

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'Institut National des Sciences Appliquées de Toulouse*
Discipline ou spécialité : *Systèmes Embarqués et Robotique*

Présentée et soutenue par *David Alberto MARQUEZ GAMEZ*
Le *26 octobre 2012*

Titre : *Towards Visual Navigation in Dynamic and Unknown Environment: Trajectory Learning and Following, with Detection and Tracking of Moving Objects*

JURY

Président :	*M. Simon LACROIX*
Rapporteurs :	*M. Fawzi NASHASHIBI*
	M. Juan ANDRADE CETTO
Examinateurs :	*M. Roland CHAPUIS*
	M. Joan SOLA
Directeur :	*M. Michel DEVY*

Ecole doctorale : *Systèmes (EDSYS)*
Unité de recherche : *Laboratoire d'Analyse et d'Architecture des Systèmes - LAAS-CNRS*
Directeur de Thèse : M. *Michel DEVY*

École Doctorale systèmes

Thése

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Institut National des Sciences Appliquées de Toulouse

Spécialité : Systèmes Embarqués et Robotique

Présentée et soutenue publiquement le 26 octobre 2012

Towards Visual Navigation in Dynamic and Unknown Environment: Trajectory Learning and Following, with Detection and Tracking of Moving Objects

David Alberto Márquez Gámez

Préparée au Laboratoire d'Analyse et d'Architecture de Systèmes du CNRS
sous la direction de M. Michel Devy

Jury

M. Simon Lacroix	Président
M. Fawzi Nashashibi	Rapporteur
M. Juan Andrade Cetto	Rapporteur
M. Roland Chapuis	Examinateur
M. Joan Solà	Examinateur
M. Michel Devy	Directeur de thèse

Towards Visual Navigation in Dynamic and Unknown Environment: Trajectory Learning and Following, with Detection and Tracking of Moving Objects

Abstract:

The global objective of these works concerns the navigation of autonomous robots on long routes in outdoor dynamic environments, more precisely on the development and the evaluation of advanced perception functions, embedded on vehicles moving in a convoy formation, on an a priori unknown route in urban or natural environments.

Three issues are tackled: first several methods from the State of the Art have been integrated in order to cope with the visual mapping and the trajectory learning problems for a vehicle A equipped with a stereovision sensor, moving in a large-scale environment, assumed static. Then it is proposed two modes for the execution of this trajectory by a vehicle B equipped by a single camera: either a delayed mode, in which B loads initially all representations learnt by A, and executes alone the recorded trajectory, or a convoy mode, in which B follows A, which sends him by a communication link, the trajectory sections as soon as they are learnt. Finally, it has been considered changing and dynamic environments, dealing with the detection of events from images acquired on a dynamic vehicle: detection of changes (disappearances or appearances of static objects, typically cars parked in a urban environment), or detection of mobile objects (pedestrians or other vehicles).

About the trajectory learning, we have adapted the Bi-Cam SLAM approach proposed by J.Solà (PhD, LAAS, 2007), in order to learn long routes. We proposed a multi-maps method, and an off-line optimization step executed from the learnt representations in order to improve the accuracy both for the trajectory and for the maps. The method proposed by H.Gibson (ACFR, 2009), based on the information filter, is based on the fusion of landmarks perceived jointly on several maps, making possible to improve the estimation of the transforms between the reference frames attached to every map.

About the trajectory execution in the delayed mode, the learnt model is first used so that the B vehicle computes its initial position with respect to the first map from which the trajectory to be executed is expressed. Then we have used the control method Tiji developed by V.Delsart (PhD, Grenoble, 2010) so that the robot gets to the trajectory and then, follows it; this method takes as input, the robot localization, computed from the active search of the map landmarks. In order to deal with changing environments, landmarks that are not matched on several successive images, are removed from the map, and replaced by new ones perceived in the same image regions. In the convoy mode, the A and B vehicles move in a coordinated way; A sends to B a submap as soon as it is learnt; B executes the trajectory sections of the received submaps, and if required, is stopped on the end position of such a section if the following one is not yet received. A possible joint perception between A and B has not been considered in our works.

