ; la distance capteur-obstacle est obtenue de manière classique par le temps de vol d'une impulsion laser, ou par le déphasage d'un signal modulé en amplitude ou en fréquence. Tous les impacts laser sont dans un plan si le balayage est seulement en azimut, ou sont dans l'espace 3D si le balayage est en site et azimut.

L'emploi de la télémétrie laser est très courant sur des véhicules haut de gamme, ou sur des robots de service, car la mesure est rapide, la portée peut être importante (plusieurs centaines de mètres selon le type de télémètres utilisé, et les taux d'échecs (faux négatifs ou faux positifs) très faibles : en figure 2.2, nous présentons les télémètres les plus répandus. Citons par exemple [30] où les auteurs utilisent les mesures acquises par un télémètre laser IBEO pour détecter les objets dans la scène. Ce capteur retourne une image de profondeur haute résolution. Dans [54] le module de détection d'obstacles emploie un filtre d'association probabiliste et des réseaux bayésiens pour estimer la probabilité de détection à l'ins-

tant suivant. Lors du DARPA Grand Challenge, de nombreux véhicules étaient équipés du capteur laser Velodyne (figure 2.2c) qui permet d'acquérir des images 3D panoramiques de très grande résolution à 30Hz. Signalons toutefois les difficultés pour traiter de telles images, en particulier du fait des problèmes de calibrage.

(a) (b) (c)

FIGURE 2.2 – a) Télémètre laser de haute résolution de la compagnie Ibeo. b) Capteur LMS200 de Sick Inc. en Allemagne ; ce capteur est fréquemment utilisé sur les robots pour la navigation à vitesse modérée. Il est intégré sur notre démonstrateur *Dala* (figure 5.10). c) Capteur Velodyne monté sur notre démonstrateur *Mana*.

Cependant, il arrive fréquemment que les systèmes robotiques soient équipés avec des télémètres laser de plus basse résolution, comme dans les travaux de Mendes [76]. Pour ces télémètres, l'information est parfois insuffisante et même de légères vibrations peuvent perturber les mesures. Ces capteurs laser sont donc excellents pour la détection des obstacles, mais leur faible résolution rend difficile l'identification des objets détectés. Pour pallier ce défaut, de nombreux auteurs ont étudié la fusion des données laser avec l'information visuelle. Par exemple, pour des applications dans la sécurité routière, Labayrade et al. [61] utilisent simultanément deux capteurs 3D qui ont des caractéristiques différentes, la télémétrie laser et la stéréovision.

On trouve surtout dans la littérature, de très nombreux travaux, qui pour différents contextes (navigation autonome d'un robot en extérieur et en intérieur, détection de piétons depuis un véhicule intelligent) proposent de fusionner les données d'un télémètre laser 2D avec un système visuel monoculaire [18], [55] et [35] ; généralement les données laser sont analysées pour la détection et l'estimation de la profondeur, tandis que les images donnent une meilleure résolution et caractérisation de l'objet, ce qui permet son suivi et son identification.

2.2.2 Détection par radar

Le radar (RAdio Detection And Ranging) consiste à mesurer les ondes radio (ondes millimétriques, 24GHz ou 77GHz) réfléchies par les objets métalliques, en exploitant la théorie de Doppler pour détecter leurs déplacements en fréquence. Le radar peut être exploité pour détecter des objets avec de très grandes portées ; il peut être utilisé en statique (par exemple, détection d'intrusion dans un environnement sensible) ou mobile (par exemple, détection d'obstacles depuis un véhicule). Cependant, comme pour les signaux lumineux émis par un télémètre laser, la portée de détection maximale d'une cible réfléchissant

des signaux hyperfréquences, dépend de la texture de la cible et de la spécularité, ou angle entre rayon incident et normale à la surface. Concernant les applications Robotique, le radar est resté longtemps peu exploité à cause de son coût, et surtout, de la difficulté pour acquérir un capteur, car cette technologie était principalement réservée aux applications de défense ; cette situation a changé depuis l'introduction du radar dans les véhicules intelligents (en particulier radars vendus par TRW). On voit se multiplier les travaux sur le radar dans la communauté robotique (citons le projet IMPALA en France à Clermont-Ferrand, les travaux de M.D. Adams à Singapour, parmi d'autres).

La résolution angulaire d'un capteur radar est d'une vingtaine de degrés, mais le faisceau du radar, lors de sa propagation, peut détecter plusieurs cibles en même temps dans une région d'intérêt, car tous les échos successifs sont détectés. Cette caractéristique est exploitée dans [67] pour la détection de piétons dans un milieu urbain. Le système présenté dans ce travail est constitué de deux capteurs radar de courte portée situés à l'avant d'un véhicule expérimental de Daimler, avec une portée de 30m environ. La difficulté du système est d'identifier les échos renvoyés par des piétons ou par d'autres objets. Pour ce faire, les auteurs proposent une fusion de données entre le radar et des capteurs thermiques ; ces derniers servent à détecter la radiation thermique émise par les humains. La fusion des données utilise une approche probabiliste, pour construire une grille d'occupation centrée sur la position du robot.

Bien que le radar ne soit pas affecté par la pluie ou la neige, il est très perturbé par des sources d'interférence magnétique. De plus, la facilité avec laquelle un radar détecte un objet à une distance donnée, est directement liée à sa surface équivalente radar (SER). Un véhicule a une SER supérieure à celle d'un piéton ($2m^2$) [63]. La plupart des petits obstacles ont une SER encore plus faible ce qui les rend indétectables. Cet inconvénient rend indispensable l'association d'autres capteurs avec le radar. Le ladar (association laser et radar) a été proposé pendant les années 70 et 80 [73] grâce à sa meilleure vitesse d'opération par rapport aux capteurs de vision. Cependant dans les années 90, l'évolution des systèmes informatiques a rendu possible, l'utilisation des algorithmes de vision en temps réel : à partir de ce moment-là, il a été possible d'intégrer système radar ou ladar avec des systèmes visuels mono ou multi-caméras pour détecter les objets pendant la navigation d'un véhicule intelligent [56]. Citons un autre système de fusion multi-capteurs, le système FADE, développé à l'école de Mines de Paris [57] exploité pour la détection de véhicules sur route depuis des capteurs embarqués sur un véhicule expérimental. Les capteurs utilisés sont une caméra (couleur ou N/B), un radar et des capteurs proprioceptifs pour la géolocalisation du véhicule (vitesse, angle volant). La détection des autres véhicules depuis les images en vision monoculaire est faite en fusionnant les résultats de plusieurs algorithmes qui détectent les ombres portées, les phares, les lignes horizontales et verticales et les symétries locales. Les résultats obtenus par ce système montrent que les véhicules présents sont correctement détectés en milieu urbain, avec un minimum de fausses détections. Cependant ce système est encore loin d'être généralisé pour être appliqué dans des environnements plus complexes comme sur terrain naturel avec végétation, où les objets sont non structurés, donc plus difficiles à caractériser.

2.2.3 Conclusion pour les capteurs actifs

Les technologies laser et radar fournissent des données sensorielles 3D en temps réel, à haute fréquence ; l'exploitation de telles données rend plus robuste la tâche de détection et de localisation d'objets par un système robotique mobile ou un véhicule intelligent, même dans des environnements complexes comme dans des scènes urbaines ou routières. Par contre, s'il y a plusieurs systèmes autonomes qui, en même temps, utilisent ces capteurs actifs, l'interférence entre eux sera inévitable. Soulignons aussi qu'un système radar a un coût élevé, ce qui doit être considéré avant sa mise en œuvre dans un projet.

D'autre part un télémètre laser plan est fortement perturbé dans ses mesures, par la pente du sol, et

par les variations d'attitude du véhicule sur lequel il est monté, en particulier l'angle de roulis (l'angle autour de l'axe Y dans le système de référence du véhicule qui porte le capteur, voir section 5.2.3) qui change dans les phases de freinage ou d'accélération : dans ces situations le plan laser n'est plus parallèle à la route. Ces problèmes ont justifié l'introduction de télémètres multi-plans (citons IBEO ALASKA), encore plus onéreux.

L. Matthies et. al. [74] indiquent que le choix entre radar, télémétrie laser ou vision pour détecter des obstacles dans une scène depuis un véhicule, dépend de la fréquence d'apparition des objets dans l'environnement de navigation et de la vitesse du véhicule qui porte les capteurs. Si la fréquence est faible et les vitesses aussi, il est plus avantageux d'exploiter la vision. Par exemple, les auteurs précisent que l'utilisation d'un système ladar pour un robot d'exploration planétaire, n'est pas nécessaire vu la très faible fréquence d'apparition d'obstacles et vu la faible vitesse du véhicule.

2.3 Détection d'objets par capteurs passifs basées sur la vision

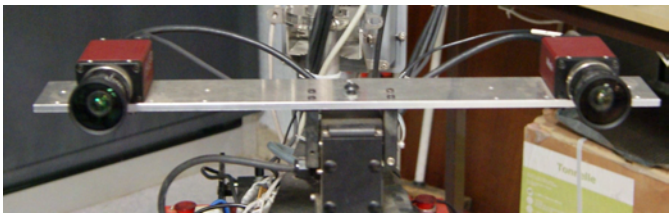
Un capteur passif ne fait que recevoir l'énergie naturellement réfléchie par les objets présents dans la scène. Plusieurs capteurs passifs sont utilisés pour la robotique ou pour la surveillance : les capteurs inertiels (détection de vibrations, de mouvement), les capteurs haptiques (contact), mais les plus connus sont les capteurs visuels, donc des caméras CCD ou CMOS [87]. En tant que capteur passif, une caméra mesure l'intensité lumineuse émise par l'environnement sous la forme d'une image digitalisée, ou d'un tableau de pixels.

Un capteur visuel devient actif dès lors qu'on lui associe un projecteur ou illuminateur qui envoie sur la scène un motif lumineux (mouchetis aléatoire, matrice de points laser, profil laser). Par exemple l'image d'une nappe laser projetée au sol permet de détecter les trous, les obstacles ou une marche. Nous pouvons ranger dans cette classe, les caméras 3D à temps de vol optique, aussi appelées Photonic Mixer Devices (PMD) (Swiss Ranger, Canesta...). Ces capteurs visuels actifs sont exploités pour la détection d'obstacles proches, mais pour l'instant, à notre connaissance, ils sont peu exploités sur véhicules. C'est la raison pour laquelle, ils restent très chers (plus de 5000 € pour un Swiss Ranger), mais le tout nouveau Kinect de Microsoft (150 €), pour l'instant exploité pour les jeux interactifs, pourrait changer la donne.

Nous ne présentons ci-dessous que la vision passive, avec les deux configurations principales utilisées pour acquérir les images : les capteurs stéréo et monoculaire, ainsi que les différentes manières d'analyser ces images. Nous commencerons par rappeler les concepts fondamentaux de la stéréovision avant de décrire les principales méthodes proposées pour détecter des obstacles mobiles à partir de données sensorielles acquises par stéréovision. Ensuite, nous décrirons l'analyse spatio-temporelle réalisée sur les images monoculaires dans le but de décrire son contenu. Finalement, une brève conclusion à propos des capteurs visuels passifs sera présentée à la fin de cette section.

2.3.1 La stéréo-vision

Les systèmes de stéréovision permettent de calculer par triangulation, la profondeur des objets dans l'espace 3D, grâce à leurs projections dans les plans des deux caméras. La différence des positions dans les deux images, de la projection d'un même point de l'espace 3D, s'appelle la disparité ; la recherche des points correspondants entre ces deux images, permet donc de produire une image de disparité. En supposant que les caméras ont des axes optiques parallèles, comme dans le capteur Bumblebee vendu par la compagnie Point Grey (figure 2.3b), alors la disparité est inversement proportionnelle à la profondeur. Ainsi la disparité est nulle pour un point à l'infini, par exemple un point sur la ligne d'horizon.



(a)

(b)

FIGURE 2.3 – a) Banc stéréo embarqué sur notre démonstrateur *Dala* (figure 5.10) b) Système de stéréovision Bumblebee2 fabriqué par Point Grey Research©.

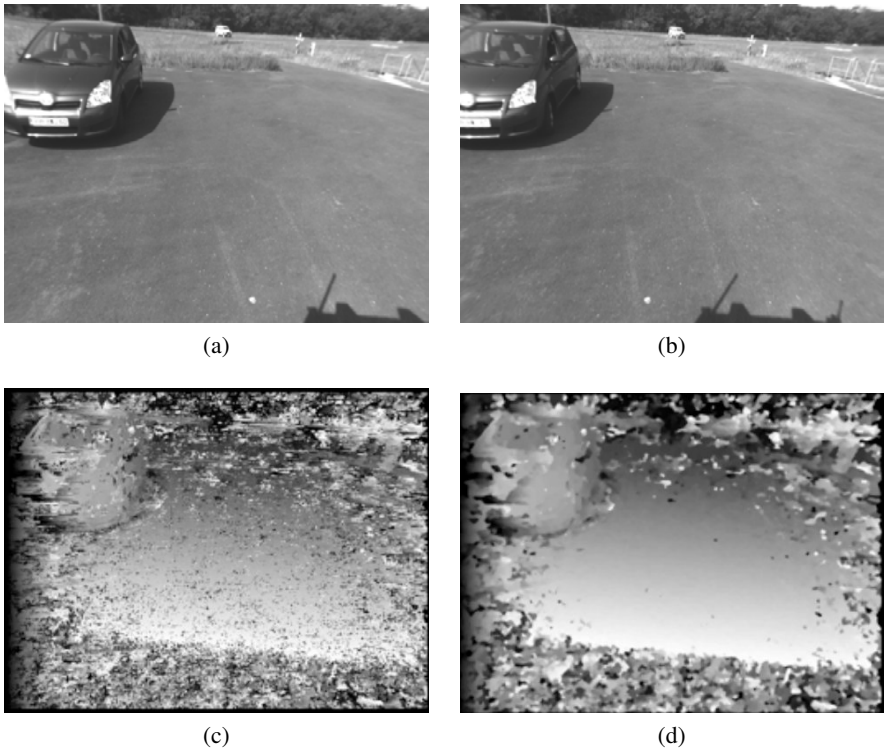
La figure 2.3a illustre un montage typique de deux caméras en configuration stéréo, configuration appelée aussi banc stéréo. Si le banc stéréo est calibré, les coordonnées du point 3D correspondant à un pixel apparié de l'image de disparité, peuvent être calculées. On sait que la précision du capteur stéréo est proportionnelle à la profondeur au carré ; de ce fait, la portée utile d'un tel capteur est limitée, typiquement quelques mètres sur des focales courtes (5m. par exemple pour la détection d'obstacles proches en site urbain), quelques dizaines de m. sur des focales longues (40 ou 50m. max pour la détection d'obstacles lointains sur route).

Les approches pour construire l'image de disparité depuis une paire de caméras sont classifiées en méthodes éparses et denses. Dans les méthodes éparses les images droite et gauche sont d'abord prétraitées pour extraire des indices visuels discriminants (points d'intérêt ou segments). Ce sont ces indices qui sont appariés et reconstruits. Dans le cas des méthodes denses, les pixels sont tous appariés en exploitant une mesure de ressemblance (corrélation par SAD, SSD, ZNCC, distance de Hamming entre code CENSUS). Ces méthodes denses peuvent être optimisées en adoptant une approche pyramidale : les disparités sont d'abord recherchées à faible résolution (images 80×60) avant d'être raffinées en descendant la pyramide jusqu'à l'image initiale (640×480 par exemple).

Le problème classique de la vision, est le temps de traitement des images, notamment pour la détection d'obstacles qui requiert une grande réactivité. Plusieurs solutions ont donc été proposées pour diminuer le temps de calcul de l'image de disparité, en particulier en employant les systèmes re-configurables principalement les FPGA (Field Programming Gate Array). L'avantage de l'implémentation d'un algorithme sur FPGA, réside en sa capacité d'exécuter plusieurs opérations en parallèle, ce qui accélère considérablement le calcul même pour un processus dense. Des systèmes stéréo sur FPGA commercialisés produisent des images de disparité à 30Hz à la résolution VGA ; ils ont des performances très convenables pour traiter des applications en robotique ou pour les transports [66].

Un algorithme de stéréovision basé sur la transformée CENSUS et la mise en correspondance par distance de Hamming, a été développé dans notre équipe sur une architecture FPGA par Ibarra et al. [52]. L'architecture proposée permet de générer une image de disparité dense à 100Hz pour la résolution 640x480 pixels. La figure 2.4c montre l'image de disparité calculée par l'architecture FPGA depuis les images stéréo 2.4a et 2.4b ; en figure 2.4d, l'image de disparité est lissée par un filtre médian ; les zones bruitées sur les bords de ces images sont des artefacts créés par l'algorithme. Les niveaux de gris dans l'image de disparité représentent le niveau de profondeur dans la scène ; les pixels gris les plus clairs

désignent les objets les plus proches des caméras ; les plus éloignés sont en noir. La voiture garée dans la scène est notablement identifiable par la disparité. Grâce à cette carte il est possible de connaitre la profondeur des points, ce qui peut permettre de segmenter les objets par leur profondeur. La navigation d'un robot mobile ou d'un véhicule autonome basée sur la vision stéréo, peut donc être effectuée en temps réel, même si les conditions de l'environnement ne sont pas très favorables [52].



(a) (b)

(c) (d)

FIGURE 2.4 – En haut les images gauche et droite acquises par un banc stéréo pendant la navigation en extérieur sur route (image des bases de donnée ARCOS). En bas images de disparité, avant et après une étape de filtrage.

Le problème de l'auto-calibration. Le besoin de calibrer un banc stéréo (comme celui montré en figure 2.3a) périodiquement après son montage sur un robot ou un véhicule, est considéré comme un des principaux problèmes liés à la stéréovision ; en effet, la performance d'un banc stéréo (nombre de pixels appariés, nombre de faux appariements, précision de la reconstruction 3D) dépend totalement du calibrage. L'auto-calibrage, c'est-à-dire l'estimation des paramètres du capteur sans exploiter de mires, est toujours un sujet de recherche très étudié, depuis l'article fondateur de O.Faugeras en 1992 [32]. C'est un problème difficile, surtout quand il faut le résoudre en temps réel, donc sans faire appel à des méthodes d'optimisation trop lourdes.

Vu cette limitation, au LAAS il a été proposé dans les travaux de J.Sola [90], une approche limitée de correction du calibrage en considérant connus les paramètres intrinsèques des caméras. L'auteur propose de ré estimer la rotation entre les deux caméras dans la boucle de localisation, sur la base d'amers visuels appris soit au préalable, soit en simultané (cas du SLAM). Cependant, aujourd'hui il est possible de se passer de cette étape, car sur le marché, il existe des bancs stéréo pré calibrés, comme le BumbleBee2

de Point Grey Research©(voir photo en figure 2.3b). Par ailleurs ce système stéréo produit une image de disparité à la résolution VGA à plus de 48Hz ; même si ces images de disparité sont assez imprécises et contiennent quelques artefacts, les grands problèmes de la stéréovision sont bien résolus par ce système. Le bémol identifié pour ce banc stéréo est qu'il continue à être cher, que sa configuration est rigide (base fixe) et surtout que l'utilisateur dépend directement de techniques préprogrammées pour l'obtention de la carte de disparité.

La détection d'objets par stéréo-vision. Citons les nombreux travaux du JPL en Californie sur la détection d'obstacles mobiles depuis un capteur stéréo embarqué sur un robot. Dans certains cas, ils estiment le mouvement propre du robot pour distinguer le mouvement apparent des composantes statiques de la scène de celui des composantes mobiles. Cette estimation est ainsi exploitée pour corriger le flot optique en "retirant" le mouvement du système, comme cela est proposé dans [93]. Dans ce travail, Talukder et al. utilisent des champs denses de disparité stéréo et ils suppriment les petites régions non connectées dans le champ du flot estimé.

D'autre part, si on considère que le banc stéréo est calibré et que sa position sur le robot est connue, alors la détection d'obstacles peut être réalisée par une rectification homographique [101]. La rectification homographique transforme une des images de la paire stéréo (par exemple, la droite), de manière à ce que le plan de la route soit identique entre image gauche et image droite rectifiée. Une simple différence entre ces deux images permet alors de discriminer les pixels correspondants à des points 3D qui ne sont pas portés par ce plan. Après que ces pixels associés à des points 3D qui ne sont pas sur la route ont été mis en évidence, les obstacles sont segmentés, puis caractérisés par une analyse de connexité et de similarité. La rectification homographique est appliquée a priori à l'aide d'une calibration préalable du système stéréo et du plan de la route. Cette méthode suppose que la route est plane, ce qui en pratique, la rend inexploitable en milieu naturel.

Par ailleurs, Lefebvre, Ambellouis et Cabestaing [64] proposent une méthode générique de mise en correspondance qui exploite des caractéristiques extraites d'un ensemble de courbes de similarité calculées sur des fenêtres 1D. Cette méthode a été appliquée dans le cadre d'une application de détection d'obstacles à l'avant d'un véhicule routier. Les auteurs proposent un procédé assez simple pour segmenter une carte de disparité : ils montrent que la détection des obstacles est plus précise qu'avec des cartes des disparités calculées sur des voisinages 2D.

Signalons aussi les approches comparées par V. Lemonde [66] au LAAS-CNRS pour la détection d'obstacles proches (à moins de 5m. par exemple), à partir de la stéréovision : on peut faire sans risque dans ce cas, une hypothèse de route plane.

◇ D'abord l'auteur a exploité le concept de v-disparité, qui permet de détecter les obstacles directement à partir de l'image de disparité. Depuis les travaux de R. Labayrade [61], il est connu que cette approche est très sensible aux erreurs sur l'attitude du capteur vis-à-vis du plan de la route.

◇ Puis V. Lemonde a proposé de détecter les obstacles à partir de l'image de points 3D reconstruites à partir de l'image de disparité, simplement par seuillage sur l'élévation de ces points au-dessus du plan de la route. Les points étiquetés *Obstacle* sont ensuite regroupés en fonction d'un critère de connexité. Finalement, la création d'enveloppes permet de modéliser ces ensembles de points par des parallélépipèdes rectangles. Ces boites englobantes permettent d'associer les obstacles détectés d'une image à la suivante, et d'estimer grossièrement leurs dynamiques s'il s'agit d'obstacle mobile.

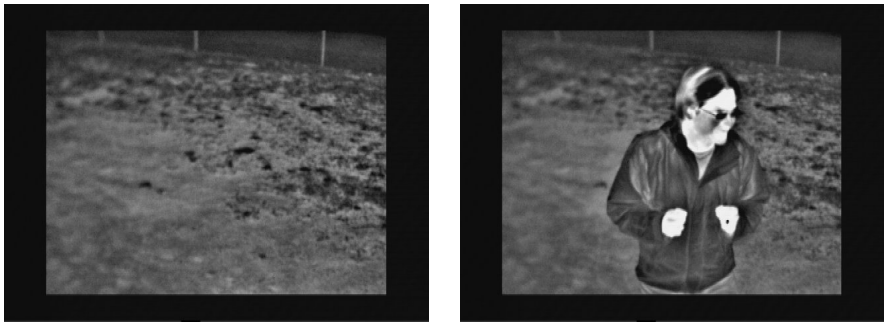
◇ Dans le cas général d'obstacles détectés sur une route, l'auteur propose une approche très inspirée de la v-disparité, sauf que les cumuls se font dans le domaine 3D, donc successivement sur les coordonnées X,Y et Z des points de l'image 3D, ce qui rend plus facile le regroupement des points Obstacle en objets disjoints, et ce qui permet également de traiter de manière plus robuste, le cas du dévers de la route ou du roulis non nul du véhicule.

Ces méthodes supposent que la route est plane, ce qui en pratique, les rendent inexploitables pour la navigation sur terrain accidenté en milieu naturel ; dans ce cas, la détection d'obstacles se mue en la construction d'une carte de traversabilité du terrain, à partir d'une carte d'élévation (ou Modèle Numérique du Terrain) [62].

2.3.2 Capteur de vision Infrarouge (IR)

Malgré leur faible résolution et le mauvais contraste des images qu'elles fournissent, les caméras infrarouges sont exploitées pour la navigation de véhicules dans des conditions de visibilité difficiles, soit la nuit, soit en cas de mauvaises conditions météo par exemple avec du brouillard ou de la neige. On trouve dans la littérature de nombreux travaux sur la détection d'objets depuis des caméras IR de nuit [73] ou avec de la fumée [36].

La bande de fréquence de l'infrarouge est divisée en 3 : bande I (1-1,7 mm), bande II (3-5 mm) et bande III (8-12 mm). Les caméras IR correspondantes ont différentes capacités et propriétés. Par exemple l'eau absorbe plus ou moins les ondes infrarouges selon la fréquence. En plus, les images infrarouges ne sont pas très texturées, ce qui empêche l'utilisation des techniques traditionnelles pour la détection de texture ou d'indices visuels. Deux images acquises par une caméra infrarouge [1] sont montrées dans la figure 2.5.



(a) (b)

FIGURE 2.5 – Exemple des images acquises par une caméra Infrarouge Raytheon Thermal-eye 2000B.

En ce qui concerne la navigation d'un véhicule intelligent dans un milieu extérieur inconnu, citons les travaux de Talukder et al. [93] réalisés avec un véhicule expérimental équipé avec une paire de caméras IR qui opèrent dans la bande 3-5 mm. Les algorithmes de stéréovision et du flot optique dense exécutés à partir d'images infrarouges, sont utilisés pour la détection d'objets dynamiques dans l'environnement. Plus récemment, Alonzo Kelly et al. [56] et Larry Matthies et al. [74] ont présenté une synthèse des résultats obtenus en matière de navigation soit sur des terrains naturels couverts de végétations, soit pour

1. Ces images font partie de "The Marulan Dataset" [92] ; elles ont été téléchargées depuis le site web [46]

l'exploration planétaire. Ils concluent que l'exploitation de capteurs infrarouges est essentielle pour l'exploration de terrain inconnu, notamment pour identifier la nature des obstacles, autre véhicule, buisson ou rocher par exemple.

D'autres applications plus spécifiques de l'imagerie infrarouge ont été présentées en 2008 par L. Gond et al. dans [36] pour la surveillance de foules dans un lieu public (gare du métro). Les auteurs utilisent un capteur infrarouge non refroidi dans la bande 8-12 mm, pour reconnaître des situations d'incendies ou d'attaques chimiques. Ils ont analysé l'impact des conditions d'illumination sur la qualité des images acquises par ces capteurs infrarouges : ils précisent que la vision dans cette bande infrarouge est sensible à certains facteurs comme la lumière du soleil sur les objets, la pluie ou la neige. Dans ce contexte de vidéo surveillance, la détection de personnes est faite en appliquant la méthode de soustraction du fond en deux temps. Après une première soustraction de fond, ils calculent l'offset entre la moyenne des modes du modèle SKDA (Sequential Kernel Density Approximation), et la moyenne des pixels classés comme appartenant au fond. Cet offset est utilisé lors d'une seconde soustraction de fond, pour obtenir une image nettement améliorée.

Les caméras infrarouges présentent donc un intérêt certain pour la détection d'obstacles : leurs coûts encore élevés empêchent leur diffusion massive sur les véhicules grand public, mais ils sont exploités, souvent en fusion avec des caméras dans le visible, dans des applications spécifiques comme la sécurité civile, les transports aériens ou la défense.

2.3.3 Vision mono-caméra

Lorsque qu'on ne dispose que d'une unique caméra, les méthodes de détection d'obstacles se fondent sur la reconnaissance de formes ou sur la vision dynamique. Les systèmes les plus avancés combinent ces deux approches :

— Les modèles peuvent aussi contenir les comportements dynamiques caractéristiques d'une classe d'objets (vitesse min et max, type de trajectoires, parmi d'autres).

— Ces modèles peuvent être partiellement acquis ou mis à jour en ligne.

Dans la suite, il est présenté une synthèses des travaux qui concernent l'analyse de séquences d'images monoculaires afin de détecter des objets statiques et dynamiques. Nous ne prétendons pas être exhaustive mais nous présentons les principales méthodes utilisées dans ce contexte car nos contributions concernent aussi la vision monoculaire. Nous trouvons dans la littérature deux grandes approches : l'approche spatiale fondée sur l'extraction d'indices visuels dans une seule image, et l'approche temporelle fondée sur la détection de mouvement sur plusieurs images successives. Les approches spatio-temporelles traitent de la détection et du suivi d'indices visuels.

Approche spatiale pour la détection d'objets.

En ce qui concerne l'analyse spatiale, les indices visuels les plus fréquemment identifiés sur les objets dans une image sont les suivants :

— *Texture*. La texture des objets sur l'image est caractérisée par des variations locales de luminance. La texture est notamment utilisée depuis le traitement d'images médicales à un niveau microsco-

pique pour la recherche de lésions au niveau cellulaire jusqu'à l'analyse d'images satellites pour analyser la nature des sols en télédétection, en passant par les images acquises depuis des caméras embarquées ou fixes [39]. Les indices de texture sont extraits couramment par les matrices de co-occurrence, par la technique de sommes et différences d'histogrammes par les réponses à des filtres de Gabor, etc.

— *Couleur*. La couleur est possiblement l'indice visuel le plus facile à exploiter dans un image. Il est normalement utilisé pour la détection et l'identification d'objets par des classifieurs paramétriques, de type bayésien, SVM ou AdaBoost. Quelques espaces de couleur utilisées dans la communauté de vision sont le RGB, le RGB-normalisé, le HIS pour Teinte/Intensité/Saturation, le CIE-Lab, le CIE- Luv, l'espace de Ohta $I_1I_2I_3$... Un résumé des caractéristiques de chacun de ces espaces couleur est présenté dans [78].

— *Symétrie horizontale*. Cet indice peut être utilisé si la géométrie des objets structurés qu'on recherche dans l'environnement est symétrique et connue par avance. Par exemple, dans les travaux de Bertozzi et al. [11], les auteurs exploitent le fait que l'image d'un véhicule observée dans une vue frontale, présente une symétrie suivant un axe vertical. Cette symétrie est généralement estimée à partir des contours extraits des images.

Nous avons testé dans nos propres travaux, une approche de classification couleur/texture ; cette méthode peut être considérée comme éparse [78], dans la mesure où elle s'appuie sur une segmentation préalable de l'image en régions uniformes en couleur. Dans ce travail l'image couleur est d'abord calculée à partir de l'image brute acquise par une caméra mono-CCD, par un algorithme de démosaïquage qui effectue des interpolations pour obtenir en chaqhe pixel, les trois composantes de la couleur dans l'espace RGB. L'image RGB doit être bien distribuée en couleur car la technique de segmentation dépend directement de l'illumination et de la qualité de la couleur. Par exemple l'image présentée dans la figure 2.6 [2], requiert une correction chromatique est . L'image à gauche est en format RGB mais il est notable qu'une dominante jaune prédomine partout dans l'image. L'image à droite est obtenue après correction ; bien qu'elle soit moins illuminée, elle a une meilleure distribution de la couleur, ce qui est requis pour obtenir de bonnes performances dans la segmentation.

Ensuite, l'image couleur corrigée est transformée vers l'espace $I_1I_2I_3$ proposé par Ohta [79] ; ce n'est pas l'espace optimal, mais c'est une transformation linéaire très rapide à appliquer, par rapport en particulier à la transformation vers CIE-Lab. Une méthode de segmentation trouve les régions connexes sur l'image $I_1I_2I_3$ qui ont les attributs couleur les plus uniformes en fonction de seuils établis a priori. L'image des régions résultant de la segmentation de l'image 2.6b est montrée sur l'image en haut à gauche de la figure 2.7 et à la même position dans la figure 2.8. Notons que le nombre des régions trouvées dans l'image est très grand (près de 50 régions). Les couleurs utilisées pour distinguer ces régions, n'ont aucune signification ; elles servent uniquement pour distinguer les régions voisines entre elles. À cette étape, il devient nécessaire d'introduire un autre indice visuel afin de caractériser ces régions. Donc, sur chaque région X segmentée dans l'image, des attributs de texture sont calculés en chaque point de X, et les valeurs moyennes de ces attributs sont associées à la région X. Ces attributs de texture sont complétés avec les attributs de couleur déjà calculés dans l'étape de segmentation, afin de former un vecteur qui caractérise les propriétés de chaque région X.

2. Cette séquence appartient à Thales Optolectronics ; elle a été acquise dans le cadre du projet TAROT

(a) Image RGB (b) Image corrigée

FIGURE 2.6 – La correction d'une image pour une meilleure distribution de couleur. a) L'image RGB acquise par la caméra, b) L'image après correction.

Les vecteurs caractéristiques des régions sont utilisés d'abord dans une phase d'apprentissage pour construire une base de données sur les attributs de régions identifiées par un opérateur comme SOL, CIEL, ARBRE, HERBE, etc. Puis ces vecteurs sont exploités dans une phase d'identification par un classifieur fondé sur l'algorithme KNN (K-Nearest Neighbors), pour reconnaître la nature des objets et du terrain perçus dans les images ; puis nous fusionnons les petites régions adjacentes identifiées dans la même classe, par exemple le sol.

Les résultats de la classification sont illustrés dans l'image à droite de la figure 2.7. La même image de régions a été testée sur un classifieur fondé sur la technique des SVM (Support Vectors Machine) ; les résultats correspondants sont présentés en figure 2.8.

Les techniques KNN et SVM sont des classifieurs supervisés qui exploitent une base d'apprentissage pour générer une stratégie de classification ; dans tous les cas, la qualité de cette stratégie dépend des propriétés de la base (complétude, redondance). La méthode SVM trouve un classifieur optimal. Dans les figures 2.7 et 2.8, les régions identifiées de chaque classe sont distinguées par des couleurs différentes. Le codage des couleurs est le suivant : bleu pour le ciel, gris pour le sol et marron pour les arbres ou le bois, et vert pour la végétation ; les régions rouges n'ont pas pu être classifiées. En bas les contours de la région SOL, donc libre d'obstacles, sont présentés.

Il est bien connu que le calcul des attributs de texture requiert de nombreuses opérations, ce qui est peu compatible avec le temps réel. L'approche précédente a été adaptée et implémentée sur FPGA dans notre groupe de travail par Ibarra-Manzano [51]. La classification couleur/texture est traitée pixel à pixel, donc sans passer par la segmentation préalable par la couleur ; l'algorithme AdaBoost, simple à paralléliser, donne le taux de reconnaissance le plus élevé. La figure 2.9 montre les résultats de classification obtenus dans un environnement intérieur en utilisant une approche dense de classification par couleur/texture fondée sur l'espace de couleur CIELaB, des attributs de texture calculés par ASDH (Adequacy of Sum and Difference Histograms). Cette approche, même si elle calcule les attributs sur chaque pixel, satisfait des contraintes de temps réel car elle est mise en œuvre sur FPGA.

Approche temporelle

Les approches temporelles exploitent le mouvement apparent dans une séquence d'images, comme caractéristique d'intérêt. Dans le cas où la caméra est embarquée sur un véhicule ou sur un robot, le but est

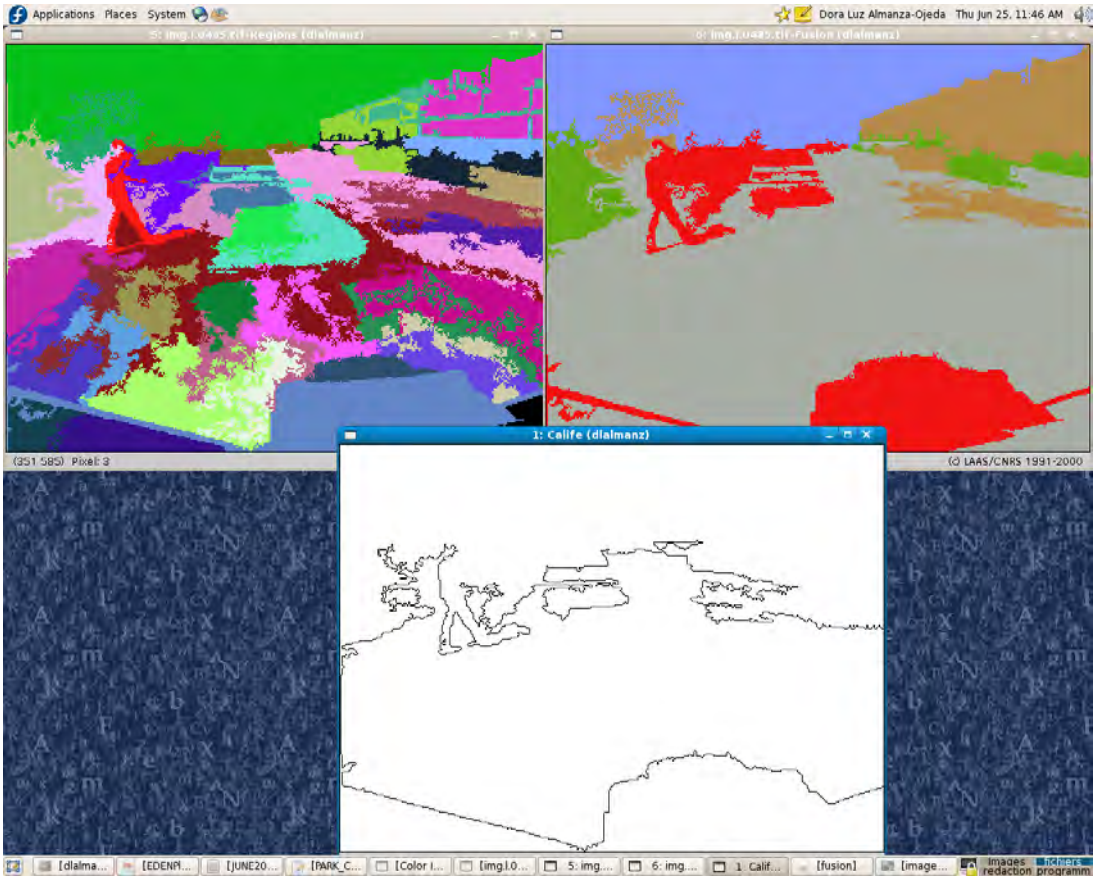


FIGURE 2.7 – Images résultant de la classification par KNN en utilisant l'information couleur-texture de l'image de la figure 2.6b. En haut à gauche l'image de segmentation par couleur uniquement, à droite cette même image classifiée par KNN. En bas contours de la région classée SOL.

d'estimer à partir d'indices visuels suivis sur les images, le mouvement d'objets mobiles présents dans la scène ; nous pouvons rencontrer certaines difficultés à distinguer le mouvement propre du véhicule de celui des éléments de la scène, puisqu'une image donne juste une projection de la scène 3D. Les effets de perspective introduits par la projection, engendrent des mouvements apparents différents selon l'endroit où se trouvent les objets détectés dans l'image. C'est ce qu'on appelle l'effet de parallaxe : pour un même mouvement 3D, les objets proches de la caméra engendreront un mouvement apparent dans l'image plus important que celui des objets lointains. Par ailleurs, une hypothèse de mouvement linéaire (ou mouvement à vitesse constante) est souvent utilisée afin d'estimer le déplacement apparent v dans le temps des indices visuels. Ce déplacement est obtenu en calculant la différence entre la position de cet indice au temps t et sa position au temps $t+1$; cela implique que cet indice a pu être apparié ou suivi entre ces images successives de la séquence.

Approche spatio-temporelle

Les meilleures performances pour la détection d'obstacles par vision monoculaire, sont obtenues par des approches hétérogènes fondées sur l'analyse spatiale et temporelle. Nous évoquons, pour ce type d'approche, le travail développé dans [93] où le vecteur v est obtenu par une minimisation aux moindres carrés sur un groupe de pixels décrivant l'indice visuel (une fenêtre gaussienne) entre les images acquises

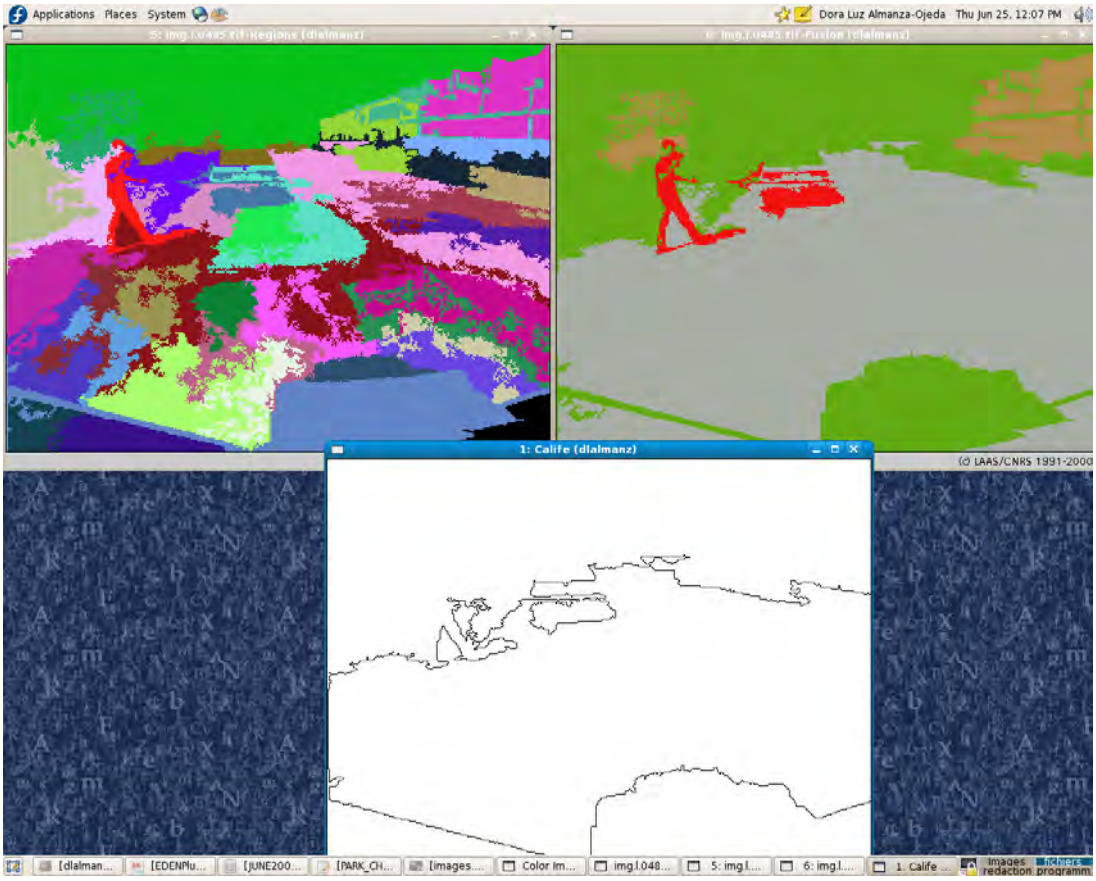


FIGURE 2.8 – Images résultant de la classification par SVM en utilisant l'information couleur-texture de l'image de la figure 2.6b. En haut à gauche image segmentée par couleur uniquement, à droite image classifiée par SVM. En bas contours de la région classée SOL.

(a) Image RGB (b) Image classifiée binaire

FIGURE 2.9 – Classification du sol et des objets dans une image d'intérieur en employant l'algorithme AdaBoost. Sur l'image binaire, en blanc : espace trouvé comme navigable, en noir : obstacles.

à deux instants successifs. Une autre approche spatio-temporelle, que d'ailleurs nous avons analysée et mise en place dans nos travaux (décrits dans le chapitre 3), a été proposée par Veit et al. [98]. Les auteurs associent trois descripteurs pour caractériser le mouvement apparent d'indices visuels suivis dans une sé-

quence d'images. Ces mouvements sont comparés avec un modèle aléatoire qui représente le mouvement apparent pour les objets statiques appartenant au fond ou arrière-plan de l'image. Cette comparaison permet au système de distinguer les mouvements d'objets du fond (y compris le va-et-vient des feuilles d'un arbre, les petits changements d'illumination dans la scène, etc.) qui n'ont pas d'importance dans la détection d'objets mobiles, et les mouvements d'un obstacle dynamique potentiel.

Nous décrirons dans la section suivante les approches spatio-temporelles qui traitent de la détection et du suivi d'objets.

2.3.4 Conclusion sur les capteurs de vision

Dans le contexte de la robotique mobile, nous avons décrit les différentes méthodes proposées en vision pour la détection d'objets et pour la classification des régions segmentées sur les images. D'une part les inconvénients de la stéréovision sont devenus de moins en moins gênants avec les nouveaux dispositifs commerciaux pré calibrés : donc ces capteurs sont des candidats très pertinents pour les applications robotiques, mais ils restent encore difficilement acceptables économiquement sur des véhicules grand public. D'autre part utiliser la vision monoculaire reste un challenge pour les applications d'extérieur du fait des conditions d'illumination et d'une excessive variation des caractéristiques de texture.

En recourant à des architectures dédiées embarquables comme les systèmes à base de FPGA (les GPU ne sont pas embarcables pour l'instant), des grandes améliorations dans les temps de traitement sont obtenues, ce qui rend possible de proposer en robotique des algorithmes de vision très consommateurs de puissance de calcul (texture, filtrage particulaire ou stéréovision dense). Cependant le coût et le temps de la conception d'un algorithme sur une architecture dédiée, restent encore plus élevés qu'une conception sur logiciel.

2.4 Suivi et estimation de l'état des objets

Les techniques pour le suivi d'objets peuvent être classifiées en deux principaux groupes : le suivi basé sur les propriétés d'apparence, et le suivi basé sur des modèles. La première approche exploite les propriétés 2D d'objets dans l'image comme la géométrie, les bords, la couleur, la texture (déjà décrits dans la section précédente) pour estimer l'état et prédire la position suivante de l'objet dans une séquence. La deuxième approche utilise un modèle de l'objet à suivre, comme un modèle CAO, éventuellement grossier, ou encore une modélisation de l'apparence sous la forme d'un motif exploité pour faire le suivi et pour estimer le mouvement. En vision monoculaire, l'estimation nécessaire pour le suivi est 2D ; il s'agit de prédire le mouvement apparent de la région ou de la primitive suivie dans l'image. Si la perception est 3D (télémétrie ou stéréo), si en vision monoculaire, est appliquée une approche de type SFM *Structure From Motion* ou si l'objet suivi est localisé dans l'espace de travail, alors, l'estimation peut se faire en 3D. Dans ce cas, la prédiction du mouvement apparent se fait en exploitant le modèle du capteur, par exemple par projection perspective pour une caméra.

Dans cette section, nous évoquerons tout d'abord les techniques de suivi dans une séquence d'images, fondées sur les propriétés d'apparence des objets : approches exploitant les points d'intérêt, les régions (Mean-Shift), le Flot Optique ou les contours : seules certaines méthodes seront décrites. Nous présenterons ensuite les méthodes bayésiennes exploitées pour réaliser ce suivi d'une part, et l'estimation du mouvement des objets suivis d'autre part.

2.4.1 Suivi de régions par un modèle statistique d'apparence

Pour commencer, nous décrirons les travaux de suivi d'objets en temps réel fondés sur des propriétés visuelles où les distributions statistiques caractérisent la région d'intérêt, projection de l'objet dans l'image. Comaniciu et Meer, dans [20], présentent la méthode MeanShift, capable de détecter, localiser et suivre une cible mobile représentée par un modèle statistique, même si les conditions d'acquisition des images sont défavorables (changements d'échelle, occultations ou bruit). La méthode d'itération de vecteurs Mean-Shift sert à trouver la cible candidate, celle qui est la plus similaire au modèle donné de la cible, en comparant leurs fonctions de densité probabiliste. La similarité est exprimée par une métrique fondée sur le coefficient de Bhattacharya. La minimisation de l'écart entre les fonctions de densité sert à trouver la nouvelle position de la cible dans l'image courante. La robustesse aux changements d'échelle est obtenue en faisant varier la taille de la fenêtre. Le système est aussi rendu robuste aux occultations, en pondérant les couleurs des pixels du centre de la région par rapport à ceux pris sur les bords.

Plus récemment, Han et al [38] proposent une technique d'approximation pour obtenir le noyau (ou kernel) de densité basée sur un mode (maximum local) d'une fonction de densité donnée. La même méthode d'itération de vecteurs Mean-Shift développée dans [20] est utilisée pour converger sur 3 types de points stationnaires : points d'inflexion (saddle), points minima locaux, et points maxima locaux. Cependant, les points maxima locaux sont les plus importants pour simplifier la fonction d'approximation du noyau. Cette méthode itérative non paramétrique est devenue une technique appelée aussi CAMSHIFT ("Continuously Adaptive Mean-Shift") qui finalement sert à trouver le mode des fonctions de distribution de probabilité d'un objet. La première partie de l'algorithme CAMSHIFT consiste à obtenir le centre et la taille de l'objet à suivre en utilisant une image qui représente la distribution de probabilité de couleur de l'objet. Cette image de probabilité est obtenue à partir d'un modèle d'histogramme couleur déjà connu. La méthode CAMSHIFT est populaire, car intégrée dans la librairie OpenCV [53].

2.4.2 Suivi d'objets basé sur un modèle géométrique

Les techniques de suivi d'objets basées sur un modèle, produisent généralement une solution robuste. La connaissance de la scène, c'est-à-dire l'information 3D implicite, permet au système d'améliorer le comportement du suivi de sorte qu'il est capable de prévoir le mouvement d'un objet caché, de détecter les occultations partielles et de diminuer l'effet des données mises en correspondance de manière erronée (outlier) pendant le suivi.

Comport et al. dans [21] proposent un algorithme pour le suivi d'objets 3D dont le modèle CAO (Conception Assistée par Ordinateur) est connu. Dans ce travail, l'obtention de la position est calculée par une méthode d'optimisation non linéaire appelée Virtual Visual Servoing (ou AVV : Asservissement Visuel Virtuel) : le calcul de la position est considéré comme une tâche itérative exploitant le formalisme de l'asservissement visuel 2D. Cette dernière tâche consiste à réguler dans l'image les indices visuels pendant le suivi pour qu'ils restent attachés au modèle de la cible. Pour ce faire, une loi de contrôle en boucle fermée qui utilise un M-estimateur est mise en œuvre. La combinaison de plusieurs indices visuels (points, lignes, cercles) est possible en fusionnant les matrices d'interaction de chaque indice dans une seule matrice. L'erreur de positionnement des indices visuels est calculée comme la somme des écarts entre chaque primitive extraite de l'image, avec les contours du modèle CAO de l'objet projeté dans l'image. Une hypothèse de base de cette méthode suppose que les contours de l'objet sur l'image peuvent être décrits comme des courbes paramétrées, droites ou quadriques.

L'appariement entre les primitives de l'image et le modèle CAO projeté, exploite l'approche EMD

proposée par Bouthemy ; il s'agit d'une approche de corrélation 1D permettant d'associer des points de gradient extraits de l'image avec les arêtes non occultées du modèle de l'objet projetées sur l'image. Les arêtes projetées sont échantillonnées en un ensemble de points pris à une distance donnée en paramètre. En chaque point échantillonné, une recherche unidimensionnelle est faite sur la normale au contour en ce point, pour trouver un point de gradient de l'image ayant les mêmes propriétés photométriques (forme du contour en ce point).

Plus récemment, dans le même cadre de recherche, Dionnet et Marchand ont appliqué cet algorithme de suivi pour réaliser des tâches en asservissement visuel dans l'espace [28]. L'expérimentation est réalisée sur l'Eurobot développé par l'ESA ; ce robot, destiné à assister l'homme lors des activités extra véhiculaires dans l'International Space Station (ISS), est constitué de trois bras. L'Eurobot est équipé d'une caméra à la fin de chaque bras et d'un système de stéréo vision à la base : l'algorithme a été adapté pour considérer les informations venant de plusieurs caméras.

2.4.3 Suivi et estimation par des méthodes probabilistes

En matière d'estimation probabiliste, nous décrirons les travaux développés pour le suivi multi-cible. Ces approches, fondées sur la théorie de Bayes, ont été exploitées dans le cadre de plusieurs projets de recherche européen, notamment le projet BACS (Bayesian Approach to Cognitive Systems) auquel participe l'INRIA Rhônes Alpes (C. Laugier, T. Fraichard et al.). Cette théorie [8] se fonde sur une méthode d'inférence et d'apprentissage probabiliste pour expliquer et comprendre la perception, et pour décider quelles doivent être les actions suivantes d'un système robotique.

Nous décrivons ici plusieurs traitements qui utilisent les approches bayésiennes : suivi sur une grille discrète, prédiction en exploitant différentes représentations des trajectoires et enfin prédiction avec plusieurs modèles de mouvement.

Suivi et estimation à partir d'une grille discrète. Dans [103] et [22] sont décrits des travaux qui utilisent des techniques de filtrage Bayésien. Ces travaux présentent un modèle qui est une extension du Filtre d'Occupation Bayésien (BOF en anglais) pour effectuer le suivi d'objets en environnements dynamiques. Le modèle est centré autour d'une représentation discrète, une grille d'occupation constituée de cellules de taille donnée. En chacune de ces cellules sont stockées une distribution de probabilité de l'occupation de la cellule et une autre distribution sur la vitesse de l'objet qui l'occupe. En plus, chaque cellule est considérée indépendante des autres ; ceci évite de devoir représenter les corrélations ou covariances entre les distribution de probabilités afférentes à des cellules voisines, mais cela empêche aussi de faire émerger la notion d'un objet physique.

En ne considérant pas cette notion d'objet, le problème classique de l'association de données entre objets n'a plus de sens ce qui simplifie l'ensemble de la chaîne de traitements. Avec pour but de diminuer le temps de calcul, une nouvelle formulation du BOF est décrite par Tay et al [94] ; à la différence du filtre utilisé dans [103] et [22], ils stockent aussi la distribution de la vélocité de propagation de la cellule occupée.

Cette approche exploite donc une grille 2D discrète d'occupation ; elle est surtout adaptée avec des capteurs télémétriques radar ou laser 2D (balayage dans un plan parallèle au sol). Néanmoins, la nouvelle formulation de Tay et. al. est utilisée pour détecter et suivre des objets dynamiques sur des environnements urbains [81], avec fusion des données laser 2D/stéréovision pour calculer les valeurs dans chaque cellule de la grille. Les mesures de distance données par le télémètre laser sont fusionnées avec les

mesures de disparité données par le système stéréo. Cela permet de détecter et filtrer les erreurs de la stéréovision d'une part, et de corriger les erreurs sur la taille des objets, erreurs connues des méthodes de stéréovision dense exploitant des fenêtres de corrélation.

Suivi fondé sur des modèles de trajectoire. En matière d'estimation de la trajectoire par des méthodes probabilistes, on trouve dans la littérature, des approches d'estimation basées sur la maximisation de la fonction de vraisemblance (en anglais Likelihood Function Maximization). Parmi ces approches citons le travail proposé par D.Vasquez, T.Fraichard et. al [96]. En supposant que les objets mobiles suivent des trajectoires typiques dans l'environnement, les auteurs proposent une technique pour estimer ces trajectoires sur la base de modèles de mouvement typique, dans un environnement structuré avec sol plat.

Pour ce faire, ils proposent un algorithme de classification non supervisée divisé en 2 étapes : une étape d'apprentissage et une étape de prédiction. Dans l'étape d'apprentissage la technique obtient les différentes classes associées à chaque modèle de mouvement typique ainsi que leurs moyennes et leurs écarts-types. Ces classes de mouvement sont liées aux trajectoires courantes des objets mobiles détectés dans l'environnement. Dans l'étape de prédiction, on obtient un écart partiel entre la trajectoire observée et chaque classe apprise. Avec cet écart partiel on estime la vraisemblance (likelihood) que la trajectoire observée appartienne à une certaine classe de trajectoires.

Cependant, cette représentation n'est pas assez flexible et ne permet pas de prendre en compte facilement les incertitudes des observations. Donc dans des travaux plus récents [97], ils ont proposé une représentation stochastique, les Modèles de Markov Cachés (ou HMMs, Hidden Markov Models), basés sur un nombre fini d'états, connectés dans un graphe de transition. A partir de ce modèle HMM, l'état courant d'une cible est estimé à l'instant t et prédit de (t+1) à (t+H) à partir des observations, par des inférences bayésiennes. La contribution principale dans ces travaux concerne l'Apprentissage Incrémental de la structure du graphe (création/destruction dynamique des états et des transitions inter-états) et des paramètres qui lui sont attachés.

La méthode est validée sur plusieurs jeux d'observations obtenus dans des environnements réels ou simulés. En particulier, la méthode a été évaluée

— en milieu intérieur dans un grand hall ; les objets suivis sont des hommes ; les trajectoires correspondent aux chemins suivis par les utilisateurs pour joindre des lieux d'intérêt dans ce hall, portes, panneaux d'affichage, guichets, etc.
— en milieu extérieur, dans un parking ; les objets suivis sont des piétons ou des voitures.

Suivi par la méthode IMM (Interacting Multiple Models). Les modèles de mouvement des objets peuvent être donnés a priori ou appris par une approche géométrique dans un espace continu ou par une approche de type HMM exploitant une représentation discrète. Pour exploiter ces modèles de mouvement possibles pour un objet, de nombreuses approches exploitent la méthode IMM, méthode introduite par Bar-Shalom [6].

Le principe de la méthode IMM, a été en particulier exploité dans [15], pour traiter le cas du suivi d'une personne en intérieur par un système de vidéo-surveillance, ou pour suivre des piétons ou des véhicules depuis des capteurs embarqués sur un véhicule. À chaque itération, les probabilités de tous les modèles de mouvement des objets suivis (par exemple, un véhicule suivi peut aller tout droit à vitesse constante, peut tourner à droite ou à gauche, peut s'arrêter...), sont ré-évaluées en considérant les possibles transitions entre ces mouvements. Au vu de l'observation courante, toutes les possibilités de

transitions sont analysées ; chaque cas est assorti d'un coefficient de vraisemblance ; l'état du véhicule est obtenu en faisant la combinaison linéaire de tous les états potentiels, en pondérant chaque possibilité par sa vraisemblance.

Les IMM modélisent donc l'interaction entre les différents filtres grâce à la Matrice de Transition de Probabilité (Transition Probability Matrix ou TPM). Cette Matrice encode la probabilité de changement du mouvement de l'objet suivi ; en général cette matrice est donnée a priori. Signalons les travaux récents de J. Burlet [14] dans lesquels les TPM sont adaptées ou apprises en ligne automatiquement. Cette adaptation permet d'augmenter significativement l'efficacité du filtrage IMM.

2.4.4 Conclusions sur les méthodes de suivi

De très nombreuses méthodes de suivi existent : nous nous sommes focalisée ici sur les algorithmes génériques, plus que sur les techniques spécifiques à un capteur particulier. En vision monoculaire, nous verrons dans les chapitres suivants que nous exploitons les méthodes classiques de suivi de points d'intérêt (KLT) et de régions (snakes ou contours actifs).

Il existe un lien fort entre le suivi, l'estimation, la fréquence d'acquisition des données sensorielles et la dynamique des objets perçus : le suivi a d'autant plus besoin d'une prédiction pour être plus robuste, que la fréquence est faible et que la dynamique des objets est élevée. Signalons aussi que si l'objet suivi a de nombreuses ruptures de dynamiques (typiquement, le mouvement d'un piéton en milieu urbain), alors il conviendra d'exploiter des méthodes multi-hypothèses (filtrage particulaire ou mélange de gaussiennes ou IMM) coûteuses en temps de calcul. Dans tous les cas, il existe un cercle vertueux bien connu : meilleure est la fréquence, meilleur est le suivi ; avec une fréquence très élevée (en vision, au delà de 30Hz), on peut se contenter d'une estimation du mouvement apparent à vitesse constante, très simple à mettre en œuvre par filtrage de Kalman.

De nombreuses méthodes décrites ci-dessus ont été proposées et sont appliquées dans un contexte de vidéo-surveillance : pour les transposer à la surveillance d'environnements dynamiques depuis des capteurs embarqués sur un véhicule ou robot, il faut considérer le lien qui existe entre la localisation du véhicule et l'estimation du mouvement des objets. Dans la section suivante, nous décrirons la technique du SLAMMOT (Simultaneous Localization and Mapping and Moving Objets Tracking) proposée pour distinguer les objets stationnaires des objets dynamiques et pour estimer la position et la vitesse de ces derniers.

2.5 Approches de localisation : SLAMMOT

Le SLAMMOT est une technique qui consiste à traiter dans un même système, la cartographie et localisation simultanée (SLAM) d'une part, et la détection et le suivi des objets mobiles et l'estimation de leurs états (identification, position, vitesse,etc.) d'autre part. Le SLAM est exécuté depuis un système mobile équipé de capteurs embarqués, typiquement un robot : il estime à la fois les paramètres liés au modèle de l'environnement, et les paramètres liés à l'état du robot, en particulier sa localisation. Pour ce faire, il intégre en temps réel les observations fournies par les capteurs extéroceptifs (télémétrie laser, caméras, radar) et proprioceptifs (estimation de l'égo-motion par odométrie ou inertiels) embarqués sur le robot mobile. Le résultat du SLAM est donc une carte stochastique qui donne :

— La modélisation des parties statiques de l'environnement,

— et la localisation du/des capteurs ou du robot qui les porte.

Le MOT rajoute à cette carte, l'estimation de l'état des composantes dynamiques de l'environnement. D'un côté, nous trouvons dans la littérature plusieurs approches pour résoudre le problème du SLAM et de l'autre côté les approches pour le MOT ou le DATMO (Detection And Tracking of Mobile Objects). Les méthodes SLAM font l'hypothèse que l'environnement est statique ; si des primitives de la carte stochastique construite par le SLAM, sont en fait portées sur un objet mobile, alors la carte sera faussée tant que cet objet sera dans la carte. Dans le cas général d'une exploration en environnement non contrôlé, le SLAM a besoin de prendre en considération que l'environnement ne contient pas uniquement des objets statiques. Donc, il est nécessaire de concevoir une stratégie qui soit capable d'adapter les méthodes SLAM en considérant aussi la détection et le suivi des objets dynamiques de la scène.

Une approche spatio-temporelle, appelée selon le cas *Visual SLAM* ou *Bearing-Only SLAM*, a été proposée par Davison [25] pour résoudre le problème du SLAM en utilisant une seule caméra. L'auteur utilise le détecteur de Harris pour initialiser des points d'intérêt qui après seront recherchés de façon active sur les images successives de la séquence acquise par la caméra. Les points sont initialement connus sous la forme d'un rayon optique dans l'espace. Après, s'ils sont réobservés dans les images suivantes, ils sont reconstruits par triangulation et deviennent des points en 3D qui aideront à la localisation de la caméra. Chaque point est représenté par son apparence, un motif de taille $n \times n$ autour de sa position dans l'image dans laquelle il a été détecté initialement. L'état du système à l'instant t, est donné par une carte stochastique : cette carte est définie par

— un vecteur stochastique qui regroupe plusieurs sous-vecteurs qui représentent les rayons optiques non encore reconstruits, la position 3D des points reconstruits, et la position et vitesse de la caméra,

— et par une matrice de variance-covariance entre toutes ces variables.

À partir de cet état, chaque observation des primitives visuelles apprises jusqu'à l'instant t, est prédite dans l'image acquise au temps $t + 1$. La projection d'un rayon ou d'un point 3D, avec son incertitude, forme une zone elliptique dans laquelle le point correspondant doit se trouver si l'état global est cohérent, c'est-à-dire si la position estimée par le SLAM de chaque primitive détectée dans l'environnement, est centrée autour de sa position réelle.

Le SLAM visuel a connu une évolution importante depuis les travaux initiaux de Davison. Les principales contributions concernent :

— la représentation des rayons optiques (ensemble de particules pour Davison, puis mélanges de gaussiennes (J.Sola [90] et T. Lemaire [65]), puis représentation appelée *Inverse Depth Parametrization* par J.Civera et al [19].

— les autres primitives visuelles, en particulier les segments, pour lesquels des paramétrisations des droites porteuses ont aussi été proposées pour les représenter avant qu'ils ne soient connus en 3D.

Nous détaillons les travaux de J. Solà [90] qui, dans une dernière partie de sa thèse, a traité du SLAM dans un environnement dynamique, donc du SLAMMOT. J.Solà est le premier à avoir inclus un rayon optique dans une carte stochastique : quand un point d'intérêt est détecté sur l'image, son incertitude est caractérisée par une fonction de densité de probabilité gaussienne. Une représentation du rayon optique par un mélange de gaussiennes avec des variances croissantes avec la profondeur, permet de représenter ce rayon optique par un cône de vue de manière probabiliste. Dans le cas monoculaire, une approche multi-hypothèses, permet de mettre à jour la carte SLAM sans retard. Ensuite, comme dans les travaux de A.Davison, une approche active (Active Search) pour l'appariement des points a été mise en œuvre ; on utilise les différentes gaussiennes pour rétroprojeter (back projection) un rayon optique en une zone

elliptique. Une nouvelle observation du point 3D correspondant à ce rayon, a une probabilité 0,99 de se trouver dans cette zone.

Par ailleurs, dans cette méthode SLAM la détection d'objets mobiles (MOT) a été aussi considérée, mais uniquement pour le cas de la stéréovision, car avec sa représentation des rayons optiques, J.Solà indique que profondeur et vitesse d'un point mobile ne sont pas observables par vision monoculaire. Chaque point extrait de l'image correspond donc à un rayon optique, représenté par N points 3D potentiels. Dans le cas stéréo, un rayon conique est construit dans les deux images pour deux points mis en correspondance au temps *t*. Il sera possible d'observer la profondeur d'un amer si l'intersection entre ces rayons coniques tombe dans la zone d'observabilité de la stéréo, sinon on traite un des points extraits comme dans le cas monoculaire. Donc pour chaque nouveau point observé, on a N hypothèses de profondeur : typiquement 6 ou 7 en monoculaire, une seule si le point est dans la zone d'observabilité de la stéréovision, seulement 2 ou 3 si ce point est en dehors de cette zone. Par ailleurs, ces points 3D potentiels peuvent être fixes ou mobiles dans le référentiel du monde. Toutes ces hypothèses sont évaluées à chaque itération, pondérées par un coefficient de vraisemblance ; la carte est mise à jour en fonction de ces coefficients. Donc, les hypothèses les moins probables sont détruites au fur et à mesure, jusqu'à trouver une seule solution (position) pour chaque amer fixe, ou (position, vitesse) pour chaque point mobile ainsi reconstruit.

Les performances temps réel de cet algorithme SLAM dépend largement de la taille de la carte et du nombre de points appariés par Active Search à chaque itération. Pour gagner du temps de calcul, seulement un ou deux amers sont rajoutés dans la carte à chaque itération ; l'image est partitionnée en zone rectangulaires de taille identique ; un amer ponctuel (rayon optique initialement, puis point 3D) est créé seulement dans une zone de l'image dans laquelle il n'y a pas d'amers rétroprojetés, et en prenant pour cette zone, le point de Harris le plus discriminant. Dans le cas des points mobiles, on peut avoir une connaissance a priori des zones dans lesquelles ces points peuvent apparaître, par exemple sur la ligne d'horizon de l'image par exemple dans le cas d'images acquises depuis un véhicule.

B.Wang et C.Thorpe [100] ont également proposé une approche SLAMMOT très souvent citée. Au lieu d'une carte stochastique décrivant des amers épars, l'environnement est représenté par une carte discrète composée de cellules de $5cm \times 5cm$, de type grille d'occupation. Un véhicule mobile est équipé avec un télémètre laser à balayage horizontal. L'algorithme de *point to point matching* par ICP est exploité pour apparier les données sensorielles courantes avec les cellules occupées de la carte des Objets Stationnaires (SO-Map), dans lesquelles ont été fusionnées les données acquises précédemment sur les composantes statiques de l'environnement. Les résultats de cette procédure d'appariement ICP, sont donc intégrés et stockés dans cette carte ; pour chaque cellule, on précise le temps qu'elle a été occupée par les mêmes points stationnaires. Si cette valeur dans une cellule est grande, ça veut dire que la cellule appartient à un objet stationnaire. Les points mobiles sont segmentés en fonction de la cohérence de leurs mouvements, et conservés dans des cartes d'Objets Mobiles (MO-Map).

Notons dans ces travaux de B.Wang et C.Thorpe sur le SLAMMOT, une discussion sur deux approches possibles du SLAMMOT. La première consiste à gérer les objets ou primitives dynamiques, directement dans la carte créée et mise à jour par l'approche SLAM. La description de ces objets est généralisée au sens où elle contient la dynamique du mouvement. Les auteurs indiquent qu'une telle approche est trop complexe pour être exécutée en temps réel. La deuxième approche est celle qu'ils ont implémentée avec ces cartes discrètes SO-map et MO-map, dans laquelle les objets dynamiques sont décrits indépendamment de la carte.

Il est admis dans la littérature qu'il faut représenter séparément le monde statique (que cela soit par une carte discrète ou une carte stochastique) et les objets dynamiques qui s'y déplacent. On néglige les corrélations entre les variables stochastiques qui décrivent leurs états, mais on peut ainsi exécuter la fonction SLAMMOT en temps réel. Signalons aussi ici les travaux de G. Gaté [34] qui définit un cadre conceptuel cohérent pour traiter à la fois de la Détection, du Suivi, du SLAM, du MOT et de l'Identification des objets perçus lors du déplacement d'un capteur dans un environnement dynamique ; mais cette approche très élégante est restée théorique, donc sans validation expérimentale.

2.6 Approche proposée

Dans ce manuscrit, nous allons proposer une approche de détection et suivi des obstacles depuis des images acquises en vision monoculaire ; ce choix a été fait car d'une part c'est un challenge scientifique, vis-à-vis des nombreuses approches existantes fondées sur la télémétrie laser, et d'autre part nous n'exploitons que des capteurs bas coût, conditions indispensables pour transférer ces méthodes sur des produits grand public (véhicules, robots de service).

Nos travaux portent essentiellement sur la détection et le suivi d'objets dynamiques détectés depuis une caméra embarquée sur un véhicule ou robot mobile. Mais nous avons fait quelques pas vers l'intégration de notre approche dans un système plus complet qui prend en compte les fonctions SLAM et MOT.

2.6.1 Notre approche de détection et suivi d'objets dynamiques

La figure 2.6.1 montre le diagramme de la procédure développée dans ce travail. Nous allons tout d'abord étudier la méthode propose par Shi-Tomasi [86] pour la détection de N_{pts} points d'intérêt dans l'image d'entrée à partir de l'analyse des gradients. Ces points sont en nombre limité, car les traitements qui les exploitent doivent être exécutés en temps réel ; ils doivent donc être sélectionnés à chque étape, pour bien représenter les objets les plus remarquables dans la scène, vu les connaissances courantes. Néanmoins, ces points seront détectés en fonction des propriétés de contraste et d'illumination sans viser la détection d'une catégorie spécifique d'objets (piétons, voitures,etc.) ; on peut considérer qu'on propose un détecteur d'objets génériques, sans modèle a priori.

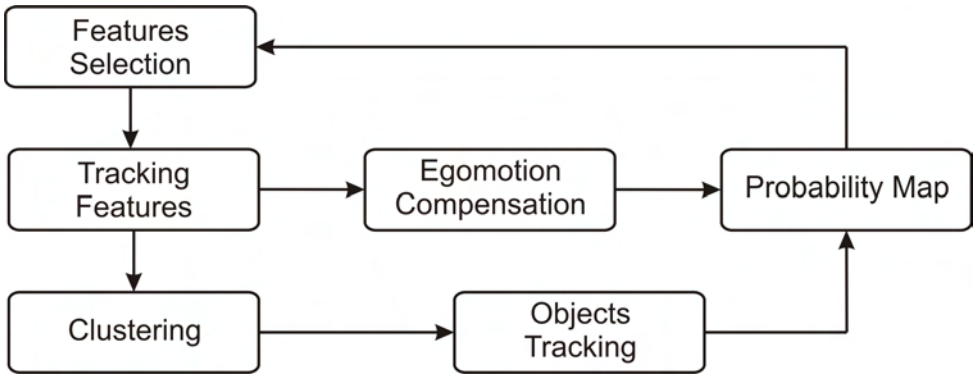


FIGURE 2.10 – Diagramme général de l'approche détection et suivi d'objets.

Ensuite chaque point sera consécutivement suivi par la technique du KLT pendant un petit nombre N d'images que nous avons appelé *temps de pistage*. Les boîtes *Features Tracking* et *Object Tracking* masquent des boucles internes de suivi de points ou d'objets sur N images successives ; la fréquence de la boucle Détection/Suivi/Clustering représentée en figure 2.6.1, est donc au moins de N fois la fréquence d'acquisition des images en traitement purement séquentiel (il est supposé que le KLT trourne à la fréquence d'acquisition, typiquement 30Hz).

Chaque point suivi sur les N images, sera caractérisé par un vecteur de quatre dimensions, une piste ou *tracklet* étant définie par sa position et sa vitesse en 2D. Les pistes des points suivis sur N images successives, sont analysées par une fonction de groupement ou "clustering" [98] qui classifiera les points comme statiques ou mobiles, et qui formera des sous-ensembles de points mobiles avec des positions proches et des vitesses similaires ; un tel sous-ensemble ou *cluster* deviendra notre modèle initial d'un objet mobile. Chaque objet mobile ainsi détecté sera suivi séparément dans les images suivantes,et son mouvement apparent sera estimé à partir des points qui le définissent, par filtrage de Kalman.

En plus une carte de probabilités mémorise les positions des points détectés et suivis avant leur regroupement comme objets. Cette carte de probabilités est très importante pour la sélection des nouveaux points qui seront détectés et suivis à l'itération suivante dans les zones de l'image les plus probables pour y trouver un objet mobile ; la carte de probabilités est mise à jour à l'aide de connaissance a priori, de l'historique et de l'état courant de l'environnement.

Pour améliorer la performance de cette méthode combinant détection, suivi et clustering de points pour la détection d'objets mobiles, nous devons supprimer les fausses détections qui apparaissent surtout sur les zones de l'image correspondant au sol proche du robot. Nous avons d'abord mis en œuvre une fonction pour compenser le mouvement du robot en s'appuyant sur la géométrie épipolaire [31], [40] et d'une variante de la méthode *Inverse Perspective Mapping* [70]. Les mesures obtenues depuis les capteurs proprioceptifs (odométrie, centrale inertielle) embarqués sur le robot sont nécessaires pour estimer le mouvement propre de la caméra et le compenser dans les modèles des objets dynamiques. Cette méthode nous permet d'augmenter la vitesse de déplacement du robot car nous corrigeons la dynamique des points mobiles détectés sur la partie basse de l'image pour savoir s'ils sont vraiment des points acquis sur des objets mobiles ou s'ils correspondent à des points fixes proches du robot sur le plan du sol, points dont le mouvement apparent est créé par le mouvement de la caméra.

Dans le contexte de la navigation sur un sol plan, avec ce module de prise en compte de l'égomotion, nous fermons la boucle qui traite la fonction MOT explicitement par vision monoculaire et mesures d'odométrie. Cette méthode de compensation ne fonctionne que sur le sol ; une compensation pourrait aussi être appliquée pour corriger le mouvement apparent de points jugés mobiles dans l'image, correspondant à des points 3D fixes lointains dans la scène (par exemple, points caractéristiques sur la ligne d'horizon).

Mais, dans le cas général, pour un environnement naturel comme celui montré sur l'image du robot Mana dans la figure 2.1, le sol n'est pas plan : donc notre méthode de compensation ne s'appliquera pas, ou fera des erreurs. Pour faire face à cette situation, dans la suite nous proposons d'intégrer notre approche de détection et suivi des objets dynamiques, avec d'autres méthodes classiques utilisées dans la navigation.

2.6.2 Vers l'intégration avec une méthode SLAM.

Nous représentons dans le diagramme 2.11, la façon d'intégrer nos travaux avec ceux menés par ailleurs sur le SLAM et le MOT. Étant donne la modularité de notre approche, nous avons commencé son incorporation avec un module SLAM développée au sein de notre équipe par D. Marquez [72]. Un diagramme global est présenté en figure 2.11. Les fonctions additionnelles à considérer afin de mettre en œuvre le SLAMMOT sont répresentées en couleur grise.

Le but de cette fusion est de corriger mutuellement les problèmes qu'ont les deux approches : pas de point mobile dans la carte SLAM, et pas de fausse détection d'objets mobiles. Notre objectif est de fusionner les modèles qu'elles construisent lors de la navigation, afin de les rendre complémentaires. Nous avons atteint cet objectif partiellement.

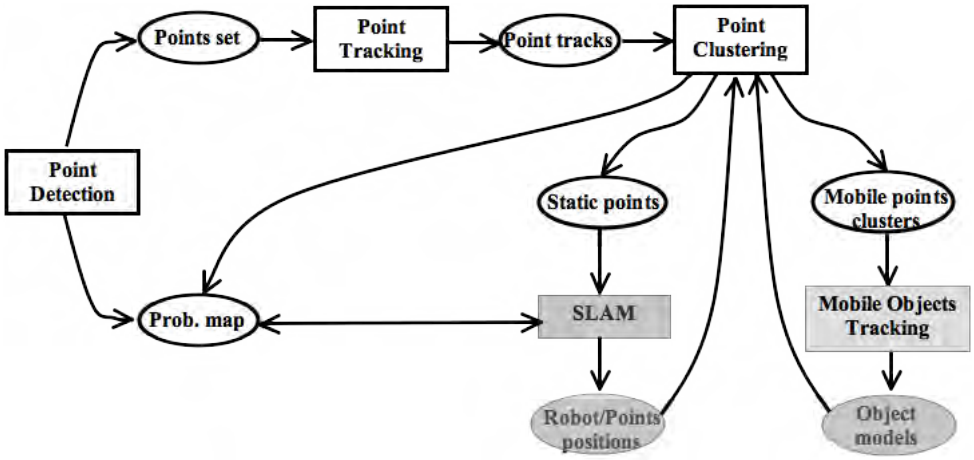


FIGURE 2.11 – Diagramme global de l'intégration d'une méthode SLAMMOT avec notre approche de suivi d'objets.

2.7 Conclusion

Nous avons présenté d'abord dans ce chapitre une étude bibliographique sur la détection d'obstacles depuis des capteurs montés sur un véhicule effectuant un déplacement. En conclusion, soulignons que malgré la grande quantité d'algorithmes et de techniques consacrés à la détection, au suivi et à l'estimation de la dynamique d'objets mobiles dans l'environnement d'un robot mobile, cette thématique de recherche dans le cadre de l'étude des systèmes de navigation autonome, est encore très ouverte. Le choix entre l'une ou l'autre des techniques possibles, exige une étude très approfondie des spécifications du système à développer ainsi que beaucoup d'expérimentations avant la mise en œuvre d'un système complet.

Il apparaît que la phase essentielle est la détection des objets mobiles, car la phase de suivi et d'estimation exécutée ensuite séparément sur chaque objet détecté, permet de différencier les composantes stationnaires et mobiles de l'environnement, donc de filtrer les faux positifs qui provoqueraient des fausses alarmes sur un système de navigation (robot) ou sur un système d'aide à la conduite (véhicule) et de renforcer la vraisemblance sur les vrais positifs. Aussi nous avons présenté comme une partie importante de cette étude bibliographique, le suivi qui intègre une fonction d'estimation de l'état (position, vitesse)

des objets mobiles, potentiellement futurs obstacles pour le robot. Même si nous ne les avons pas exploitées par manque de temps, signalons l'importance des différentes méthodes d'estimation bayésienne. Concernant la fonction de modélisation et localisation simultanées, nous avons analysé quelques stratégies, notamment celle déjà exploitée au LAAS par Joan Solà pour le SLAM monocaméra, car elle sera à terme intégrée avec notre approche MOT.

Nous avons, finalement, présenté la structure générale de la méthode qui nous semble la plus exploitable au vu des capacités de calcul disponibles sur un système embarqué et des contraintes temps réel propres à cette fonction de détection et suivi des objets mobiles. D'abord on exploite des points d'intérêt, faciles à détecter et à suivre, puis le regroupement des points pour faire émerger des connaissances sur les objets dynamiques. Les critères de ce regroupement (ou clustering), décrit dans le chapitre suivant, sont d'abord liés à la cohérence de la dynamique spatiale et temporelle des points détectés sur la surface de l'objet mobile ; nous verrons dans le chapitre 4 qu'un objet sera ensuite caractérisé par une région homogène au sens de l'illumination, afin d'être suivi par une méthode basée région.

Chapitre 3

Détection d'obstacles mobiles par une approche spatio-temporelle

Sommaire

3.1 Introduction

Le calcul du flot optique est la méthode choisie dans ce travail pour calculer la vitesse des points d'intérêt projections de points 3D appartenant aux objets mobiles, le long d'une séquence d'images. Si le flot optique est calculé pour tous les points de l'image, alors le champ de mouvement résultant est appelé flot optique dense. Une approche dense a été proposée par Horn and Schunck [42] où le champ de mouvement de l'image entière permet d'avoir le mouvement apparent en tout pixel de l'image. Cependant cette estimation du flot optique dense a un coût important en temps de calcul. Dans le cas où uniquement les vitesses apparentes de certains points de l'image sont calculées, alors ces vecteurs de mouvement constituent un flot optique épars. C'est cette dernière alternative que nous avons choisie pour développer les

fonctions en charge de la détection et du suivi des caractéristiques mobiles. Ces fonctions correspondent aux deux premiers blocs du diagramme général de notre approche, présentée en figure 2.6.1. Le calcul d'un flot optique épars est un algorithme fondamental bien connu dans la communauté de vision par ordinateur. Parmi les avantages de cette méthode, nous soulignons essentiellement les points suivants :

— il est rapide en temps de calcul
— il ne s'intéresse qu'aux points les plus caractéristiques de l'image
— il a une faible complexité numérique favorisant son implémentation sur des systèmes parallélisables, comme un GPU ou un FPGA [52], [69].

Notre méthode va ensuite exploiter un vecteur associé à chaque point d'intérêt, vecteur donnant sa position initiale ainsi que sa vitesse. La vitesse est obtenue en utilisant le flot optique calculé sur un certain nombre d'images au cours de la séquence vidéo. Nous avons appelé cette période d'accumulation des points, le *temps de pistage*. Ce temps est essentiel dans la performance des fonctions suivantes de notre approche, parce qu'il détermine la quantité des données qui devront être traitées pour identifier les objets mobiles, ainsi que le temps de latence pour détecter la présence d'un objet mobile dans une scène à partir d'une séquence d'images. Ainsi, plusieurs *temps de pistage*, définis par différents nombres d'images sur lesquelles les points sont suivis, sont testés et montrés sur des séquences acquises depuis un robot mobile à la fin de la section 3.2 ; ceci nous permettra de trouver la valeur adaptée à notre problématique.

Une fois que les points d'intérêt sont extraits et suivis pendant ce *temps de pistage*, il est important de distinguer parmi les points suivis, ceux qui correspondent à des points 3D portés par des objets mobiles. Les techniques de groupement ou *clustering* peuvent être proposées pour répondre à cette question, mais malheureusement la plupart d'entre elles nécessitent une connaissance *a priori* sur la scène comme, par exemple, le nombre de groupes à trouver. Le succès de ces méthodes dépend fortement de ces paramètres d'initialisation. T. Veit, F. Cao, et. al. [98] ont fait face à la même problématique pour l'analyse de courtes séquences vidéo. Ils ont présenté un algorithme de groupement basé sur une méthode *a contrario* [27] qui n'a pas besoin de paramètre ou d'information initiale sur la scène, pour trouver dans une séquence d'images, des groupes de points qui sont les projections de points 3D portés par des objets mobiles. Pour tirer parti de ces avantages, nous avons inclus la méthode de groupement *a contrario* dans notre approche de détection et de suivi, ce qui la rend utilisable dans le contexte d'un environnement inconnu *a priori*. Une description plus détaillée de la méthode *a contrario* est présentée dans la section 3.3 ainsi que des résultats pour trois situations différentes de détection d'objets qui peuvent survenir lors de la navigation du robot.

Étant donné le caractère épars de l'algorithme de calcul du flot optique, le choix des points les plus représentatifs de la scène est primordial pour le bon fonctionnement de l'algorithme. Une stratégie qui permet une recherche active des points à chaque itération de la boucle de détection, devient nécessaire. Pour cela, une carte donnant en chaque pixel, la probabilité qu'il corresponde à un point mobile de la scène, est mise à jour à chaque itération, puis exploitée pour la sélection active des points. L'idée principale de cette carte consiste à affecter une probabilité d'apparition d'un point dynamique en tout pixel de l'image en fonction de son contenu précédent. Ainsi la valeur de probabilité de chaque pixel de la carte, est entièrement déterminée par les observations précédentes faites dans le voisinage de ce pixel. Cette carte est initialisée par le module de détection et ensuite mise à jour par les autres modules à chaque itération de la boucle de détection. La description du fonctionnement de la carte est présentée dans la section 3.4.

Le regroupement des points mobiles par la méthode *a contrario* permet de détecter des objets dynamiques mais, dans le cas où l'objet rentre dans la scène, seule une partie est identifiée. Afin de détecter un l'objet en sa totalité dans l'image, nous devons exécuter de façon continue les fonctions de sélection-suivi-groupement avec de nouveaux points d'intérêt à chaque début de cycle. Ainsi à la fin de chaque cycle, de nouveaux groupes de points dynamiques seront détectés, sachant que certains des groupes détectés à l'instant t, correspondent à des parties d'objets mobiles détectés aux instants précédents. Dans la section 3.5 nous proposons une règle afin de tester si la fusion de ces groupes détectés à des instants différents est possible. Enfin, nous montrons la performance de ces différentes fonctions sur des séquences d'images pour parvenir à la détection complète d'objets mobiles rigides et non-rigides.

3.2 Flot optique

Le flot optique est un moyen de détecter le mouvement présent dans un environnement intérieur ou extérieur à partir d'une séquence d'images acquises depuis une caméra fixe ou mobile. Sa théorie est fondée sur l'hypothèse de conservation spatio-temporelle de la luminance de la scène. Soit $I(\mathbf{x}, t)$ la luminance d'un point $\mathbf{x} = (x, y)^T$ sur le plan image au temps t, alors l'expression mathématique qui conserve invariante la luminance d'un point est :

$$\frac{dI(\mathbf{x}, t)}{dt} = 0 \tag{3.1}$$

Cette conservation spatio-temporelle de l'intensité permet de calculer le vecteur de mouvement $\mathbf{d} = [d_x, d_y]^T$ du point $\mathbf{x}$, à partir des techniques basées sur le gradient. Dans le cas où le mouvement $\mathbf{d}$ est purement translationnel entre les images consécutives, l'équation qui décrit ce mouvement de la scène est donné par :

$$I(\mathbf{x}, t) = I(\mathbf{x} + (\mathbf{d}), t + 1) \tag{3.2}$$

En développant $I(\mathbf{x} + (\mathbf{d}), t + 1)$ au premier ordre et en prenant en compte l'invariance spatio-temporelle, le flot optique d des points est donné par l'équation :

$$\nabla I(\mathbf{x}, t)\mathbf{d} + I_t(\mathbf{x}, t) = 0 \tag{3.3}$$

où $\nabla I(\mathbf{x}, t) = (I_x(\mathbf{x}, t), I_y(\mathbf{x}, t))^T$ est la dérivée spatiale de l'image I dans la direction x et y respectivement, et $I_t(\mathbf{x}, t)$ représente la dérivée partielle temporelle de l'image. Ainsi, les composantes du vecteur $\mathbf{d}$ sont obtenues en utilisant les hypothèses de cohérence spatiale et temporelle locale. Cependant, il est nécessaire de définir une contrainte additionnelle afin d'enlever certaines perturbations des données au moment d'évaluer l'équation 3.1, comme par exemple le bruit sur l'image. Pour ce faire, normalement une notion de similarité ϵ est utilisée, dans laquelle on cherche à minimiser sa valeur résiduelle dans un petit voisinage spatial W autour du point $\mathbf{x}$:

$$\epsilon(\mathbf{d}) = \int\int_W (I_0(\mathbf{x}) - I_1(\mathbf{x} + \mathbf{d}))^2 \omega(\mathbf{x}) d\mathbf{x} \tag{3.4}$$

Dans cette équation, I_0 et I_1 sont, respectivement, la première et la deuxième image consécutive traitée, $\omega(\mathbf{x})$ est une fonction gaussienne qui pondère les valeurs du centre par rapport aux valeurs de la périphérie sur le voisinage W de taille $2w + 1$. La solution à l'équation 3.4, développée dans [9], est la suivante :

$$\mathbf{d} = \mathbf{A}^{-1}\mathbf{b} \tag{3.5}$$

où

$$A = \sum_{x-w}^{x+w} \sum_{y-w}^{y+w} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_x^2 \end{bmatrix}, \; b = \sum_{x-w}^{x+w} \sum_{y-w}^{y+w} \begin{bmatrix} (I_1 - I_2) I_x \\ (I_1 - I_2) I_y \end{bmatrix} \tag{3.6}$$

Une étude descriptive et comparative des diverses approches pour estimer le flot optique à partir d'une séquence d'images a été faite par Baker et Matthews dans [5]. Parmi ces méthodes, la plus couramment utilisée est celle développée par Shi et Tomasi [86] qui, au lieu de suivre tous les points pour obtenir un flot optique dense, se sont plutôt focalisés sur la détection des points les plus intéressants sur l'image ; leur papier s'intitule *Good Features to Track* . Nous avons choisi cette approche car elle semble appropriée à nos objectifs. Dans la suite nous décrirons les conditions de sélection des points et le processus complet pour le suivi des ces points caractéristiques.