Finally, we cope with the Mobile Object Detection and Tracking from a mobile vehicle. Two methods have been combined for the visual detection of mobile obstacles either pedestrians or vehicles; the method proposed by D.L.Almanza (PhD LAAS, 2011), based on the optical flow, provides hypothesis, which are verified by a classification method based on a learning data base, used to identify a pedestrian or a vehicle in each image region assumed to be dynamic. A MOT module (Mobile Object Tracking) based on The Extended Kalman Filter, estimates the 3D positions of points detected on every obstacle; filtering, fusion and aggregation functions are used to update these sets of 3D points, and the including 3D boxes used as models of the tracked obstacles, known with respect to the robot reference frame.

This PhD was prepared in the context of the RINAVEC project (Reconnaissance of routes and convoy navigation of communicating vehicles), in close collaboration with Thales Optronique S. A and the LASMEA Laboratory; a number of experiments or simulations prepared with Thales or LAAS robots, validates the contributions presented in this PhD manuscript.

Vers une navigation visuelle en environnement dynamique inconnu : apprentissage et exécution de trajectoire avec détection et suivi d'objets mobiles

Résumé :

L'objectif de ces travaux porte sur la navigation de robots autonomes sur de grandes distances dans des environnements extérieurs dynamiques, plus précisément sur le développement et l'évaluation de fonctions avancées de perception, embarquées sur des véhicules se déplaçant en convoi sur un itinéraire inconnu a priori, dans un environnement urbain ou naturel.

Nous avons abordé trois problématiques : d'abord nous avons exploité plusieurs méthodes de l'état de l'art, pour qu'un véhicule A, équipé d'un capteur stéréoscopique, apprenne à la fois une trajectoire et un modèle de l'environnement supposé d'abord statique. Puis nous avons proposé deux modes pour l'exécution de cette trajectoire par un véhicule B équipé d'une simple caméra : soit un mode différé, dans lequel B charge toute la trajectoire apprise par A, puis l'exécute seul, soit un mode convoi, dans lequel B suit A, qui lui envoie par une communication HF, les tronçons de la trajectoire au fur et à mesure qu'ils sont appris. Enfin nous avons considéré le cas des environnements évolutifs et dynamiques, en traitant de la détection d'événements depuis les images acquises depuis un véhicule mobile : détection des changements (disparition ou apparitions d'objets statiques, typiquement des véhicules garés dans un milieu urbain), ou de la détection d'objets mobiles (autres véhicules ou piétons).

Sur l'apprentissage de trajectoires, nous avons adapté la méthode dite Bi-Cam SLAM proposée dans les travaux de J.Sola (thèse LAAS, 2007), pour l'appliquer afin d'apprendre des itinéraires longs. Nous proposons une approche multi cartes, et une phase d'optimisation hors ligne en fin d'apprentissage afin d'améliorer la précision de la trajectoire et des cartes apprises. La méthode de H.Gibson (ACFR, 2009), exploitant un filtre d'information, permet de fusionner les perceptions d'amers communs à plusieurs cartes, et ce faisant de ré estimer les transformations

inter-cartes. Concernant l'exécution des trajectoires en mode différé, le modèle appris est exploité d'abord pour que le véhicule B calcule d'abord sa position initiale par rapport à la première carte qui permet de définir la trajectoire à exécuter; puis nous exploitons la méthode Tiji développée par V.Delsart (thèse Grenoble 2010) afin de rejoindre la trajectoire, puis de la suivre; ce module prend en entrée la localisation du robot, calculée à partir de la recherche active des amers de la carte. Pour pouvoir se déplacer dans des environnements évolutifs, des amers non retrouvés sur plusieurs images successives, sont supprimés et remplacés par d'autres amers détectés dans la même zone de l'image. En mode convoi, les véhicules A et B se déplacent de manière coordonnée : A envoie vers B une sous-carte une fois qu'elle est apprise; B exécute successivement les tronçons attachés à chaque sous-carte, et si nécessaire, s'arrête sur la dernière position d'un tronçon si le suivant n'est pas reçu. Une possible perception conjointe entre A et B n'est pas considérée dans nos travaux.