3.2.1 Sélection des points d'intérêt

La procédure de flot optique utilisée dans ce travail est basée sur la technique initiale proposée par Lucas et Kanade [68] et ensuite modifiée par Shi et Tomasi [86]. Cette technique est largement connue sous le nom de KLT (Kanade-Lucas-Tomasi). Elle est employée dans la communauté robotique parce qu'elle propose de calculer le flot optique seulement sur les points de plus haute saillance car ce sont les plus faciles à suivre le long de la séquence, ce qui permet de réduire fortement le temps de traitement. Par ailleurs, cette méthode permet de contrôler les occultations et les erreurs d'appariement dans l'image en utilisant un modèle d'évolution affine pour les points appariés dans le temps, modèle qui vérifie que l'on suit toujours le point initialement détecté.

Étant donnée une image initiale I_0 acquise au temps t_0, la sélection des points caractéristiques est faite en analysant les valeurs du gradient spatial de l'image. L'image initiale peut être traitée avec un filtre gaussien avant de démarrer le calcul des gradients afin d'enlever le bruit et de lisser l'image. La figure 3.1 montre une représentation graphique de la procédure mise en œuvre afin de sélectionner les points les plus pertinents.

Deux images de gradient spatial sont obtenues pour deux directions orthogonales de l'image I_0. Ensuite, on construit une matrice de gradient pour chaque point de l'image initiale, autour d'une petite fenêtre de recherche en utilisant les deux images des gradients. Étant données λ_1 et λ_2 les valeurs propres de la matrice de gradient en chaque point, ce point de l'image sera considéré comme point caractéristique si :

$$min(\lambda_1, \lambda_2) > \lambda \tag{3.7}$$

où λ est un seuil, assigné selon les caractéristiques de l'image. Ainsi, tous les points de l'image seront rangés de façon décroissante selon la valeur de sa plus petite valeur propre. Les premiers N_{pts} points caractéristiques de cette liste sont sélectionnés comme les plus significatifs. Nous utilisons cette définition éparse du flot optique parce que nous devons distribuer le temps de traitement avec d'autres fonctions essentielles pour notre approche complète, donc le processus de sélection des points doit être le plus rapide possible.

Après la sélection des N_{pts} meilleurs points à traiter, ils seront suivis pendant un certain nombre d'images consécutives, que nous appelons *temps de pistage* (section 3.2.4). Ainsi, le processus de sélection des points est exécuté à chaque début d'un *temps de pistage* où une nouvelle sélection de N_{pts} nouveaux points est mise en œuvre. Cette nouvelle sélection des points sera conditionnée par l'information fournie par une carte de probabilités qui affecte à chaque position de l'image, une probabilité que se projette en ce pixel, un point acquis sur un objet mobile. L'obtention de la carte de probabilités et sa mise

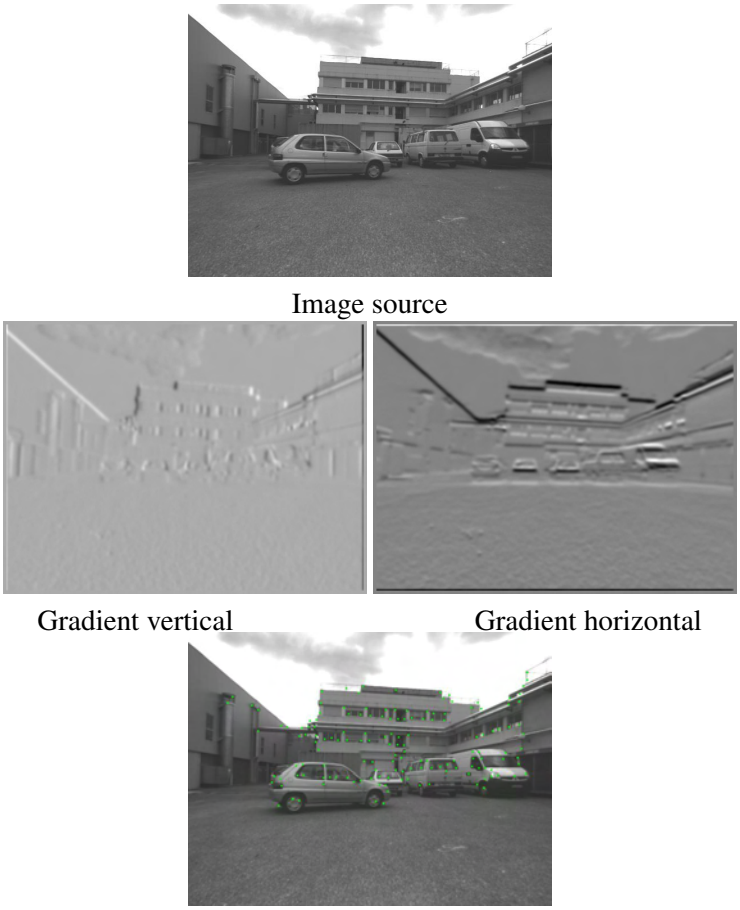


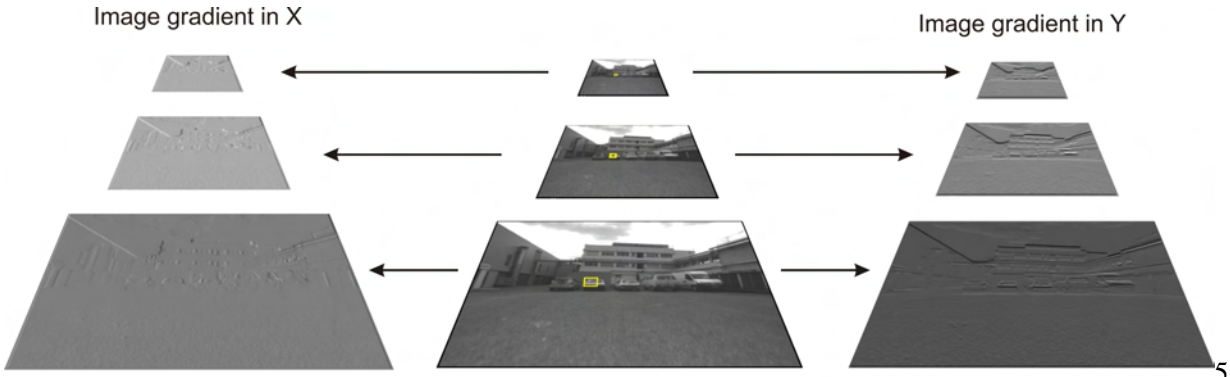
FIGURE 3.1 – Processus pour la sélection des meilleurs points d'intérêt.

à jour seront décrites dans la section 3.4. Cependant, dans le cas où un ou plusieurs points ne sont pas retrouvés dans les images suivantes, cette fonction de sélection sera exécutée pour les remplacer, donc pour ne détecter que le nombre de points perdus afin de conserver un nombre constant de points détectés le long du *temps de pistage*.

3.2.2 Suivi des points d'intérêt

Les positions des N_{pts} points dans l'image suivante de la séquence I_1 sont obtenues en maximisant une mesure de corrélation sur une petite fenêtre, définie dans l'équation 3.4. Le processuss de recherche est accéléré en construisant une pyramide avec des versions à différentes échelles de l'image précédente et de l'image courante, et en obtenant les images de gradients en x et y pour chaque niveau. La figure 3.2 montre une représentation graphique de cette pyramide avec trois échelles et les images correspondantes de gradients. Nous trouvons l'image à son échelle réelle dans le niveau le plus bas de la pyramide, ensuite un sous-échantillonnage permet de construire les échelles suivantes de l'image. D'abord, une fenêtre de taille W est centrée sur la position sous-échantillonnée du point d'intérêt à suivre dans l'image au niveau le plus haut de la pyramide. Ainsi sera obtenue une position grossière du point sur l'image I_1, ensuite, le même point est recherché aux échelles les plus fines en ayant comme position de départ la position grossière issue des échelles précédentes. Finalement la position à l'échelle réelle est plus rapidement

trouvée grâce aux approximations aux plus petites échelles.



518

FIGURE 3.2 – Représentation pyramidale de l'image afin d'accélérer la recherche des points d'intérêt. A gauche et à droite les images de gradient pour chaque niveau de la pyramide, pour les directions x et y respectivement, calculées à partir de la pyramide d'images centrale.

Un point détecté dans I_0 peut ne pas être retrouvé sur l'image I_1, soit parce qu'un certain nombre minimal d'itérations est atteint, soit parce que la condition de similarité n'est pas encore obtenue ou parce que la position du point trouvée sur l'image suivante est en dehors d'une certaine limite de déplacement. Dans ces cas, le point est labellisé comme "not found" (non trouvé), donc un nouveau point doit être ajouté à la liste pour le remplacer. Pour ce faire, la procédure décrite dans 3.2.1 est à nouveau exécutée, uniquement pour remplacer les points non trouvés. Cette procédure ne prend pas trop de temps car pour cette nouvelle image I_1 les images de gradient ont été déjà obtenues. Ainsi, les gradients nécessaires dans la procédure, graphiquement représentée sur la figure 3.1 sont directement obtenus de la procédure de suivi des points. C'est aussi cette caractéristique qui fait le succès de la sélection et du suivi des points proposés par Shi-Tomasi, car les points sont sélectionnés là où on peut facilement les suivre, c'est à dire en favorisant certaines valeurs du gradient des images.

Pour la sélection des points, il y a déjà des positions de l'image qui contiennent des points d'intérêt en cours de suivi. Donc, afin de ne pas choisir une de ces positions, la carte de probabilités (décrite dans la section 3.4) qui mémorise les positions des points suivis au cours du temps, est consultée. Finalement, cette procédure de suivi est itérée sur la séquence obtenue pendant le *temps de pistage* ; les vecteurs de déplacement (le flot optique) sont obtenus pour tous les points caractéristiques initiaux ; leurs vitesses est calculée en se basant sur ces vecteurs de déplacement.

Pour rendre plus robuste le suivi, la rotation, l'échelle et la déformation de chaque point sont pertinemment gérés en calculant les paramètres de transformation spatiaux linéaires correspondants pendant le processus itératif, transformations qui sont décrites dans la section suivante.

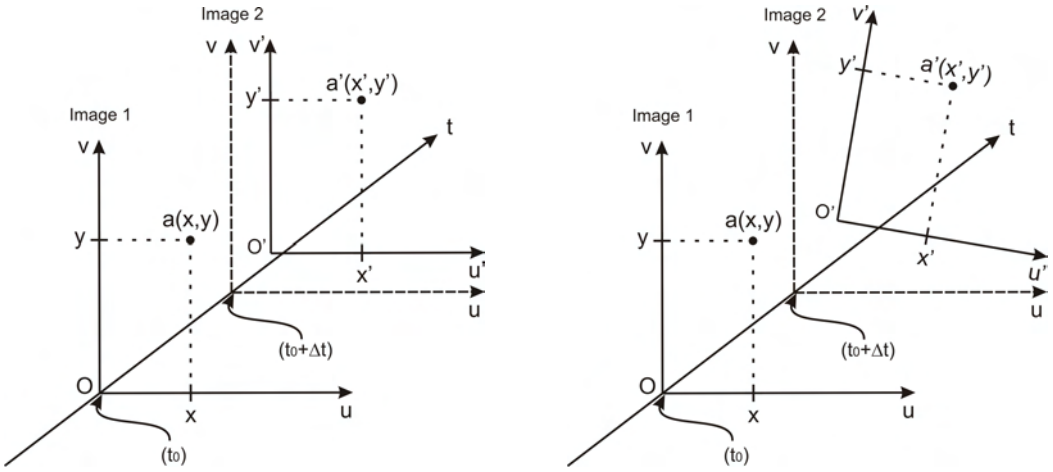
3.2.3 Modèle de translation et de transformation affine

Le choix des propriétés qui caractérisent au mieux les points à suivre devient plus complexe quand l'objet qui porte ces points est partiellement caché ou quand il a une réflexion de lumière sur sa surface. Ce problème a été abordé par Shi et Tomasi dans [86], où les auteurs présentent une méthode pour surveiller la qualité des propriétés des points dans une séquence d'images en utilisant une mesure de dissimilarité. Cette quantité évalue les changements d'apparence des points entre la première et la dernière image d'une séquence, afin d'assurer que l'on suit une cible toujours avec les caractéristiques les plus

proches de celles qu'elle avait au départ. Pour ce faire, ils proposent un modèle fondé sur une transformation affine de l'image.

Le modèle de mouvement de translation d'un point dans l'image est présenté dans l'équation 3.2. Ce modèle suppose un déplacement x et y du point $a(x, y)$ sur l'image au temps t_0 vers l'image au temps $t_0 + \Delta t$, voir figure 3.3a. Avec ce mouvement, le calcul du flot optique $(\bar{x}, \bar{y})$ du point a est donné par :

$$\begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix} - \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.8}$$



(a) Mouvement de translation (b) Translation et transformation affine

FIGURE 3.3 – Mouvement apparent d'un point a de l'image au temps t_0 vers l'image au temps $t_0 + \Delta t$.

Le modèle de translation est largement utilisé car il est simple et rapide. De plus, dans le cas de séquences acquises par une caméra fixe, il est suffisant pour prédire le mouvement des points. Par contre, les images acquises depuis une caméra mobile contiennent, la plupart du temps, des mouvements plus complexes que seulement, un mouvement de translation. Une transformation affine permet d'ajouter une certaine déformation aux points dans la fenêtre de recherche, qui va être plus proche des changements de l'apparence des propriétés dus au mouvement de la caméra. L'équation 3.9 modifie l'équation 3.2 afin de permettre un déplacement plus complexe du point $a(x, y)$, en plus de la translation d. Ainsi, la matrice carrée **A** déforme une région de recherche en conservant les propriétés de parallélisme. La figure 3.3b montre une représentation graphique de cette transformation affine pour un point a du temps t_0 au $t_0 + \Delta t$

$$I(\mathbf{x}, t) = I(\mathbf{A}\mathbf{x} + (\mathbf{d}), 0) \text{ où } \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \tag{3.9}$$

Les paramètres de la matrice **A** ajoutent 4 inconnues de plus, donc l'équation 3.5 passe d'un système de taille 2×2 à un autre de 6×6 avec le vecteur $\mathbf{d} = [a_{11}, a_{12}, a_{21}, a_{22}, d_x, d_y]^T$. De la même manière, le vecteur **b** et la matrice **A** passent respectivement à la dimension 6×1 et 6×6, voir [86]. De cette façon les N_{pts} points d'intérêt détectés sont suivis et validés qualitativement.

Ensuite, notre approche consiste à accumuler les positions et vitesses pour tous les points suivis jusqu'à avoir une quantité d'information qui puisse mettre en évidence le comportement des points caractéristiques. Le nombre d'images utilisées pour accumuler l'information sur le mouvement des points

est essentiel car s'il n'est pas suffisant, les données ne seront pas représentatives et aucun objet mobile ne pourrait être détecté. Dans le cas contraire, si ce nombre est trop élevé, des fausses détections pourraient apparaître dans la méthode de groupement. Cette situation est analysée dans la sous-section suivante où nous décrivons le problème de la perception des objets en fonction des "traces" ou "pistes" laissées dans le temps par ces points d'intérêt : une telle piste est souvent appelée *tracklet* dans la littérature.

3.2.4 Le temps de pistage

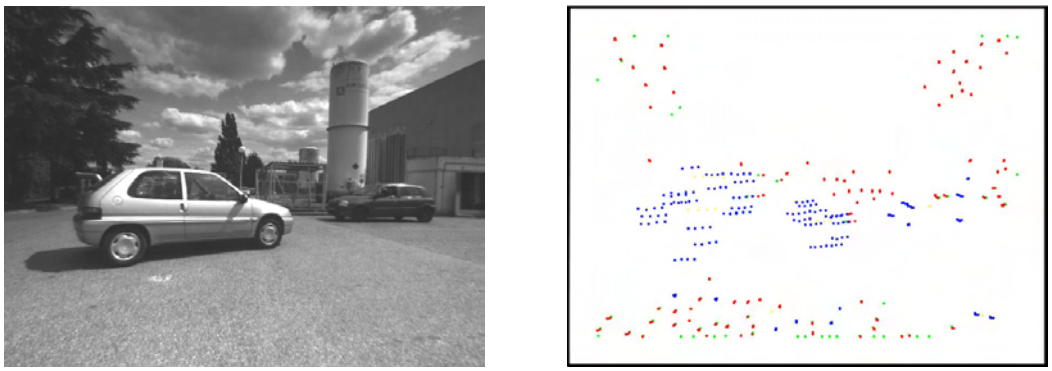
La détection d'un objet mobile sur une séquence d'images a besoin, dans notre approche, d'un suivi continu des points les plus distinctifs de l'objet pendant un certain temps. Cette continuité est clairement représentée sur la figure 3.4, dans laquelle nous accumulons la position des points suivis sur différents nombres d'images consécutives N_{im}. Nous exploitons ici la séquence présentée dans la figure 3.8 qui montre la navigation du robot dans un parking sans aucun objet en mouvement. Alors, le mouvement apparent de tous les points suivis dans la séquence est le résultat du mouvement du robot.

FIGURE 3.4 – Les pistes pour $N_{pts} = 100$ points accumulées à partir du moment où le robot tourne à gauche. Le flot optique accumulé le long de (a) $N_{im} = 8$ images, (b) $N_{im} = 18$ images et (c) $N_{im} = 30$ images.

La figure 3.4a montre la position accumulée de 100 points initialement détectés et suivis pendant 8 images consécutives. Sur cette figure les points d'intérêt semblent statiques puisque pratiquement leur position reste presque la même que celle du départ. Après 18 images (image 3.4b) et sans connaissance initiale à propos de l'environnement, nous percevons un déplacement constant de la position des points, provoqué par le mouvement de la caméra. Cette situation est beaucoup plus détectable si nous augmentons la valeur de N_{im}, par exemple 30 pour l'image 3.4c. En analysant le comportement du flot optique sur les images acquises depuis une caméra mobile, nous déduisons que plus le nombre d'images N_{im} traitées est grand, meilleure sera la perception du déplacement des objets.

La valeur de N_{im} utilisée pour accumuler la position et la vitesse des points caractéristiques est ce que nous appelons le *temps de pistage*. Nous utilisons normalement 5 images, car cela est suffisant pour estimer le mouvement apparent d'un point ; en fait une piste sera composée uniquement de l'information obtenue dans les quatre dernières images car la première sert à détecter les points, et les deux premières sont nécessaires pour calculer leurs vitesses initiales. Dans la figure 3.5, nous présentons un exemple qui montre que le choix d'utiliser 5 images pour accumuler les pistes, est suffisant pour détecter un objet mobile rigide tout en évitant d'avoir des pistes significatives pour les points statiques à cause du mouvement du robot.

L'image 3.5a est la dernière d'une séquence pour un *temps de pistage* de 5 images. La scène présente



(a) (b)

FIGURE 3.5 – Dans (a) : une des cinq images utilisés pour accumuler le flot optique. Dans (b) : la position accumulé des points où les pistes correspondants à la voiture sont visibles.

deux voitures qui traversent le champ de vision du robot, pendant que le robot est lui même en mouvement. La voiture la plus proche du robot à gauche est détectée avec plus de points d'intérêt que celle qui est au fond à droite. Sur la première image, $N_{pts} = 150$ points d'intérêt sont sélectionnés ; puis ils sont suivis sur les 4 images suivantes ; la figure 3.5b montre la position des points accumulés. Les points en bleu représentent les points avec un flot optique plus long. Donc, en analysant ce comportement des points, l'hypothèse qu'ils appartiennent à un objet dynamique est hautement probable. Ainsi, avec un *temps de pistage* relativement petit, nous mettons en évidence les mouvements apparents dûs aux mouvements des objets mobiles dans la scène, et en même temps nous minimisons l'accumulation pour des points statiques de la scène dûs au mouvement du robot.

D'autre part, le choix du *temps de pistage* doit répondre à une autre contrainte sur la minimisation du temps d'échantillonnage pour la détection des objets mobiles. Donc nous essayons de ne pas accumuler beaucoup de données en nous contentant d'une quantité d'information minimale pour la formation des pistes sur les objets mobiles. Cette valeur représente aussi le retard (ou temps de latence) dans la détection des objets, car c'est justement à la fin de chaque *temps de pistage* que le vecteur de données est complet pour commencer la méthode de groupement des points. Ainsi, avec un traitement séquentiel, la fréquence de détection des objets mobiles est fonction du nombre d'images N_{im} choisi pour le *temps de pistage* ; nous verrons en chapitre 5 qu'un temps de pistage plus grand peut être choisi, sans accroître le temps de latence, si on peut paralléliser les traitements.

Dans la section 3.3 nous expliquons la méthodologie de groupement *a contrario* qui permet de grouper les points caractéristiques acquis sur les objets mobiles à partir des vecteurs accumulés le long d'un *temps de pistage*.

3.3 Groupement des points mobiles par la méthode a contrario

La perception visuelle est une fonction complexe qui exige la présence de modèles saillants pour identifier des éléments qui "cassent" la tendance continue ou uniformément distribuée des données. Ce "modèle saillant" est appelé par la méthode *a contrario* comme un "événement significatif" [16]. Les concepts fondamentaux de la méthode de groupement *a contrario* sont issus de la théorie de Gestalt exposée dans [26] et plus en détail dans [27]. De manière générale, la théorie de Gestalt établit que des

groupes d'éléments saillants dans un ensemble, pourraient être formés en se basant sur une ou plusieurs caractéristiques communes de ces éléments. A partir de cette affirmation, la technique *a contrario*, appliquée en vision par Veit et. al. [98], considère un groupe comme significatif si tous ses éléments montrent une distribution différente de celle d'un modèle de base aléatoire déjà établi. Cette approche a été également utilisée dans [83] pour détecter les objets mobiles dans de courtes séquences d'images. De plus, les auteurs proposent un système plus complet, avec estimation des positions 3D des points suivis, ce qui leur permet de mieux détecter les objets mobiles, mais ce qui rend le calcul plus lourd. Par ailleurs ce travail présente des résultats expérimentaux sur des images réelles, acquises depuis une caméra fixe ; par conséquent les problèmes fondamentaux de segmenter les mouvements apparents dûs aux objets de ceux dûs au mouvement de la caméra, ne sont pas considérés dans ce travail.

Contrairement à la plupart des techniques de groupement, par exemple l'algorithme K-means, le nombre initial de groupes n'est pas nécessaire et aucun paramètre ne doit être réglé. C'est un point très important dans notre cadre de travail qui concerne la navigation d'un robot autonome dans un environnement inconnu. Donc nous avons adopté cette méthode *a contrario* pour la détection d'objets dynamiques ; l'algorithme de groupement est décrit par la suite, avant de présenter quelques résultats de simulation.

3.3.1 Description de l'algorithme

L'objectif de la méthode de groupement *a contrario* consiste à grouper les points d'intérêt ayant un mouvement "cohérent" le long d'une courte séquence d'images. Ici le critére de cohérence fait référence à des vecteurs de mouvement (pistes ou tracklets) qui ont des magnitudes et directions à peu près similaires pour tous les points du groupe.

D'abord, la méthode reçoit un ensemble V de vecteurs d'entrée $(x, y, v, \theta|t)$ définis dans R^4, qui contient le flot optique épars accumulé des points d'intérêt dans le temps, selon la procédure décrite dans 3.2. Dans le vecteur V la variable t est rajoutée juste pour indiquer le moment où ces points ont été sélectionnés (début du temps de pistage) ; cela rappelle le caractère temporel des données. Alors, le premier objectif consiste à évaluer quels éléments de V présentent une distribution particulière et contraire à celle établie par le modèle du fond qui sera décrit dans 3.3.3. Pour éviter l'évaluation élément par élément, d'abord un arbre binaire est construit avec les éléments de V, en utilisant la méthode de liaison simple (en anglais "single linkage") pour avoir tous les groupes qui peuvent être formés à partir de ces éléments. La figure 3.6 montre une représentation graphique d'un arbre binaire construit à partir de 8 points. Dans la racine de l'arbre binaire, on trouve le groupe qui intègre tous les éléments de V (8 dans pour le cas de la figure) ; sur les feuilles de l'arbre, on trouve les éléments de façon individuelle, où chaque groupe contient un seul point. Chaque nœud dans l'arbre représente un groupe candidat de points $G(x, y, v, \theta|t) \subset V$ qui sera comparé avec le modèle du fond en utilisant un ensemble de régions préétablies dans R^4. Du fait de cette dimension, ces régions sont des hyper-rectangles dont la taille est fonction des valeurs des points sur chaque dimension. Ces hyper-rectangles sont définis dans la section 3.3.2.

De cette première évaluation, uniquement certains groupes seront considérés comme significaticativement différents du modèle du fond ; ils seront utilisés dans une deuxième étape dite de fusion. Pendant cette étape, on cherche à trouver les groupes qui, étant labellisés comme significatifs, peuvent être encore inclus dans d'autres le long de l'arbre binaire ; ainsi les groupes de taille maximale seront extraits. Les équations exploitées dans la première étape pour juger du caractère significatif d'un groupe, puis dans la deuxième étape, pour la fusion, sont présentées en section 3.3.3. Finalement une recherche des groupes les plus significatifs est faite : on va privilégier les groupes pères plutôt que les groupes fils le long de l'arbre binaire. Par exemple, dans la figure 3.6, les groupes $G''11$ et $G''12$ sont dérivés (sont les fils)

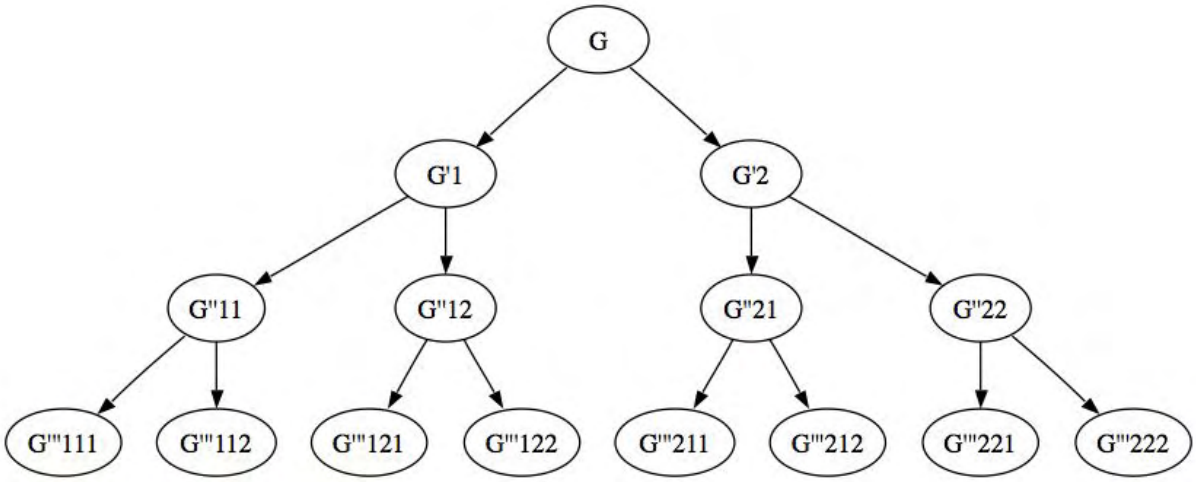


FIGURE 3.6 – Arbre binaire construit par single-linkage, pour 8 points. Tous les points sont trouvés individuellement dans les feuilles de l'arbre, puis en remontant l'arbre ils se regroupent jusqu'à la racine où ils forment un seul groupe G.

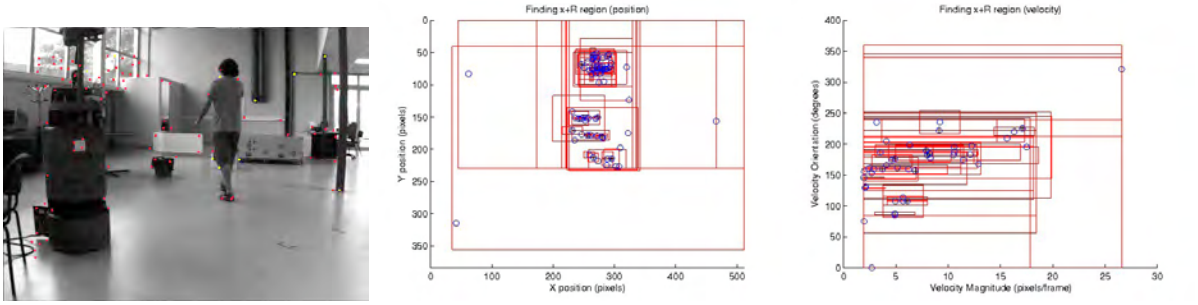
du groupe $G'1$; donc, si les groupes "fils" comme le groupe "père" sont significatifs, nous allons garder uniquement le groupe $G'1$ et éliminer les deux autres qu'il inclue.

3.3.2 L'espace des régions de test dans R^4

D'abord, un ensemble de régions de test doit être établi afin d'évaluer la fonction de distribution de chaque groupe G de points suivis issus de l'accumulation du flot optique, où $G \subset V$. L'ensemble des régions de test, représenté ici par $\mathcal{H}$, est formé par des hyper-rectangles de différentes tailles sur chaque dimension. Ainsi, si la vitesse (longueur d'une piste) pour tous les points dans V reste inférieure à 10 pixels, la taille de la dimension correspondant à la vitesse sur l'hyper-rectangle sera de 10 pixels maximum, car avec cette taille tous les points de V peuvent être inclus dans cette région, sur la dimension "vitesse". La même condition doit être vérifiée pour les autres dimensions ; donc des hyper-rectangles de différentes tailles seront utilisés pour chaque groupe G. La raison d'utiliser des hyper-rectangles répond au besoin de calculer des distribution et de comparer des données dans un espace de caractéristiques en R^4, avec des valeurs d'échelles différentes par dimension.

Une représentation graphique de l'usage des hyper-rectangles est montrée sur la figure 3.7. La figure 3.7a présente dans la première image le temps de pistage avec $N_{pts} = 150$ points d'intérêt. Ensuite ces N_{pts} points sont suivis par la méthode KLT pendant un *temps de pistage* composé de 8 images. La position et la vitesse accumulées des points dynamiques sont représentées dans les images 3.7b et 3.7c, respectivement. Sur ces mêmes figures les rectangles rouges, dessinés séparément dans l'espace des positions et des vitesses car il est impossible de représenter un hyper-rectangle dans R^4, montrent les différentes tailles des hyper-rectangles projetés sur les sous-espaces (X, Y) ou $(magnitude, orientation)$ pour évaluer les groupes G dans l'arbre binaire. L'objectif consiste à trouver l'hyper-rectangle qui contient tous les points dans chaque groupe G de l'arbre binaire en R^4. Pour ce faire, chaque région en R^4 est centrée en chaque point $X(x, y, v_x, v_y) \in G$ jusqu'à trouver la région H_X où $H \in \mathcal{H}$ contient les éléments de G et, en même temps, rend minimale la probabilité de suivre la distribution du modèle du fond. Des tailles différentes d'hyper-rectangles sont utilisées en fonction du domaine de données ; dans nos expériences nous utilisons 20 tailles différentes par dimension. Ainsi, le nombre de régions de test

pour chaque point X est de 20^4.



(a) Image initiale (b) Régions dans le domaine de la position (c) Régions dans le domaine de la vitesse

FIGURE 3.7 – Image (a) : plusieurs objets statiques et une personne traversant la pièce sont perceptibles. Image (b) : les positions X et Y dans l'image représentées dans l'espace bidimensionnel. Image (c) : la magnitude de la vitesse et son orientation représentées dans l'espace bidimensionnel. Les rectangles rouges en b) et c) représentent les régions de test qui doivent contenir tous les points du groupe testé aussi bien en position qu'en vitesse.

Par ailleurs, cet ensemble de régions $\mathcal{H}$ dans R^4 est supposé avoir une distribution identique et indépendante pour chaque composante, comme le modèle du fond qui sera présenté dans la section 3.3.3. Cette propriété de la région permet de calculer la distribution p de chaque région comme le produit de quatre distributions indépendantes, une pour chaque composante dans les données. Ainsi, pour les dimensions correspondantes aux positions des points et à l'orientation de la vitesse, leur distribution sera uniforme parce que la position et la direction du mouvement de l'objet mobile sont arbitraires. C'est à dire qu'aucune information à propos de la position initiale ou de l'orientation de déplacement d'un objet mobile ne sont connus. D'autre part, la distribution de la magnitude de la vitesse est obtenue directement à partir de l'histogramme empirique des données observées. Ainsi la distribution jointe p sera le produit de ces quatre distributions. Ensuite, chaque fois que la région est centrée sur un point différent $X \in G$, sa distribution va changer en fonction des points dynamiques qu'elle contient. D'ailleurs, c'est cette recherche de la meilleure région qui va permettre d'identifier le groupe de test G comme significatif par rapport au modèle du fond qui est obtenu par la méthode de groupement *a contrario*.

3.3.3 Évaluation du modèle du fond

L'espace des régions $\mathcal{H}$ est utilisé pour calculer la probabilité que la distribution de chaque groupe dans l'arbre binaire soit similaire à la distribution d'un modèle pour les objets du fond. Ce modèle propose que tous les pixels correspondant à des points qui sont statiques sur le fond de la scène, aient un mouvement apparent sans direction précise et commune entre eux. En effet, si des pixels ont des mouvements allant "dans tous les sens", il y a très peu de possibilités qu'il s'agisse d'un objet mobile indépendant de la scène. Plus précisément, nous utilisons le modèle du fond proposé dans [98] qui établit une organisation aléatoire des observations, distribuées de façon identique et indépendante et qui suivent une distribution p. Cette nature indépendante des éléments du modèle du fond, se fonde donc sur l'absence de mouvement cohérent, ce qui rend ces éléments certainement pas intéressants à suivre. Ainsi, pour détecter les objets mobiles, tous les nœuds dans l'arbre binaire ainsi que l'espace des régions $\mathcal{H}$ sont analysés afin d'évaluer l'hypothèse suivante :

Hypothèse : tout groupe de pixels qui ne suit pas la distribution aléatoire du modèle du fond est considéré comme un groupe doté d'un mouvement indépendant.

Afin d'obtenir une valeur quantitative de l'évaluation de cette hypothèse, nous utilisons une mesure appelée Nombre de Fausses Alarmes (NFA) pour chaque groupe dans l'arbre binaire. Elle est obtenue par l'équation 3.10 :

$$NFA\left(G\right) = N^2 \cdot \left|\mathcal{H}\right| \min_{\substack{X \in G, \\ H \in \mathcal{H}, \\ G \subset H_X}} B\left(N-1, n-1, p\left(H_X\right)\right) \tag{3.10}$$

Dans cette equation, N représente le nombre d'éléments du vecteur initial des données V, $|\mathcal{H}|$ est la cardinalite des régions et n est le nombre d'éléments dans le groupe de test G. Le terme qui apparaît dans la fonction minimale est la loi binomiale accumulée, cela représente la probabilité qu'au moins n points, en incluant le point $X(x, y, v_x, v_y)$ du centre, soient à l'intérieur de la région H_X, X étant le centre de la région. La distribution p est composée de quatre distributions indépendantes, déjà décrites dans la section 3.3.2. Un groupe G est dit significatif, c'est à dire qu'il correspond possiblement à un objet dynamique sur la scène, si $NFA\left(G\right) \leq 1$.

Ensuite, deux groupes significatifs disjoints au même niveau dans l'arbre binaire pourraient appartenir au même objet mobile. Cette situation peut arriver dans le cas de deux voitures en mouvement dans la même direction ou de plusieurs piétons dans la scène. Dans ces cas, plusieurs régions mobiles peuvent être détectées mais certaines de ces régions peuvent en fait appartenir au même objet. Alors, une deuxième évaluation ne prenant en compte que les groupes significatifs sera effectuée. L'équation 3.11 décrit cette deuxième évaluation :

$$NFA_G\left(G_1, G_2\right) = N^4 \cdot \left|\mathcal{H}\right|^2 \mathcal{T}\left(N-2, n_1-1, n_2-1, p_1, p_2\right) \tag{3.11}$$

où le terme $\mathcal{T}$ représente la loi trinomiale accumulée et est défini par l'équation 3.12 :

$$\mathcal{T}\left(N, n_1, n_2, p_1, p_2\right) = \sum_{\substack{i \geq n_1 \\ j \geq n_2}} \left(\begin{array}{c} N \\ i, j \end{array} \right) p_1^i p_2^j \left(1 - p_1^i - p_2^j\right)^{N-i-j} \tag{3.12}$$

avec le coefficient trinomial donné par :

$$\left(\begin{array}{c} N \\ i, j \end{array} \right) = \frac{N!}{i!j!(N-i-j)!} \tag{3.13}$$

Afin d'obtenir cette nouvelle mesure, les deux groupes de test G_1 et G_2 doivent être disjoints, c'est-à-dire $G_1 \cap G_2 = \emptyset$. Mais ils ne le sont pas forcément. Alors, les deux groupes sont d'abord évalués ensemble comme un unique groupe : $G = G_1 \cup G_2$; une nouvelle région et sa probabilité doivent être retrouvées en utilisant la même procédure et le même ensemble de régions $\mathcal{H}$ que pour les groupes de l'arbre binaire. Avec cette procédure nous calculons la valeur de $NFA(G)$. Ensuite, nous réutilisons l'information déjà obtenue sur la région de chaque groupe (les dimensions et la probabilité) pour calculer $NFA_G(G_1, G_2)$. Dans cette equation, les valeurs de n_1 et n_2 représentent le nombre de points dans

les groupes G_1 et G_2 respectivement, qui ne sont pas dans l'intersection entre les régions G_1 et G_2. Mathématiquement, cette condition est représentée par l'équation suivante :

$$n_1 = |G_1 \backslash (X_2 + R_2)| \; et \; n_2 = |G_2 \backslash (X_1 + R_1)| \tag{3.14}$$

où $(X_2 + R_2)$ est la région R_2 centrée sur le point X_2 avec $X_2 \in G_2$, $(X_1 + R_1)$ a la même signification pour le groupe G_1.

De la même façon, p_1 et p_2 représentent la probabilité de la région du groupe G_1 et G_2 respectivement, hors de l'intersection entre les deux régions. Cette condition est aussi représentée par l'équation suivante :

$$p_1 = p((X_1 + R_1) \backslash (X_2 + R_2)) \; et \; p_2 = p((X_2 + R_2) \backslash (X_1 + R_1)) \tag{3.15}$$

où p_1 est la probabilité de la région R_1 centrée sur le point X_1 sans l'intersection avec la région $X_2 + R_2$, p_2 a la même signification pour le groupe G_2. Les deux mesures 3.10 et 3.11 donnent une valeur quantitative aux groupes dans l'arbre binaire. Les groupes finaux sont trouvés en explorant tout l'arbre binaire et en comparant s'il est plus significatif d'avoir deux objets mobiles G_1 et G_2 ou de fusionner les deux dans un groupe "père" G. Mathématiquement, $NFA(G) < NFA_G(G_1, G_2)$ où $G_1 \cup G_2 \subset G$.

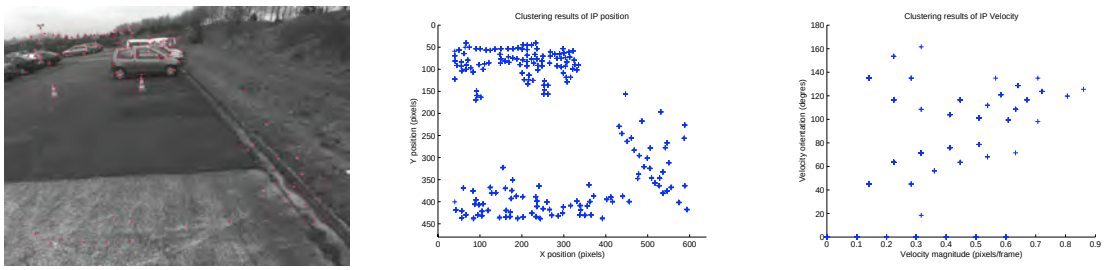
3.3.4 Résultats de la méthode de groupement

La méthode de groupement *a contrario* a été initialement implémentée sur Matlab, afin de tester plus rapidement sa performance sur différents types d'images. Nous présentons trois cas de séquences d'images acquises par une caméra montée sur un robot mobile. Les points d'intérêt accumulés en R^4 sont représentés sur des espaces bidimensionnels séparés, en $pixels$ pour les coordonnées x et y dans l'image, en $pixels/image$ pour la magnitude de la vitesse et en $degrs$ pour l'orientation de la vitesse.

Premier cas : environnement sans objet mobile

La première séquence de résultats montre la scène d'un parking où plusieurs objets fixes sont présents, principalement des voitures. Le robot qui porte la caméra est en mouvement ; le mouvement apparent des pixels est le résultat du mouvement du robot. Dans ce contexte, nous nous attendons à ce que la méthode de groupement de points dynamiques ne trouve pas de groupe cohérent car en réalité il n'y en a pas. La figure 3.8 confirme cette bonne performance attendue de la méthode. D'abord l'image 3.8a montre une des 7 images utilisées pour accumuler le flot optique des points d'intérêt. Ensuite, l'image 3.8b montre la position de tous les points accumulés. L'image 3.8c présente la magnitude de la vitesse et son orientation. Après l'évaluation sur les quatre dimensions (position et vitesse) le résultat est que la distribution de toutes ces données est plutôt conforme au modèle du fond, c'est à dire, aléatoire, donc on ne va pas détecter d'objet mobile dans la scène, et on ne continuera pas à suivre ces points.

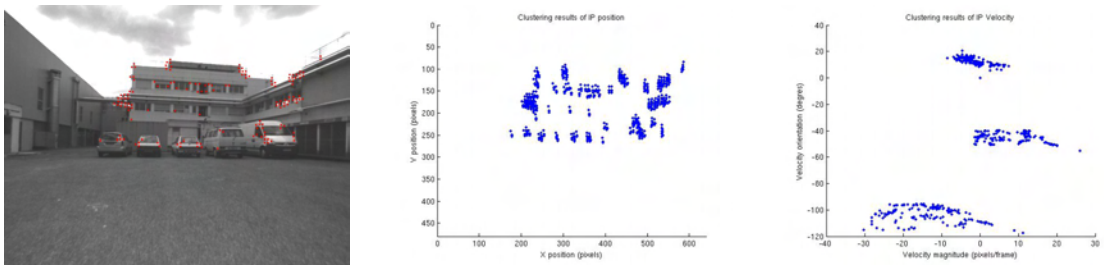
Dans ce premier cas, la vitesse de mouvement du robot est très faible et aucune perturbation sur le chemin du robot n'est présente. Pourtant, dans notre contexte de navigation d'extérieur, très possiblement, le robot trouvera des petites pierres ou certaines discontinuités sur son chemin, ce qui sera directement reflété par des mouvements apparents parasites lors de l'acquisition des images. Donc un test est réalisé dans les conditions où un robot parcourt un parking, mais cette fois nous avons ajouté des mouvements non désirables, comme cela pourrait être le cas si le sol était plus accidenté. Quelques images



(a) Image initiale (b) Groupement des positions (c) Groupement des Vitesses

FIGURE 3.8 – Image (a) : les points rouges représentent les cent meilleurs points intéressants dans l'image. Dans ce cas seul le robot est en mouvement. Les points mobiles dans l'espace R^4 sont représentés dans les deux sous-espaces bidimensionnels pour la position dans l'image (b) et pour la magnitude et l'orientation de la vitesse dans l'image (c). Aucun groupe de points mobiles n'est détecté le long du *temps de pistage* composé de 7 images.

résultantes de cette simulation sont présentées dans la figure 3.9. La première image de la séquence est représentée dans la figure 3.9a. D'abord 150 points sont sélectionnés et suivis pendant 5 images consécutives. Dans ce cas la vitesse du robot est bien supérieure à celle de la première séquence présentée, par conséquent, les points suivis auront une plus grande magnitude de vitesse, même s'ils correspondent à des points du fond. En plus, nous avons ajouté des mouvements perturbants, c'est à dire des mouvements sans direction cohérente. Dans l'image 3.9b on peut apercevoir des traces des mouvements verticaux des points. En réalité, ces points décrivent des mouvements de haut en bas et vice-versa. Ce fait est confirmé en regardant les valeurs de la vitesse et l'orientation des points suivis, extraits depuis les tracklets. Du côté de la magnitude de la vitesse, sa valeur oscille entre -30 et $+30$ pixels par image. La méthodologie pour calculer cette magnitude de la vitesse des points est détaillée dans la sous-section 3.2.3. D'autre part, l'orientation montre 3 directions principales aux alentours de 20°, -40° et -110°. Ces valeurs indiquent une direction aléatoire des points et favorisent finalement la distribution proposée par le modèle du fond de la méthode *a contrario*, donc aucun objet dynamique n'est détecté malgré l'ajout de mouvements parasites dans les images.



(a) Image initiale (b) Groupement des positions (c) Groupement des vitesses

FIGURE 3.9 – Dans cette séquence le robot est en mouvement dans un environnement statique. Des mouvements perturbants sont intentionnellement ajoutés pendant l'acquisition des images consécutives. Malgré cela, aucun groupe de points mobiles n'est détecté depuis les graphiques (b) et (c) malgré la magnitude de la vitesse, supérieure à 20 pixels par image.

Deuxième cas : environnement avec objets mobiles rigides

Étant donnée la bonne performance de la méthode sur des séquences d'images avec des points dynamiques mais sans objets réels en mouvement, nous avons continué nos expérimentations sur la détection des objets mobiles rigides. La figure 3.10 montre une scène où une voiture rentre sur le champ de vue de la caméra de notre robot mobile. D'abord, sur l'image 365 de la séquence, 150 points d'intérêt sont détectés (représentés en rouge sur l'image 3.10a). Ensuite ces points sont suivis le long de 4 images consécutives jusqu'à l'image 369. L'image 3.10c montre en bleu la position sur l'image de tous les points accumulés et en vert le seul groupe de points mobiles identifié comme un objet dynamique. La position de ces points correspond exactement à la position des points sur la voiture qui rentre dans le champ de vue. L'image 3.10d montre la magnitude et l'orientation de la vitesse des points. Les points verts qui correspondent à l'objet détecté, ont tous la même valeur d'orientation puisque l'orientation vaut autour de 0 et 360 degrés, donc ils correspondent à la même direction.

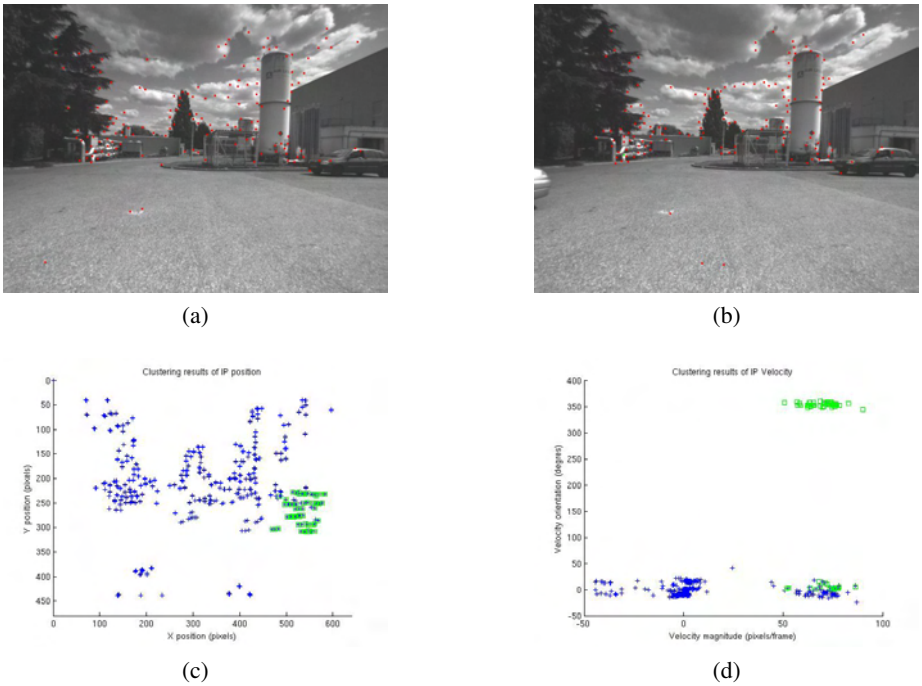


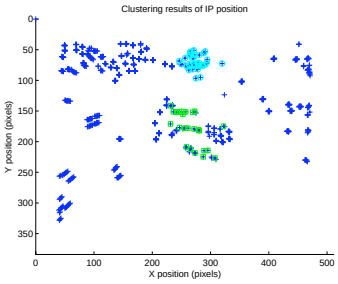
FIGURE 3.10 – En haut : deux des cinq images traitées ; dans ce cas le robot ainsi que la voiture sont en mouvement. En bas : les deux sous-espaces bidimensionnels à gauche pour la position et à droite pour la magnitude et l'orientation de la vitesse. Un seul objet dynamique est détecté (en vert), la voiture qui rentre dans la scène.

Le temps que prend la détection d'un objet dynamique sur l'image est fonction principalement du nombre d'images utilisées pour le suivi de points, 5 dans le cas présenté. Alors, la détection est faite correctement mais c'est justement ce retard, bien qu'indispensable pour la détection des mouvements indépendants et cohérents, qui est le point faible de cette approche, même si dans nos expérimentations, la détection d'un objet mobile rigide ne nécessite jamais plus de 8 images après sa première apparition. Nous trouvons plus de complications pour détecter les objets mobiles non rigides, notamment les piétons. Cette situation est analysée dans la suite.

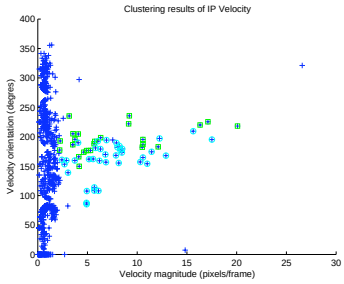
Troisième cas : environnement avec des objets mobiles non-rigides

La détection d'objets dynamiques devient plus compliquée avec la présence de piétons sur la trajectoire du robot. Cette situation a été testée dans un environnement intérieur (figure 3.11) où les conditions d'illumination peuvent être mieux contrôlées. Pour ce test, nous avons sélectionné initialement 100 points d'intérêt, qui ont été suivis pendant 10 images consécutives. La position des points ainsi que les deux groupes des points dynamiques trouvés sont présentés sur la figure 3.11b. Ainsi, deux groupes sont trouvés malgré le fait qu'il y a sur la scène un seul piéton. La tête et les bras de la personne sont identifiés comme un seul objet, représenté en couleur cyan, pendant que les jambes sont détectées comme un autre objet qui apparaît en vert. En analysant ce résultat, nous trouvons effectivement que les points correspondant à la partie haute du corps ont des directions de mouvement différentes de ceux correspondant à la partie basse. D'ailleurs, les positions des deux groupes dans l'image ne sont pas connexes à cause du manque de points d'intérêt sur le tronc de la personne et de la proximité de la personne avec la caméra du robot ; ceci empêche aussi la fusion des groupes.

(a) Image initial



(b) Position dans l'image

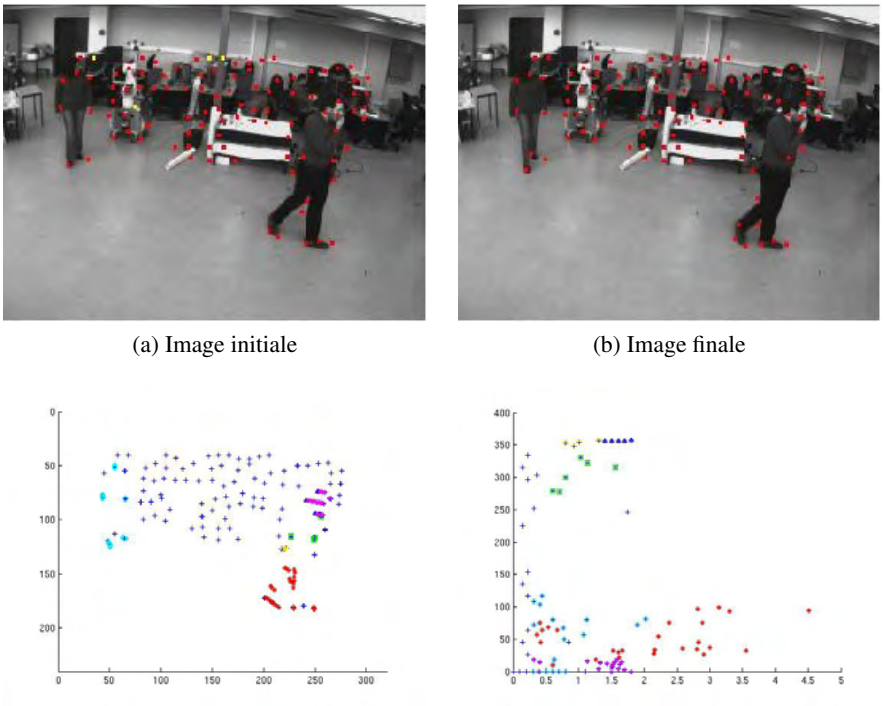


(c) Vitesse

FIGURE 3.11 – Résultats de la procédure de groupement *a contrario* pour les points accumulés le long de 10 images pendant la navigation en intérieur d'un robot. En bas, les deux sous-espaces bidimensionnels. Deux groupes sont détectés : le groupe en couleur cyan représente le mouvement de la tête et des bras de la personne, le groupe en vert représente le mouvement des pieds et des jambes.

Pour confirmer ce comportement, nous avons fait d'autres tests, toujours dans le contexte d'un environnement intérieur, mais cette fois sans point mobile produit par le mouvement du robot. Alors, c'est sur une séquence acquise depuis une caméra fixe que nous avons testé à nouveau la détection de piétons. La figure 3.12 montre l'image initiale et l'image finale de six images consécutives traitées pour l'accumulation du flot optique. Dans ces images nous observons sur le plan arrière de l'image, une personne qui marche dans le sens de l'axe optique de la caméra (vers la caméra) et une deuxième personne sur le premier plan de l'image qui marche en direction orthogonale à l'axe optique. Dans la figure 3.12c la pre-

mière personne est complètement détectée, ce qui confirme que le mouvement parallèle à l'axe optique est beaucoup plus facile à détecter. D'autre part, la deuxième personne est détectée comme 4 groupes dynamiques différents. En analysant la vitesse de ces points, il y a de fortes différences dans la magnitude de la vitesse, notamment entre celles des points sur les pieds (en rouge) et celles des points sur les bras que la personne colle à son corps. Dans tous les cas, même si ces points avaient eu des mouvements avec à peu près la même orientation, cela n'aurait pas permis de fusionner le groupe dynamique des pieds avec celui de la tête, parce que, certes, ils ont une vitesse et une orientation de mouvements similaires mais en position, ils ne sont pas suffisamment proches.



(a) Image initiale (b) Image finale

(c) Position dans l'image (d) Vitesse

FIGURE 3.12 – Résultats sur une séquence acquise depuis une caméra fixe. En haut : les première et la dernière images traitées où 2 personnes sont en mouvement dans la scène. En bas : les deux sous-espaces bidimensionnels. Plusieurs objets dynamiques sont détectés : le groupe en couleur cyan représente une des personnes détectée comme un objet mobile. La deuxième personne est détectée comme plusieurs objets mobiles, notamment avec les pieds et la tête séparés du corps.

Une caractéristique à noter dans tous les résultats expérimentaux présentés est que le nombre d'images utilisées pour accumuler les points d'intérêt est différent. Nous avons trouvé que pour détecter des objets rigides, le traitement de 5 images est suffisant. Par contre, dans le cas d'objets non-rigides, plus d'images sont nécessaires afin de bien représenter les pistes, notamment le cas des piétons est plus compliqué à cause du mouvement de va-et-vient des pieds.

Une fois que la tâche de groupement est achevée, un nouveau cycle de sélection-suivi-groupement des points KLT commence le long d'un *temps de pistage*. Au début de chaque nouveau cycle, la sélection de N_{pts} nouveaux points d'intérêt différents de ceux choisis dans le cycle précédent est faite. Une carte

de probabilités dans l'image est mise à jour lors du processus du suivi des points afin de choisir ces nouveaux points d'intérêt sur les zones les plus représentatives de l'image. L'idée générale de cette carte et sa mise en œuvre est décrite dans la section 3.4.

3.4 Carte de probabilités

Dans notre approche pour la détection et le suivi d'objets, il est important à chaque cycle d'ajouter de nouveaux points, d'une part pour enrichir lesobjets mobiles déjà détectés et d'autre part qui permettent de détecter de nouveaux objets entrants. Initialement, dans un contexte d'environnement inconnu, les points caractéristiques sont répartis dans toute l'image sans privilégier aucun endroit. Cependant, après le processus d'initialisation, la sélection des points dans la suite de la navigation du robot ne doit plus être faite de façon uniforme car les données issues des images précédentes apportent déjà une information sur le contenu de la scène. Donc, une carte de probabilités sur les positions spatiales et temporelles des points suivis jusqu'alors est construite afin d'indiquer les endroits les plus favorables pour la sélection de nouvelles caractéristiques. Cette carte est traitée comme une sorte de grille d'occupation, dans le sens où les plus hautes probabilités représentent les endroits de l'image qui ne sont pas importants pour chercher un nouveau point, par exemple les positions occupées par un autre point. Ainsi, les endroits dans l'image avec les valeurs de probabilité les plus basses de la carte, seront d'abord utilisés pour chercher de nouveaux points.

3.4.1 Initialisation de la carte

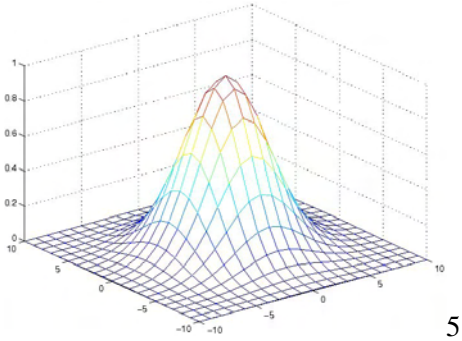
La carte de probabilités d'occupation que nous proposons garde les coordonnées des points KLT dans l'image en leur assignant une probabilité et une petite cellule centrée sur chaque point d'intérêt $X = (x_i, y_i)$. Nous avons utilisé une fonction de distribution gaussienne bidimensionnelle présentée dans l'équation 3.16 pour assigner la valeur de probabilité à chaque position de la cellule :

$$\mathcal{G}(x_i, y_i|\mu_x, \mu_y, \sigma_x, \sigma_y) = \frac{1}{2\pi\sigma_x\sigma_y} \cdot e^{-\frac{1}{2}\left(\left(\frac{x_i-\mu_x}{\sigma_x}\right)^2+\left(\frac{y_i-\mu_y}{\sigma_y}\right)^2\right)} \tag{3.16}$$

où $x_i \in \mathbf{X}$ et $y_i \in \mathbf{Y}$ définies par 3.17 :

$$\mathbf{X} = \{x|\ |x - x_i| \leq r\}\ ;\ \mathbf{Y} = \{y|\ |y - y_i| \leq r\} \tag{3.17}$$

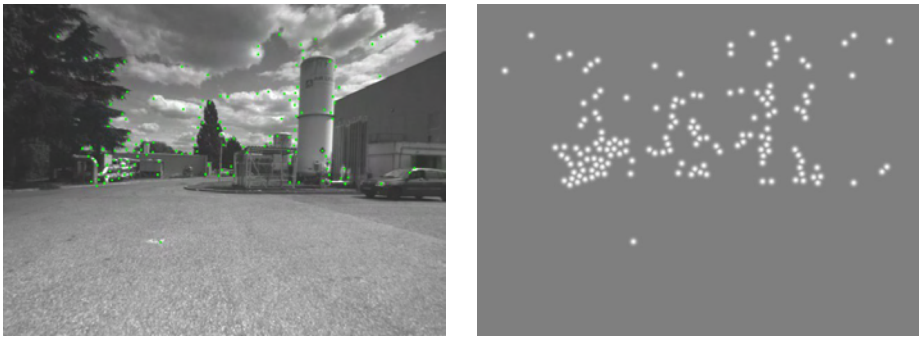
Ainsi, toutes les positions dans l'image contenues par les vecteurs **X** et **Y** qui satisfont la condition établie dans l'équation 3.17 (centré sur le point (x_i, y_i)), seront donc modifiées en accord avec cette fonction gaussienne. Les paramètres μ_x et μ_y représentent la moyenne de la distribution dans une cellule de taille $r \times r$, nous leurs assignons les valeurs des coordonnées du point d'intérêt (x_i, y_i) dans les deux directions. Ensuite, les paramètres σ_x et σ_y qui représentent l'écart-type dans la cellule sont établis à la valeur supérieure résultant de la division $\frac{r}{3}$. Étant donné que la taille des cellules est la même pour tous les points le long de la séquence, les valeurs de l'écart-type sont fixes. Ainsi, la distribution sur chaque position d'un point d'intérêt de l'image aura la forme présentée dans la figure 3.13 pour une cellule de taille $r = 21$. Avec cette fonction nous établissons la valeur de probabilité la plus haute au centre de la cellule, c'est-à-dire à la position du point d'intérêt, ensuite cette probabilité diminue à fur et à mesure que nous nous éloignons du centre. Cela veut dire, que même si les positions autour du point d'intérêt X ne sont pas occupées, elles sont considérées comme telles, du fait de leurs proximités avec un point d'intérêt.



518

FIGURE 3.13 – Graphique de la fonction gaussienne bidimensionnelle définie par l'équation 3.16 pour une cellule de taille 21×21.

La figure 3.14 montre le moment de l'initialisation de la carte de probabilités pour la séquence d'images présentée dans la figure 3.10. L'image de gauche montre en vert les 150 points initialement détectés par la méthode présentée dans la section 3.2.1. Cependant, avant de traiter l'image suivante, l'algorithme envoie les positions de ces points à la carte de probabilités pour leur donner une valeur de probabilité d'occupation, tout le reste des positions de l'image auront une même valeur de probabilité. Dans l'image de droite de la figure 3.14, cette valeur uniforme de probabilité est représentée en gris, les cellules établies autour des points sont représentées en blanc et indiquent une valeur de probabilité d'autant plus haute que la cellule est occupée. La carte de probabilités a la même taille que les images de la séquence et ses valeurs de probabilité sont calculées en suivant un modèle d'évolution proposé dans la suite.



(a) (b)

FIGURE 3.14 – L'image (a) est la première image traité de la séquence sur la figure 3.10, en vert les 150 points d'initialisation détectés. L'image (b) est une représentation graphique de l'état initial de la carte de probabilités, en blanc les cellules centrées sur les points d'intérêt.

3.4.2 Modèle d'évolution de la carte de probabilités

D'abord, nous mettons en évidence 3 valeurs de probabilité les plus représentatives dans les cellules : $p_o = 1.0$, $p_e = 0.5$, et $p_v = 0$. La valeur de probabilité d'une cellule de la carte sera obtenue dépendamment de l'état précédent de la cellule entière ou de la position du point sous évaluation, à chaque image traitée. Nous considérons trois états pour les cellules : *occupée*, *vide* et *équiprobable*. *Équiprobable* est

l'état d'une position ou d'une cellule qui n'a jamais été occupée depuis l'initialisation de l'algorithme. Au début, toutes les positions de l'image ont une probabilité équiprobable, c'est-à-dire, sans connaissance *a priori* avec une probabilité p_e. Ensuite pour le temps $t \geq 0$, *occupée* est l'état d'une cellule centrée sur un point d'intérêt. À partir du temps $t \geq 1$, *vide* est l'état d'une cellule qui contenait un point d'intérêt dans une image précédente et actuellement le point est localisé ailleurs.

Procédure :

1. Avant de commencer la procédure de mise à jour, la carte de probabilités est initialisée à p_e dans toutes ses positions (x, y).
2. À l'image initiale (temps $t = 0$), les coordonnées (x_i, y_i) des points KLT détectés deviennent les centres des cellules *occupées* et leurs valeurs correspondantes dans la carte sont données par l'équation 3.18. Pour ce cas d'initialisation, la valeur de la fonction α est p_e, cela implique que l'état précédent de tous les points dans l'image est équiprobable. Ainsi, la valeur de p pour chaque élément de la cellule i ($x \in \mathbf{X}_i$ et $y \in \mathbf{Y}_i$) sera donné par la fonction gaussienne plus la valeur constante p_e. La valeur de probabilité pour toutes les positions de l'image dans la carte, en dehors des points d'intérêt et ses cellules, est de p_e car elles conservent sa valeur d'initialisation dans la carte.
3. Ensuite, une cellule devient vide en fonction des déplacements des points d'intérêt à chaque itération : si un point suivi laisse la cellule qu'il utilisait au temps précédent $t - 1$, sa probabilité changera au temps actuelle t, en plus cette cellule est étiquetée comme *vide* lors de la mise à jour des points. La nouvelle cellule dans laquelle le point suivi est trouvé au temps t devient maintenant une cellule *occupée*.

   Ainsi, à l'image t_n ($n \geq 1$), la mise à jour de la probabilité est faite en deux étapes. D'abord nous calculons la probabilité pour les cellules centrées sur des points précédents, $(x_i, y_i)_{t-1}$. Pour ce faire, nous utilisons l'équation 3.18 pour toute cellule $(\mathbf{X}, \mathbf{Y})$ centrée sur le point $(x_i, y_i)_{t-1}$ (cellule déjà étiqueté comme *vide* au temps t). Donc la valeur de α est donnée par le deuxième cas de l'équation 3.19 et la valeur de $\mathcal{G}(x_i, y_i)_{t-1}$ est nulle car il s'agit d'une cellule qui n'est plus occupée. Nous analysons d'abord les positions de ces cellules parce que sa probabilité ne doit pas rester comme celle d'une cellule *occupée*, mais principalement parce que la probabilité de la nouvelle cellule du point d'intérêt sera modifiée par cette valeur si elles sont à proximité. Ensuite la deuxième étape consiste à mettre à jour la probabilité sur l'actuelle position des points. Alors, l'équation 3.18 est utilisée à nouveau avec le premier cas de l'équation 3.19.
4. La procédure de l'étape 3 est exécutée sur plusieurs images consécutives, spécifiquement pendant la durée de $2 \times$ *temps de pistage*. Après deux *temps de pistage*, la carte est réinitialisé (procédure du point 1) pour éviter que la carte garde pour longtemps les mêmes valeurs de probabilité en certains endroits de l'image, alors le processus commence à nouveaux. Si certaines positions actuelles et précédentes de la carte correspondent à des objets mobiles dans la scène au moment de la réinitialisation, donc sa probabilité restera la même et la valeur d'équiprobabilité sera donnée au reste de positions.

La figure 3.15 représente des résultats simulés de l'évolution de la carte de probabilités pour un point suivi le long de cinq images successives. Une cellule de taille 11×11 centrée sur le point (x, y) est utilisée. La valeur de cette cellule est donnée par la fonction de distribution gaussienne bidimensionnelle de l'équation 3.18. L'effet inverse de la fonction gaussienne, qui donne le deuxième cas de l'équation 3.19, est appliqué à la position précédente du point. Cet effet inverse appliqué aux positions précédentes est clairement identifié dans la figure 3.15. Avec cette carte de probabilités nous privilégions l'hypothèse que les nouveaux points d'un objet qui rentre dans la scène apparaîtront derrière les positions actuelles

des points détectés. Alors, nous assignons la probabilité la plus bas $p_v \approx 0$ ceux qui indique des bonnes positions pour trouver des nouveaux points.

$$p(x, y, t) = \alpha(x, y)_t + \mathcal{G}(x, y|\mu_x, \mu_y, \sigma_x, \sigma_v) \tag{3.18}$$

où la fonction $\alpha(x, y)$ représente la probabilité précédente dans la cellule en fonction de son état, donnée par l'équation :

$$\alpha(x, y)_t = \begin{cases} p_e & \text{si Cell}(\mathbf{X}_i, \mathbf{Y}_i) = \textit{occupée} \\ p_o - p(x, y)_{t-1} & \text{si Cell}(\mathbf{X}_i, \mathbf{Y}_i) = \textit{vide} \end{cases} \tag{3.19}$$

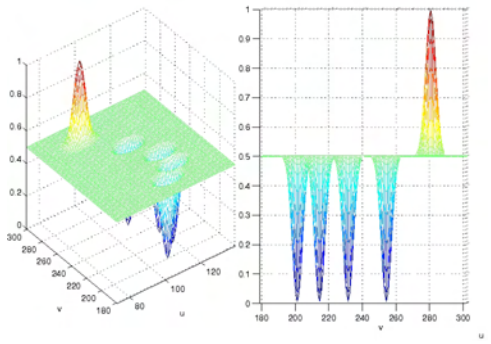


FIGURE 3.15 – Évolution de la carte de probabilités pour un point suivi pendant 5 images consécutives. Les formes inverses représentent la probabilité lors des quatre images précédentes, la forme gaussienne représente la position du point actuel.

3.5 Enrichissement du modèle des objets mobiles dans le temps

Les cas les plus communs qui se présentent dans notre thématique de navigation dans un environnement inconnu sont quand l'objet mobile rentre dans la scène. La méthode de sélection des points va trouver des points sur l'objet, mais normalement, pour le cas d'objets rigides et grands, ces points seront localisés seulement sur une partie de l'objet. Après, les groupes dynamiques résultants de la méthode de groupement *a contrario* contiendront seulement un certain nombre de points qui, pour le moment, ne seront utilisés que pour initialiser l'objet dynamique. Cependant, nous savons que ces points ne sont pas suffisants pour modéliser notre objet car même s'ils sont les plus représentatifs, ils ne sont pas distribués sur tout l'objet. Ainsi, l'algorithme proposé a besoin d'une méthodologie qui puisse permettre d'incrémenter le nombre des points et en même temps de les identifier sur d'autres parties de l'objet où il n'y en a pas. Donc, dans l'intérêt d'enrichir notre modèle de l'objet, une règle de fusion qui évalue les positions des différents objets détectés le long du temps, est présentée dans cette section avec des résultats sur des images avec objets rigides et non rigides.

3.5.1 Description du problème

La figure 3.16 représente les objets mobiles détectés après deux *temps de pistage* consécutifs de 5 images chacun. Le rectangle jaune montre le groupe de points dynamiques détecté dans le premier *temps de pistage* et l'ellipse rouge indique les différents groupes dynamiques résultants du deuxième *temps de pistage*, donc les résultats actuels. Au moment de la détection des points dans le rectangle jaune, nous avons détecté uniquement la partie frontale de la voiture. Sur la figure 3.16 nous sommes déjà au temps

$t + 5$ depuis que ce premier objet a été détecté ; un deuxième objet apparait, nous ne savons pas s'il correspond en fait à une autre partie du premier objet. Il est donc nécessaire d'analyser si les deux groupes font partie du même objet dynamique.

FIGURE 3.16 – Cette image montre les résultats de la méthode de groupement le long de deux temps de piste. Dans le rectangle le groupe dynamique détecté lors du premier *temps de pistage*, dans l'ellipse les résultats lors du deuxième *temps de pistage*.

D'abord, les deux objets sur la figure n'ont pas la même vitesse car ils ne sont pas issus du même processus. C'est à dire que la moyenne de vitesse de l'objet dans l'ellipse est obtenue à partir du flot optique de chaque point. Par contre, pour l'objet dans le rectangle, à partir du moment où il a été détecté comme objet mobile, sa vitesse a été estimée par un filtre de Kalman et ensuite corrigée par la véritable position des points (voir la section 4.4). Malgré cela, les deux objets peuvent être comparés en vitesse et orientation parce que, finalement, leurs vitesses sont calculées en se basant sur le déplacement des pixels de l'objet.

Afin d'enrichir le modèle initialement détecté, nous avons mis en œuvre un processus d'évaluation parmi les groupes de points déjà présents dans la scène et ceux qui sont détectés dans l'image courante. Cette méthode de fusion ainsi qu'une analyse des résultats obtenus sont présentées dans cette section.

3.5.2 Règle d'évaluation pour la fusion des objets mobiles

La tâche de fusion d'objets dans le temps est exécutée seulement si des objets mobiles ont déjà été détectés. Nous définissons O comme l'ensemble total des M objets dynamiques, détectés à n'importe quel instant, et qui serons évalués. Donc, $O = O_T \cup O_C$ où O_T contient $(1, 2, ..., k)$ objets mobiles antérieurement détectés (et suivis par le filtre de Kalman qui sera décrit dans le chapitre 4), et O_C qui contient $(1, 2, ..., l)$ nouveaux groupes dynamiques. Les éléments compris dans le groupe O_C peuvent être interprétés comme de nouveaux objets dynamiques, qui apparaissent pour la premier fois dans la scène, ou comme une partie d'objets déjà détectés. Pour chaque objet dans O, son vecteur de vitesse est modélisé par la moyenne des ses composantes de vitesse en X et Y, représentés par μ_{v_X} et μ_{v_Y}, respectivement. Par exemple, l'objet dynamique présenté dans le rectangle jaune de la figure 3.16 est formé par 8 points d'intérêt (en bleu), c'est la moyenne de la vitesse de tous ces points qui définira la valeur de μ_{v_X} et μ_{v_Y} pour cet objet. La même procédure est considérée pour le reste des objets dans O. Donc, nous utilisons ces valeurs pour modéliser les objets à fusionner, modèle nécessaire lors de l'évaluation de l'équation 3.20.

$$\min_{\substack{i,j \in M, \\ i \neq j, \\ O_i, O_j \subset O}} \left(\begin{bmatrix} s(\mu_{v_X}(O_i), \mu_{v_X}(O_j)) \\ s(\mu_{v_Y}(O_i), \mu_{v_Y}(O_j)) \end{bmatrix} \right) < \begin{bmatrix} d_{v_X} \\ d_{v_Y} \end{bmatrix} \tag{3.20}$$

Nous évaluons la mesure de similarité s entre les deux modèles de vitesse des objets de test. Cette mesure est calculée pour chaque objet dans O, en considérant un premier objet et en le comparant avec tout le reste des objets dans O. Les paramètres d_{v_X} et d_{v_Y} sont deux constantes qui représentent le seuil d'écart parmi les vitesse des objets évalués, la valeur de ces deux constantes est de 1 pixel. Cette valeur est choisie en accord avec le seuil établi pour la sélection des caractéristiques mobiles dans le processus de KLT décrit dans 3.2.1. Avec cette valeur on arrive à conserver le meilleur compromis entre le seuil des points mobiles dans le module KLT et la tendance de l'écart attendue des vitesses des différents objets dans la scène. Si cette première évaluation de similarité est satisfaisante avec l'équation 3.20, alors une deuxième validation est mise en œuvre où on vérifie si les zones des objets identifiés sont suffisamment proches ou même encore si elles s'intersectent. Grâce à cette évaluation nous évitons de fusionner des objets avec la même vitesse et direction de mouvement mais qu'il peut forcement s'agir de deux objets différents.

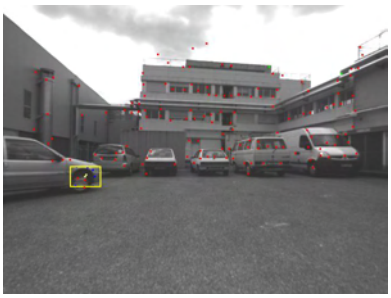
L'évaluation des groupes en O est réalisée à la manière d'une liste chainée, où les groupes fusionnés sont enlevés de O et ajoutés comme un nouvel objet à la fin de la liste avec, évidemment, un nouveau modèle de vitesse. Cette stratégie permet de fusionner les objets détectés auparavant et de former un modèle plus riche.

Dans la suite, nous présentons les résultats de cette détection incrémentale pour un objet mobile rigide et un autre objet mobile non-rigide pendant la navigation du robot.

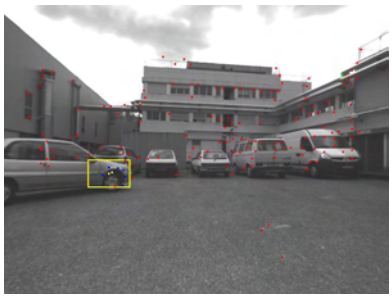
3.5.3 Résultats de la fusion

Afin d'exposer les résultats de notre approche de fusion temporelle d'objets dynamiques, nous avons repris la séquence présentée dans la figure 3.9, au moment où une voiture croise le chemin de navigation du robot. La figure 3.17 montre les résultats de 5 *temps de pistage* consécutifs, chacun traite 5 images afin d'accumuler le flot optique de 150 points d'intérêt. Dans l'image 3.17a, seule la partie frontale de la voiture est initialement détectée. Ensuite, dans 3.17b la taille de ce groupe est élargie grâce à la fusion avec un autre groupe de points d'intérêt détectés au niveau des vitres qui est apparu lors du deuxième *temps de pistage*. La fusion entre ces deux groupes est possible en raison de leur proximité, ce qui garantit la même vitesse et orientation des deux groupes. D'autres groupes de points dynamiques sont détectés sur la voiture dans 3.17c et 3.17d mais pour le moment, ils sont détectés comme des objets indépendants. Ces groupes ne peuvent pas être fusionnes parce que, tout d'abord, la position des objets n'est pas très proche et par conséquence, la vitesse mais surtout l'orientation ne pas sont suffisamment similaires pour former un seul objet dynamique avec ceux détectés au deuxième *temps de pistage*. Au cinquième *temps de pistage*, l'algorithme a détecté 3 groupes comme objets mobiles, en plus un autre groupe issu du cinquième *temps de pistage* apparaît sur la partie arrière de la voiture, ainsi l'équation de fusion 3.20 est évaluée pour quatre groupes dynamiques. Finalement un seul objet dynamique est détecté avec succès sur la scène (image 3.17e).

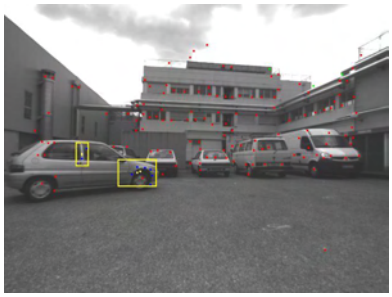
Nous avons aussi évalué le processus de fusion pour une séquence d'images contenant un objet mobile non-rigide. Nous avons déjà fait allusion dans la sous-section 3.3.4 à la complexité de cette tâche, due, dans le cas de piétons, aux mouvements des pieds, très différents de ceux de la tête. Cependant dans la figure 3.18 nous avons traité la même séquence que dans la figure 3.11, sauf que nous avons réduit le temps de piste à 5 images avec plus de points, 150 dans ce cas. Dans l'image 3.18a seul un objet mobile est détecté au niveau de la tête de la personne. À la fin du deuxième temps de pistage, sur la figure 3.18b, un autre objet est détecté au niveau des jambes. Ces deux objets sont très distants, donc l'information n'est pas suffisante pour les fusionner. Dans l'image 3.18c un troisième objet est détecté mais toujours aucun objet ne peut être fusionné. L'image 3.18d montre la première fusion d'objets au niveau du corps



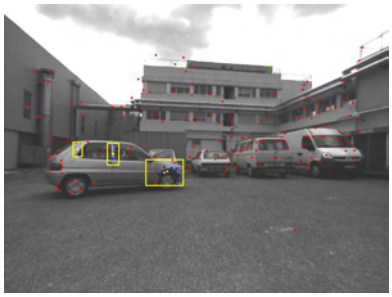
(a) À la fin du premier *temps de pistage*



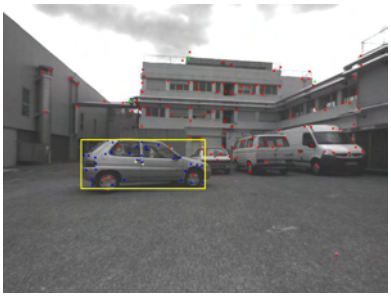
(b) À la fin du deuxième *temps de pistage*



(c) À la fin de troisième *temps de pistage*



(d) À la fin de quatrième *temps de pistage*



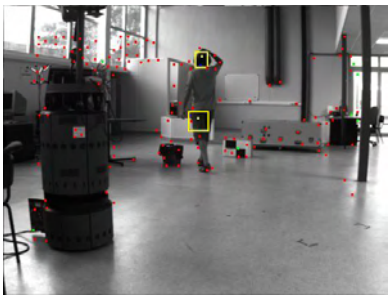
(e) À la fin de cinquième *temps de pistage*

FIGURE 3.17 – Objets dynamiques détectés à chaque fin de *temps de pistage*. Un seul objet de mouvement rigide est détecté après 5 *temps de pistage* grâce à la fusion des objets dynamiques antérieurement détectés sur la voiture.

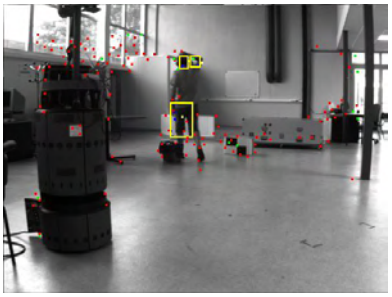
mais il y a toujours un deuxième objet au niveau de la tête. Finalement à l'image 3.18e et après 5 *temps de pistage* un seul objet est détecté sauf que les jambes ne seront jamais incluses dans le modèle à cause du mouvement de va-et-vient qui ne correspond pas au mouvement du corps.

La méthode proposée de détection d'objets mobiles présente un inconvénient lorsque la scène contient des objets mobiles non-rigides. Par exemple dans la figure 3.18 la personne est détectée pour la première fois après 15 images depuis qu'elle est rentrée dans le champ de vue de la caméra. Cela représente un grand retard au niveau de la détection, en plus, seulement une très petite quantité de points d'intérêt sont détectés au niveau de la tête. Un nombre suffisant de points sont détectés le long du corps de la personne,

(a) À la fin du premier *temps de pistage*

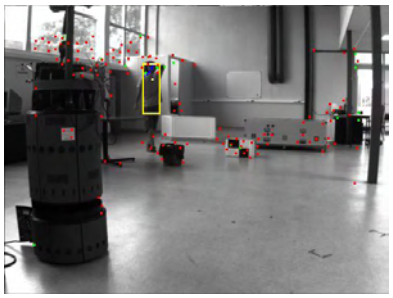


(b) À la fin du deuxième *temps de pistage*



(c) À la fin du troisième *temps de pistage*

(d) À la fin du quatrième *temps de pistage*



(e) À la fin du cinquième *temps de pistage*

FIGURE 3.18 – Objets dynamiques détectés à chaque fin de *temps de pistage*. Un seul objet de mouvement non rigide est détecté après 5 *temps de pistage* grâce à la fusion des objets dynamiques antérieurement détectés.

mais, ils montrent de fortes différences en magnitude et en orientation sans possibilité de fusionner en un seul groupe. Pour cette raison, la méthode de groupement ne trouve pas des groupes dynamiques et il faut attendre au *temps de pistage* suivant pour essayer de rattraper des points avec une vitesse plus similaire, ce qui produit le retard en la détection.

3.6 Conclusion

Ce chapitre a présenté la détection et le suivi des points d'intérêt, puis le regroupement en ensembles de points dynamiques qui appartiennent au même objet physique de la scène. Les points de chaque groupe sont connexes et ont un mouvement cohérent non aléatoire qui contraste avec le mouvement des points qui sont dans l'arrière plan. Le détecteur de points d'intérêt est basé sur l'approche de Shi-Tomasi ; ensuite la vitesse de ces points est obtenue à partir de leur suivi par le tracker KLT (Kanade-Lucas-Tomasi). Etant donné que cet algorithme ne détecte que les points les plus représentatifs, cela permet de diminuer le temps de calcul en assurant une performance d'exécution plus proche de la fréquence de la séquence vidéo.

Afin de contrôler cette sélection des caractéristiques éparses dans chaque image, nous avons proposé de construire une carte de probabilités qui prenne en compte les positions précédentes des points d'intérêt pour estimer les endroits les plus probables où trouver de nouveaux points mobiles. Cette carte est essentielle dans la détection incrémentale des objets mobiles car au lieu de détecter des points n'importe où, il est très important de chercher d'abord sur les positions les plus probables où il peut y avoir un point mobile. La dernière section montre l'exemple d'une séquence où après quelques images, plusieurs groupes de points mobiles sont détectés sur une voiture qui rentre dans le champ de vue du robot : mais ces groupes sont proches et ils ont des mouvements similaires donc ils sont fusionnés et une détection intégrale de la voiture est possible.

Jusqu'à maintenant, la détection, le suivi et le groupement des points mobiles ont été réalisés en utilisant l'information acquise par une seule caméra embarquée sur le robot, sans connaissance a priori sur l'environnement de navigation et les objets qu'il contient. Ainsi, dans le chapitre suivant nous allons voir comment utiliser le résultat de cette approche de groupement pour construire un modèle stable de l'objet détecté et pour mieux le caractériser.

Chapitre 4

Caractérisation et suivi des objets mobiles

Sommaire

4.1 Introduction

Le contexte de navigation d'un robot dans des environnements inconnus implique que le robot exécute certaines fonctions essentielles de façon autonome. Une de ces fonctions est la capacité de prédire le mouvement des objets dynamiques qui se déplacent dans le même environnement, principalement pour minimiser le risque de collision avec eux. Cela est une tâche bien connue et développée depuis des données sensorielles acquises par des caméras fixes (intelligence ambiante) ou mobiles (embarquées) et, parfois, avec l'aide d'autres capteurs.

Dans notre cas nous étudions cette fonction en ayant pour seule information, des images acquises depuis une caméra montée sur le robot mobile ou sur le véhicule. Pour ce faire nous devons exploiter le maximum de propriétés visuelles possibles. Nous avons traité dans le chapitre précédent la détection

des objets mobiles à partir de points d'intérêt. Ce chapitre propose tout d'abord deux types de modèles possibles pour représenter et suivre les objets mobiles détectées, et aussi pour améliorer la connaissance que le système en a : le groupe de points d'intérêt et la région qu'il recouvre dans les images successives. Ensuite nous décrivons le filtre du Kalman utilisé pour le suivi de ces objets et évaluons plusieurs méthodes de suivi d'un objet, représenté par un groupe de points ou par une région..

4.2 Modèle basé sur des points d'intérêt

Un des principaux objectifs de ce travail consiste à suivre tous les objets mobiles détectés sans aucune connaissance *a priori* sur le type d'objet que l'on suit. Dans un contexte réel et dynamique, nous nous attendons à trouver des objets mobiles rigides comme par exemple des voitures. Pour ce type d'objets, nous avons montré dans la section 3.3 la bonne performance de la méthode de groupement *a contrario* pour trouver des ensembles de points dynamiques qui correspondent à des points 3D portés par de tels objets mobiles dans la scène. De ce fait, nous prenons l'ensemble des points d'intérêt (KLT) pour initialiser un modèle de l'objet.

Notons que si un tel objet est statique (une voiture garée), il ne sera pas détecté par notre méthode ; mais il ne représente pas de danger de collision. Par contre, il serait intéressant de savoir que des points statiques correspondent à des points 3D qui sont portés sur un objet potentiellement dynamique, afin d'éviter de les prendre comme amers de localisation. Mais ceci est une autre histoire !

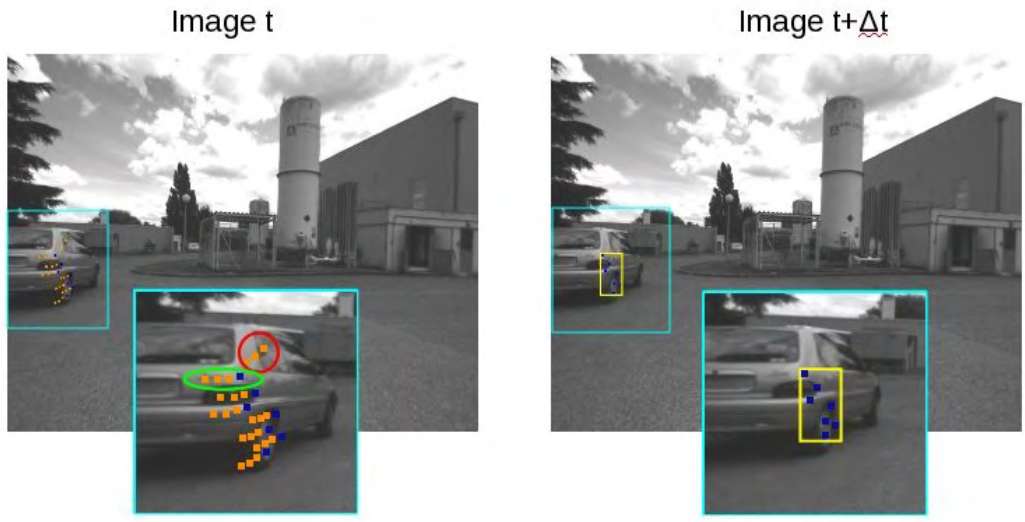


FIGURE 4.1 – Initialisation du modèle de l'objet. À gauche, le groupe de points identifié comme un objet dynamique : les points en orange désignent les positions lors du temps de pistage, tandis que les points en bleu représentent les positions sur l'image t. À droite dans le rectangle jaune, seuls les points en bleu de l'image t sont retenus pour générer le modèle de l'objet et le suivre à partir de l'image $t + \Delta t$.

La figure 4.1 à gauche illustre l'exemple d'un groupe G avec 31 points identifiés comme dynamiques par la méthode de groupement. À titre d'illustration, nous avons représenté en bleu la position des points sur l'image t, et en orange les positions précédentes le long d'un *temps de pistage* composé de 4 images. Étant donné le caractère temporel des points, nous ne pouvons pas utiliser les 31 points contenus dans le groupe G pour initialiser un modèle de l'objet. Nous ne sélectionnons que les points dans l'image

courante, donc, dans le cas de la figure, nous prenons seulement les points en bleu, correspondant à l'instant t, dernière itération du temps de pistage.