Enfin, nous avons traité de la détection d'obstacles mobiles depuis un véhicule lui-même mobile. Deux approches sont combinées pour la détection visuelle des obstacles mobiles, soit de type piétons, soit de type véhicules; la méthode proposée par D.L.Almanza (thèse LAAS, 2011) fondée sur le flot optique, permet de générer des hypothèses, qui sont ensuite confirmées par une méthode de classification exploitant une base d'apprentissage, pour identifier un piéton ou un véhicule en chaque région de l'image supposée dynamique. Un module MOT (Mobile Object Tracking) exploite le filtrage de Kalman étendu afin d'estimer la position 3D des points suivis sur chaque obstacle; des procédures de fusion, filtrage, agrégation, permettent ensuite de gérer ces ensembles de points 3D, et leurs boîtes englobantes exploitées comme modèles de ces obstacles, connus relativement au véhicule.

Cette thèse a été préparée dans le contexte du projet RINAVEC (Reconnaissance d'Itinéraires et NAvigation en convoi de VEhicules Communicants), en collaboration avec Thales Optronique S. A et le laboratoire LASMEA: de nombreuses expérimentations et simulations préparées pour ce projet avec les démonstrateurs de Thales ou du LAAS, ont permis de valider les approches présentées dans cette thèse.

The only way to do great work is to love what you do. If you haven't found it yet, keep looking. Don't settle. As with all matters of the heart, you'll know when you find it.

Steve Jobs

Contents

1 Introduction **15**

1 Motivation 16

2 Aims and contributions of this Thesis 20

2.1 Visual trajectory learning and following in unknown routes for autonomous navigation 20

2.1.1 Proposed system overview 22

2.2 The problem of dynamic environments 23

2 Visual trajectory learning and world mapping **25**

1 Introduction 26

2 Bi-Camera SLAM 27

2.1 System Overview 28

2.2 Overall Architecture 29

3 Multi-Map Bi-Camera SLAM 30

3.1 System Overview 32

3.1.1 Local Level 33

3.1.2 Global Level. 34

4 Trajectory Optimization based submap joining 35

4.1 Refinement of the global map 35
5 From local maps to convoy navigation 37
6 Experiments 38
6.1 Real-data Experiments 39
7 Conclusion 40

3 Visual trajectory replay and map update 43

1 Introduction 45
2 Replaying a learned trajectory 46
2.1 Patch search and data association 46
2.2 Matching points from 3D landmarks projections 47
2.3 Replacing points and update of the map 48
3 Localization from the learned map 49
3.1 Initial position 49
3.2 On-line Localization 50
3.2.1 Prediction step 51
3.2.2 Data association 52
3.2.3 Estimation step 52
4 Path tracker: trajectory following 54
4.1 Trajectory generation: from robot position to reference path 56
5 Visual trajectory learning and replay: system overview 56
6 Experiments 57
6.1 Replaying the learnt trajectory 58
6.2 Tracking the reference path 60
7 Conclusion 61