Cette stratégie implique quatre cas à considérer pour chaque point KLT suivi :

1. Le point est suivi le long de tout le *temps de pistage* de N_{im} images.

2. Le point est suivi pendant $N_{im} - 1$ images consécutives du *temps de pistage*.

3. Le point est perdu à la dernière image du *temps de pistage*.

4. Le point est suivi uniquement pendant la moitié du *temps de pistage*.

Les meilleurs cas sont le 1 et le 2, puisqu'ils impliquent que le point a bien été trouvé sur chaque image du *temps de pistage*. Le cas 2 apporte aussi une bonne information sur le point KLT, même s'il n'a pas été suivi depuis le début du *temps de pistage*, mais il est présent sur l'image courante au moment du processus de groupement. La figure 4.1 représente avec une ellipse verte le premier cas, qui est le cas pour presque tous les points dans cet exemple. Le cercle rouge de la même figure illustre le cas 3 où le point a été suivi pendant 3 des 4 images consécutives, mais il n'a pas été retrouvé sur la dernière image. Dans ce cas le point d'intérêt ne sera pas pris en compte pour le modèle car il n'est plus détecté sur l'image courante. La même situation se présente dans le cas 4 où les points ne sont détectés que la moitié du *temps de pistage*.

Alors, on ne valide pour initialiser le modèle $\mathcal{M}_O$ associé à un objet O, que les points présents sur l'image au moment du processus de groupement (fin du temps de pistage) et aussi, les points présents au moins sur les $N_{im} - 2$ images précédentes.

D'abord, les données pour calculer le modèle doivent être sélectionnées parmi les points inclus dans le vecteur G délivré par la méthode de groupement. Dans la section 3.3, nous avons défini qu'un groupe $G(X_n|t)$ contient n points $X(x, y, v_x, v_y)$ avec $X \in X_N$, $n \leq N$ et $t = [t_1, ..., t_{N_{im}}]$. Nous avons d'ailleurs défini que les points X_N forment le vecteur d'entrée $V(X_N|t)$, donc $G \subset V$ car les éléments des deux vecteurs sont issus de l'accumulation des positions jusqu'à l'instant t.

Étant donné un groupe de points dynamiques $G(X_n|t_1, ..., t_{N_{im}})$ issu de l'accumulation des points le long de N_{im} images, et soit $g(X|t = t_{N_{im}})$ un sous-ensemble de $G(X_n|t_1, ..., t_{N_{im}})$ qui contient uniquement les points de G qui respectent les deux premiers cas du suivi de points. Alors, nous allons initialiser un modèle d'objet sur la base des équations suivantes :

$$\bar{x}(t) = \frac{\sum_m g(X(x)|t=t_{N_{im}})}{m} \quad \bar{y}(t) = \frac{\sum_m g(X(y)|t=t_{N_{im}})}{m} \tag{4.1}$$

$$\bar{v}_x(t) = \frac{\sum_m |g(X(v_x)|t=t_{N_{im}})|}{m} \quad \bar{v}_y(t) = \frac{\sum_m |g(X(v_y)|t=t_{N_{im}})|}{m} \tag{4.2}$$

$$\mathcal{M}_O(t) = [\bar{x}, \bar{y}, \bar{v}_x, \bar{v}_y]^T \tag{4.3}$$

Après avoir obtenu le modèle, une région qui délimite dans l'image courante, l'ensemble des points détectés sur l'objet, est calculée pour représenter les limites de l'objet. Ainsi, les limites inférieure et supérieure de la région sont données respectivement par les valeurs minimales et maximales des points en X et Y. Nous représentons cette région avec un rectangle jaune sur l'image à droite de la figure 4.1.

La figure 4.2 présente un deuxième exemple sur l'initialisation d'un objet mobile, les points en bleu caractérisent le modèle de la voiture dans la scène. La figure illustre l'extraction d'une petite zone autour de la position de chaque point, fixée à une taille de 11×11 pixels. Ces motifs caractéristiques pour chaque point sont stockés ; ils seront utilisés lors de la recherche des points sur l'image suivante.

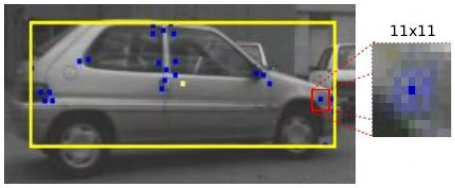


FIGURE 4.2 – Les points dynamiques en bleu forment le modèle de l'objet mobile, délimité par le rectangle. Un motif de 11×11 pixels est extrait autour de chaque point détecté sur l'objet mobile.

Cette même procédure est mise en œuvre pour initialiser chaque groupe dynamique G. Dans le reste de cette section nous décrirons la méthode de mise en correspondance des points sur l'image suivante, la procédure de prédiction sur la position de l'objet dans cette image, sera expliquée en section 4.4.

4.2.1 La mise en correspondance des points

La recherche d'un point sur l'image suivante démarre à partir de sa position prédite, obtenue à partir de sa position précédente par un filtre de Kalman (voir section 4.4). Le motif extrait pour chaque point est une fenêtre carrée de $2 \times w_m + 1$ pixels fixée avec $w_m = 5$ pixels. Ensuite, ce motif sera recherché par corrélation à l'intérieur d'une région de taille $2 \times w_r + 1$ autour de sa position prédite sur l'image courante. Cette procédure est illustrée en figure 4.3. À l'image t la position du point et la fenêtre qui délimite son motif sont représentées en rouge, la prédiction de ce point donnée par filtrage de Kalman est dessinée en cyan. Dans l'image $t + \Delta t$ la nouvelle position du point est trouvée (en bleu) après la mise en correspondance effectuée sur toute la zone de recherche (en rose) centrée sur la position prédite en couleur cyan.

Nous choisissons la taille de la zone de recherche comme $2 \times w_r + 1$ où w_r est donné par :

$$w_r = w_m + \textit{ marge de sécurité} \tag{4.4}$$

avec une *marge de sécurité* fixée à 5 pixels, donc avec $w_r = 10$. Dans cette zone centrée sur la prédiction du point KLT, nous cherchons la position $\mathbf{x} = (x_t, y_t)$ qui correspond le plus à l'apparence du motif du point extrait à l'image précédente. Donc la nouvelle position sera obtenue par l'équation 4.5 qui décrit le flot optique dans l'équation 3.2, dont les valeurs pour d_x et d_y sont obtenues par les équations 3.5 et 3.6

$$\begin{aligned} x_t &= x_{t-1} + d_x \\ y_t &= y_{t-1} + d_y \end{aligned} \tag{4.5}$$

Le plus grand risque au moment d'exécuter la procédure, est que la paire (d_x, d_y) trouvée en optimisant le score SSD entre les motifs, corresponde à une position sensiblement différente de la position prédite. Ce problème arrive quand le point d'intérêt est localisé sur une zone homogène de l'objet. Par exemple sur le coffre ou le toit d'une voiture où le meilleur correspondant peut tomber n'importe où

FIGURE 4.3 – Processus de mise en correspondance d'un point de l'image t vers l'image $t + \Delta t$. À gauche, la position courante du point (en rouge) et sa prédiction (en cyan) donnée par le filtre de Kalman. À droite, le point retrouvé (en bleu) est un peu décalé par rapport à sa position prédite (en cyan) mais il appartient à la zone de recherche (en jaune).

dans la zone de recherche, trop homogène. Afin d'éviter ce problème, nous choisissons une zone de recherche relativement petite pour rechercher le point suivi autour de sa position prédite. Par contre, cette amélioration implique que le point suivi ne peut pas sortir de la zone de recherche ; dans le cas contraire le point ne sera pas retrouvé. Cela signifie que (1) la vitesse des objets, et surtout du robot qui porte la caméra doit être faible par rapport à la fréquence d'exécution de la procédure de suivi (ici nous visons la fréquence vidéo, soit 25Hz) et (2) que la prédiction doit être la plus précise possible.

Une fois que toutes les nouvelles positions des points inclus dans l'objet suivi,sont retrouvées sur l'image courante, la dernière tâche consiste à mettre à jour la nouvelle apparence des motifs des points. Nous avons testé principalement deux stratégies : sans et avec comparaison avec le motif du départ. La description de ces stratégies sera abordée dans la sous-section suivante.

4.2.2 Mise à jour du modèle de l'objet

Au moment de l'initialisation du modèle de l'objet à partir d'un groupe de points issu de la méthode *a contrario*, nous avons décrit que nous prenons un motif autour de chaque point pour le représenter. Le processus de mise en correspondance de ces points dans les images successives, est fondé sur une évaluation de la similarité de ces motifs dans chaque nouvelle image. Mais quel motif faut-il conserver pour chaque point ? Le motif initial issu de la première image où cet objet a été détecté, ou le motif qui entoure ce point suivi dans la dernière image où ce point a été suivi ?

Nous avons effectué une première étape des tests en comparant uniquement le motif précédent avec le motif courant dans chaque image de la séquence. Étant donné le caractère dynamique des cibles, très facilement le point d'intérêt du départ perdait sa véritable position sur l'objet. C'est-à-dire que le point "glissait" sur la surface de la cible, en gardant des propriétés similaires au niveau du motif, mais en représentant une partie différente de celle du départ sur l'objet. D'après ces résultats, il est toujours important de conserver la meilleure ressemblance des propriétés prises au départ avec celles qu'on trouve sur les images suivantes, mais sans forcer le seuil de similarité à une valeur trop petite.

Au moment d'évaluer la ressemblance de l'apparence entre le motif de l'image courante et le motif

extrait au départ, nous avons besoin d'évaluer les transformations subies par le motif. La transformation affine est la plus couramment utilisée. Dans la section 3.2.3, nous avons présenté le modèle de transformation affine par l'équation 3.9. Dans cette équation la matrice carrée A contient quatre paramètres qui permettent de déformer le motif de départ du point d'intérêt, en lui donnant un aspect plus proche de celui qu'il a sur l'image courante. Tous les points qui satisfont le test de la transformation affine, seront utilisés pour recalculer le modèle de l'objet défini par l'équation 4.3. Par contre, les points qui ne satisfont pas le test seront enlevés du modèle de l'objet et ils ne seront pas remplacés comme dans le cas du suivi KLT. En conséquence, le modèle de l'objet aura moins de points et le seul moyen de compléter le modèle de l'objet avec de nouveaux points, sera la fusion de l'objet avec de nouveaux clusters issus de la fonction de groupement.

4.2.3 Les contraintes du modèle des points

Le fait de n'utiliser que des points pour construire un modèle de l'objet dynamique, limite la performance de notre approche quand plusieurs de ces points sont perdus. La cause principale de la perte des points provient de la grande vitesse apparente des objets dans la séquence ou quand l'objet tourne sur lui-même. Dans la séquence d'images acquises depuis la caméra mobile, à l'instant courant, un objet avec un nombre initial de points n est identifié par une région qui contient tous ces points. Comme la taille de la région est directement définie par les points qu'elle contient, à chaque fois qu'un point est perdu, la taille de la région diminue aussi et le modèle initial ne peut pas être conservé le long de son suivi.

Un exemple est montré sur la figure 4.4. Initialement à l'image 536 nous détectons la voiture avec une région qui la contient entièrement. À cause des mouvements non désirés de la caméra sur le robot mobile, la région de l'objet commence à diminuer jusqu'à la moitié de la taille de départ en à peine 9 images consécutives.

Afin de minimiser les inconvénients constatés en n'utilisant que des points, un modèle basé sur des régions est décrit dans la section suivante ; d'autres propriétés comme l'intensité moyenne de la région, et surtout, la forme du contour qui la délimite, apportent plus d'informations exploitées pendant le suivi des objets le long de la séquence.

4.3 Modèle basé sur des régions

Le suivi d'objets en n'utilisant que des points d'intérêt, nous a permis de développer un algorithme simple, déjà exploité pour la détection, et surtout rapide en temps de calcul. Cependant, comme vu dans la sous-section précédente, le modèle d'un objet par des points, peut très facilement être perdu à cause de légères rotations de l'objet mobile, ou de vibrations de la caméra.

Plutôt que de prendre une fenêtre englobante, il est possible de délimiter une région dans l'image qui contient les points dynamiques détectés par la méthode de groupement. En faisant une analyse d'intensité à l'intérieur de la région, nous proposons la mise en correspondance et la mise à jour de ces régions le long de la séquence afin de suivre les objets non-rigides qui ont des mouvement plus compliqués, typiquement des objets qui correspondent à des piétons.

D'abord pour délimiter la région nous initialisons un contour actif dans l'image. Un contour actif ou *snake* est une courbe qui évolue soit dans une seule image à partir d'une position initiale (segmentation) soit dans une séquence d'images à partir d'une position prédite (suivi d'image en image), afin de minimiser une énergie définie à partir de forces restrictives internes et de forces externes dans l'image,

FIGURE 4.4 – Suivi d'un objet basé sur des points d'intérêt. À l'image 536 l'objet mobile est entièrement identifié. Ensuite sa taille est réduite de moitié à cause des points non retrouvés au moment de la mise en correspondance.

typiquement calculées à partir des bords, du gradient, etc. Un *snake* n'est pas pensé, en principe, pour résoudre le problème de recherche automatique des contours proéminents de l'image, mais plutôt pour raffiner le contour d'une forme, à partir d'une position initiale proposée par d'autres mécanismes ; si le contour de départ est relativement proche de la solution (par exemple un contour défini manuellement par un opérateur ou obtenu grâce à une autre méthode), le contour évolue jusqu'à minimiser la fonction d'énergie définie à partir des forces internes et externes. À cause des contraintes temps réel, nous exploitons seulement certaines statistiques globales de la région qui décrit l'objet, comme les propriétés principales, pour faire sa mise en correspondance à l'image suivante.

Nous souhaitons compléter l'information de l'objet avec d'autres attributs comme l'intensité de l'objet. Soit un objet dynamique détecté par un groupe de points : nous avons mis en œuvre une méthode pour extraire la région qui caractérise l'objet, à partir de ce groupe de points, qui permette une meilleure rcaractérisation que la simple boîte et les motifs autour de chaque point comme proposé dans la section précédente.

Cette procédure permet de générer un modèle plus stable de la cible parce qu'il contient beaucoup plus d'information que le modèle basé sur des points. Pour ce faire, nous utilisons la théorie de contours actifs afin de délimiter la région de l'objet mobile, ensuite, nous caractérisons cette région par son intensité lumineuse, que nous utiliserons comme mesure de similarité entre régions dans chaque image de la séquence. La prédiction de la région sur l'image suivante est toujours donnée par le filtre du Kalman. Au long de cette section nous décrirons essentiellement les détails de la théorie des contours actifs, et la procédure de suivi des objets représentés par des régions.

4.3.1 Caractérisation de la région par un contour actif

Les contours actifs ou déformables «snakes» permettent de modéliser les frontières complètes d'un objet, et de le séparer du fond et du reste des objets dans l'image [23]. Le modèle d'un contour actif consiste en une courbe élastique qui est initialisée par une information a priori sur la forme des objets et qui évolue comme résultat de l'action combinée d'énergies externes E_{ext} et internes E_{int}. Le contour initial peut être arbitraire, c'est-à-dire qu'il peut complètement contenir l'objet que l'on cherche à segmenter ou à suivre (le contour doit se rétracter jusqu'à atteindre les bords de l'objet), être à l'intérieur de celui-ci (le contour doit être expansé comme un ballon) ou le contenir partiellement (chevauchement). La forme du contour va évoluer par un processus de minimisation d'énergie. Il est souhaitable de minimiser l'énergie externe E_{ext} pour guider la courbe vers des caractéristiques discriminantes de l'image, les bords de l'objet à segmenter ou suivre, tandis qu'on utilise la minimisation de l'énergie interne E_{int} pour jouer sur l'élasticité de la courbe. Les lignes, les bords et les "coins" sont les principales caractéristiques de l'image employées pour interagir avec les contours actifs. Pour ce faire, certaines pondérations sont assignées à ces caractéristiques par rapport au comportement souhaitable du «snake».

La fonction de base B-Spline. Le modèle déformable est décrit en employant un ensemble de courbes flexibles qui formeront le contour global avec des paramètres qui contrôleront les variables cinétiques de la courbe. Ces paramètres contrôlent par exemple les tailles des différentes parties du contour et les angles d'union entre elles.

Pour ne pas être contraint sur la forme du contour, nous adoptons une forme paramétrique ; le contour dans le plan XY est représenté comme une courbe fermée $u(\tau) = (x(\tau), y(\tau))$, $\tau \in [0, 1]$, avec $u(0) = u(1)$. Donc, les énergies sont décrites en termes de petits changements de la position $u(\tau) = (x(\tau), y(\tau))$ sur chaque point du contour actif. τ désigne l'abscisse curviligne le long de sa frontière, normalisée entre 0 et 1. Les fonctions paramétriques unidimensionnelles $x(\tau)$ et $y(\tau)$ sont représentées par une combinaison linéaire de fonctions de base B-spline $B_i(\tau)$, définies à partir de points

de contrôle ou nœuds V_j. Les noeuds sont définis par un ensemble de valeurs pour le paramètre τ, $(0, \tau_1, ..., \tau_j, ..., \tau_N)$ [10]. Si la fonction de base est de degré n, chaque fonction de base $B_i(\tau)$ est non nulle seulement sur $n + 1$ intervalles consécutifs entre points de contrôle.

Les fonctions B-spline sont compactes, permettent un contrôle local de la courbe et il est possible d'inclure des coins en permettant plusieurs multiplicités dans les nœuds. Nous avons utilisé les fonctions de base de spline cubique pour obtenir le contour de notre objet. Ensuite, à chaque itération ce contour est modifié afin de le faire converger vers la silhouette de l'objet que l'on veut segmenter ou suivre.

Initialisation du contour actif. Depuis l'ensemble de points qui sont déjà détectés sur un objet dynamique par la méthode de groupement (section 3.3), nous sélectionnons uniquement les points qui sont les plus à l'extérieur pour définir la frontière de l'objet. Nous pourrions aussi partir de la fenêtre englobante (notre modèle d'objet dans la section précédente), mais il est plus robuste pour définir les caractéristiques initiales de la région, de partir le l'enveloppe convexe du groupe des points.

Un exemple est illustré sur la figure 4.5 : un ensemble de points dynamiques (en bleu) sont détectés sur la personne, qui est en mouvement sur cette séquence. De cet ensemble de points nous avons sélectionné ceux illustrés en couleur magenta pour décrire la frontière de l'objet illustrés en jaune car ce sont les points les plus lointains du centre et pourtant les plus proches de la frontière de l'objet. Pour ce cas particulier, nous avons 8 points sur la courbe et nous utilisons entre chacun, 4 points de contrôle . La valeur de 4 points de contrôle est fixe pour chaque intervalle sur la frontière afin d'introduire des coins ou des discontinuités dans la courbe [71].

FIGURE 4.5 – Contour actif initial dérivé des points les plus éloignés du centre.

4.3.2 Formulation de l'énergie de «snakes»

Une fois établie l'approximation des contours par une fonction B-Spline cubique, la solution au problème de la détection ou du suivi du contours d'un objet réside en la minimisation de l'énergie totale de ce contour représenté par une courbe paramétrique :

$$E_{snake} = \int_0^1 [E_{int}(u(\tau)) + E_{ext}(u(\tau))]\, d\tau \tag{4.6}$$

où E_{int} est composé principalement par un premier terme qui pondère l'élasticité et un deuxième qui

FIGURE 4.6 – Contour initial de l'objet dynamique et le masque correspondant.

pondère la flexibilité, définie par l'équation suivante :

$$E_{int} = \alpha \int_0^1 |u_\tau(\tau)|^2 d\tau + \beta \int_0^1 |u_{\tau\tau}(\tau)|^2 d\tau \tag{4.7}$$

L'indice τ, et $\tau\tau$ sous le terme $u(\tau)$ impliquent un premier et deuxième ordre de différentiation, respectivement. Nous revenons sur l'équation 4.6 pour introduire le terme E_{ext} ou l'énergie de l'image [85], définie à partir d'un champ de potentiel P donné par l'équation :

$$E_{ext} = \int_0^1 P(u(\tau)) d\tau \tag{4.8}$$

Le potentiel $P(u(\tau))$ inclut différents termes définis à partir des caractéristiques des images, notamment, les bords, les lignes, etc. Le plus couramment utilisé est l'énergie des bords obtenue en fonction de la magnitude de l'intensité du gradient $|\nabla I|$, ce qui donne :

$$E_{bord} = -w_{bord} \int |\nabla I|^2 d\tau \tag{4.9}$$

où w_{bord} est le facteur de pondération du bord. Sans une bonne initialisation, l'énergie des bords ne sera pas suffisante pour localiser les objets sur des images bruitées ou de bas contraste. Donc un potentiel additionnel des régions est ajouté à celui du bord. Généralement, le potentiel des régions est défini par la moyenne (μ) et la variance (σ^2) de l'intensité des pixels internes à la région ; ceci ne permettra pas de distinguer des régions avec les mêmes valeurs de μ et σ^2, mais avec des histogrammes différents.

Notons que d'autres contraintes pourraient bien être ajoutés comme la vitesse des objets ou d'autres statistiques dans la région [12].

4.3.3 Mise à jour du modèle de l'objet

Une fois que nous avons obtenu le contour de l'objet, nous obtenons la région à l'intérieur du contour. La figure 4.6 montre ce contour initial sur l'image à gauche et la région à l'intérieur de ce contour à droite. L'image à droite est construite à la fois pour la donner comme entrée au processus qui recherche le contour actif, mais aussi pour rajouter cette région sur la carte de probabilités pour éviter de détecter des nouveaux points d'intérêt sur cette région dans les images suivantes.

Ensuite, nous avons testé une méthode basée sur le travail de Chan et Vese [17] afin de trouver la silhouette de l'objet. Dans ce travail, les auteurs donnent une région d'initialisation qui peut contenir l'objet soit dans sa totalité soit partiellement. Cette région doit être définie dans un masque au format

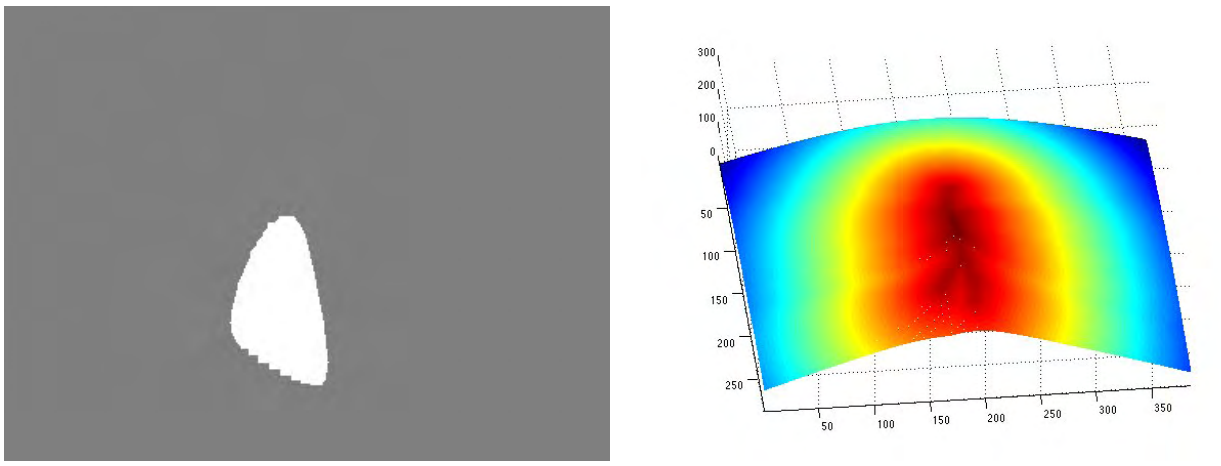


FIGURE 4.7 – À gauche le masque d'entrée obtenue par la détection des points par notre approche. À droite la représentation graphique du calcul de la distance entre chaque pixel et le pixel '1' le plus proche sur ce masque.

binaire : la valeur '0' indique l'extérieur de l'objet et '1' l'intérieur. À partir de ce masque, on calcule la distance euclidienne entre chaque pixel et le pixel '1' le plus proche. Les valeurs résultants de cette procédure vont permettre d'évaluer les conditions de minimisation de l'énergie à l'intérieur et à l'extérieur du contour. La figure 4.7 à droite montre une image type obtenue du calcul des distances depuis le masque montré à gauche. La zone occupée par l'objet est segmentée (en rouge) du fond et ce sont ces valeurs plus intenses à l'intérieur de l'objet qui vont permettre d'adapter le contour initial à la véritable silhouette de l'objet.

Nous avons pu réaliser des expérimentations préliminaires pour le moment, avec les travaux de Chan et Vese par échange de données entre notre module de détection et leur module de suivi. Notre approche indique le nombre d'objets et, pour chacun, le masque qui indique la zone où se trouve cet objet dans l'image. Ensuite une recherche des vrais contours de l'objet est réalisé en équilibrant les forces à l'extérieur et à l'intérieur du contour. L'adaptation du contour actif est totalement obtenu par la méthode de Chan et Vese, restreinte à un nombre fixe d'itérations, établi à 30.

Bien que ce nombre d'itérations ne permette pas d'obtenir la silhouette exacte de l'objet, nous utilisons cette valeur comme un compromis qui permet d'obtenir une très bonne approximation sur les limites de l'objet, tout en réduisant le temps de calcul. Nous favorisons l'utilisation d'un nombre petit d'itérations principalement parce que le but est de suivre l'objet le long d'une séquence où à chaque fois l'objet changera sa forme (surtout dans le cas de piétons) et aussi parce qu'on peut arriver à mieux obtenir la silhouette de l'objet dans le temps.

La figure 4.15 illustre l'amélioration de la silhouette en exploitant 10 images consécutives dans lesquelles nous suivons une personne, à raison de 30 itérations sur chaque image dans le but de trouver la frontière de l'objet. Notons qu'à chaque image, le contour de l'objet donne plus de détails sur la silhouette de notre objet mobile. Au début l'initialisation de l'objet n'est pas bien détaillée ; malgré cela, nous utilisons ce contour pour démarrer la recherche du nouveau contour sur la position prédite par le filtre du kalman, comme cela sera expliqué en section 4.4.

Nous avons étendu nos expérimentations sur le suivi de régions pour le cas de deux objets mobiles

Image 270 Image 275

Image 290 Image 295

FIGURE 4.8 – Suivi de deux objets mobiles sur le base des régions.

dans la même image. Ici, de la même façon que pour le test précédent, les régions d'initialisation sont données par notre approche de détection et groupement d'objets. La figure 4.8 montre 4 images lors de la détection de deux personnes sur l'image. L'homogénéité des régions n'est pas très visible sur la fille et est perturbé par le fond dans le cas du garçon. L'initialisation de la fille sur l'image n'est pas complète, mais au fur et à mesure cette région s'élargit. Cependant à cause des forts contrastes sur ses vêtements, la méthode détecte seulement des régions séparées. Bien que ces zones ne sont pas jointes elles représentent une même région. Dans le cas du garçon, les vêtements sont plus homogènes en couleur et c'est plutôt l'environnement du fond qui est très similaire, donc une extension vers des zones du fond, est inévitable en n'exploitant pas d'autres forces pour faire évoluer notre contour actif.

4.3.4 Les contraintes du modèle de régions

Les contours actifs sont peu sensibles aux petites discontinuités sur la frontière des objets ou sur des bords de bas-contraste parce que l'énergie est configurée globalement. Toutefois, en analysant les résultats obtenus sur la figure 4.8, nous avons trouvé que le contour commence à s'éloigner de la véritable silhouette de notre objet dynamique car l'intensité, en particulier de la zone sur le plan arrière du garçon, est très similaire. Cela représente pour le moment une des contraintes que nous avons rencontrées sur cette méthode.

Afin de faire une comparaison entre les modèle basés sur des points et sur des régions, nous avons analysé la même séquence vidéo en utilisant que les points d'intérêt. Les résultats obtenues pour 1000 images traitées sont montrés dans la figure 4.14. Nous montrons sur ces images, la détection de presque toutes les personnes qui sont en mouvement le long de la séquence ; cependant la perte des points à cause des mouvements apparents erratiques, fait que très facilement aucun point n'est retrouvé sur l'image suivante et donc le modèle de l'objet disparaît. Dans la mesure où la mise en correspondance du modèle de régions permet de suivre cet objet et même de mettre à jour (ici par une expansion) sa région de caractérisation à chaque image, le suivi d'objets dynamiques non rigides est mieux réalisé par des régions. Mais nous devrons ajouter à cela d'autres contraintes comme possiblement, la vitesse afin d'éviter l'extension sur des objets du fond.

4.4 Estimation du mouvement des objets mobiles

Jusqu'à maintenant, nous avons décrit la détection d'objets mobiles en utilisant l'information depuis une seule caméra et sans connaissance *a priori* de l'environnement. Cependant, une fois la cible dynamique détectée, l'objectif du robot consiste à prédire sa position à chaque instant de temps t. Cette tâche peut être considérée comme un sous-problème d'une autre tâche plus complète comme la planification d'une trajectoire pour le robot afin d'éviter la cible. Nous considérons le mouvement de chaque cible détectée comme un modèle d'état linéaire gaussien, ainsi toutes les variables aléatoires sont continues, donc l'algorithme du filtre de Kalman peut être utilisé.

4.4.1 Le filtre de Kalman

Le filtre de Kalman est un algorithme d'estimation statistique développé par Rudolf Emil Kalman à la fin des années 50 [37]. Il est utilisé pour prédire et corriger l'état courant d'un système dynamique linéaire dont l'état est caractérisé par un vecteur aléatoire, dont on connaît à l'instant t, la valeur moyenne et une matrice de covariance avec un modèle de bruit additionnel. La prédiction utilise des mesures linéaires perturbées aussi par du bruit [75]. Dans le temps, les perturbations localisées sur les mesures sont représentées par du bruit gaussien avec une matrice de covariance connue à chaque temps de mesure.

Dans la suite nous présenterons les principales équations du filtre du Kalman ainsi que le modèle d'état que nous utilisons. Dans les équations suivantes, nous utilisons le sous-script k pour indiquer la valeur d'une donnée (état, mesure...) au temps discret t_k.

D'abord, nous définissons la fonction d'évolution du vecteur d'état $\mathbf{x}$ du temps t_{k-1} vers le temps t_k comme un système linéaire gaussien :

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{G}_k v_{k-1} \quad \text{avec} \quad k = 1, 2, ... \tag{4.10}$$

où $\mathbf{x}_k$ dépend directement de la dimension D des données, donc de façon générale $\mathbf{x}_k \in \mathcal{R}^{D_x \times 1}$. La matrice $\mathbf{F}_k$ représente le modèle de mouvement de dimension $\mathcal{R}^{D_x \times D_x}$. $\mathbf{G}_k$ est une matrice connue de dimensions $\mathcal{R}^{D_x \times D_v}$, parfois utilisée pour regrouper des coefficients de pondération appliqués sur le processus de bruit de dynamique gaussien v_k de moyenne zéro, décrit par :

$$v_k \sim \mathcal{N}(0, \mathbf{Q}) \tag{4.11}$$

avec matrice de covariance $\mathbf{G}_k \mathbf{Q} \mathbf{G}_k'$ et $\mathbf{Q} \in \mathcal{R}^{D_v \times D_v}$. Si la matrice $\mathbf{G}_k$ n'apparait pas comme coefficient dans l'équation d'état, la matrice de covariance de v_k est simplement $\mathbf{Q}$.

Ensuite, les mesures $\mathbf{z}$ au temps t_k, sont directement obtenues à partir de l'état donné par 4.10 , mais elles sont perturbées par du bruit ω_k :

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \omega_k \tag{4.12}$$

où $\mathbf{H}$ est la matrice de transformation de l'état vers l'espace des mesures, de dimension $\mathcal{R}^{D_z \times D_x}$. Cette matrice est connue, mais peut-être variable dans le temps. Le vecteur ω_k est un bruit gaussien de dimension $\mathcal{R}^{D_z \times 1}$ donné par :

$$\omega_k \sim \mathcal{N}(0, \mathbf{R}) \tag{4.13}$$

avec la covariance $\mathbf{R}$ de dimension $\mathcal{R}^{D_z \times D_z}$. Les deux processus de bruit gaussien (pour la dynamique et pour les mesures) ainsi que l'état initial du système, sont supposés mutuellement indépendants, donc les termes de covariance croisée seront nuls, ce qui facilite ainsi l'estimation de l'état.

Jusqu'ici nous avons présenté les équations du vecteur d'état $\mathbf{x}$ et du vecteur de mesures $\mathbf{z}$ avec leurs respectives fonctions de bruit additionnelles. Le modèle de dynamique et le modèle d'observation sont exploités pour la prédiction et la correction de l'état du système, mises en œuvre en utilisant les équations du filtre de Kalman.

Prédiction du vecteur d'état et des mesures

Soit $\hat{\mathbf{x}}_{k-1|k-1}$ le vecteur d'état estimé et $\mathbf{P}_{k-1|k-1}$ sa matrice de covariance correspondante au temps t_{k-1}, dont le deuxième argument $k-1$ indique le caractère conditionnel aux informations des mesures obtenues à l'instant $t-1$ $\mathbf{z}_{k-1}$ [6]. On trouve aussi des formulations où l'état est conditionné aussi :

— par les données de contrôle $\mathbf{u}_{k-1}$, non explicitées dans cette formulation générale ;
— par les observations acquises depuis l'origine $\hat{\mathbf{x}}_{k-1|0:k-1}$, mais en précisant que les prédictions et corrections ne dépendent que des données acquises à l'instant courant.

La prédiction de l'état et sa matrice de covariance au temps t_k sont données par le modèle dynamique :

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1|k-1} \tag{4.14}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}_{k-1}^T + \mathbf{G}_{k-1}\mathbf{Q}\mathbf{G}'_{k-1} \tag{4.15}$$

où la matrice $\mathbf{P}$ est de dimension $\mathcal{R}^{D_x \times D_x}$ et le deuxième terme dans l'équation 4.15 correspond à la covariance du vecteur de perturbation de l'état. Ensuite, la prédiction des mesures $\hat{\mathbf{z}}$ faites à l'instant t_k depuis cet état prédit, l'innovation η et la matrice de covariance sur l'innovation $\mathbf{S}$ sont respectivement données par :

$$\hat{\mathbf{z}}_{k|k-1} = \mathbf{H}\hat{\mathbf{x}}_{k|k-1} \tag{4.16}$$

$$\eta_k = \mathbf{z}_k - \hat{\mathbf{z}}_k \tag{4.17}$$

$$\mathbf{S}_k = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R} \tag{4.18}$$

Dans nos expérimentations en utilisant les régions de l'objet, nous avons exploité la prédiction de l'objet sur l'image suivante, de façon à indiquer l'endroit où démarrer le processus de mise en correspondance par la méthode du contour actif. La figure 4.9 montre cette position du barycentre (en rouge) et la région prédite sur l'image suivante d'après le filtre du Kalman. Nous utilisons toujours un modèle à vitesse constante pour le mouvement apparent des régions dans l'image (en 2D) : dans les méthodes SLAMMOT (voir chapitre suivant), cette prédiction se fonde plutôt sur le mouvement réel de l'objet correspondant à la région suivie (en 3D).

Que le modèle s'applique au mouvement réel ou apparent, il est nécessaire de bien positionner les paramètres de démarrage du filtre pour les adapter aux différents types de mouvement des objets, cela afin d'éviter un glissage trop accéléré de la position prédite. Les méthodes multi-modèles de type IMM [14] permettent de mettre en compétition plusieurs modèles dynamiques et de choisir à chaque itération, quel est le meilleur. Nous n'avons pas exploité de telles méthodes, pour éviter d'alourdir les temps de calcul.

Correction de l'état

Pour la mise à jour du vecteur d'état estimé $\hat{\mathbf{x}}$ et sa covariance $\mathbf{P}$, il est nécessaire d'obtenir le gain du filtre $\mathbf{K}_k$ qui dépend directement des matrices de covariance obtenues pour l'état prédit et pour l'innovation.

Ainsi, les équations 4.19 de 4.21 régissent la correction de l'état a posteriori, tenant compte de l'observation faite à l'instant courant :

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T(\mathbf{S}_k)^{-1} \qquad \textit{(Gain du filtre)} \tag{4.19}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\eta_k \qquad \textit{(Vecteur d'état estimé)} \tag{4.20}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} + \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T \qquad \textit{(Covariance de l'état)} \tag{4.21}$$

Dans la suite, nous détaillons les valeurs utilisées lors de l'exécution du filtre du Kalman, pour faire le suivi des objets dynamiques le long d'une séquence d'images.

FIGURE 4.9 – Prédiction par le filtre du Kalman. Le contour en jaune représente la silhouette à l'image courante mais elle est déplacée sur la position prédite pour le barycentre de la région sur l'image suivante ; la prédiction se fonde sur un modèle à vitesse constante du mouvement apparent de la région.

4.4.2 Le modèle à vitesse constante

Le processus de prédiction démarre pour $k = 1$ avec une estimation initiale $\hat{\mathbf{x}}_{0|0}$ de $\mathbf{x}_0$ et sa matrice de covariance associée $\mathbf{P}_{0|0}$. Nous avons déjà défini au début de ce chapitre la valeur initiale du vecteur d'état $\mathbf{x}_0$ que nous utilisons dans le filtre du Kalman pour chaque objet dynamique détecté. Afin de faire le lien avec la notation de cette section, nous re-écrivons ce vecteur qui initialise l'état du système depuis l'équation 4.3 :

$$\mathbf{x}_0 = [\bar{x}, \bar{y}, \bar{v}_x, \bar{v}_y]^T \tag{4.22}$$

où les valeurs du vecteur ont été définies par les équations 4.1 et 4.2 comme la moyenne de la position et de la vitesse de la cible détectée en coordonnées bidimensionnelles. Ce vecteur est utilisé comme la condition initiale du filtre ; c'est donc la moyenne du processus gaussien qui décrit la fonction de densité de probabilité du vecteur d'état. D'autre part, la matrice de covariance initiale pour ce vecteur d'état est choisie en fonction de la confiance en la position et la vitesse des mesures, $r = 10$ pixels et $r_v = 20$ pixels/sec, respectivement. Cette valeur est à peu près la moitié de la taille d'une cellule sur la carte de probabilités (voir 3.4). Ainsi $\mathbf{P}_0$ est une matrice carrée diagonale établie comme :

$$diag(\mathbf{P}_0) = \begin{bmatrix} r^2 & 0 & 0 & 0 \\ 0 & r^2 & 0 & 0 \\ 0 & 0 & r_v^2 & 0 \\ 0 & 0 & 0 & r_v^2 \end{bmatrix} \tag{4.23}$$

L'équation d'état du système 4.10 dans sa forme matricielle devient :

$$\mathbf{x}_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{k-1} + \begin{bmatrix} \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} v_{k-1} \tag{4.24}$$

avec la matrice **F** qui fait évoluer le système avec un modèle à vitesse constante. Le bruit gaussien v_{k-1} perturbe toutes les composantes du vecteur d'état. D'autre part, la fonction d'observation 4.12 prend en compte uniquement les composantes de la position du vecteur d'état, perturbées par du bruit, comme sa représentation matricielle l'atteste :

$$\mathbf{z}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} \omega_{1,k} \\ \omega_{2,k} \end{bmatrix} \tag{4.25}$$

Donc, étant données les valeurs de mesure **z** à chaque temps t_k, il est possible de corriger le vecteur d'état et sa fonction de densité au temps suivant en utilisant le filtre du Kalman. Dans la suite nous présentons les résultats de cette estimation lors du suivi d'un objet mobile rigide sur une séquence d'images, ainsi que le graphique montrant la consistance (ou l'intégrité) du filtre.

4.4.3 Résultats de la performance du filtre

La position dans le temps de la cible détectée dans la scène de la figure 4.10 est prédite en utilisant les paramètres décrits dans la sous-section précédente. Néanmoins, cette position correspond uniquement au barycentre de la cible ; une zone autour de cette position doit être initialisée pour rechercher le modèle de la cible à l'image suivante. Les dimensions de cette zone de recherche, illustrées en vert à chaque image, sont calculées grâce à l'équation 4.26. Les variables $LimiteX\_cible$ et $LimiteY\_cible$ sont directement la taille respectivement en X et Y de la fenêtre englobante autour de l'objet (zone jaune qui délimite la cible). Le rectangle en blanc représente la zone de recherche de la cible sur l'image suivante, centrée sur la prédiction du filtre de Kalman.

$$\begin{gathered} Zone\_cible = L \times A \\ L = LimiteX\_cible + Marge\_confiance + Mouvement\_maximal \\ A = LimiteY\_cible + Marge\_confiance + Mouvement\_maximal \end{gathered} \tag{4.26}$$

Dans le suivi, le temps initial t_0 est établi au moment de la détection de la cible et de l'initialisation de son filtre du Kalman (image 4.10a). Les axes de l'image, centrés sur le coin supérieur gauche, sont pris comme les axes de référence. En conséquence, les coordonnées du barycentre de la cible ne donnent pas l'origine, comme dans la plupart des cas lors du suivi d'un objet.

Dans l'exemple montré en figure 4.10f, la procédure mise en œuvre sur les 41 images consécutives, où la cible est dans le champ de vue de la caméra embarquée sur le robot, est la suivante :

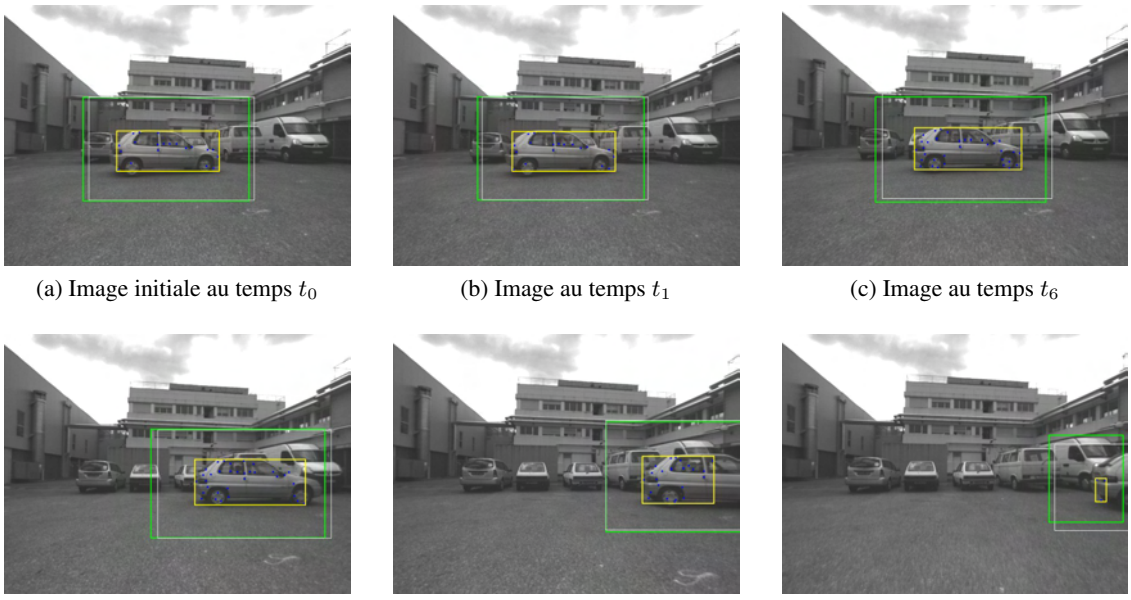
— Au temps $t \to t_k$, prédire la position du barycentre à l'image $t_k + \Delta t$

— Délimiter la zone de recherche du modèle centré sur la prédiction du barycentre

— Au temps $t \to t_k + \Delta t$, identifier la position réelle du modèle de la cible

— Mettre à jour la position et le modèle courant de la cible et du filtre du Kalman, pour une nouvelle procédure de prédiction.

La première ligne d'images de la figure 4.10 montre le début du suivi de la cible où les deux premières images sont consécutives et la troisième est le résultat au temps $t_0 + 6 \times \Delta t$. Dans ces trois images le modèle de la cible reste le même, c'est-à-dire qu'aucun élément n'a été perdu, même si le filtre de Kalman montre ses plus grandes valeurs d'erreur au début de la procédure (voir figure 4.12). À l'image 4.10d le modèle de la cible commence à perdre des points jusqu'au moment où la cible est



(a) Image initiale au temps t_0 (b) Image au temps t_1 (c) Image au temps t_6

(d) Image au temps t_{16} (e) Image au temps t_{26} (f) Image au temps t_{40}

FIGURE 4.10 – Résultats obtenus avec le filtre du Kalman lors du suivi d'un objet le long de 37 images. Les limites de la cible sont montrées en jaune ; en vert est représentée la région de recherche de la cible, en blanc la région prédite centrée sur la valeur donnée par l'équation 4.14. Uniquement les images a) et b) sont consécutives.