4 Visual exploration in dynamic environments 63

1 Introduction 65

2 Related Works 66

3 Moving Objects: Model and Representation 68

3.1 Moving Objects Model 69

3.2 Moving Objects Representation 69

4 System Description 70

4.1 Feature Selection 70

4.2 Feature classification 71

4.3 Motion Grid 72

4.3.1 Motion grid initialization 73

4.3.2 Motion grid update 75

4.4 Classification approach 76

4.4.1 Rectangular filters or Haar-like features 77

4.4.2 Histogram of oriented gradient 78

4.4.3 AdaBOOST 79

4.4.4 Weak classifier-Haar 80

4.4.5 Weak classifier-HoG 80

4.5 Moving Objects detection 81

4.5.1 Object Hypotheses: classification based approach 81

4.5.2 Final Object Hypotheses fusion rule 82

4.6 Moving Objects tracking 84

4.6.1 Moving points and Objects creation 84

4.6.2 Moving Objects and Robot Motion compensation 86

4.6.3 Moving Objects motion 87

5 Experiments 88

6 Conclusion 90

5 Applications **93**

1 Preliminary 94

2 The MORSE Simulator 94

3 Learning and Replay: a real robot experiment 95

3.1 Learning step 96

3.2 replay step 97

4 Convoy Navigation 101

5 Active Visual-based Detection and Tracking of Moving Objects . 104

5.1 Indoor Experiment: 2D detection and tracking task . . . 104

5.2 Outdoor Experiment: Bi-Camera SLAM-DATMO 105

6 Conclusions and Future Directions **109**

1 Conclusions 109

2 Future Directions 112

A Appendix A **113**

B Appendix B **125**

Bibliography **129**

List of Figures

1.1 An overview of the major processing blocks in our proposed system 22

2.1 An overview of the learning-mapping approach in our proposed system 26
2.2 Main objects in our Bi-Camera SLAM context 29
2.3 Multi-map SLAM representation 33
2.4 The Segway platform and an overhead view of the learned trajectory in the experiment 39
2.5 Global map obtained using our hierarchical/hybrid BiCam SLAM approach 40
2.6 Robot trajectory estimate: robot trajectory learned path and true robot trajectory (GPS) 41

3.1 An overview of the replay-localization approach in our proposed system 45
3.2 Two screen shots of a sequence in a parking lot, in the morning at 10 am and in the afternoon at 6 pm 48
3.3 Results of initial position from two cameras mounted on an on board vision bench 50
3.4 An overview of the path tracker processing block in our proposed system 55
3.5 Visual trajectory learning and replay system 57

3.6 The Segway platform and an overhead view of the learned and replayed trajectory in the experiment. 58
3.7 LAAS experiment: global map; trajectory estimate; trajectory learned path and trajectory replayed path 59
3.8 An experiment at LAAS: replay-localization works in spite of bad matchings on occluded landmarks 60
3.9 MORSE experiment: trajectory estimate; trajectory learned path; trajectory replayed path 61
3.10 MORSE experiment. Experimental results: replay-localization step 62

4.1 SLAM with DATMO: the major processing blocks in our system and their connections 71
4.2 The *Feature classification* process 72
4.3 *Motion Grid* principle 74
4.4 Set of Haar-like features. Black and white areas have negative and positive weights respectively 77
4.5 Original image and the output obtained by applying the vertical and horizontal Haar filters 78
4.6 Example tracking results and performance plot of our system for the first sequence 89
4.7 Experiment results: image plane and top views of the produced 3D map 91

5.1 The MORSE Simulator 95
5.2 Learning and Replay: an overview of the major processing blocks in our proposed system 96
5.3 Learning and Replay: a real robot experiment. View of the learned and replayed trajectory 97
5.4 Learning and Replay: a real robot experiment. The Segway platform used in the experiment 97
5.5 Learning and Replay experiment: $2D$ and $3D$ plots of the global map obtained using our approach 98
5.6 Learning and Replay experiment: trajectory estimate; trajectory learned path; trajectory replayed path 99

5.7 Learning and Replay experiment: images acquired on the same area. Learning and replay step 100
5.8 Convoy experiment 102
5.9 Convoy experiment: robot trajectory estimate; robot trajectory learned path; robot trajectory replayed path . . . 103
5.10 The major processing blocks in our SLAM-DATMO system and their connections 104
5.11 Indoor Experiment: detection clustering and tracking objects 106
5.12 Outdoor Experiment: *Motion Grid* evolution 107
5.13 Outdoor experiment results: image plane and top views of the produced 3D map 108

A.1 The RINAVEC project. V_0 leader robot, V_1, V_2, V_3 followers robots 114
A.2 Satellite view of the experiment site 116
A.3 R-trooper and VELAC; the robotics vehicles used in the project 117
A.4 Trajectory executed in the learning and replay experiment 117
A.5 Trajectory estimation in the learning step and 3D-points map 118
A.6 Trajectory estimate results of the replay procedure . . . 118
A.7 Trajectory executed in the replay step 119
A.8 View of the terrain travelled in the learning step 119
A.9 Map and trajectory estimated in the learning step 120
A.10 The terrain travelled in the replay step 120
A.11 Map updated in the learning step. A sample of the new patchs added to the map 121
A.12 Template-Based Tracking 122
A.13 Detecting the leader vehicle: Representative snapshots of the sequence 123

B.1 Overview of the SLAM problem with the principal data structures. 126

1 Introduction

The Answer to the Great Question... Of Life, the Universe and Everything... Is... "Forty-two," said Deep Thought, with infinite majesty and calm.

Douglas Adams, The Hitchhiker's Guide to the Galaxy

Contents

1 Motivation 16

2 Aims and contributions of this Thesis 20

2.1 Visual trajectory learning and following in unknown routes for autonomous navigation . . . 20

2.2 The problem of dynamic environments 23

1 Motivation

During the last few years, the number of robots in operation is exploding, under the combined effects of technical progress, to lower costs and the emergence of new socio-economic needs (security, defence, assistance to elderly, transportation, etc.). Domestic robots (such as automatic vacuum cleaners and lawn machines) have already invaded our homes and gardens. Thus, the actual growth in the number of robots in our environment is remarkable.

We can define mobile robotics as the discipline that investigates and attempts to develop systems capable of moving robots autonomously in their environment. This autonomy means no operator intervention of any human in the process of robot motion.

To be considered as such, a mobile robot must have a drive system that allows him to move. Depending on the applicative context, the drive system may be based on wheels, tracks or legs if the robots are terrestrial, but there are also aerial or underwater robots which drive system is based on fluid mechanical drive. In the same way, a mobile robot must have sensors that allow him to perceive the environment, such as ultrasonic sensors, laser range finders or cameras, and processing capabilities (computers, microprocessors,...) to execute the algorithms to accomplish a particular task. These algorithms that form an intelligent connection between perception and action, are those that implement the capabilities of autonomous navigation for a mobile robot.

The problem of autonomous navigation of a mobile robot can be summarized by three key questions: “Where am I?”, “Where am I going?”, and “How should I get there?” [Leonard & Durrant-Whyte 1991] . Probably the first question is the most important, which focuses the major research efforts of many groups around the world. The second one has not been addressed deeply enough and is an active research topic, since for most of the known applications, the destination or target to be achieved by the robot is defined by a user system, and only in applications of exploration of unknown environments in a very rudimentary level. The third question is answered with the path planning algorithms. These al-

gorithms are mature enough for effective implementation and operation in a mobile robot as well as the various techniques of low-level control, suitable for real time implementation.

Searching for the answer to the question "Where am I?", the robotic community has been developed different localization algorithms. We can find different scenarios possible: first, a prior information is available (e.g. map of its surrounding environment), in this case, the robot tries to localize itself (in an existing map) based in this knowledge; second, no previous information available, in this case the robot tries to localize itself and build a map of the environment without any prior information.

The problem of robot localization in a known map can be approached from different perspectives: (1) continuous localization or position tracking, (2) global localization and (3) relocalization. Continuous localization assumes that the initial position of the robot is relatively well known. Localization methods that solve the this problem continue to provide good estimations to robot position as the robot moves through the environment despite uncertain motion and error filled sensor observations. The global localization problem assumes that the initial position of the robot is unknown. Localization methods that solve this issue typically need to deal with parts of the environment that are indistinguishable from other parts of the environment due to structural similarity. The relocalization problem occurs when a robot may be certain of its position, but given further observations and motion, realizes that it is completely lost. This is the case when a localization method estimates the incorrect robot position, which may be due to environment similarities. The relocalization problem is also known as the kidnapped robot problem.

The first algorithms to address the problem of build or update a map of the environment while the robot moves in the it, were solved separating the problem of map construction and localization. So that, they build a map with the assumption that the robot position is correct and try to compute a localization assuming that the map is correct. Very soon it is demonstrated that the rigorous solution of the problem is not possible without considering both aspects simultaneously.