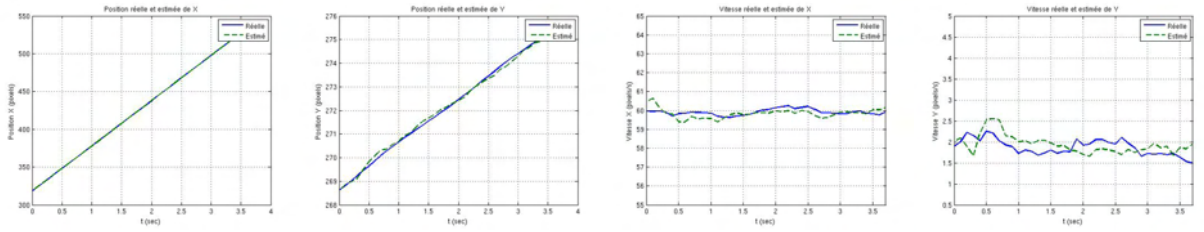
totalement perdue au temps $t_0 + 41 \times \Delta t$. Cette séquence a été acquise à 10 Hz donc $\Delta t = 0.1$, ce qui fait un temps de suivi de 4.1 s.

La figure 4.11 montre les graphiques de l'estimation et de l'observation réelle pour chaque élément du vecteur d'état pour la scène de la figure 4.10. La position de la cible en *X* dans l'image (voir 4.11a) montre une erreur minimale entre la valeur prédite et la valeur réelle. La position en *Y* sur le graphique 4.11b montre une erreur d'estimation sur les mesures à cause des valeurs de covariance du bruit blanc, ce qui n'a pas lieu sur la position en *X*. Cette différence est plus notable en Y car les valeurs assignées au bruit blanc sont les mêmes dans l deux directions (terme $(\Delta t^2)/2$ de l'équation 4.24) mais finalement le mouvement se développe plutôt dans la direction *X*. Donc, la prédiction calcule une valeur plus grande de mouvement de ce qui est fait en réalité. Dans le cas des graphiques de la vitesse en *X* et en *Y*, l'erreur n'est pas importante même si la valeur de la vitesse n'est pas mesurée par un capteur (valeur de la matrice H sur l'équation 4.25), mais est seulement déterminée à partir du modèle dynamique.

Même si les graphiques de la figure 4.11 nous montrent une faible différence entre les valeurs estimées et réelles du filtre, un test statistique plus formel a été réalisé afin de vérifier la consistance du filtre. Ce test vérifie la "crédibilité" du résultat donné par un filtre, en particulier l'hypothèse que le filtre est consistant avec les covariances S_k calculées (équation 4.18) ; cette hypothèse est violée lorsque les non linéarités sont trop importantes. Pour cela, l'innovation quadratique normalisée (NEES pour *normalized estimation error squared* en anglais) est calculée par l'équation suivante ([6]) :

$$NEES_k = \eta_k^{'} S_k^{-1} \eta_k \tag{4.27}$$

où l'innovation η_k est donnée par l'équation 4.17. Comme la distance de Mahalanobis, la variable aléatoire $NEES_k$ suit une distribution χ^2, avec D_z degrés de liberté (où D_z est la dimension du vecteur des



(a) X (b) Y (c) Vitesse X (d) Vitesse Y

FIGURE 4.11 – Graphiques des résultats numériques du vecteur d'état pour la cible suivie dans la figure 4.10 le long de 4.1 secondes. Les graphiques montrent les valeurs prédites et estimées, en utilisant le filtre de Kalman.

mesures **z**). Afin de vérifier l'hypothèse de consistance de notre filtre avec un risque α) (ou une confiance $(1-\alpha)$), il est nécessaire de calculer les bornes qui délimitent la valeur obtenue de la variable $NEES$. Pour ce faire, nous choisissons $\alpha = 0.05$, avec $D_z = 2$ degrés de liberté. En utilisant une table des valeurs de distribution χ^2, les bornes sont :

$$\begin{aligned} Borne_{inf} &= \chi^2_2(0.025) = 0.051 \\ Borne_{sup} &= \chi^2_2(0.975) = 7.38 \end{aligned} \tag{4.28}$$

Les graphiques de résultats obtenus pour le test NEES sont montrées sur la figure 4.12. Notez que uniquement 2 valeurs sont hors des bornes 4.28, en sachant que nous avons réalisé 40 fois notre processus de Kalman, cela nous laisse 38 valeurs dans les bornes, donc 95% des cas. Ceci satisfait le test de consistance en accord avec les valeurs de confiance établies au préalable.

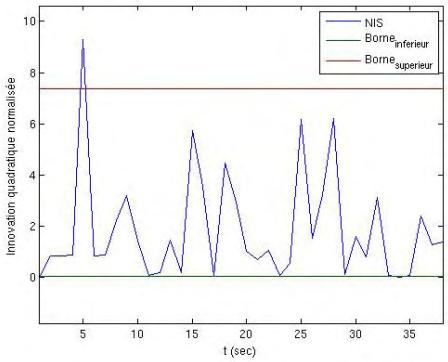


FIGURE 4.12 – Graphique de l'innovation quadratique normalisée estimée pour les résultats du filtre sur la figure 4.10. Les bornes sont obtenues en supposant que le filtre suit une distribution χ^2 avec D_z degrés de liberté avec un risque de 0.05, donc une probabilité de consistance de $0,95$.

4.5 Résultats expérimentaux du suivi des objets mobiles

Afin de montrer la performance du module de suivi et de fusion d'objets mobiles dans le temps, nous avons réalisé plusieurs tests sur de longues séquences d'images acquises sur des scènes dynamiques en extérieur et en intérieur. Le défi consiste à montrer l'efficacité du module de détection et de suivi de plusieurs objets dynamiques dans des environnements divers, avec des objets qui ont des directions de

mouvement différentes et variables. Les expérimentations ont été réalisées sur des séquences récupérées sur le Web, acquises dans des environnements d'extérieur pour des objets rigides (véhicules) et d'intérieur pour des objets non rigides (piétons).

Notons que ces séquences mises à disposition pour valider des approches de vidéo-surveillance, ont été acquises depuis des caméras fixes. Néanmoins nous n'exploitons pas cette hypothèse dans notre approche ; en particulier nous ne modélisons pas le fond, comme cela est fait généralement dans ce contexte.

4.5.1 Détection d'objets rigides.

La séquence présentée dans la figure 4.13 montre 10 images sur les 1000 de la séquence acquise à 30 Hz depuis une caméra fixe dans un carrefour [3]. Dans cette séquence nous trouvons des voitures lointaines, donc avec une petite échelle, et se déplaçant à des vitesses apparentes bien supérieures à celles traitées précédemment. La détection des voitures qui ont des mouvements dans le sens vertical aussi bien que dans le sens horizontal, est indiquée par des rectangles en jaune. Le suivi est effectué en exploitant le modèle basé sur des points d'intérêt.

Notons qu'au niveau de la troisième ligne, l'image à gauche ne contient plus de zones d'objets bien qu'il y ait des voitures dans l'image. Les images précédentes de la séquence montrent que les voitures précedemment détectées comme mobiles se sont arrêtées complètement, ce qui implique que les points détectés sur les voitures, ont vu leurs vitesses diminuées et s'éloigner de la vitesse originale de l'objet lorsqu'il a été détecté. À cause de cela tous les points sont éliminés du modèle parce que leurs vitesses ne respectent plus celles apprises au départ et réévaluées à chaque fois lors de la mise en correspondance. Aussitôt que les voitures reprennent leurs mouvements, elles sont de nouveau détectées et suivies dans les images suivantes.

Certains objets mobiles sur cette séquence sont identifiés avec deux ou plus rectangles jaunes, dont chacun est censé représenter un objet mobile différent. La fusion de ces régions dynamiques, bien qu'elles soient sur le même objet physique, n'est pas possible à cause des positions éloignées auxquelles elles se trouvent sur la même voiture. Une possible solution à ce problème serait le passage d'un modèle de points à un modèle de régions lors du suivi, mais il faudrait inclure un terme d'énergie lié à la vitesse car le test sur l'intensité comme cela a été évalué dans la section 4.3, risque de faire déborder chaque contour actif sur d'autres objets proches très similaires.

Nous concluons de l'expérimentation sur cette séquence que notre approche est capable de détecter et de suivre plusieurs objets dynamiques simultanément. Par ailleurs, il a permis d'évaluer aussi la possible fusion des objets existants avec les nouveaux objets détectés à chaque *temps de pistage*, ce qui garantit un bon départ pour l'extension vers le suivi d'objets depuis une caméra embarquée sur un vecteur lui-même mobile.

4.5.2 Détection d'objets non-rigides.

Un second test a été réalisé dans le cas des objets dynamiques non rigides. Pour cette expérience, nous avons choisi un environnement d'intérieur pour ne pas avoir de changements d'intensité importants qui perturbent le suivi des objets dynamiques. La séquence présentée en figure 4.14 a été réalisée depuis une caméra fixe à 30 Hz sur le couloir d'un centre commercial [4]. Nous présentons sur cette figure, le résultat de notre méthode sur 20 images, sur plus de 1000 images traitées dans cette séquence.

3. Séquence d'images téléchargée du site web [45] mises à disposition par KOGS/IAKS Universität Karlsruhe
4. Séquence d'images téléchargée du site web [43], mise à disposition par le *EC Funded CAVIAR project/IST 2001 37540*

Nous avons déjà présenté quelques résultats expérimentaux sur cette séquence dans la sous-section 4.3.3, mais en utilisant un modèle de régions au lieu des points. Ces tests étaient réalisés uniquement sur quelques images consécutives et pas sur toute la séquence, car des modifications sur la procédure de détection de régions doivent être réalisées. Alors nous avons traité la séquence sur 1000 images avec des résultats satisfaisantes dans la mesure où nous détectons tous les piétons qui rentrent et sortent du champ de l'image. Sauf que nous ne pouvons pas garantir le suivi du plus nombre possible des points sur le modèle des objets.

Au niveau des images 85 et 145 le personnage est bien détecté et suivi ; cela est illustrée par les rectangles en jaune. Au niveau de l'image 200 sa région de détection est déjà réduite à cause des mouvement de rotations que fait la personne. Malgré cela nous avons une très bonne performance dans le cas du suivi des deux personnages lorsqu'ils traversent la scène avec des directions de mouvement opposées. Donc les occultations ne sont pas un inconvénient dans cette méthode. Puis au niveau de l'image 480, il n'y a pas d'objet dynamique détecté bien qu'il y ait un personnage sur l'image. L'explication est la même que pour le cas de la séquence sur le carrefour, car la personne est debout en face d'une vitrine et elle n'a pas de mouvement ; ensuite, elle est détectée quelques images après dès qu'elle bouge de nouveau et suivie jusqu'à ce qu'elle sorte du champ de vue de la caméra. Finalement, dans la dernière ligne, sur la partie supérieure de l'image il y a un groupe de jeunes qui rentrent dans la scène : ils sont aussi identifiés avec en premier plan le personnage qui était déjà là au début de la séquence.

Nous constatons de bonnes performances de notre méthode sur cette séquence ; ce test était rendu complexe du fait du suivi de plusieurs objets dynamiques. Il est important de signaler que nous avons pu tester notre méthode sur des séquences acquises aussi bien en extérieur qu'en intérieur sans aucune modification dans les paramètres internes des différentes fonctions, principalement au niveau du nombre des points à analyser et du temps de pistage.

4.6 Conclusions

Dans ce chapitre nous avons étudié une méthode de suivi de plusieurs cibles définies à partir des groupes de points d'intérêt extraits par la méthode décrite dans le chapitre précédent. Ce suivi utilise le filtre de Kalman : l'évaluation de sa performance a été validée en utilisant le test $NEES$ d'innovation quadratique normalisée. Nous avons ainsi analysé le modèle de mouvement apparent à vitesse constante utilisé pour estimer le déplacement de l'objet dans les séquences d'images.

Principalement dans ce chapitre, nous avons proposé un modèle adapté pour caractériser les régions dans l'image, afin de les suivre et les mettre à jour en fonction de ce qui survient dans l'environnement. L'originalité de ce modèle consiste à ne pas utiliser la sortie de la méthode de *clustering*, c'est-à-dire l'ensemble des points présents dans l'image dans laquelle l'objet a été détecté. Ainsi ces points d'intérêt sont uniquement utilisés pour former les pistes, puis pour la segmentation initiale de régions correspondantes à des objets mobiles dans la scène : ensuite, ces points ne sont pas inclus dans le modèle. Cela évite les problèmes d'occultation, c'est-à-dire d'associer au modèle de l'objet des points qui ne sont plus visibles sur l'image courante. Pour avoir une meilleur fiabilité, le modèle d'un objet dynamique, détecté par un ensemble de points d'intérêt, est ensuite caractérisé par une région de l'image décrite par :

— un contour actif, défini par des points de contrôle et des B-splines cubiques,
— un modèle de mouvement de la région, appliqué uniquement sur le barycentre,
— et les attributs d'apparence pour tous les pixels internes à cette région ; l'apparence est actuellement décrite uniquement par l'intensité ; de nouvelles caractéristiques (couleur, variance de

l'intensité, ou histogrammes) devraient être évalués .

Chaque région est initialisée à partir du groupe de points sortant du *clustering*, puis déformée pour atteindre les limites les plus proches de la silhouette de l'objet. La convergence est obtenue par optimisation vers une courbe qui équilibre des forces définies à partir des intensités des pixels internes et externes à la région. C'est ce modèle que nous mettons à jour à chaque itération du suivi, à partir d'une position prédite par le filtre de Kalman.

Enfin nous avons intégré la méthode de suivi des objets avec la méthode de détection décrite au chapitre précédent ; au final notre système permet de suivre simultanément plusieurs cibles différentes en détectant des nouveaux objets mobiles et en les évaluant pour une fusion avec les cibles existantes.

Les performances de cette méthode exécutée continuellement le long de la séquence ont été présentées sur des images acquises par des caméras fixes. La première séquence a été acquise en milieu urbain, dans un carrefour où il n'y a que des objets rigides. La plupart des voitures sont détectées malgré la faible résolution. La seconde séquence a été acquise dans un centre commercial où il n'y a que des objets non rigides. Les piétons sont bien détectés tant qu'ils sont proches de la caméra mais il reste à traiter le cas des personnes décrites par plusieurs régions mobiles, car souvent les pieds sont détectés comme des objets différents du reste du corps.

Nous avons fait face à ce problème en adoptant le modèle de région pour chaque objet mobile. Les résultats obtenus sur chaque image traitée sont continuellement améliorés le long de la séquence, sans pour le moment avoir une mesure plus discriminante qui permette d'éviter les possibles fusions avec les objets du fond. Cependant, nous pouvons conclure que l'extension de notre modèle de suivi d'objets en utilisant des régions, permet d'obtenir des résultats plus stables, surtout dans le cas d'objets rigides.

FIGURE 4.13 – Résultats de la détection et du suivi d'objets dynamiques rigides qui se déplacent autour d'un carrefour. Plusieurs cibles avec des directions de mouvement différentes et variables sont détectées au long de 1000 images, avec $\Delta t = 0.03$.

FIGURE 4.14 – Résultats de la détection et du suivi d'objets dynamiques non-rigides sur des images acquises à 30 Hz par une caméra fixe. Plusieurs cibles avec des directions de mouvement différents et variables, sont détectées au long de 1000 images, avec $\Delta t = 0.03s$. (acquisition à 30Hz).

FIGURE 4.15 – Résultats de la détection et du suivi d'un seul objet dynamique non-rigide, dans ce cas un piéton sur la même séquence qu'en figure 4.14.

Chapitre 5

Intégration de notre approche dans le système de navigation d'un véhicule autonome

Sommaire

5.1 Introduction

La méthodologie proposée dans les chapitres précédents implique certaines contraintes à satisfaire lors de l'exécution de nos algorithmes ; la plus importante concerne la limitation de la vitesse pendant la

navigation du robot. Cette valeur maximale de vitesse dépent de la fréquence de traitement des images, plus exactement de la latence entre acquisition d'une image et moment où un objet potentiellement à risque est détecté. Cette contrainte est importante dans des environnements naturels non contrôlés, mais aussi pour l'aide à la conduite de véhicules (ADAS) sur autoroutes ou en milieu urbain. Dans le but de permettre des applications plus générales de notre algorithme de détection et de suivi d'objets mobiles, la prise en compte de l'estimation des déplacements de la caméra pendant l'exécution d'un mouvement par la robot, est le principal sujet évoqué dans ce chapitre.

La tâche de détection et de suivi d'obstacles est essentielle en robotique dans différents contextes, par exemple pour des robots collaboratifs avec interaction homme-robot, ou encore pour des robots autonomes avec exploration d'un environnement *a priori* inconnu. L'approche que nous proposons peut être intégrée dans un module indépendant qui détecte des ensembles de points d'intérêt et qui valide que ces ensembles correspondent à des points 3D situés sur des composantes dynamiques ou statiques dans l'environnement. Cette modularité rend possible la coopération de notre algorithme avec un module de cartographie et localisation simultanée, bien connue dans la littérature comme SLAM-MOT (Simultaneous Localization and Mapping-Moving Objets Tracking). La stratégie de coopération que nous proposons est basée sur celle développée par Migliore et al [77] qui traite séparément la tâche du SLAM de celle du MOT en les considérant comme deux processus qui fonctionnent parallèlement ; l'échange de données 2D produites par notre module et de données 3D issues de module SLAM, est réalisé au moment de faire la mise à jour de la carte stochastique. À la différence du système proposé par Migliore, cet échange de données effectué lors du traitement de la séquence d'images acquises sur le robot, est cadencé à la fréquence de notre module de détection. Ce sont les points considérés comme statiques depuis les deux modules qui seront partagés ; nous proposons donc une stratégie de synchronisation et de couplage entre les deux modules afin que les données 3D soient correctement ajoutés à la carte stochastique à partir des observations données par les données 2D.

Ce chapitre présente d'abord un bref état de l'art des différents travaux dédiés à l'estimation du mouvement du robot. Cela nous permet de décrire ensuite la stratégie de compensation de ce mouvement, compensation nécessaire pour valider les objets dynamiques détectés par notre méthode. Puis, nous présentons une brève analyse des méthodologies proposées pour résoudre le problème du SLAM-MOT, avant de décrire notre proposition sur un système complet de navigation avec détection et suivi d'obstacles mobiles. Ce système doit satisfaire des contraintes sévères de réactivité : nous proposons donc une architecture parallèle mise en œuvre avec l'environnement de simulation *Jafar*. Finalement, nous illustrons nos contributions par les résultats de nos algorithmes de détection d'objets mobiles avec compensation du mouvement, appliqués sur des séquences d'images acquises sur des scènes dynamiques réelles depuis une caméra embarquée sur un robot se déplaçant d'abord avec une vitesse faible, puis à une vitesse plus élevée.

5.2 Compensation du mouvement du robot

L'objectif de cette section consiste à estimer le mouvement propre du robot (appelé aussi ego-motion) et à le compenser sur les images acquises par la caméra lors de la navigation du robot. Nous avons pu constater que la détection d'objets mobiles sans compensation du mouvement fonctionne correctement uniquement si la vitesse du robot ne dépasse pas une certaine valeur ; une augmentation de cette vitesse de déplacement implique de compenser le mouvement de la caméra. En effet, si l'orientation relative entre le robot et l'environnement, change continuellement avec une magnitude non maîtrisée, les mesures de déplacement des points détectés et suivis dans les images, sont trop perturbées par le mouvement du

robot, ce qui rend difficile de décider si ces points correspondent à des entités statiques ou dynamiques dans la scène, et donc ce qui induit des fausses détections d'objets mobiles. Un exemple de ces fausses détections est illustré sur la figure 5.1 ; cette séquence d'images a été acquises depuis le robot *Dala* lors de sa navigation autour du laboratoire à la vitesse maximale du robot (2 m/s).

Dans cette figure, le robot observe une voiture en mouvement ; elle a été bien détectée par notre approche. Cependant un grand objet "dynamique" est aussi détecté, bien que le seul objet avec mouvement propre soit la voiture. Cette erreur est due aux larges déplacements des points d'intérêt entre les images successives, donc le flot optique créé par le mouvement du robot.

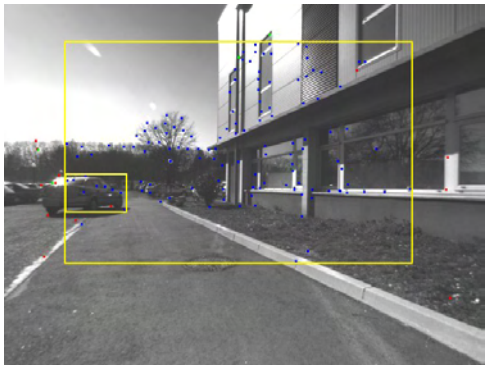


FIGURE 5.1 – Résultats de la détection sans compensation de mouvement : un grand objet dynamique est détecté, alors que la voiture est le seul objet en mouvement dans la scène.

Nous avons choisi d'estimer le mouvement en utilisant la centrale inertielle montée sur le robot exploité dans nos expérimentations, car les mesures de déplacement du robot sont estimées par un module SLAM, avec lequel nous allons partager les informations extraites de chaque image. Ce module SLAM utilise aussi les données inertielles (procédure décrite dans la section 5.3.2 de ce chapitre). Pour faire cette estimation nous faisons plusieurs hypothèses, par exemple, que la caméra est montée sur le robot de manière fixe et que sa vitesse est la même que celle du robot. Ainsi, le modèle que nous exploitons dans nos travaux, considère uniquement les degrés de liberté du robot ce qui rend l'estimation du mouvement propre plus stable et exact.

Dans cette section, nous décrirons d'abord la méthode de rétroprojection des points extraits du plan image vers le monde, plus précisément la rétroprojection des points extraits en dessous de la ligne d'horizon vers le plan du sol ; cette opération appliquée à l'image acquise à l'instant t,permet de reconstruire les points 3D supposés être sur le plan du sol, car leur hauteur est proche de zéro. Puis nous appliquerons le processus inverse, soit la projection des points 3D reconstruits sur le plan du sol vers le plan image acquis à l'instant suivant $t + 1$. Grâce à cette transformation les points 3D statiques qui sont au niveau du sol seront identifiés et donc ils ne seront pas considérés comme des possibles points dynamiques.

Un état de l'art sur l'estimation du mouvement (ou *egomotion*) sur des véhicules intelligents sera tout d'abord présentée. Puis, nous donnerons les informations essentielles sur les transformations géométriques induites par rétroprojection et projection depuis une image et sur la cinématique des corps rigides. Nous expliquerons comment appliquer la transformation de perspective inverse (ou *IPM : inverse perspective mapping*), pour corriger la détection de faux objets dynamiques. En fin de cette section seront analysés les résultats obtenus par cette stratégie mise en œuvre dans notre approche.

5.2.1 Différentes approches pour l'estimation du mouvement

Le déplacement de la caméra entre deux instants t et $t+1$ peut être estimé en utilisant directement les images pour évaluer la contrainte épipolaire [40]. Si cette contrainte est définie en fonction de la matrice de rotation et de translation alors la procédure est appelée comme discrète, alors qu'elle est différentielle si la contrainte est définie en fonction des vitesses linéaire et angulaire [4] du déplacement. L'expression de la contrainte épipolaire pour un point dans le plan image à l'instant de temps t vers le point correspondant dans le plan image à l'instant $t+1$ nécessite le calcul de la matrice fondamentale. Une étude des différentes méthodes pour calculer la matrice fondamentale est présentée dans [3].

De nopmbreux travaux sur la vision par ordinateur sont consacrés à l'aide à la conduite de véhicules (ADAS), notammemt sur les autoroutes, à partir de données extraites depuis une caméra embarquée. La stratégie couramment utilisée dans ce contexte applicatif, consiste à estimer le mouvement en calculant la matrice fondamentale depuis des données éparses en ayant comme objectif de détecter les objets mobiles [102]. Avec le même objectif il existe des approches qui utilisent le flot optique dense dans le calcul de la matrice fondamentale, pour ensuite évaluer d'autres contraintes de la géométrie épipolaire [59]. Dans [60] les auteurs ont étendu cette approche à un système stéréo, pour construire une carte de disparité, puis une image 3D décrivant la scène perçue. Plus récemment, cette méthodologie continue a être utilisée dans un cadre de filtrage par Kalman étendu (EKF) [84] en parallélisant la mise en correspondance des points sur une carte graphique (GPU). D'autres travaux proposés par Onkarappa et al. [80] utilisent une technique de flot optique dense variationnel ($TV-L^1$ Optical Flow) sur seulement une zone identifiée du sol et modélisent le problème d'estimation du déplacement comme une estimation de la ligne d'horizon. Néanmoins, la stratégie d'estimation du mouvement depuis les images a l'inconvénient de ne pas bien fonctionner dans toutes les situations où les données sont trop éparses ou trop bruitées, par exemple pour des images acquises dans la nuit ou avec de mauvais conditions météorologiques.

D'autre part, les déplacements de la caméra peuvent être obtenus à l'aide d'une centrale inertielle (*Inertial Mesurements Units, IMU*) montée sur le robot. Il est alors nécessaire que la centrale inertielle soit fixée de manière rigide et connue à la caméra, ce qui peut nécessiter des travaux mécaniques compliqués.

Afin de compenser les inconvénients liés à la centrale inertielle et à l'estimation basée sur des images, récemment dans Hwangbo et al. [50], les auteurs proposent un système caméra-IMU pour acquérir les images et mesurer les déplacements. Ensuite, ils proposent une stratégie pour fusionner l'information des deux dispositifs mis en œuvre sur un GPU. Malgré les bonnes performances, nous trouvons que cette stratégie n'est pas très adaptée à notre problématique de navigation robotique car en plus du fait que l'unité GPU requiert beaucoup de puissance, le logiciel n'est pas compatible avec celui que nous utilisons.

La solution que nous avons trouvée la plus adaptée à notre méthode consiste à récupérer les mesures de mouvement du robot depuis la centrale inertielle. Nous avons choisi cette option principalement afin de compléter les informations données sur la scène courante par la luminosité mesurée par la caméra, avec d'autres caractéristiques. Avec les mesures inertielles, il est possible d'estimer à court terme la nouvelle position de la caméra, même dans les cas où il y a des mouvements brusques exécutés par le robot ; un tel mouvement rapide peut donner une image floue à partir de laquelle il est difficile d'estimer les paramètres de la matrice fondamentale. Dans la suite nous expliquons la méthode utilisée pour estimer les déplacements en utilisant la centrale inertielle montée sur le robot.

5.2.2 Modèle de projection de la caméra

Le modèle classique d'une caméra se fonde sur des transformations :

1. D'abord, la projection des points 3D de la scène via des rayons de lumière incidents passant au centre de l'objectif de la caméra, vers des points sur le plan photosensible du capteur.
2. Puis ces projections sont converties en une luminance (ou une couleur) dans une image digitale construite la caméra.

Du point de vue purement de la perspective, un rayon est créé depuis un point 3D du monde vers le centre de la caméra, cette procédure est appelée projection centrale. Ensuite chaque rayon est propagé par le système optique jusqu'à l'intersection avec le plan image où il forme une image du point 3D [33]. La construction d'une image est normalement mise en œuvre en supposant une projection idéale avec une distorsion optique minimale. La figure 5.2 montre le modèle *pinhole*, très répandu dans la littérature de vision par ordinateur, modèle fondé sur cette projection centrale. Ce modèle considère que la distance focale f est équivalente à la distance entre le centre optique C et le plan image. Dans le cas où cette focale f du modèle *pinhole* est l'unité, i.e. $f = 1$ alors le modèle est considéré comme *normalisée* et on obtiendra une image *normalisée*. Le centre optique se trouve sur le plan focal, ce plan est perpendiculaire à l'axe optique et parallèle au plan image. Le point d'intersection de l'axe optique avec le plan image est appelé point principal.

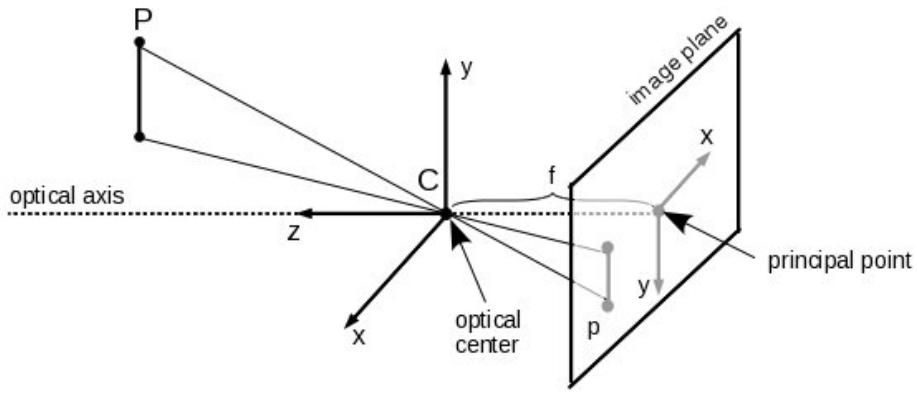


FIGURE 5.2 – Modèle de projection *pinhole* de la caméra.

Deux transformations nous permettent d'associer la position d'un point de l'espace 3D aux cordonnées pixel de sa projection dans l'image :

1. La projection géométrique du point 3D sur le plan image 2D.
2. La transformation affine qui change les coordonnées d'un point du plan image du repère métrique de la caméra en un repère pixel lié à l'image.

La première transformation est définie à partir des triangles semblables montrés sur la figure 5.2 où le point $\mathbf{p} = (x, y)^T$ sur le plan image associé au point 3D $\mathbf{P} = (X, Y, Z)^T$, avec des coordonnées

exprimées dans le repère de la caméra positionné sur le centre optique C, est donné par :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{5.1}$$

Ensuite, cette projection perspective du point *P*, peut être exprimée en utilisant les coordonnées homogènes $p = (x, y, 1)^T$ sur le plan image et $P = (X, Y, Z, 1)^T$ dans l'espace, obtenues à partir des cordonnées euclidiennes en leur rajoutant un 1. Ainsi, la projection de l'équation 5.1 est réécrite par l'équation matricielle suivante :

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim s \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{M}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{5.2}$$

où $\mathbf{M}$ est la matrice de projection dans l'espace $\Re^{3\times4}$. Normalement, cette matrice est décomposée en :

$$M = M_f M_0 = \underbrace{\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{M_f}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{M_0}} \tag{5.3}$$

où M_0 est la matrice de projection normalisée qui permet la projection du plan image avec une distance focale de $z = 1$ pour tous les points du plan image. La matrice $\mathbf{M_f}$ contient le facteur d'échelle pour avoir le plan image à une distance $z = f$ différente de l'unité. L'équation 5.2 qui transforme le point 3D $P = (X, Y, Z, 1)^T$ en un point $p = (x, y, 1)^T$ sur le plan image est réécrite comme :

$$s\mathbf{p} = M_f M_0 \mathbf{P} \tag{5.4}$$

En outre, le modèle *pinhole* est une projection idéale qui ne considère pas les erreurs dans la formation d'une image depuis un système optique réel, spécialement quand l'objectif monté sur la caméra est de basse qualité ou a une focale courte (appelé *grand angle*). La figure 5.3 montre une image avec une distorsion typique, notamment sur les zones éloignées du centre de l'image, par exemple le poteau qui est au premier plan : il s'agit d'une distorsion radiale [40].

FIGURE 5.3 – Déformation radiale lors de la formation de l'image par une caméra avec une focale courte, principalement sur les bords de l'image.

Les coordonnées distordues d'un point de l'image $\mathbf{p}_d = (x_d, y_d)^T$ exprimées en fonction des coordonnées obtenues pour une projection centrale parfaite $\mathbf{p} = (x, y)^T$, sont données par l'équation 5.5 avec des coefficients de distorsion k_i, $\| \mathbf{p} \|$ étant la distance du pixel au centre de l'image :

$$\mathbf{p}_d = \left(1 + k_1 \| \mathbf{p} \|^2 + k_2 \| \mathbf{p} \|^4\right) \mathbf{p} \tag{5.5}$$

L'image délivrée par la caméra est donnée par un tableau, composée de pixels, désignés par des coordonnées entières à partir du coin supérieur gauche. Les pixels ne sont pas forcément carrés et parfois non-orthogonaux ; par conséquent des facteurs d'échelle en ligne et colonne, et un point principal, projection du centre optique sur le plan image, doivent être considérés. Ces répercussions sont prises en compte pour formuler un modèle de transformation affine afin de convertir les coordonnées métriques sur le plan image (ici coordonnées distordues) $\mathbf{p_d} = (x_d, y_d, 1)^T$ en coordonnées pixels $\mathbf{p_{ima}} = (u, v, 1)^T$,

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \underbrace{\begin{bmatrix} s_u & s_\theta & u_0 \\ 0 & s_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{M_s}} \begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix} \tag{5.6}$$

où u_0 et v_0 sont les coordonnées en pixels du point principal ; s_u et s_v (pixels par mètre) sont respectivement les facteurs d'échelle vertical et horizontal ; le paramètre d'asymétrie ou "*skew*" s_θ traduit le fait que lignes et colonnes ne sont pas strictement orthogonales ; ce paramètre est souvent proche de zéro. Si toutes les distorsions sont négligeables, alors $\mathbf{p_d} = \mathbf{p}$ et l'équation 5.6 est une transformation affine représentant un changement d'échelle, une rotation et une translation. La relation entre les coordonnées $(X, Y, Z)^T$ du point $\mathbf{P}$ et les coordonnées image (u, v) du point $\mathbf{p_{ima}}$ est donnée par :

$$u = s_u f \frac{X}{Z} + u_0 \qquad v = s_v f \frac{Y}{Z} + v_0 \tag{5.7}$$

En multipliant l'équation de transformation affine 5.6 avec celle de la projection perspective 5.2 nous pouvons obtenir la matrice des paramètres intrinsèques $\mathbf{K}$:

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \underbrace{\begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{I}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{5.8}$$

où les paramètres $\alpha_u = s_u f$, $\alpha_v = s_v f$, u_0 et v_0 sont les paramètres intrinsèques de la caméra et $\mathbf{I}$ est appelée la matrice intrinséque ; notons que, pour pouvoir inverser cette matrice, la formulation $\mathbf{I}$ est souvent remplacée par une formulation $\mathbf{K}$ dans laquelle la dernière colonne $(000)^T$ est retirée.

Une calibration préalable de la caméra sera nécessaire pour estimer les paramètres intrinsèques. Ce modèle fondamental de la caméra permet de calculer le rayon optique associé à un quelconque pixel dans l'image parfaire, ou corrigée des distorsions. Le rayon optique est une droite, associée à un pixel de coordonnées $(uv)^T$ de l'image, sur laquelle se trouve le point 3D $(XYZ)^T$ correspondant. Cependant, aucune information par rapport à la profondeur Z du point n'est connue parce qu'au moment de la projection, une dimension est perdue. Le modèle inverse de projection perspective de la caméra (on parle de rétroprojection) est donc utilisé pour transformer le point $\mathbf{p_i} = (u, v)^T$ dans l'image en point 3D $\mathbf{P} = (X, Y, Z)^T$.

Ce modèle inverse permet de déterminer, par exemple, la pose d'un objet connu par un modèle, par rapport au repère de la caméra. Cependant, quand on ne dispose pas de connaissance sur la scène observée, cette transformation ne donne pas le facteur d'échelle correspondant à la dimension perdue. Pour

cette raison, la projection du point 3D est faite sur le plan image normalisé, puis le pixel est rétroprojeté vers une profondeur connue ou estimée. Le modèle inverse est fait en deux étapes, toujours en supposant que la distorsion est négligeable ou qu'elle est corrigée a priori :

1. La projection inverse est la transformation du pixel $\mathbf{p_i} = (u, v, 1)^T$ connu par ses coordonnées dans l'image, dans le point $\mathbf{p} = (x, y, 1)^T$ exprimé en coordonnées métriques du plan image normalisé. Cette transformation est définie par l'inverse de l'équation 5.8 comme :

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \underbrace{\begin{bmatrix} 1/\alpha_u & -\alpha_\theta/(\alpha_u\alpha_v) & (\alpha_\theta v_0 - \alpha_v u_0)/(\alpha_u\alpha_v) \\ 0 & 1/\alpha_v & -v_0/\alpha_v \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K^{-1}}} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{5.9}$$

2. La projection inverse du point depuis le plan image normalisé est utilisée pour transformer le point $\mathbf{p} = (x, y, 1)^T$ sur le plan image avec $f = 1$ au point 3D $\mathbf{P} = (X, Y, Z)^T$. Cette projection est définie pour l'équation de la ligne droite qui passe par le point du centre optique et le point (x, y) sur le plan image (voir figure 5.2). L'équation de projection est écrite sous sa forme paramétrique comme :

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = s \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{5.10}$$

où s représente la "*profondeur*" du point 3D $\mathbf{P} = (X, Y, Z)^T$ par rapport au plan focal. Cette distance peut être calculée à partir de l'équation de la ligne droite, soit si l'on dispose de connaissance *a priori*, soit par estimation de la profondeur quand on dispose de plusieurs images (stéréovision ou *Structure From Motion*).

Dans notre application, notre connaissance *a priori* de la profondeur est réduite aux points qui se trouvent sur le sol. Ainsi, les points de l'image détectés comme mobiles, mais qui appartiennent au sol, ne seront pas associés à des points 3D portés par des objets dynamiques ; ils seront donc éliminés du module de suivi.

5.2.3 Mouvement de corps rigides

Nous nous sommes intéressée à l'analyse du mouvement des corps rigides, sans prendre en compte la masse ou la force de l'objet, donc en considérant uniquement sa cinématique [41]. Un repère est défini sur un corps rigide afin de décrire ses mouvements dans le temps. Les mouvements comportent deux transformations fondamentales : la translation et la rotation.

Dans le cas de notre robot, le mouvement sera mesuré par la centrale inertielle. La figure 5.4 montre les trois axes définis pour le robot X_R, Y_R, Z_R et les angles de rotation $\Delta\varphi, \Delta\alpha, \Delta\Psi$ (*roll, pitch, yaw*) correspondant respectivement aux rotations autour des trois axes. Ainsi un mouvement correspond à un déplacement et une rotation du repère du robot, ce qui donne six degrés de liberté.

Le mouvement de translation du robot est positif en direction des axes XYZ de la figure 5.4 ; notons que l'axe X correspond à l'axe longitudinal du robot, orienté vers l'avant . Par contre les angles sont considérés comme positifs en accord avec les conventions suivantes :

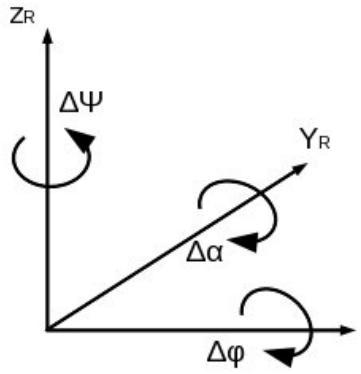


FIGURE 5.4 – Système de coordonnées du robot avec six degrés de liberté.

— $\Delta\varphi$ est positif si le côté droit du robot va vers le bas et le côté gauche se lève, autour de l'axe X.

— $\Delta\alpha$ est positif si le devant du robot, autour de l'axe Y, pointev vers le bas.

— $\Delta\Psi$ est positif si le robot tourne à gauche, autour de l'axe Z.

La position et l'orientation d'un objet par rapport à un repère de référence (par exemple le robot par rapport au monde) peuvent être décrites par le couple $\{\mathbf{d}, \mathbf{R}\}$, où $\mathbf{d}$ appelé *vecteur de translation*, décrit le vecteur liant l'objet et l'origine du repère, et $\mathbf{R}$ décrit l'orientation de l'objet par rapport au repère ; elle est appelée la *matrice de rotation*.

Le vecteur de translation est composé des trois coordonnées, une pour chaque axe du repère, ce vecteur est défini par l'équation suivante :

$$\mathbf{d} = \begin{pmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{pmatrix} \tag{5.11}$$

La matrice de rotation peut être représentée de différences façons : citons entre autres, les *angles d'Euler* et le *quaternion*. Nous utilisons la première de ces représentations pour manipuler les orientations.

Les angles d'Euler sont trois angles qui permettent de représenter une orientation dans l'espace 3D. Nous utilisons la convention ZYX correspondant à l'ordre dans lequel les rotations autour des trois axes sont exécutées. Cette convention prend en compte l'axe Z (l'angle yaw $\Delta\Psi$) comme l'axe autour duquel effectuer la première rotation, puis la deuxième rotation se fait autour de l'axe Y (l'angle pitch $\Delta\alpha$) qui s'est déjà déplacé du fait de la rotation autour de l'axe Z ; enfin, la troisième rotation se fait autour de l'axe X (l'angle roll $\Delta\varphi$) qui a subi déjà deux déplacements, du fait des rotations autour des axes Z et Y.

Si $\mathbf{e} = (\Delta\varphi, \Delta\alpha, \Delta\Psi)^T$ exprime une rotation par les angles d'Euler correspondant aux rotations en

roll, pitch et yaw, la matrice de rotation est obtenue à partir de l'équation suivante :

$$\mathbf{R}\left(\mathbf{e}\right) = \begin{bmatrix} \cos\Delta\alpha\cos\Delta\Psi & \sin\Delta\varphi\sin\Delta\alpha\cos\Delta\Psi - \cos\Delta\varphi\sin\Delta\Psi & \cos\Delta\varphi\sin\Delta\alpha\cos\Delta\Psi + \sin\Delta\varphi\sin\Delta\Psi \\ \cos\Delta\alpha\sin\Delta\Psi & \sin\Delta\varphi\sin\Delta\alpha\sin\Delta\Psi + \cos\Delta\varphi\cos\Delta\Psi & \cos\Delta\varphi\sin\Delta\alpha\sin\Delta\Psi - \sin\Delta\varphi\cos\Delta\Psi \\ -\sin\Delta\alpha & \sin\Delta\varphi\cos\Delta\alpha & \cos\Delta\varphi\cos\Delta\alpha \end{bmatrix} \tag{5.12}$$

Les angles d'Euler sont une représentation minimale pour les rotations. Ces angles ont l'inconvénient de présenter des discontinuités ; de ce fait on a besoin de prendre en compte un intervalle approprié pour chacun des angles :

$$\Delta\varphi \in [-\pi, \pi], \qquad \Delta\alpha \in [-\pi/2, \pi/2], \qquad \Delta\Psi \in [0, 2\pi]. \tag{5.13}$$

La figure 5.5 illustre la transformation linéaire 3D du repère robot au repère monde avec un vecteur de translation $\mathbf{d}$ et une matrice de rotation $\mathbf{R}$ qui est obtenue à partir des angles d'Euler ou des quaternions. Cette transformation linéaire de repère est effectuée en utilisant des coordonnées homogènes. Le point $\mathbf{P} \in \Re^3$ dans la représentation homogène est défini par l'équation 5.14 et la transformation linéaire du repère robot au repère monde et son inverse sont décrites par les équations 5.15 et 5.16 respectivement. Les équations 5.17 et 5.18 illustrent la notation de façon compacte pour les transformations linéaires homogènes.

$$\widehat{\mathbf{P}} = \begin{pmatrix} \mathbf{P} \\ 1 \end{pmatrix} \in \Re^4 \tag{5.14}$$

$$\begin{pmatrix} \mathbf{P}^M \\ 1 \end{pmatrix} = \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix}}_{\mathbf{H}^{MR}} \begin{pmatrix} \mathbf{P}^R \\ 1 \end{pmatrix} \tag{5.15}$$

$$\begin{pmatrix} \mathbf{P}^R \\ 1 \end{pmatrix} = \underbrace{\begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix}}_{\mathbf{H}^{RM}} \begin{pmatrix} \mathbf{P}^M \\ 1 \end{pmatrix} \tag{5.16}$$

$$\widehat{\mathbf{P}}^M = \mathbf{H}^{MR}\widehat{\mathbf{P}}^R \tag{5.17}$$

$$\widehat{\mathbf{P}}^R = \mathbf{H}^{RM}\widehat{\mathbf{P}}^M \tag{5.18}$$

La figure 5.6 illustre la transformation linéaire 3D du repère robot R au repère lié au plan image de la caméra C ; cette transformation est obtenue par l'équation suivante :

$$\widehat{\mathbf{P}}^F = \mathbf{H}^{FC}\mathbf{H}^{CR}\widehat{\mathbf{P}}^R \tag{5.19}$$

où $\mathbf{H}^{FC}$ et $\mathbf{H}^{CR}$ représentent les matrices des paramètres extrinsèques. La matrice $\mathbf{H}^{CR}$ est obtenue à partir de la position et l'orientation de la caméra qui est sur le robot avec une position de (X_C, Y_C, Z_C) et une orientation (φ, α, Ψ). La matrice $\mathbf{H}^{FC}$ est utilisée pour changer seulement l'orientation du repère caméra, en fonction des conventions utilisées pour définir un repère lié au plan image de cette caméra : axe Z sur l'axe optique, axes X et Y respectivement parallèles aux lignes et colonnes du plan image. Les deux matrices sont statiques, une fois qu'elles sont obtenues elles ne changent pas, réduisant ainsi le coût de calcul dans le traitement.