Since the works of [Chatila & Laumond 1985, Smith & Cheeseman 1987,

Ayache & Faugeras 1988, Durrant-Whyte 1988, Crowley & Crowley 1989, Smith *et al.* 1990] among others, the problem of building a map and simultaneously compute the robot's position on that map is known as the "SLAM" (Simultaneous Localization and Mapping) problem. Thus, SLAM, also called *Concurrent Mapping and Localization* (CML), has become a major scientific research topic, in which several researchers are working around the world.

So, being considered the SLAM problem as one of the most important pieces in the search for effective autonomy in mobile robotics, is still an open problem, not completely solved, although in recent years there has been considerable progress in the topic.

The work described in this thesis has been carried out in the framework of the RINAVEC[1] project funded by ANR, the french Association Nationale de la Recherche. The development of this project was performed in close collaboration with Thales Optronique S. A (France) and Lasmea Laboratory in Clermont-Ferrand (France). The main goal of RINAVEC is to develop and evaluate advanced features perception and modelling environment for vehicles moving on a route unknown a priori, open environment (suburban or natural) with the objective of implement a "convoy" of communicating vehicles able to accurately follow a path created by a leader vehicle using only low cost sensors.

Thus, the global objective of this thesis, addresses the cooperative visual mapping and localization problem for multiple vehicles, moving on a priori unknown routes. A large-scale learning-mapping approach and a map-based replay-localization method are combined to achieve cooperative navigation. The learning-mapping approach is based on a proposed multi-map Bi-Camera EKF-SLAM and the replay-localization approach, exploits a localization method based on an active search procedure. Our proposed system can be generalized to be executed on multiple vehicles moving as a convoy. Thus, a global 3D map maintains the relationships between a series of local maps built by the first vehicle of the convoy (leader), defining a path that all other vehicles (followers) must stay on. Only single camera setups are considered.

[1] http://www.lasmea.univ-bpclermont.fr/Rinavec/

The world is assumed static, but with moved objects between two passages. The robots are equipped with low cost proprioceptive sensors (odometer, gyro) or GPS in order to compute an initial coarse estimate of its position, and with cameras (monocular or stereo vision) in order to acquire image sequences and to extract and characterize 3D features from the perceived scenes. These 3D landmarks will be tracked and identified between successive images acquired during robot motions, making the robot able to locate itself accurately.

On the other hand, we address the problem of navigation in dynamic environments, understanding as "dynamic", an environment with moving objects. Thus, this thesis describes a method proposed for the detection, the tracking and the identification of moving objects, detected from a mobile camera, typically a camera embedded on a robot. A global architecture is presented, using only vision, in order to solve simultaneously several problems: the camera (or vehicle) localization, the environment mapping and the detection and tracking of moving objects. The goal is to build a convenient description of a dynamic scene from vision: what is static? what is dynamic? where is the robot? how do other mobile objects move?. In this way, it is proposed to combine two approaches: first a clustering method allows to detect static points, to be used by the SLAM algorithm and dynamic ones, to segment and estimate the status of moving objects. Second a classification approach allows to identify objects of known classes in image regions. These two approaches are combined in an active method based in a Motion Grid in order to select actively where to look for moving objects.

It has been shown [Strasdat *et al.* 2010] that visual SLAM methods based on non linear optimization (Incremental Sparse Bundle Adjustment, or SBA) converge better than EKF-SLAM or more generally, methods based on filtering. Here why do we use an EKF-SLAM method?

First in this application, loop closure is not mandatory: the method must both build the map and the trajectory, but the evaluation criterion will be based on the capability for a robot to replay a learned trajectory, whatever the drift with respect to the exact localization in a global frame. Non linear optimization is only used off line in order to refine the map and

the learned trajectory. Then for the convoy configuration, off line refinement is forbidden. Thus, in our system, EKF-SLAM with a Bi-Camera approach has been selected has the best way to generate submap's that could be communicated from the leader robot to the followers ones.