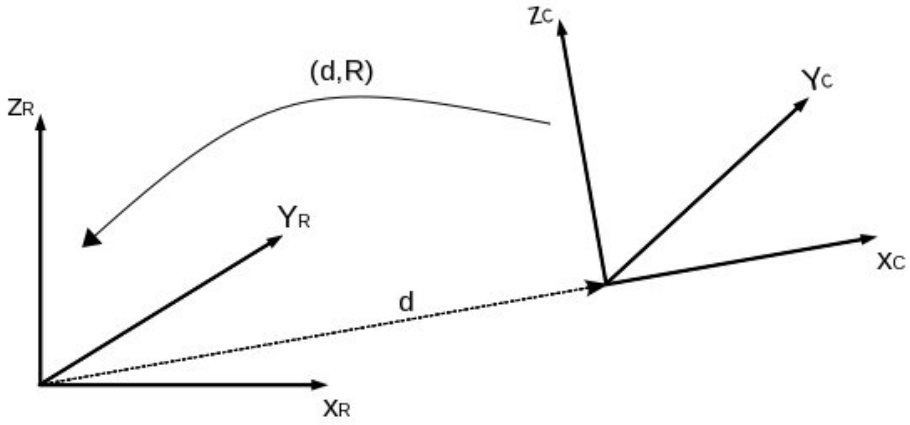


FIGURE 5.5 – Transformation 3D avec un vecteur de translation $\mathbf{d}$ et une matrice de rotation $\mathbf{R}$ du repère caméra au repère robot.

Une transformation linéaire 3D complète d'un point $\mathbf{P}^O$ qui appartient à un objet O, observé par la caméra C, est représentée par l'équation 5.20. Un point P a des coordonnées notées $\mathbf{P}^F$ dans le repère lié au plan image de la caméra C, et des coordonnées notées $\mathbf{P}^O$ par rapport au repère de l'objet.

$$\widehat{\mathbf{P}}^F = \mathbf{H}^{FC}\mathbf{H}^{CR}\mathbf{H}^{RM}\widehat{\mathbf{P}}^O \tag{5.20}$$

où $\mathbf{H}^{RM}$ représente la position du robot par rapport au repère du monde ; la caméra étant rigidement liée au robot, sa position dans le repère du monde, dépend de la position du robot. La figure 5.6 illustre les axes de référence utilisés pour la configuration de la scène pendant la navigation du robot. Le robot est un véhicule à 4 roues, non-holonome, donc il est limité à deux types de mouvement : de translation sur le plan XY et de rotation autour de l'axe Z.

5.2.4 Transformation Perspective Inverse

La perception d'une scène depuis la caméra dépend de l'angle de vue de la caméra au moment d'acquérir l'image et de la distance entre la caméra et chaque objet présent dans la scène. Étant donné que la caméra est montée à une position et une orientation fixes sur le robot, l'angle de vue est donné par l'orientation et la position du robot dans la scène plus l'orientation et la position de la caméra par rapport au robot. La distance entre la caméra et chaque objet ne peut pas être récupérée directement par une seule image, à cause de l'effet de perspective au moment de l'acquisition de l'image par la caméra. Cet effet se traduit par un facteur d'échelle à prendre en compte lorsque les images seront traitées. En particulier si ce traitement applique une approche géométrique, ce facteur influencera les informations contenues dans chaque pixel [7].

Pour faire face à ce problème, nous avons utilisé une approche géométrique. Cette approche consiste à projeter les pixels dans l'image acquise à l'instant $(t-1)$ vers l'image acquise à l'instant (t), en supposant que les points 3D sont au niveau du sol (au dessous de la ligne d'horizon). Donc notre méthode a pour but de détecter des pixels statiques au niveau du plan du sol ; étant statiques, ces pixels peuvent néanmoins être détectés comme dynamiques à cause d'un mouvement apparent important entre deux

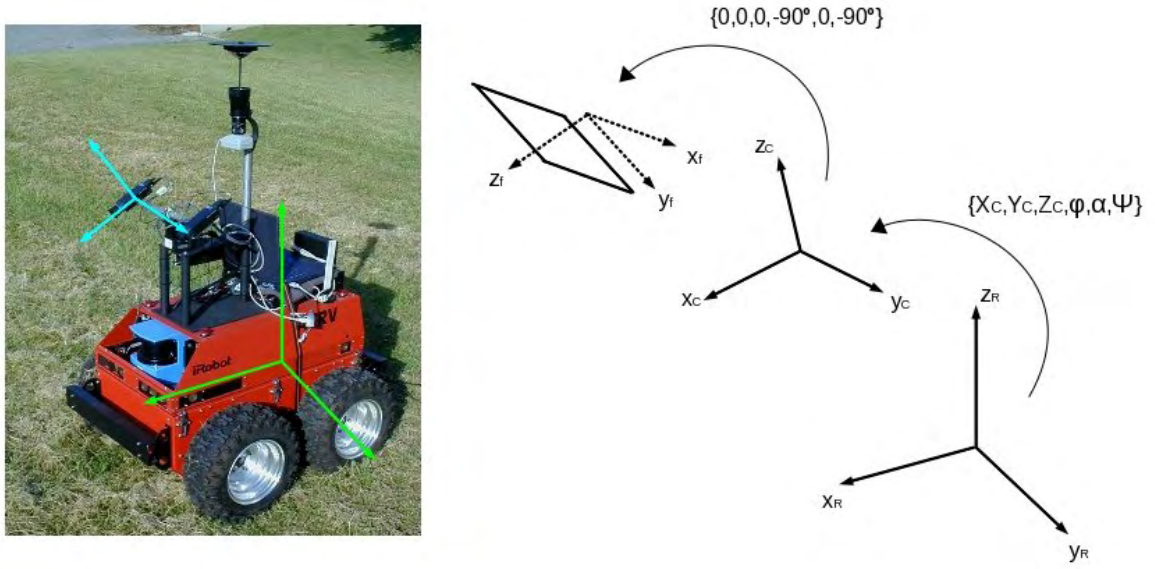


FIGURE 5.6 – L'image à gauche montre les repères du robot et de la caméra. Le schèma à droite montre la transformation 3D du repère robot au repère lié au plan image en passant par le repère de la caméra.

images successives. Les pixels qui satisfont une contrainte géométrique propre aux points du sol, seront considérés comme de faux points dynamiques et seront exclus de la méthode de suivi.

Soit $\widehat{\mathbf{p}}^i\left(t-1\right)=\left(u\left(t-1\right),v\left(t-1\right),1\right)^T$ les coordonnées d'un pixel sur l'image acquise au temps $t-1$, alors si on suppose que ce pixel correspond à un point 3D sur le sol, et si on connait le mouvement de la caméra entre les instant $t-1$ et t, il est possible de calculer la projection de ce point 3D vers l'image acquise au temps t, donc de donner les coordonnées estimées du pixel $\widetilde{\mathbf{p}}^i\left(t\right)=\left(\widetilde{u}\left(t\right),\widetilde{v}\left(t\right),1\right)^T$. Ces coordonnées sont calculées par l'équation suivante :

$$\widetilde{\mathbf{p}}^i\left(t\right)=\mathbf{K}\mathbf{H}^{FC}\mathbf{H}^{CR}\mathbf{H}^{ODO}\left(t-1,t\right)\mathbf{H}^{RC}\mathbf{H}^{CF}\mathbf{K}^{-1}\widehat{\mathbf{p}}^i\left(t-1\right) \tag{5.21}$$

où $\mathbf{K}$ et $\mathbf{K}^{-1}$ sont la matrice de paramètres intrinsèques et son inverse, respectivement. Les matrices $\mathbf{H}^{CF}$ et $\mathbf{H}^{FC}$ sont la rotation nécessaire pour passer du repère caméra au repère lié au plan image, et son inverse, respectivement. Les matrices $\mathbf{H}^{RC}$ et $\mathbf{H}^{CR}$ sont la matrice de transformation du repère caméra au repère robot et son inverse, respectivement. La matrice $\mathbf{H}^{ODO}\left(t-1,t\right)$ est la matrice de transformation du repère robot du temps $t-1$ au repère robot du temps t. Cette dernière matrice est obtenue par l'estimation du déplacement du robot, estimation qui change à chaque acquisition. De ce fait cette matrice est la seule qui doit être recalculée à chaque acquisition, puisque les paramètres intrinsèques et la position de la caméra par rapport au robot sont fixes.

Afin de réduire le temps de calcul nécessaire durant la navigation du robot, nous utilisons l'hypothèse que le sol est plat et nous appliquons une restriction sur le mouvement de notre robot, c'est-à-dire seulement trois degrés de liberté. Ainsi le calcul nécessaire pour obtenir la matrice $\mathbf{H}^{ODO}\left(t-1,t\right)$ est réduit seulement au déplacement sur le plan *XY* et à une rotation autour de l'axe *Z*.

Nous avons d'abord validé ces relations en simulation. La figure 5.7 présente la transformation géo-

métrique pour les quatre coins d'un rectangle détecté au niveau du sol, perçu depuis deux positions différentes du robot. Dans cette figure, à gauche est montrée la simulation de la scène en 3D : le rectangle sur le sol en couleur marron, les deux positions du repère du robot et de la caméra sont en couleur bleu et rouge respectivement, les plans image correspondants en couleur jaune, les projections des quatre coins du rectangle sur les plans images en couleur cyan et finalement les champs de vue sur le plan du sol en couleur verte ; ces champs de vue, projection du plan image sur le sol, ont une forme de trapéze.

La caméra est placée sur le robot dans la position $(X_C, Y_C, Z_C) = (0.5, 0, 0.4)$ avec un angle d'inclinaison de $\Delta\alpha = 20°$. Pour la première position simulée, le robot est à l'origine à l'instant $(t-1)$, nous utilisons U_{t-1} pour faire référence à cette position. Puis au temps (t), le robot se déplace à la position $U_t = (0.7, 1.2)$ sur le plan *XY* avec une orientation de $\Delta\Psi = -30°$.

À droite dans la même figure, le premier graphique en haut montre l'image du rectangle sur le sol, acquise depuis une caméra embarquée sur le robot à la position U_{t-1}. Les quatre cercles en couleur rouge représentent les quatre coins du rectangle. Le deuxième graphique est l'image du rectangle vue par la caméra depuis la deuxième position U_t. Ici les quatre coins du rectangle sont représentés par les carrés en couleur bleu. Le troisième graphique montre en rouge la transformation géométrique appliquée sur les coins observés en position U_{t-1}, pour estimer où ils seront observés en position U_t. On peut voir que puisque l'objet est sur le sol, les coordonnées estimées des coins du rectangle à partir des coordonnées obtenues en U_{t-1} et du mouvement de la caméra entre les instants t et $t-1$, correspondent avec les coordonnées projetées de ces coins à la position U_t ; la correspondance est presque exacte parmi les quatre coins du rectangle obtenus soit par la transformation exploitant l'estimation du mouvement de la caméra, soit par projection.

Il est donc possible de valider ces pixels comme appartenant au sol.

Le quatrième graphique est juste une vérification. Il a été fait aussi à partir de la même transformation, mais appliquée dans le sens inverse, c'est-à-dire, les coins observés à la deuxième position U_t sont transformés pour estimer où ils étaient observés dans l'image acquise depuis U_{t-1}. Donc, puisque nos hypothèses sont satisfaites, les coordonnées des coins obtenus dans le graphique 4, correspondent à ce qu'ils étaient dans le graphique 1.

Les simulations présentées dans la figure 5.7 ont permis de développer une stratégie afin d'identifier les points qui sont sur le sol parmi l'ensemble des points détectés comme dynamiques par notre algorithme. Pour ce faire, l'étape suivante consiste à projeter ces points vers le plan du sol. Alors, il faut refaire une transformation qui avait déjà été mise en œuvre dans la transformation géométrique. Afin de réduire le calcul nécessaire, nous avons reformulé notre transformation géométrique en utilisant la transformation Perspective Inverse (Inverse Perspective Mapping, IPM) qui a été développée par Mallot et al [70]. Cette transformation change l'angle de vue depuis lequel une scène plane est observée en permettant de supprimer l'effet de perspective dans les images [89] ; par exemple, on peut générer une image vue de dessus (*bird's eyes view*) d'une scène plane observée depuis une caméra montée sur un véhicule. Le changement de l'angle de vue permet de changer le domaine de perception des pixels de sorte que le contenu de l'information est distribuée de façon plus homogène sans effet perspective. Cela permet la mise en œuvre plus efficace d'autres traitements [7] comme la détection des bords d'une route ou la détection des marquages, qui deviennent parallèles dans l'image transformée. L'application de cette transformation a besoin de conditions spécifiques d'acquisition (connaissance de la position et l'orientation de la caméra sur le véhicule, des paramètres intrinséques...) et une hypothèse de planarité pour la scène observée dans l'image, hypothèse généralement définie comme une connaissance *a priori*) [82].

Cette approche divise notre première transformation géométrique en trois : les deux premières trans-

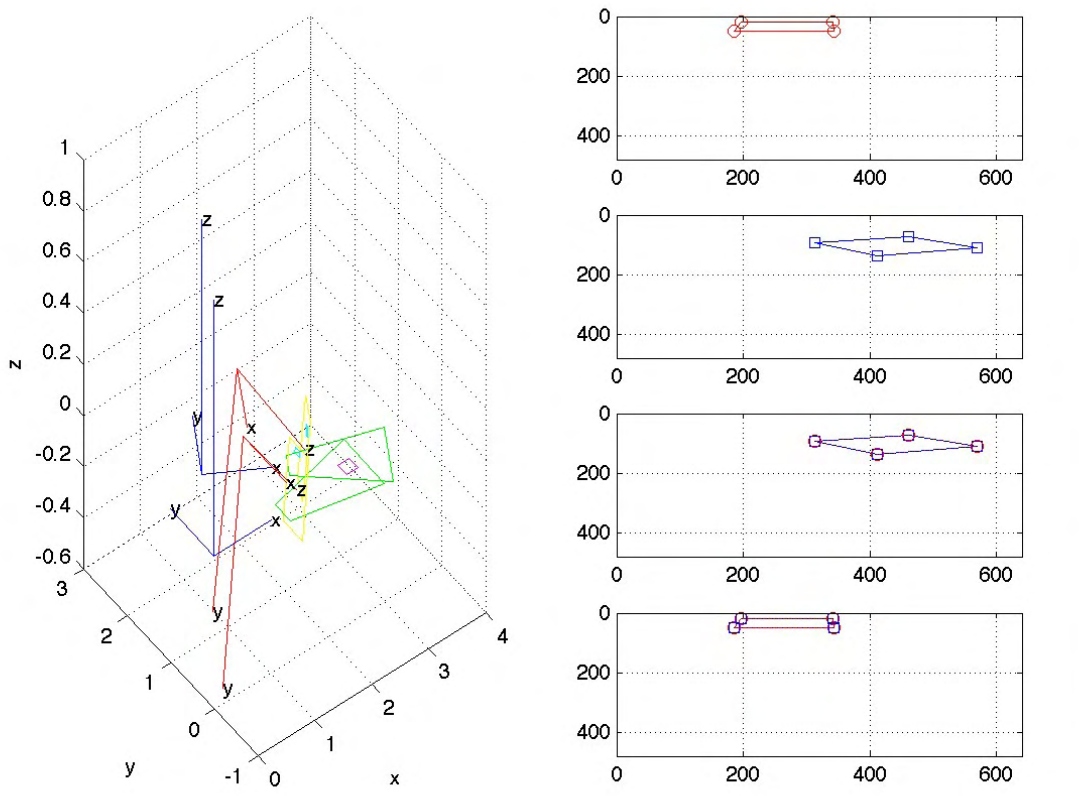


FIGURE 5.7 – Simulation de la transformation géométrique d'un rectangle au niveau du sol perçu depuis deux points de vue différents de la caméra.

formations sont les rétroprojections de chaque pixel des deux images sur le plan du sol, tandis que la troisième consiste à un mouvement 2D des points sur le plan du sol, avec une translation et une rotation. L'équation 5.22 présente les deux premières transformations qui ne dépendent pas de la position du robot ou du temps d'acquisition ; les deux transformations sont donc les mêmes. Dans l'équation 5.22, $\widehat{\mathbf{p}}^s$ représente les coordonnées homogènes du pixel $\widehat{\mathbf{p}}^i$ sur le plan du sol, c'est-à-dire, le sous-index *s* indique les coordonnées sur le sol tandis que le sous-index *i* indique les coordonnées sur l'image. Concernant la troisième transformation, il faut considérer la translation et la rotation de l'image acquise au temps $(t-1)$ déjà projeté sur le sol. Ces deux mouvements sont mesurés en fonction du déplacement du robot entre les acquisitions de l'image $(t-1)$ vers l'image (t) par la centrale inertielle du robot. L'équation 5.23 calcule les coordonnées estimées du pixel dans l'image au temps (t) en ajoutant la transformation odométrique [5] du robot aux coordonnées $\widehat{\mathbf{p}}^s\,(t-1)$ au temps précédent.

$$\widehat{\mathbf{p}}^s = \mathbf{H}^{RC}\mathbf{H}^{CF}\mathbf{K}^{-1}\widehat{\mathbf{p}}^i \tag{5.22}$$

$$\widetilde{\mathbf{p}}^s\,(t) = \mathbf{H}^{ODO}\,(t-1,t)\,\widehat{\mathbf{p}}^s\,(t-1) \tag{5.23}$$

5. En fait, nous n'utilisons pas les odomètres, mais une centrale inertielle pour estimer les mouvements du robot

Si $\widetilde{\mathbf{p}}^s(t)$ est égale à l'incertitude près à $\widehat{\mathbf{p}}^s(t)$, alors, le point 3d correspondant à ce pixel suivi entre les instants $t-1$ et t, est sur le plan du sol.

5.2.5 Estimation des déplacements du robot

La centrale inertielle du robot est équipée avec 3 accélératomètres qui mesurent les accélérations linéaires selon trois axes X, Y, Z et 3 gyromètres, qui mesurent les vitesses de rotation autour de ces mêmes trois axes. L'intégration simple (rotation) ou double (translation) des mesures successives obtenues de ces capteurs, permet d'estimer les déplacements effectués par la centrale depuis le temps initial. Si la centrale est positionnée sur le robot avec ces axes alignés sur ceux du robot, et en un point correspondant au centre de rotation (cela dépent du système de locomotion du dit robot), alors une transformation simple permet de mesurer la position et l'orientation du robot, donc son état donné par six degrés de liberté.

Nous avons validé notre approche en différé, en traitant des séquences d'images acquises depuis notre robot *Dala*. Les images sont stockées et, pour chacune, l'état du robot à l'instant de l'acquisition, est également mémorisé dans un fichier. Nous obtenons les mesures dans les six directions depuis 3 axes de référence différents [47] :

— **robotToSensor** : indique la position du capteur par rapport au robot

— **originToRobot** : indique la position du robot par rapport au monde

— **baseToRobot** : indique la position du robot dans une frame de base.

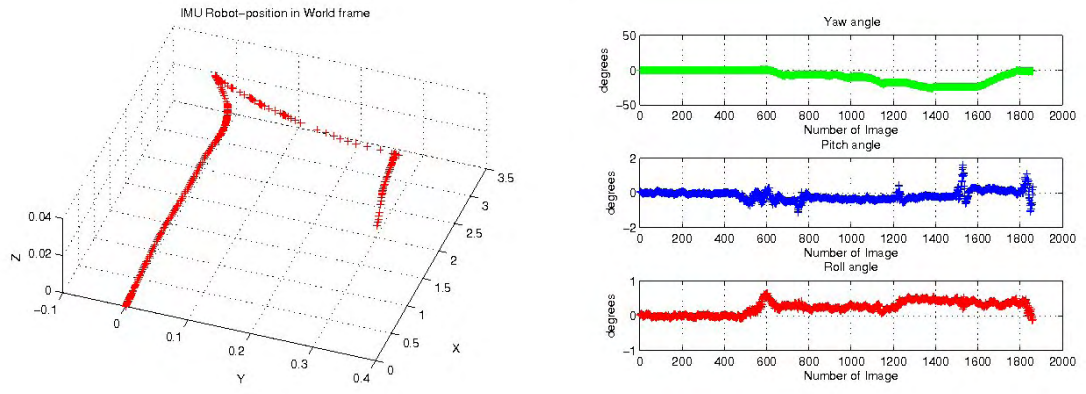
Pour la séquence présentée en figure 5.13 utilisée pour valider notre approche de détection et suivi des objets mobiles, avec compensation des mouvements du robot, la figure 5.8 montre les positions du robot sur le plan XY (le sol) données par la centrale inertielle lors de la navigation à l'extérieur du laboratoire. Cette information de la centrale inertielle nous permet de calculer la vitesse moyenne de navigation, qui pour cette séquence est d'environ 1 m/s.

Bien que la centrale inertielle soit une bonne solution, dans la pratique, nous sommes confrontée à deux problèmes. Le premier est que la centrale doit être couplée rigidement à la caméra ou au robot, autrement elle ne reproduira pas le mouvement de ces corps ; le robot, la centrale inertielle ainsi que la caméra doivent être calibrées entre eux ; en pratique la centrale est fixée sur le robot, avec ses axes alignés avec ceux du robot. Le deuxième est le problème de la synchronisation entre les acquisitions d'images et les acquisitions inertielles ; en pratique, les mesures inertielles sont acquises à 100Hz, alors que les images sont acquises à la fréquence vidéo, soit 25Hz. L'estimation courante de la position du robot est enregistrée à chaque acquisition. Une synchronisation plus fine, sur la base des temps précis d'acquisition des mesure, n'a pas été mise en oeuvre.

Comme nous l'avons déjà mentionné dans la section 5.2 une façon alternative d'estimer le mouvement propre d'une caméra, ou du robot sur laquelle cette caméra est embarquée, utilise uniquement les images et la théorie de la géométrie épipolaire.

5.3 La stratégie globale pour la navigation extérieure

La fonction robotique de haut niveau dans laquelle notre approche sera intégrée, concerne l'exploration d'un environnement non contrôlé, c'est-à-dire partagé avec d'autres objets, statiques ou dynamiques,



(a) (b)

FIGURE 5.8 – L'information obtenue depuis la centrale inertielle pour la séquence d'images de la figure 5.13. L'image (a) montre la trajectoire du robot réalisée depuis le repère du monde. L'image (b) donne l'information des trois angles de rotation (*yaw*, *pitch*, *roll*) .

donc le robot ne connaît pas à l'avance les comportements. Une fonction dédiée à la cartographie et la localisation simultanée (SLAM) doit être exécutée par le robot pendant qu'il se déplace ; il est nécessaire de prendre en compte que l'environnement ne contient pas uniquement des objets statiques. Cette considération rend beaucoup plus complexe cette fonction sur un système autonome, mais en même temps, elle la rend plus robuste et mieux adaptée pour être exécutée dans des environnements réels.

Un des objectifs de notre algorithme de détection et de suivi d'objets mobiles, consiste à identifier les points statiques les plus représentatifs (on parle de saillance pour qualifier la particularité d'un indice visuel, ce qui augmente la capacité de le retrouver dans les images successives) qui pourront être exploités par un autre module dédié au SLAM. Nous nous sommes intéressée à développer une stratégie de coopération afin de fusionner une approche de SLAM avec notre approche de détection et suivi d'objets mobiles, puis d'estimation de leurs positions et vitesses. Cette technique est connue sous le nom de SLAMMOT.

Dans la suite différentes méthodes pour résoudre le problème de SLAMMOT seront brièvement analysées, avant de présenter notre approche de coopération avec un module SLAM et les résultats des expérimentations.

5.3.1 État de l'art du SLAM mono-caméra

Une solution au problème de SLAMMOT depuis un robot mobile, avec un cadre mathématique bien posé, a été initialement proposée par Wang et al [99] et ensuite dans [100]. Dans ces travaux les auteurs montrent qu'il est possible de résoudre le problème en séparant les estimations des états *a posteriori* pour les données statiques (état donné par une carte stochastique, dont le premier élément donne l'état du robot) et les objets mobiles (états indépendants pour chaque objet). Ainsi, il est proposé une méthode de suivi basée sur la mise en correspondance entre les données actuelles de la carte SLAM et les données mesurées afin de (1) détecter les nouveaux objets mobiles par rapport à la liste courante des objets déjà détectés et aussi (2) pour estimer la position relative du robot. Leur algorithme est validé en analysant les données acquises depuis un télémètre laser 2D (balayage dans un plan horizontal) dans un environnement urbain. Les résultats de l'expérimentation exploitent la méthode *Feature matching* (appariements

de segments extraits de chaque coupe laser), mais ils ne présentent pas la robustesse nécessaire pour détecter les objets dans un environnement non structuré. Cet inconvénient provient du fait que la méthode extrait d'abord les indices d'intérêt des données sensorielles afin d'utiliser moins d'information et de satisfaire des contraintes temps réel. Les mesures sont donc des primitives éparses, ce qui rend moins discriminantes les données sensorielles exploitées pour détecter des objets mobiles.

Si le problème du SLAM est attaqué en utilisant une caméra, les algorithmes conçus sont appelés *Visual SLAM*. Dans la communauté robotique, la première approche de SLAM mono-caméra a été présenté par Davison et al. [24], avec un premier exposé qui a fait date en 2003. Les auteurs proposent, avec l'information acquise par une caméra, d'exploiter les informations d'orientation (on parle de *bearing-only SLAM*) fournies par des points d'intérêt extraits de l'image, afin d'estimer et de mettre à jour un état donnant à la fois la structure 3D de l'environnement et la position de la caméra ; cette méthode se fonde sur la triangulation afin de trouver les positions des points 3D observés par la caméra depuis différents points de vue. Un filtre de Kalman étendu accomplit la tâche de SLAM pour estimer en temps réel les six degrés de liberté définissant position et orientation de la caméra. L'initialisation de la profondeur de chaque point observé (limitée au maximum à 5 mètres) est faite par une approche non paramétrique ; la profondeur est d'abord estimée en utilisant un filtre à particules ; une fois que la distribution de la profondeur est proche d'une distribution normale, alors elle est intégrée dans la carte stochastique mise à jour par filtrage de Kalman étendu. Le principal inconvénient de cette stratégie retardée (*delayed*) est la non exploitation de points 3D qui sont au delà de 5 m. par rapport à la caméra, ce qui interdit l'exploitation de points à l'infini, pourtant de très bons indices pour extimer l'orientation de la caméra (évoquons les étoiles qu'exploitaient nos ancêtres, ou les capteurs stellaires montés sur les satellites).

Une autre méthode de *bearing-only SLAM* qui évite ce problème de retard, a été mise en œuvre par J.Solà [91], en intégrant une méthode non paramétrique fondée sur des sommes de gaussiennes, soit par vision monoculaire ou soit par vision binoculaire (Bicam SLAM). Dans ces travaux, les indices visuels sont détectés par l'opérateur de Harris ; un point d'intérêt est caractérisé par son voisinage afin de pouvoir le retrouver dans les images suivantes. Pour gagner en temps de calcul et pour garantir une bonne couverture des scènes perçues, l'image est partitionnée en zones rectangulaires de taille identique (par exemple, 10×10 fenêtres) ; à chaque itération, de nouveaux amers sont créés afin que les observations des amers existants couvrent au mieux l'image.

Pour chaque point détecté, une représentation du rayon optique par un mélange de gaussiennes à variances croissantes avec la profondeur, permet de représenter sans délai (*undelayed*) l'amer correspondant de manière probabiliste.

— Dans le cas monoculaire, cette approche multi-hypothèses permet de mettre à jour la carte SLAM sans retard. Chaque point extrait de l'image correspond à N points 3D potentiels. Dans une version plus complexe, chacun peut être considéré comme fixe ou mobile.

— Dans le cas stéréo, on détecte les indices visuels et on crée les rayons coniques dans une des images uniquement (généralement, la gauche). Mais cet amer est immédiatement projeté sur l'autre (donc la droite), en exploitant la géométrie épipolaire. Cela donne une zone de recherche de cet amer visuel. Il sera possible d'observer la profondeur d'un amer dès qu'il est observé la première fois, si l'intersection entre ces rayons coniques issus des deux images, tombe dans la zone d'observabilité de la stéréo, sinon on traite cet amer comme dans le cas monoculaire.

Pour chaque point ainsi reconstruit, on a N hypothèses de profondeur (typiquement 6 ou 7 en monoculaire, 2 ou 3 en stéréo, en dehors de la zone d'observabilité). Pour traiter l'image suivante, la nouvelle

position de la caméra (donc du robot si elle est embarquée) est d'abord prédite selon différents modèles de mouvement (typiquement vitesse constante, ou position/vitesse estimées par les capteurs inertiels ou par odométrie . . .).

Une approche dite de recherche active (*Active search*) est appliquée pour rechercher et mettre à jour à chaque itération, un nombre limité d'amer ; ceci permet de satisfaire les contraintes temps réel. Un amer sélectionné pour être corrigé est rétroprojeté (back projection) avec ces différentes profondeurs possibles. Cela donne des zones de recherche, définies avec des fonctions de densité de probabilité gaussienne, de sorte que l'observation de cet amer a une probabilité de 0,99 de s'y trouver.

A chaque itération, quand un amer est observé, les hypothèses sur sa profondeur sont évaluées, pondérées par un coefficient de vraisemblance. La carte est mise à jour en fonction de ces coefficients ; les hypothèses les moins probables sont détruites au fur et à mesure, jusqu'à trouver une seule solution pour chaque point. Cette solution est définie par une seule profondeur ; dans la version traitant des points dynamiques, elle donne aussi une seule vitesse (0 si le point est statique).

Un autre algorithme représentatif du SLAM visuel, a été proposé par Silveira et al. [88]. Les auteurs proposent une méthode directe qui suppose que l'image de la scène est localement plane. Les paramètres, comme le vecteur de la normale plus la distance au plan, pour chaque point d'intérêt extrait dans l'image, sont estimés simultanément avec les paramètres du mouvement propre en utilisant la mesure d'erreur SSD (Sum of Squared Differences). De plus, les changements d'illumination affine sont modélisés pour chaque point. Les estimations résultantes sont envoyées au filtre de Kalman pour compléter le processus du SLAM en ligne.

Les travaux récemment présentés sur le SLAM montrent des évolutions intéressants concernant ce problème ; les méthodes s'inspirant de la Vision, fondées sur des technique d'Optimisation (*Structure From Motion, Bundle Adjustment. . .*) sont considérées plus précises et plus robustes que les approches traditionnelles de la Robotique fondées sur des approches de filtrage (Kalman, Particle Filter...).

Cependant l'analyse approfondie d'une méthode SLAM spécifique est au delà des objectifs de cette thèse. Il est plutôt de notre intérêt de proposer une stratégie de coopération entre un algorithme SLAM et notre approche de détection et de suivi d'objets mobiles, et de démontrer que cette stratégie fait que la tâche de localisation et de cartographie génère moins d'incertitude et permet au robot de se localiser plus facilement. Dans la suite de cette section nous décrirons l'approche de coopération entre ces modules, avec une description simple d'un module SLAM et notre proposition pour la coopération avec notre module de détection et suivi d'objets mobiles.

5.3.2 Le module de SLAM

La méthode SLAM considérée ici applique une stratégie probabiliste exécutée sur un système robotique expérimental, équipé de capteurs de technologie spécifique. Cette tâche peut être vue comme une boucle où le robot acquiert une carte de son environnement et où simultanément il se localise de façon relative [95]. Soit $\mathbf{x}_t = (x, y, z, \alpha, \Psi, \varphi)^T$ la position et l'orientation à l'instant t du robot relativement à un système des coordonnées fixe choisi en fonction de l'application, qu'on appelle en général, le repère du monde. Un module SLAM est une métode itérativen qui consiste à estimer en ligne, la probabilité *a posteriori* de $\mathbf{x}_t$, et d'une carte m. La carte m généralement peut avoir différentes représentations ; cela peut être une liste d'amers (*landmarks*) ou une grille discrétisée associé au monde (grille 2D dite d'occupation, ou grille 3D dite d'élévation, selon la nature du sol plat ou accidenté).

La probabilité *a posteriori* est mise à jour en fonction des observations $\mathbf{z}_{1:t}$ données par les capteurs extéroceptifs et des entrées de contrôle $\mathbf{u}_{1:t}$, données par des capteurs proprioceptifs (centrale inertielle, codeurs, odométrie. . .) ou par les consignes transmises aux actionneurs. L'indice $1 : t$ dénote l'intervalle

des données de l'instant de temps 1 jusqu'à t. En tenant compte de ces paramètres, la tâche consiste à estimer :

$$p(\mathbf{x}_t, m|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{5.24}$$

ainsi la position courante la plus probable est celle qui maximise la probabilité *a posteriori* :

$$\hat{\mathbf{x}}_t = \arg\max_{\mathbf{x}_t, m} \; p(\mathbf{x}_t, m|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{5.25}$$

Depuis que la problématique du SLAM existe, donc depuis 1990 environ, la solution la plus fréquente au problème du SLAM en ligne est le filtre de Kalman étendu qui propose de bonnes performances si les "landmarks" utilisés sont suffisamment distincts. Pour le SLAM visuel, les approches par Optimisation gagnent en popularité, d'autant plus que les capacités de calcul disponibles sur les robots augmentent.

5.3.3 Le partage des données et la synchronisation des modules

L'intérêt de notre approche de suivi d'objets mobiles est de proposer une stratégie de coopération de données avec un module SLAM. La fonctionnalité du module SLAM sera vue par notre approche comme une "boite noire" avec des entrées et des sorties, et son exécution peut être faite de manière parallèle à notre approche avec un seul point de "rendez-vous" à la fin de chaque *temps de pistage* qui dicte la fin de chaque processus de détection dans notre algorithme. Cet schéma de fonctionnement est illustré dans la figure 5.9. Cette stratégie de coopération est similaire à celle proposée par Migliore et al. [77], à la différence près que les auteurs proposent une méthode différente pour la détection d'objets.

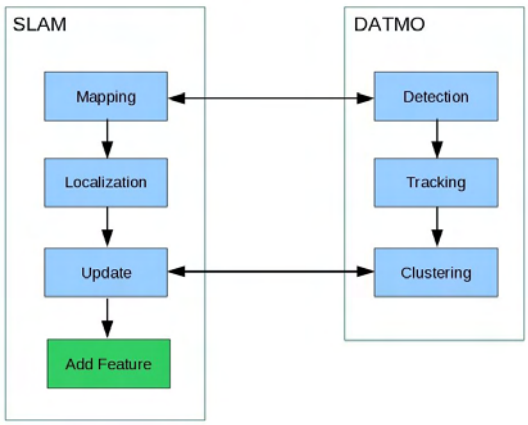


FIGURE 5.9 – Diagramme à blocs de la stratégie de coopération entre le module SLAM, qui construit le modèle pour les composantes statiques de l'environnement, et le module MOT qui s'occupe des composantes dynamiques.

5.4 Architecture matérielle et logiciel

5.4.1 Le robot d'expérimentation *Dala*

Une majorité des séquences présentées ici ont été réalisées sur le véhicule robotique tout terrain *Dala* (photo dans la figure 5.10). *Dala* est un robot d'extérieur de la société *iRobot* [48] équipé avec de

FIGURE 5.10 – *Dala*, véhicule robotique tout terrain.

multiples capteurs pour acquérir des mesures proprioceptives et extéroceptives. C'est un *rover* de type "*skid-steering*", qui peut être conduit manuellement en utilisant un *joystick* ou qui peut être commandé automatiquement en envoyant des consignes sur les micro-contrôleurs depuis les modules intégrés dans le système embarqué. Son châssis permet des déplacements en milieu accidenté mais sans grandes amplitudes. Le tableau 5.1 décrit plus en détails ses caractéristiques [66].

TABLE 5.1 – Table de caractéristiques de *Dala*.

Motricité	4 roues motrices
Vitesse maximale	2 m/s
Batteries	24V, 1440 W*hr*
Bus	PCI
Processeur	1 Intel Pentium IV
Système d'exploitation	Linux
Communication	*IEEE*802.11*b*
Capteurs	1 odomètre 12 capteurs ultra-sons 2 caméras *IEEE*1394 1 caméra panoramique
Dimensions	$105cm \times 80cm \times 100cm$
Poids	120 *kg*

Dans le cas de nos expérimentations et lors de l'acquisition des données, nous considérons la configuration suivante :

— **capteurs de vision** : un banc stéréo est situé à l'avant du robot à 0.975m du niveau du sol, l'axe

optique des caméras est parallèle avec l'axe *X* du robot et perpendiculaire à l'axe *Z*. Nous avons exploité et traité les images déjà rectifiées issues de la caméra gauche en niveaux de gris, de taille 640×480 pixels. Notons que ces images rectifiées sont aussi corrigées des distorsions optiques.

— **odomètre** : nous exploitaons en fait, une centrale inertielle, située au milieu de la partie supérieure du robot. Ellel mesure les déplacements de rotation et de translation du robot par rapport aux axes de référence du monde.

5.4.2 Description de la plateforme d'intégration robotique *Jafar*

Jafar est un outil de développement mis en œuvre au LAAS-CNRS, composé d'un ensemble de modules ayant la même hiérarchie [44]. Chaque module est composé par des fonctions et des classes programmées en C/C++ et il contient les librairies statiques et dynamiques correspondantes. Une action de "wrapping" permet de générer un package TCL, ce qui donne l'accès aux fonctions et classes spécifiques dans des scripts écrits en langage TCL. Le module principal de *Jafar* est le module *Kernel* qui contient les classes de base et les outils standards comme les librairies *BOOST*, *Opencv*, entre autres. Parmi les modules de base, il y en a un pour l'acquisition et le management des images, pour la lecture et l'interprétation des fichiers de mesures données par l'odomètre, par le télémètre laser, et de manière générale, des applications liées à la navigation visuelle.

Un des principaux avantages est la disponibilité des modules déjà développés sur la plateforme. En plus, la structure modulaire de *Jafar* permet que tous les développements puissent être testés sur la plateforme du robot sans modification interne. Cependant, afin de mettre en place toute l'organisation des modules sur le robot réel, il est nécessaire de tester si les ressources de calcul disponibles sur le robot *Dala* sont suffisantes pour permettre à tous les algorithmes de notre approche de travailler en parallèle à une fréquence acceptable dans un environnement dynamique. Nous avons été contraint par les capacités de calcul limitées de *Dala* ; pour la suite de ces travaux, un nouveau robot tout terrain, *Mana* (voir figure 2.1), en développement avec des capteurs de plus haute résolution et un système de calcul plus puissant, pourrait être utilisé.

En tant qu'utilisateurs de *Jafar* nous utilisons six modules de base en plus du kernel : *images, camera, jmath, datareader, filter, geom*. En exploitant les services rendus disponibles par ces modules, nous avons développé quatre modules qui implémentent notre méthode de détection et de suivi pendant la navigation d'un robot : **klt, cluster, tracker, snake**. La section suivante décrit la distribution de ces modules dans l'architecture complète et la configuration des paramètres utilisés pendant leur exécution.

5.5 Description de l'architecture complète pour la détection et le suivi d'objets

L'algorithme de détection et de suivi des points d'intérêt ainsi que l'algorithme pour la détection et le suivi d'objets mobiles ont été testés sur la plateforme de simulation *Jafar*. La figure 5.11 montre la distribution des tâches de l'architecture développée. Les quatre modules sont illustrés en couleurs différentes, chaque module est exécuté séparément de façon asynchrone car le temps d'exécution de certains modules dépend du nombre de données qu'il reçoit à chaque fois.

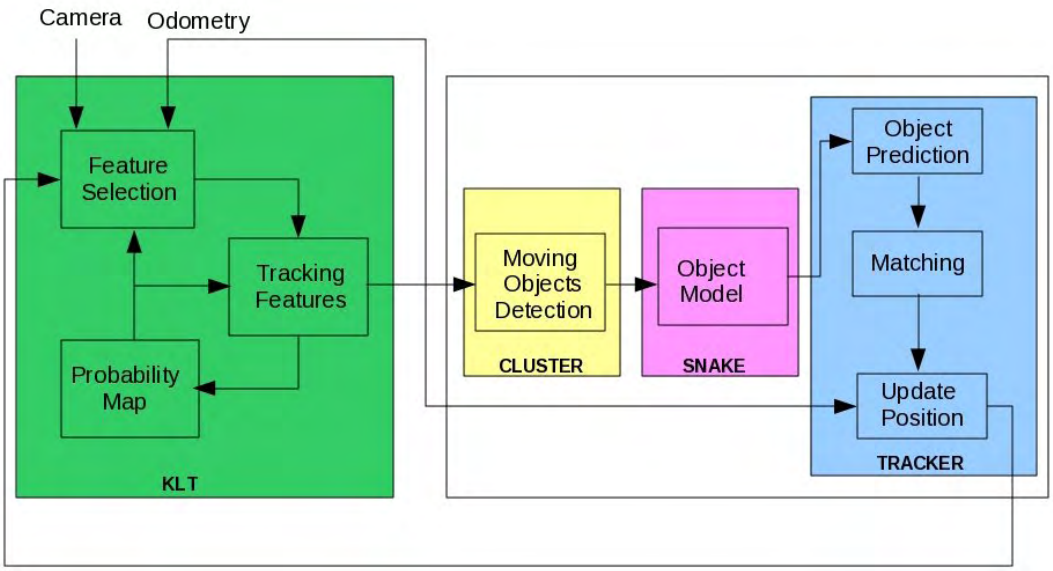


FIGURE 5.11 – Architecture globale de la méthode.

5.5.1 Module klt

Ce module est dédié à l'analyse des images acquises par la caméra gauche du robot et de l'information issue de la centrale inertielle. La sortie de ce module est un ensemble de points d'intérêt caractérisés en $\Re^4$ obtenus à partir des sorties partielles des fonctions :

— *Feature Selection* : Donne la position (x, y) sur l'image des *N* meilleurs points d'intérêt.

— *Tracking Features* : Cherche la position (x, y) des points dans l'image suivante et obtient leur vitesse dans les directions x et y (v_x, v_y).

— *Probability Map* : Garde la position de la cellule centrée sur chaque point d'intérêt (x, y) détecté dans l'image. Une valeur de probabilité p_{ij} est assignée à chaque pixel dans la cellule selon une distribution gaussienne bidimensionnelle et son état dans le temps. Cette carte est re-initialisée chaque deux *temps de pistage*.

Même si ce module réalise trois fonctions principales (*Feature Selection* ensuite *Tracking features* et *Probability Map*) elles ne sont pas exécutées à chaque image. Cela est illustré dans la figure 5.12, les flèches indiquent chaque image acquise, les flèches en rouge signalent le début du *temps de pistage* (section 3.2.4). Notez que la fonction de *Feature Selection* travaille uniquement au début de chaque *temps de pistage* tandis que les autres deux fonctions sont exécutées à chaque image (à partir de la deuxième image pour la fonction *Tracking features*).

Les paramètres d'exécution : Nous utilisons généralement 150 points d'intérêt à sélectionner dans toute l'image à chaque temps de pistage. Les points trouvés doivent être séparés par 10 pixels au minimum. Afin de sélectionner les points qui seront traités par le module **cluster** il faut que ces points soient suivis pendant 3 images consécutives au minimum et en même temps que les vitesses v_x et v_y soient

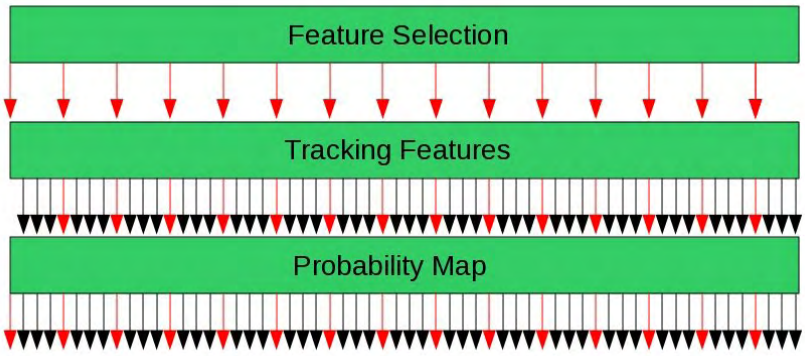


FIGURE 5.12 – Diagramme d'exécution des fonctions à l'intérieur du module **klt**. Les flèches en noir représentent chaque nouvelle image traitée, les flèches en rouge indiquent le début de chaque *temps de pistage*. Notez que la tâche *Tracking Features* commence à partir de la deuxième image acquise.

supérieures à 1 pixel.

5.5.2 Module cluster

Ce module analyse les points d'intérêt caractérisés comme les quadruplets (x, y, v_x, v_y), qui donnent leurs positions et vitesses respectives le long du *temps de pistage*. Il retourne le même ensemble mais caractérisé comme (x, y, v_x, v_y, C) où C représente l'identificateur du groupe auquel ce point appartient. Si $C = 0$, alors le point ne fait pas partie d'un objet avec un mouvement cohérent ou défini. Ce module est exécuté à la fin de chaque *temps de pistage*. Dans le diagramme de temps de la figure 5.12 cela correspond à la même fréquence que la fonction de *Feature Selection*. D'autre part, le temps de calcul de ce module est fonction de nombre des points reçus à chaque exécution. Donc, en utilisant les fonctions *tic-toc* de C le temps d'exécution est de 1 ms pour traiter 40 points mais augmente à quelques secondes à partir de 300 points reçus en entrée.