2 Aims and contributions of this Thesis

2.1 Visual trajectory learning and following in unknown routes for autonomous navigation

[Márquez-Gámez & Devy 2010, Márquez-Gámez & Devy 2011a, Márquez-Gámez & Devy 2011b, Márquez-Gámez & Devy 2012c]

Many robotic missions can be more efficiently and robustly achieved by a team of robots. This thesis is focused on the cooperation of several vehicles, moving alone or as a convoy in open environments (urban periphery or countryside), on a priori unknown routes. Although most existing mobile robotic applications involve a single robot, a wide range of potential applications require multiple robots to execute joint tasks, e.g. rescue robotics (several unmanned or driven vehicles cooperating to help rescue workers), cooperative monitoring (several aerial and terrestrial robots used to detect fire in a large forest or intruders in a forbidden zone) [Yamashita *et al.* 2003, Franchi *et al.* 2009, Vig & Adams 2006]

Localization is a key technology to address how the robots localize themselves in the operating unknown environment and how they know their local poses with respect to other objects. A variety of approaches have been reported for localization of multirobot formations. In [Mariottini *et al.* 2005], the localization problem of a leader-follower system, is based on EKF to estimate each follower's pose with respect to the leader. In [Balch & Arkin 1998], it is proposed a behavior-based approach for maintaining formation of a team of robots, with experiments performed on outdoor unmanned ground vehicles equipped with vision, GPS and hazard sensors. In [Madhavan *et al.* 2004], a scheme for distributed outdoor localization for a team of robots is based on heterogeneous sensors such as local area differential global positioning system

(LADGPS) and cameras. Most of the above approaches focused on the relative localization only, and a few of them discussed how to globally localize robots. Many methods simply assume that robots are equipped with absolute positioning capabilities.

In this thesis, we present a complete system for large scale autonomous operation of a team of robots in a "convoy" formation navigating in a dangerous, unknown and changing outdoor environment, typically on routes that could be mined. The proposed system consists of 2 steps: (1) learn first a safe path using an unmanned robot, and (2) follow this same path by other vehicles of the fleet. The two steps involve different functions: the initial path learning requires SLAM, i.e. to simultaneously memorize successive robot positions on the path and landmark positions that are used by the robot to locate itself. When the leader's path is traveled again by other vehicles of the convoy, the map will be exploited to localize and control a vehicle so that it is maintained on the same path, with a tolerance that must be minimized.

It is assumed that all vehicles are only equipped by low-cost sensors, and that differential or RTK GPS receivers are not available. Vehicles could only use embedded sensors (cameras, odometers, low-cost GPS or IMU) during the learning or replay steps.

The temporal gap between the two steps depends on the mission:

1. the learning and replay functions could be executed independently, i.e. the map and the path are acquired at time A, and the safe trajectory is followed again later at time B, with $B - A$ equal to several hours or days. In such a situation, the environment could change between the two steps: recorded landmarks could be removed during the interval, or the path itself could get blocked. Moreover vehicles used at time A and B could be the same, or could be different, involving the use of different cameras, so possibly different radiometric information.

2. these functions could be executed on different communicating vehicles navigating in a convoy formation. A leader vehicle records the map and the path, and sends these information to the second ve-

hicle of the convoy using a wireless network; the first follower stays on the path taken by the leader, updating the map from its own observations and sends the updated map and the path to the next follower. Mutual localization (the follower sees the leader) is possible, but not mandatory. The leader and followers are equipped with different cameras, involving possible radiometric differences on images.

2.1.1 Proposed system overview

The major processing blocks of our system are depicted in Fig. 1.1. The system consists of 2 steps:

1. the learning-mapping step, where a leader vehicle, builds several $3D$-points maps and learn a safe path, and
2. the replay-localization step, where the information learned and transmitted by the leader, allows other vehicles of the fleet, to follow the learned safe path.