Les paramètres d'exécution : La méthode de groupement *a contrario* n'a pas besoin de paramètres à régler de manière "astucieuse" ; c'est son grand avantage. Cependant certaines valeurs doivent être définies avant son exécution en ligne, par exemple le nombre et les tailles des régions que nous voulons tester. Il s'agit de sélectionner diverses tailles des régions pour tester tous les groupes de points dans l'arbre binaire. De ce paramètre dépend directement le temps d'exécution ; mais il ne faut pas qu'il soit trop petit car les résultats sont aussi dépendants de cette valeur. Alors un bon compromis consiste à fixer différentes tailles par dimension de telle façon que l'on aille d'une taille minimale (nous avons choisi la plus petite de 10 pixels) jusqu'à la taille de l'image. Ainsi, 20 tailles sont choisies par dimension ; les données étant exprimées dans un espace à quatre dimensions, cela fait un total de 20^4 régions.

5.5.3 Module snake

Ce module associe un contour actif à un objet détecté précédemment, défini par un ensemble de points dynamiques classifiés comme faisant partie du même objet. La fonction principale prend tous les points et sélectionne uniquement ceux qui sont sur la périphérie du nuage de points. Ces points doivent être au minimum au nombre de 4 afin de trouver les fonctions B-spline les plus convenables pour décrire le contour actif.

Le module prend en entrée un tableau de structures de taille égale au nombre des objets dynamiques. Chaque structure contient le nombre des points identifiés sur l'objet n_e et ses valeurs correspondantes $(x_i, y_i, v_{x_i}, v_{y_i}, C_i)$ où $i = 1, 2, ..., n_e$. Ce module ajoute à la structure de l'objet un tableau avec les points identifiés comme apartenant aux bords et les paramètres du contour actif. Une variable "flag" est mise à la valeur '1' si on a bien identifié un contour pour l'objet à tester. Quand elle est mise à '0' alors au lieu d'un contour actif, l'objet est délimité par un rectangle, sans chercher à trouver une silhouette précise de l'objet.

Ce module est exécuté à chaque image acquise, dès qu'il existe un objet dynamique détecté, et uniquement le temps que l'objet est présent dans l'image. Donc nous ne connaissons pas exactement les instants où ce module s'exécute ; mais quand il est actif, il doit tourner à la fréquence d'acquisition. Par ailleurs, il serait intéressant d'incorporer ici la fonction qui réalise la recherche du contour, dont les résultats sont présentés dans les figure 4.8 et 4.15). Pour le moment cette fonction de suivi d'un contour actif, est réalisée en dehors de *Jafar* en partageant des données de façon séquentielle avec Matlab.

Les paramètres d'exécution : La taille du tableau en entrée du module est fonction du nombre d'objets identifiés. Il n'y a pas apparemment une limite pour ce nombre d'objets détectés, le plus grande nombre que nous ayons trouvé sur une séquence a été de 17 objets. Dans tous les cas c'est la mémoire du système qui limite cette valeur. À l'intérieur du module le nombre de points de contrôle est établi à 4 au minimum, puis entre chaque point de contrôle, la courbe est discrétisée avec 8 points.

5.5.4 Module tracker

Ce module prend comme entrée le tableau des structures avec les éléments des objets mobiles et leurs contours. La sortie de ce module est l'état de chaque objet mobile dans l'image courante. D'abord une étape de prédiction permet de sélectionner une zone où l'objet se trouvera à l'image suivante. L'état du barycentre des objets est obtenu à la sortie de ce module. Ce module est exécuté à chaque image à partir de la détection d'un objet dynamique par le module **cluster**, donc sa fréquence est la même que la fonction de *probability map* du diagramme de temps dans la figure 5.12.

— *Object Prediction* : Donne l'état (x, y, v_x, v_y) du barycentre de chaque objet dynamique sur l'image. La prédiction utilise un modèle à vitesse constante.

— *Matching* : Cette fonction reçoit l'image suivante et fait la mise en correspondance du modèle au temps précèdent sur la position prédite. Un motif autour de chaque point d'intérêt est apparié autour d'une fenêtre de corrélation jusqu'à trouver le meilleur score de similarité.

— *Update Position* : Garde l'état (x, y, v_x, v_y) du barycentre de chaque objet dynamique à l'image courante. Ces données sont ajoutées à la carte de probabilités pour savoir les régions de l'image déjà occupées par les objets mobiles.

Ces trois fonctions principales du module sont exécutées à chaque image d'entrée à partir du moment où un objet mobile est détecté.

Les paramètres d'exécution : Les valeurs de configuration du filtre du Kalman, comme par exemple celle donnée pour la moyenne et l'écart type du bruit gaussien, sont données en fonction des vitesses attendues des objets. Par ailleurs le vecteur d'état $[X, Y, V_x, V_y]$ est directement pris du modèle de l'objet.

5.6 Résultats expérimentaux

5.6.1 Résultats de l'approche de compensation du mouvement.

Nous avons pris une séquence de 1900 images avec les informations correspondantes concernant l'estimation des déplacements, sous la forme de translations et rotations exprimées dans le repère du monde, obtenues depuis la centrale inertielle. La figure 5.8 montre la position du robot dans le repère du monde ainsi que les valeurs de rotation des angles *yaw*, *pitch*, *roll* données par la centrale inertielle. En analysant les graphiques des angles on peut distinguer le comportement du robot lors de la navigation. Notons que les angles ont une variation négligeable lors des premières 400 images, donc nous avons choisi 8 images semblant les plus représentatives à partir de ce moment dans la séquence d'images, afin de montrer des mouvements plus fréquents et réels comme des rotations et translations simultanées.

Ces images sont présentées sur la figure 5.13 où deux voitures en mouvement croisent la trajectoire du robot. D'abord une première voiture de couleur foncée traverse de droite à gauche dans l'image, donc perpendiculaire à la trajectoire du robot, elle est complètement détectée et suivie lors de son parcours. Ensuite une deuxième voiture grise ne traverse pas le champ de vue du robot, ce qui rend plus compliquée sa détection car ses déplacements sont parallèles à l'axe optique de la caméra (axe X du robot). Cette voiture est initialement détectée dans une zone au niveau de la roue avant droite, ensuite quand cette partie n'est plus visible dans l'image, la partie arrière de la voiture est détectée et suivie jusqu'au moment où son mouvement n'est plus aperçu à l'horizon. En même temps que cette détection le robot commence à tourner à gauche (voir les changements significatifs de l'angle *yaw*, notamment au niveau de l'image 620) ce qui démontre la bonne performance de notre méthode dans des situations réelles plus complexes. La voiture est aussi détectée quand elle bouge vers le robot. Finalement sur cette figure est aussi illustrée la détection des deux voitures juste avant leur occultation.

5.6.2 Séquence d'images acquise avec plus de vitesse

Une extension d'expérimentations a été mise en œuvre pour évaluer le développement de notre algorithme quand le robot se déplace plus rapidement. La figure 5.14 montre quelques images de la séquence acquise lors de la navigation du robot autour du laboratoire. La vitesse de navigation du robot est d'environ 2 m/s à la différence de la séquence présentée dans la figure 5.13 qui a été prise à une vitesse d'environ 1 m/s. La figure 5.14b illustre la détection d'un camion qui bouge parallèlement à l'axe optique de la caméra vers le robot. Cependant, dans l'image 5.14a un faux objet dynamique est trouvé sur le sol.

Cette image montre ainsi la sensibilité de la contrainte que nous avons utilisée pour compenser le mouvement propre du robot. Ainsi, même si cette stratégie de compensation (voir section 5.2) a été proposée afin de filtrer les objets statiques sur le sol, mais classifiés à tord comme dynamiques à cause du mouvement du robot, cet objet dans l'image est détecté et non filtré, parce qu'il est sur le trottoir. Donc il n'est pas au même niveau que le plan XY du robot, et de ce fait son mouvement apparent n'est pas suffisamment compensé.

5.7 Conclusions

Ce chapitre a été dédié aux problèmes posés par l'intégration de notre approche de détection d'obstacles dans un système complet de navigation. Il nous a permis d'introduire une méthode pour la compensation du mouvement du robot ; nous présentons ensuite quelques résultats obtenus durant nos expérimentations.

FIGURE 5.13 – Résultats expérimentaux sur une séquence de 1900 images avec compensation de mouvement. Les deux voitures sont bien détectées et suivies.

(a) Image 159 (b) Image 169 (c) Image 189

FIGURE 5.14 – Résultats expérimentaux dans une séquence d'extérieur à plus grande vitesse. Deux objets dynamiques sont détectés : un faux objet mobile sur le sol et un camion qui bouge vers le robot (voir le texte pour plus d'information).

Dans sa version non intégrée, notre module de détection et suivi d'objets mobiles atteint sa meilleure performance si le déplacement de la caméra est assez lent. Cette contrainte de mouvement faible a été utile pour tester notre module de façon poussée. Mais dès lors que nous voulons le tester quand la caméra est embarquée sur un robot en mouvement dans des environnements d'extérieur, alors, notre approche basée uniquement sur la vision requiert d'autres mesures pour arriver à compenser le mouvement de la caméra dans les images.

Pour ce faire, nous avons tout d'abord étudié différentes approches utilisées normalement pour l'estimation du mouvement entre des instants t et $t+1$. Nous avons trouvé intéressante la compensation basée sur l'analyse des images acquises à ces instants, mais cela implique des méthodes lourdes qui calculent la matrice fondamentale à partir de points suivis dans ces deux images. Lorsque ces méthodes pourront être exécutées au moins à la fréquence vidéo, elles seront une option très attrayante car depuis les images, on peut estimer exactement le mouvement suivi par le robot. En effet, si on mesure le mouvement du robot depuis les images, afin d'appliquer la méthode de compensation du mouvement entre ces mêmes images, on évite le processus de synchronisation nécessaire lors de l'utilisation d'une centrale inertielle ou d'autres capteurs ; nous aurons donc une mesure plus fidèle du vrai mouvement du robot.

Ensuite nous avons décrit la plateforme expérimentale *Jafar* utilisée pour traiter nos séquences d'images, ainsi que les différents modules développés pour nos expérimentations, mais qui pourraient être utilisées plus largement dans le cadre de la robotique et du traitement d'images. Nous avons évalué expérimentalement la détection et le suivi d'objets sur une séquence acquise lors de la navigation du robot à plus grande vitesse. Ainsi nous avons pu démontrer la performance de la méthode de compensation de mouvement incorporée dans notre approche. Plus précisément, la validation d'objets statiques sur le sol s'est montrée efficace dans le cas où ces objets sont effectivement intégrés dans le plan du sol, typiquement des marquages routiers.

Enfin nous avons aussi proposé une méthode de coopération pour la navigation et la génération d'une carte hétérogène. Avec cette proposition nous espérons réduire l'incertitude dans la localisation du robot en envoyant les points que nous considérons comme étant statiques à un module SLAM, qui s'exécute simultanément à notre approche de détection et de suivi d'objets mobiles.

Chapitre 6

Conclusion et perspectives

Sommaire

6.1 Conclusion Générale

Dans ce travail nous nous sommes intéressée à la détection et au suivi d'objets mobiles dans un environnement dynamique extérieur. Pour cela nous avons développé un module **KLT**, pour réaliser une méthode de détection et de suivi des indices visuels les plus représentatifs, à savoir des points d'intérêt ; ces points sont suivis sur N_{im} images, N_{im} correspondant au *temps de pistage*. Nous avons ensuite analysé les mouvements des points suivis pour détecter la présence d'objets mobiles dans la scène. Nous pouvons ainsi détecter des objets rigides (véhicules) qui ont des mouvements apparents à vitesse constante mais aussi des objets non rigides (piétons) avec des mouvements faibles. Nous avons développé un module **cluster**, pour regrouper des points de l'image acquis sur différents objets mobiles. La méthode se fonde sur une analyse spatio-temporelle en considérant un espace dans $\mathcal{R}^4$ dans lequel sont décrits les positions et les mouvements apparents des points suivis pendant le temps de pistage. Dans cet espace, nous évaluons un modèle aléatoire de mouvement, modèle affecté aux points détectés dans des zones statiques de la scène ; ces mouvements sont créés par le mouvement apparent créé par le propre mouvement de la caméra. Dans le cas où ce modèle aléatoire n'est pas respecté, alors nous détectons un groupe de points qui ont un mouvement non aléatoire, donc cohérent, et qui correspondent dans la plupart des cas, à un objet dynamique de la scène.

Dans le but de proposer une méthode sans connaissance a priori sur le nombre d'objets dynamiques présents dans la scène courante, nous avons inséré la méthode de groupement dans une boucle, de telle façon que de nouveaux objets puissent être identifiés sur chaque image analysée à chaque *temps de pistage*, puis fusionnés, si possible, avec ceux déjà existants. Pour ce faire, nous avons choisi de réaliser l'implémentation en employant des listes chaînées ; ce type de structure rend plus performante la tâche d'ajouter ou d'enlever des objets dynamiques, tout en limitant le nombre de paramètres à régler. Nous avons également mis en place une méthode pour le suivi de multiples objets simultanément en utilisant le filtre de Kalman qui permet la prédiction du mouvement de plusieurs entités suivies dans une séquence d'images, ces mouvements suivant un modèle de mouvement à vitesse constante.

Une itération de notre procédure, si elle est séquentielle, a une période égale au *temps de pistage*, donc temps d'acquisition et de traitement de N_{im} image, soit par exemple environ 130ms pour $N = 4$ avec des acquisitions à 30Hz. Le fait que le nombre de points sélectionnés à chaque itération soit fixe, et faible par rapport à celui des approches denses, pourrait avoir comme conséquence que certains objets dynamiques ne sont pas détectés. Afin d'éviter cela, nous avons proposé de mettre à jour à chaque itération une carte de probabilités afin d'indiquer les endroits où il y avait des objets mobiles détectés sur les images précédentes, ainsi que les zones non inspectées depuis plusieurs itérations et qui pourtant, pourraient contenir de nouveaux points d'intérêt correspondant à des points 3D mobiles de la scène. Cette méthode gère les points sélectionnés par le module **KLT** et permet de sélectionner les zones de l'image les plus intéressantes pour extraire de nouveaux points au temps courant.

Pour valider notre approche, nous avons tout d'abord évalué expérimentalement le module de détection et de suivi des points d'intérêt sur des séquences d'images fixes et mobiles prises à l'intérieur et à l'extérieur. Nous avons montré que l'accumulation du flot optique des points permet de commencer à distinguer la trajectoire suivie dans l'image par les objets mobiles. Le mouvement propre de la caméra peut être négligé si on n'accumule pas longtemps le flot optique des points, donc si le *temps de pistage* N reste petit, $N = 4$ dans la plupart de nos expérimentations. De ce fait nous avons trouvé la valeur la plus adaptée pour la fréquence d'exécution du module **cluster** ; nous avons ainsi fixé un des rares paramètres à régler de notre approche.

Des tests intensifs ont été réalisés sur le module **cluster** afin d'évaluer sa performance face à diverses situations : scènes fixes ou dynamiques, scènes avec un nombre quelconque d'objets mobiles rigides ou non-rigides, mouvement plus ou moins rapide de la caméra... De plus nous avons testé le processus qui ferme la boucle *détection-groupement-suivi* surtout dans le cas de multiples objets en arrivant à une performance satisfaisante dans la détection et la fusion sur des images acquises depuis une caméra embarquée sur un robot. Ceci est une contribution par rapport à l'approche originale de T. Veit, F. Cao et P. Bouthemy, qui n'ont traité que de la détection.

La carte probabiliste proposée permet d'identifier plus facilement de nouvelles régions de l'image correspondantes à des objets mobiles, et donc d'arriver à une détection plus complète. Les expérimentations ont permis de valider notre approche pour la majorité des objets mobiles qui ont été correctement détectés et suivis. Mais nous avons pu remarquer également que les objets non rigides ne sont pas identifiés comme un seul objet ; on a trouvé souvent 2 régions apparemment différentes pour chaque piéton détecté, une sur la tête et le tronc, une sur les jambes. La vitesse et la position des points identifiés sur ces objets non rigides jouent un rôle très significatif sur la fiabilité du groupement des points. Cela est notre plus grande difficulté à résoudre : il est difficile de détecter des points d'intérêt dans toute la zone occupée dans l'image par le piéton, c'est-à-dire qu'il est difficile d'extraire des points bien distribués sur l'image du piéton. De plus les points trouvés sur les pieds n'ont pas la vitesse apparente de ceux trouvés dans le reste du corps.

Pour résoudre ce problème, ou du moins pour minimiser son importance, il faut absolument un double compromis, d'abord que la taille du piéton soit grande dans l'image, et deuxièmement que le fond de la scène ne soit pas trop encombré.

— Pour le premier il faut une résolution plus importante de l'image, ce qui va directement augmenter notre temps de calcul ; dans le cas où notre méthode peut être portée sur des architectures dédiées pour exécution en temps réel (à fréquence vidéo), la contrainte de l'échelle ne serait plus un problème.

— Pour le problème d'un environnement encombré, dans le cas des environnements naturels ou urbains, cela ne peut pas être évité. Alors dans ce cas on pourrait considérer d'autres descripteurs comme les SIFT ou SURF ou d'autres propriétés comme la silhouette afin d'ajouter plus de connaissances sur les objets et construire un modèle plus complet.

L'implémentation de notre approche sur une plateforme de simulation robotique, et non sur nos démonstrateurs, s'explique par une performance non compatible pour l'instant avec un traitement en ligne ; donc nos tests ont été faits hors ligne sur des séquences enregistrées. L'utilisation d'une carte graphique (GPU) ou d'une architecture dédiée (FPGA) pourrait accélérer grandement les calculs principalement pour les algorithmes qui traitent directement l'image (détection et suivi des points par KLT) et qui sont pourtant à paralléliser. Notons que de telles architectures existent déjà pour des projets spécifiques : citons la caméra NPAL développée pour des applications spatiales par EADS Astrium, ou les travaux menés au LAAS sur la localisation d'un aéronef sur un aéroport.

Néanmoins ce simple transfert ne suffirait pas pour réduire le temps de latence, actuellement égal au moins à 2 fois N images ; dans une analyse faite récemment dans un article soumis au IFAC World Congress, nous proposons comment réduire ce temps de latence au temps d'acquisition d'une image en exécutant en parallèle N tâches de suivi et de clustering. Une telle solution pourrait être implémentée à terme sur un FPGA assez puissant.

Une approche de collaboration avec une méthode de SLAM a été analysée dans le chapitre 5, pour le moment sans implémentation en ligne mais en proposant une coopération étroite entre les deux approches. Les résultats de la méthode de SLAM proposé par J.Solà pourraient fournir le point de départ de nouvelles recherches dans le domaine de la navigation incluant le *SLAMMOT*. Signalons par exemple que la carte stochastique construite par cette approche peut être perturbée par l'incertitude rajoutée par des points dynamiques ; nous proposons de réduire cet effet en détectant les zones dans les images qui contiennent des objets dynamiques pour qu'ils ne soient pas sélectionnés par la fonction SLAM.

6.2 Perspectives

Nous sommes consciente que notre travail n'a pas abouti à une solution directement exploitable sur un robot. Dans l'optique d'une continuité des recherches sur la détection et le suivi d'objets dynamiques depuis un véhicule en vision monoculaire, plusieurs études pourraient être proposées :

— **Caractérisation des régions**. Le modèle de l'objet proposé ici utilise uniquement des propriétés d'intensité à l'intérieur et à l'extérieur de la région. Dans le cas des environnements très encombrés dans le fond, cela se traduit par une réduction dans la performance du suivi. Ainsi d'autres paramètres sont envisageables : nous pensons à la couleur, la vitesse et la direction de mouvement de la région. W.Ait Fares, doctorante dans l'équipe, poursuit ce travail sur le suivi des objets dynamiques, avec une approche de suivi de régions inspirée des travaux de A.Herbulot (compétition entre la région et le fond).

— **Navigation sur des terrains non plans**. La compensation du mouvement que nous avons proposée pour la navigation du robot fonctionne lorsque le sol est complètement plan, ce qui est le cas dans un parking ou dans la plupart des autoroutes. Cependant, un environnement naturel a un sol irrégulier, avec des pentes ; de ce fait notre approche de compensation ne sera pas suffisante.
Dans le but de corriger ce problème, plusieurs solutions sont envisagées : la génération d'une

carte d'élévation comme cela est proposé dans [13, 62] pour les environnements naturels. Cette approche semble appropriée pour résoudre notre problème ; cependant elle implique l'utilisation d'un capteur 3D tel que les télémétres laser Sick ou Velodyne ; or, dans les objectifs de l'approche ici présentée nous nous sommes intéressée uniquement à l'exploitation de données visuelles.
Une autre solution possible envisageable est l'utilisation d'une carte de disparité générée par un système de stéréovision en temps réel. Le fait qu'on puisse obtenir la carte de disparité de l'environnement à des hautes fréquences nous permet d'envisager la reconstruction 3D des points. Cette reconstruction donnera des informations à propos de l'environnement du robot que nous n'avons pas avec notre approche éparse actuelle. D'abord une extension à l'espace 6D de la méthode de groupement a contrario est possible en ajoutant la valeur de la profondeur et la vitesse de cette troisième coordonnée. Cela donne le vecteur de caractérisation (x, y, z, v_x, v_y, v_z) pour chaque point d'intérêt. Par contre il faudra bien trouver un compromis sur le temps d'évaluation car les régions aussi devront évoluer dans l'espace 6D et cela risque d'affecter largement le temps de calcul uniquement dans le module **cluster**. En analysant les points d'intérêt dans l'espace 6D au lieu de 4D, nous aurions plus de possibilités pour discriminer les faux positifs qui sont les points dynamiques détectés sur le sol.

— **SLAM visuel**. La méthode du RT-SLAM (pour Real Time SLAM) qui est développée au LAAS à partir des travaux de J. Solà, ne manipule pas le concept d'objet rigide ; elle traite des indices visuels épars, des points ou des lignes. Hors notre méthode de détection et de suivi d'objets dynamiques, génère des objets, représentés par un ensemble de points d'intérêt, mais aussi par une silhouette et des attributs d'apparence dans l'image courante. Comment prendre en compte ce modèle dans le MOT ? Pour chaque objet rigide, nous pourrions générer une sous-carte afin de reconstruire les points suivis sur cet objet, et exploiter ces points pour localiser l'objet vis-à-vis du robot ou d'un repère global dans lequel ce robot est localisé. Cette localisation pourrait être considérée comme une observation à fusionner dans une carte stochastique de plus haut niveau.

Finalement, pour évaluer notre architecture complète sur un robot réel, il est encore nécessaire de tester si les ressources disponibles sur le robot sont suffisantes pour permettre à tous les algorithmes d'être exécutés à une fréquence acceptable dans un environnement dynamique. Si cela n'est pas possible, la tendance actuelle est le développement de logiciels embarqués en "co-design". Il faut être capable de câbler certains algorithmes qui seront exécutés sur un système matériel dédié, en partageant des informations avec le reste des algorithmes, soit via un réseau, soit via des mémoires communes.

Bibliographie

[1] D.L. Almanza-Ojeda, M. Devy, and A. Herbulot. Incremental detection and tracking of moving objects by optical flow and a contrario method. In *Proc. International Conference on Computer Vision Theory and Applications (VISAPP 2010)*, Angers, France, may 2010.

[2] D.L. Almanza-Ojeda, M. Devy, and A. Herbulot. Visual-based detection and tracking of dynamic obstacles from a mobile robot. In *Proc. 7th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2010)*, Madeira, Portugal, june 2010.

[3] X. Armangu and J. Salvi. Overall view regarding fundamental matrix estimation. *Image and Vision Computing*, 21(2) :205–220, 2003.

[4] X. Armangue, H. Araujo, and J. Salvi. Differential epipolar constraint in mobile robot egomotion estimation. In *Proc. 16th International Conference on Pattern Recognition (ICPR 2002)*, pages 599–602, 2002.

[5] S. Baker and I. Matthews. Lucas-kanade 20 years on : A unifying framework : Part 1. Technical Report CMU-RI-TR-02-16, Robotics Institute, Pittsburgh, PA, july 2002.

[6] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley-Interscience, 1 edition, june 2001.

[7] M. Bertozzi, A. Broggi, A. Fascioli, and R. Fascioli. Stereo inverse perspective mapping : Theory and applications. *Image And Vision Computing Journal*, 8 :585–590, 1998.

[8] P. Bessiere, C. Laugier, and R. Siegwart. *Probabilistic Reasoning and Decision Making in Sensory-Motor Systems*. Springer Publishing Company, Incorporated, 2008.

[9] J.-Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm, 2000.

[10] P. Brigger, J. Hoeg, and M. Unser. B-spline snakes : a flexible tool for parametric contour detection. *IEEE Transactions on Image Processing*, 9(9) :1484–1496, september 2000.

[11] A. Broggi, M. Bertozzi, A. Fascioli, and S. Nichele. Stereo vision-based vehicule detection. *Intelligent Vehicules Symposium*, pages 34–44, october 2000.

[12] T. Brox, M. Rousson, R. Deriche, and J. Weickert. Colour, texture, and motion in level set based segmentation and tracking. *Image Vision Comput.*, 28(3) :376–390, 2010.

[13] Wolfram Burgard and Martial Hebert. World modeling. In *Springer Handbook of Robotics*, pages 853–869. Springer Berlin Heidelberg, 2008.

[14] J. Burlet. *Suivi Multi-Objets Adaptatif : Application à la Classification de Mobiles*. PhD thesis, Institute National Polytechnique de Grenoble, 2007.

[15] J. Burlet, O. Aycard, A. Spalanzani, and C. Laugier. Pedestrian tracking in car parks : an adaptive interacting multiple models based filtering method. *Proc. IEEE International Conference on Intelligent Transportation Systems (ITS 2006)*, pages 462–467, 2006.

[16] F. Cao, J. Delon, A. Desolneux, P. Musé, and F. Sur. A unified framework for detecting groups and application to shape recognition. *Journal of Mathematical Imaging and Vision*, 27(2) :91–119, 2007.

[17] T.F. Chan and L.A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2) :266–277, 2001.

[18] W. Chieh-Chih, C. Thorpe, and A. Suppe. Ladar-based detection and tracking of moving objects from a ground vehicle at high speeds. In *Proc. IEEE Intelligent Vehicles Symposium (IV2003)*, pages 416–421, june 2003.

[19] J. Civera, A. Davison, and J. Montiel. Inverse depth parametrization for monocular slam. *IEEE Transactions on Robotics*, 24(5), 2008.

[20] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000)*, volume 2, pages 142–149, 2000.

[21] A. Comport, E. Marchand, and F. Chaumette. Robust model based tracking for robot vision. *Proc. IEEE/RSJ International Conference on Intelligents Robots and Systems (IROS 2004)*, september 2004.

[22] C. Coue, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessière. Bayesian occupancy filtering for multitarget tracking : An automotive application. *International Journal of Robotics Research*, 2006.

[23] E. R. Davies. *Machine Vision, Third Edition : Theory, Algorithms, Practicalities*. M. Kaufmann, 3 edition, january 2005.

[24] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM : Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6) :1052–1067, 2007.

[25] A.J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. Ninth IEEE International Conference on Computer Vision (ICCV 2003)*, pages 1403–1410, october 2003.

[26] A. Desolneux, L. Moisan, and J.-M. Morel. A grouping principle and four applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(4) :508–513, 2003.

[27] A. Desolneux, L. Moisan, and J.-M. Morel. *From Gestalt Theory to Image Analysis A Probabilistic Approach*, volume 34. Springer Berlin / Heidelberg, 2008.

[28] F. Dionnet and E. Marchand. Robust stereo tracking for space applications. *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, pages 3373–3378, 2007.

[29] E. Eade and T. Drummond. Scalable monocular slam. In *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, pages 469–476, Washington, DC, USA, 2006. IEEE Computer Society.

[30] A. Ewald and V. Willhoeft. Laser scanners for obstacle detection in automotive applications. *Proc. IEEE Intelligent Vehicles Symposium (IV2001)*, june 2001.

[31] O. Faugeras, Q.-T. Luong, and T. Papadopoulo. *The Geometry of Multiple Images : The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications*. MIT Press, march 2004.

[32] Olivier D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig. In *ECCV*, pages 563–578, 1992.

[33] S. Florczyk. *Robot Vision : Video-based Indoor Exploration with Autonomous and Mobile Robots*. Wiley-VCH, april 2005.

[34] G. Gate. *Reliable Perception of Highly Changing Environments : Implementations for Car-to-Pedestrian Collision Avoidance Systems*. PhD thesis, Ecole Nationale Superieure des Mines de Paris, 2009.

[35] G. Gate, A. Breheret, and F. Nashashibi. Centralised fusion for fast people detection in dense environments. In *Proc. IEEE International Conference on Robotics Automation (ICRA 2009), Kobe, Japan*, 2009.

[36] L. Gond, Q.C. Phanm, Q. Begard, N. Allezard, and P. Sayd. Imagerie infrarouge pour la surveillance de foules. *Reconnaissance des Formes et Intelligence Artificielle, RFIA*, pages 750–758, 2008.

[37] M. Grewal and A. Andrews. *Kalman Filtering : Theory and Practice Using MATLAB*. John Wiley & Sons Inc., january 2008.

[38] B. Han, D. Comaniciu, Y. Zhu, and L.S. Davis. Sequential kernel density approximation and its application to real-time visual tracking. *IEEE Transactions on Pattern Analysis Machine Intelligent (PAMI)*, june 2007.

[39] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision (Volume I)*. Addison Wesley Longman, 2002.

[40] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN : 0521540518, second edition, 2004.

[41] J. Hol. *Pose Estimation and Calibration Algorithms for Vision and Inertial Sensors*. PhD thesis, Department of Electrical Engineering, Linköping University, 2008.

[42] B. Horn and B. Schunk. Determining optical flow. *Artificial Intelligence*, 17 :185–203, 1981.

[43] http ://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1/, may 2010.

[44] http ://homepages.laas.fr/croussil/doc/jafar/index.html, october 2010.

[45] http ://i21www.ira.uka.de/image_ sequences/, may 2010.

[46] http ://sdi.acfr.usyd.edu.au/datasets/IRcameraCalibration/, october 2010.

[47] https ://intranet.laas.fr/intranet/robots/wiki/Pelican, september 2010.

[48] http ://www.irobot.com, august 2010.

[49] http ://www.laas.fr/morse, september 2010.

[50] M. Hwangbo, J.-S. Kim, and T. Kanade. Inertial-aided KLT feature tracking for a moving camera. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*, pages 1909–1916, 2009.

[51] M.-A. Ibarra-Manzano, M. Devy, and J.-L. Boizard. Real-time classification based on color and texture attributes on an fpga-based architecture. In Adam Morawiec and Jinnie Hinderscheit, editors, *Proc. 2010 Conference on Design and Architecture for Signal and Image Processing (DASIP 2010)*, pages 53–60. Electronic Chips and Systems design Initiative, October 2010.

[52] M.A. Ibarra-Manzano, D.L. Almanza-Ojeda, M. Devy, J.L. Boizard, and J.Y. Fourniols. Stereo Vision Algorithm Implementation in FPGA Using Census Transform for Effective Resource Optimization. In *Proc. 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools*, pages 799–805. IEEE Computer Society, 2009.

[53] Intel. *Open Source Computer Vision Library*, 1999-2001. Reference Manual.

[54] B. Jida, R. Lherbier, J.-C. Noyer, and M. Wahl. Multiple target detection and tracking by interacting joint probabilistic data association filter and bayesian networks : Application to real data. In *Proc. 12th IEEE International Conference on Intelligent Transportation Systems (ITS 2009)*, pages 1–8, 2009.

[55] R. Katz, B. Douillard, J. Nieto, and E. Nebot. A self-supervised architecture for moving obstacles classification. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008)*, pages 155–160, september 2008.

[56] A. Kelly, A. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, P. Rander, S. Thayer, N. Vallidis, and R. Warner. Toward reliable off road autonomous vehicles operating in challenging environments. *International Journal of Robotics Research*, pages 449–483, may-june 2006.

[57] A. Khammari. *Système embarqué de détection multi-sensorielle de véhicules : application à la gestion intelligente des interdistances*. PhD thesis, École des Mines de Paris, 2006.

[58] Junmo Kim, III Fisher, J.W., Jr. Yezzi, A., M. Cetin, and A.S. Willsky. Nonparametric methods for image segmentation using information theory and curve evolution. volume 3, pages 797–800, june 2002.

[59] J. Klappstein, F. Stein, and U. Franke. Monocular motion detection using spatial constraints in a unified manner. pages 261–267, june 2006.

[60] J. Klappstein, T. Vaudrey, C. Rabe, A. Wedel, and R. Klette. Moving object segmentation using optical flow and depth information. In *Proc. 3rd Pacific Rim Symposium on Advances in Image and Video Technology (PSIVT '09)*, pages 611–623, Berlin, Heidelberg, 2008. Springer-Verlag.

[61] R. Labayrade, D. Gruyer, C. Royere, M. Perrollaz, and D. Aubert. Obstacle detection based on fusion between stereovision and 2d laser scanner. *Mobile Robots : Perception And Navigation*, pages 91–110, 2007.

[62] S. Lacroix, A. Mallet, D Bonnafous, G. Bauzil, S. Fleury, M. Herrb, and R. Chatila. Autonomous rover navigation on unknown terrains : Functions and integration. *International Journal of Robotics Research*, 21, 2002.

[63] D. Langer. *An Integrated MMW Radar System for Outdoor Navigation*. PhD thesis, Carnegie Mellon University, The Robotics Institute, Pittsburg, Pensylvania, 1997.

[64] S. Lefebvre. *Approche monodimensionnelle de la mise en correspondance stéréoscopique par corrélation : Application à la détection d'obstacles routiers.* PhD thesis, Université de Lille 2, 2007.

[65] T. Lemaire, C. Berger, I.-K. Jung, and S. Lacroix. Vision-Based SLAM : stereo and monocular approaches. *International Journal of Computer Vision*, 74(3) :343–364, 2007.

[66] V. Lemonde. *Stéréovision Embarquée sur Véhicule : de l'Auto-Calibrage à la Détection d'Obstacles.* PhD thesis, INSA-LAAS-CNRS, 2005.

[67] D.T. Linzmeier, M. Skutek, M. Mekhaiel, and K.C.J. Dietmayer. A pedestrian detection system based on thermopile and radar sensor data fusion. *Proc. International Conference on Information Fusion*, pages 1272–1279, 2005.

[68] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. DARPA Image Understanding Workshop*, pages 121–130, april 1981.

[69] V. Mahalingam, K. Bhattacharya, N. Ranganathan, H. Chakravarthula, R.R. Murphy, and K.S. Pratt. A vlsi architecture and algorithm for Lucas–Kanade-Based optical flow computation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(1) :29–38, 2010.

[70] H. A. Mallot, H. H. Bülthoff, J. J. Little, and S. Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological Cybernetics*, 64(3) :177–185, 1991.

[71] A. Marin-Hernandez. *Vision dynamique pour la navigation d'un robot mobile.* PhD thesis, INPT-LAAS-CNRS, 2004.

[72] D.A. Marquez-Gamez, T. Féraud, M. Devy, P. Checchin, and R. Chapuis. Visual navigation for a convoy of communicating vehicles in unknown environment. Technical report, Laboratoire d'Analyse et d'Architecture des Systèmes LAAS-CNRS, 2009.

[73] L. Matthies, T. Litwin, K. Owens, A. Rankin, K. Murphy, D. Coombs, J. Gilsinn, T. Hong, S. Legowik, M. Nashman, and B. Yoshimi. Performance evaluation of ugv obstacle detection with ccd/flir stereo vision and ladar. *Proc. ISIC/CIRA/ISAS Joint Conference*, pages 658–670, september 1998.

[74] L. Matthies, M. Mainmone, A. Johnson, Y. Cheng, R. Willson, C. Villalpando, S. Goldberg, A. Huertas, A. Stein, and A. Angelova. Computer vision on mars. *International Journal of computer Vision*, pages 67–92, march 2007.

[75] P.S. Maybeck. *Stochastics Models, Estimation, and Control : Introduction*, volume 1. Academic Press, Inc., 1979.

[76] A. Mendes, L.C. Bento, and U. Nunes. Multi-target detection and tracking with a laser scanner. In *Proc. IEEE Intelligent Vehicles Symposium (IV 2004)*, pages 796–801, june 2004.

[77] D. Migliore, R. Rigamonti, D. Marzorati, M. Matteucci, and D. G. Sorrenti. Use a single camera for simultaneous localization and mapping with mobile object tracking in dynamic environments. In *Proc. ICRA09 workshop on Safe navigation in open and dynamic environments – Application to autonomous vehicles*, 2009.

[78] J. G. Avi na Cervantes. *Navigation visuelle d'un robot mobile dans un environnement d'extérieur semi-structuré.* PhD thesis, LAAS-CNRS, 2005.

[79] Y. Ohta, T. Kanade, and T. Sakai. Color information for region segmentation. *Computer Graphics and Image Processing*, 13(1) :222–241, july 1980.

[80] N. Onkarappa and A. Sappa. On-Board monocular vision system pose estimation through a dense optical flow. *Proc. 7th International Conference Image Analysis and Recognition (ICIAR 2010)*, pages 230–239, 2010.

[81] M. Perrollaz, A. Spalanzani, and D. Aubert. Probabilistic representation of the uncertainty of stereo-vision and application to obstacle detection. In *Proc. IEEE Intelligent Vehicles Symposium (IV 2010)*, pages 313–318, 2010.

[82] D. Pomerleau. RALPH : rapidly adapting lateral position handler. In *Proc. Intelligent Vehicles Symposium (IV 1995)*, pages 506–511, 1995.

[83] H. S. Poon, F. Mai, Y. S. Hung, and G. Chesi. Robust detection and tracking of multiple moving objects with 3d features by an uncalibrated monocular camera. In *Proc. 4th International Conference on Computer Vision/Computer Graphics Collaboration Techniques*, pages 140–149, Berlin, Heidelberg, 2009. Springer-Verlag.

[84] M. Schweitzer, A. Unterholzner, and H. J. Wuensche. Real-time visual odometry for ground moving robots using gpus. In *Proc. International Conference on Computer Vision Theory and Applications (VISAPP 2010)*, Angers, France, may 2010.

[85] S. C. Sekhar, F. Aguet, S. Romain, P. Thévenaz, and M. Unser. Parametric b-spline snakes on distance maps—application to segmentation of histology images. *Proc. 16th European Signal Processing Conference (EUSIPCO 2008)*, august 2008.

[86] J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 1994)*, pages 593–600, june 1994.

[87] R. Siegwart and I. R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. Bradford Company, Scituate, MA, USA, 2004.

[88] G. Silveira, E. Malis, and P. Rives. An efficient direct method for improving visual SLAM. In *Proc. IEEE International Conference on Robotics and Automation (ICRA 2007)*, pages 4090–4095, 2007.

[89] N. Simond and M. Parent. Obstacle detection from IPM and super-homography. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, pages 4283–4288, 2007.

[90] J. Sola. *Towards Visual Localization, Mapping and Moving Objects Tracking by a Mobile Robot : a Geometric Probabilistic Approach*. PhD thesis, LAAS-CNRS, 2007.

[91] J. Sola, A. Monin, and M. Devy. BiCamSLAM : two times mono is more than stereo. In *Proc. IEEE International Conference on Robotics Automation (ICRA 2007), Rome, Italy*, pages 4795–4800, 2007.

[92] S. Terho T. Peynot and S. Scheding. Sensor data integrity : Multi-sensor perception for unmanned ground vehicles. Technical report acfr-tr-2009-002, Australian Centre for Field Robotics (ACFR), The University of Sydney, 2009.

[93] A. Talukder and L. Matthies. Real-time detection of moving objects from moving vehicles using dense stereo and optical flow. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, pages 3718–3725, september 2004.

[94] C. Tay, K. Mekhnacha, C. Chen, M. Yguel, and C. Laugier. An Efficient Formulation of the Bayesian Occupation Filter for Target Tracking in Dynamic Environments. *International Journal Of Autonomous Vehicles*, 2007.

[95] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[96] D. Vasquez and T. Fraichard. Motion prediction for moving objects : A statistical approach. *Proc. IEEE International Conference on Robotics and Automation (ICRA 2004)*, pages 3931–3936, april 2004.

[97] A. D. Vasquez-Govea. *Incremental Learning for Motion Prediction of Pedestrians and Vehicles*. PhD thesis, Institut National Polytechnique de Grenoble, 2010.

[98] T. Veit, F. Cao, and P. Bouthemy. Space-time a contrario clustering for detecting coherent motion. In *IEEE International Conference on Robotics and Automation (ICRA 2007)*, pages 33–39, Roma, Italy, april 2007.

[99] C.C. Wang. *Simultaneous Localization, Mapping and Moving Object Tracking*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2004.

[100] C.C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *International Journal of Robotics Research*, 2007.

[101] T. Williamson. *A High-Performance Stereo Vision System for Obstacle Detection*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, september 1998.

[102] K. Yamaguchi, T. Kato, and Y. Ninomiya. Vehicle ego-motion estimation and moving object detection using a monocular camera. In *Proc. 18th International Conference on Pattern Recognition (ICPR 2006)*, volume 4, pages 610–613, 2006.

[103] M. Yguel, C. Tay, K. Mekhnacha, and C. Laugier. Velocity estimation on the bayesian occupancy filter fo multi-traget tracking. Technical Report 5836, Institut Nacional de Recherche en Informatique et en Automatique INRIA, january 2005.

Résumé

Cette thèse traite de la détection et du suivi d'objets mobiles dans un environnement dynamique, en utilisant une caméra embarquée sur un robot mobile. Ce sujet représente encore un défi important car on exploite uniquement la vision mono-caméra pour le résoudre. Nous devons détecter les objets mobiles dans la scène par une analyse de leurs déplacements apparents dans les images, en excluant le mouvement propre de la caméra. Dans une première étape, nous proposons une analyse spatio-temporelle de la séquence d'images, sur la base du flot optique épars. La méthode de *clustering a contrario* permet le groupement des points dynamiques, sans information a priori sur le nombre de groupes à former et sans réglage de paramètres. La réussite de cette méthode réside dans une accumulation suffisante des données pour bien caractériser la position et la vitesse des points. Nous appelons *temps de pistage*, le temps nécessaire pour acquérir les images analysées pour bien caractériser les points. Nous avons développé une carte probabiliste afin de trouver les zones dans l'image qui ont les probabilités la plus grandes de contenir un objet mobile. Cette carte permet la sélection active de nouveaux points près des régions détectées précédemment en permettant d'élargir la taille de ces régions. Dans la deuxième étape nous mettons en œuvre une approche itérative pour exécuter détection, clustering et suivi sur des séquences d'images acquises depuis une caméra fixe en intérieur et en extérieur. Un objet est représenté par un contour actif qui est mis à jour de sorte que le modèle initial reste à l'intérieur du contour. Finalement nous présentons des résultats expérimentaux sur des images acquises depuis une caméra embarquée sur un robot mobile se déplaçant dans un environnement extérieur avec des objets mobiles rigides et non-rigides. Nous montrons que la méthode est utilisable pour détecter des obstacles pendant la navigation dans un environnement inconnu a priori, d'abord pour des faibles vitesses, puis pour des vitesses plus réalistes après compensation du mouvement propre du robot dans les images.

Mots-clés: Objets mobiles, détection, suivi, groupement, vision monoculaire.

Abstract

This dissertation concerns the detection and the tracking of mobile objets in a dynamic environment, using a camera embedded on a mobile robot. It is an important challenge because only a single camera is used to solve the problem. We must detect mobile objects in the scene, analyzing their apparent motions on images, excluding the motion caused by the ego-motion of the camera. First it is proposed a spatio-remporal analysis of the image sequence based on the sparse optical flow. The *a contrario* clustering method provides the grouping of dynamic points, without using a priori information and without parameter tuning. This method success is based on the accretion of sufficient information on positions and velocities of these points. We call *tracking time*, the time required in order to acquire images analyzed to provide the points characterization. A probabilistic map is built in order to find image areas with the higher probabilities to find a mobile objet ; this map allows an active selection of new points close the previously detected mobile regions, making larger these regions. In a second step, it is proposed an iterative approach to perform the detection-clustering-tracking process on image sequences acquired from a fixed camera for indoor or outdoor applications. An object is described by an active contour, updated so that the initial object model remains inside the contour. Finally it is presented experimental results obtained on images acquired from a camera embedded on a mobile robot navigating in outdoor environments with rigid or non rigid mobile objects ; it is shown that the method works to detect obstacles during the navigation in a priori unknown environments, first with a weak speed, then with more a realistic speed, compensating the robot ego-motion in images.

Keywords: Mobile objects, detection, tracking, clustering, monocular vision