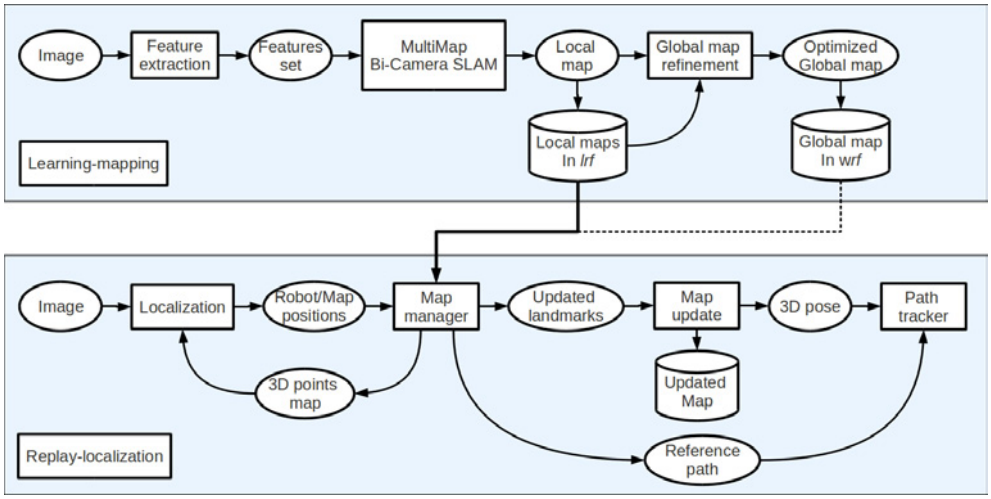


Figure 1.1: An overview of the major processing blocks in our proposed system.

The two steps involve different functions: the initial path learning is based on a proposed Multi-map SLAM using only vision in a Bi-Camera

configuration and the replay procedure exploits a localization method based on an active search procedure to localize and control a vehicle so that it is maintained on the same path than during the learning step, with a tolerance that must be minimized.

Chapters 2 and 3 present in detail the learning and replay approaches implemented in our proposed system.

2.2 The problem of dynamic environments

[Márquez-Gámez & Devy 2012a, Márquez-Gámez & Devy 2012b]

SLAM systems are capable of estimating both vehicle trajectory and world structure. It is now worth mentioning that one of the strongest hypothesis that has to be made to put SLAM systems into work is the fact that world landmarks are static. This hypothesis provides two important advantages: on one hand, landmark position estimates can be progressively refined; on the other hand, they can be used as external references for robot localization. In this thesis we are going to introduce some moving objects in the perceived world, while still trying to solve the SLAM problem and, additionally, recover each moving object's dynamics.

The SLAM with detection and tracking of moving objects problem is not only to solve the SLAM problem in dynamic environments, but also to detect and track these dynamic objects. In order to keep SLAM solvable, we will keep a significant amount of fixed landmarks. Thus, on the one hand the problem remains the same to existing SLAM algorithms with static landmarks and on the other hand to deal with moving objects new ideas are proposed

In this way, we have developed a complete system to solve the SLAM with Detection and Tracking of Moving Objects problem with the aid of vision. The goal is to achieve a convenient description of the visually perceived dynamic scene: how the static world is, where the robot is in this world, and how other existing bodies move. In order to achieve detection and tracking of moving objects as a whole, we need to solve

two problems of different nature: *Moving Objects Detection* (MOD) and *Moving Objects Tracking* (MOT).

Thus, on the one hand, the MOT problem is solved by filtering techniques, using the Bi-Camera algorithm developed in Chapter 2, and on the other hand, to deal with the MOD problem, in Chapter 4, we propose a method that, with a similar objective as active feature search, exploits the knowledge we have of the system to anticipate where potential moving objects may appear in the image, thus focusing the detecting procedure to those areas, allowing both fast and efficient detection, recognition and tracking of moving objects.

Visual trajectory learning and world mapping

2

The real act of discovery consists not in finding new lands,but in seeing with new eyes.

Marcel Proust

Contents

1 Introduction 26

2 Bi-Camera SLAM 27

2.1 System Overview 28

2.2 Overall Architecture 29

3 Multi-Map Bi-Camera SLAM 30

3.1 System Overview 32

4 Trajectory Optimization based submap joining 35

4.1 Refinement of the global map 35

5 From local maps to convoy navigation 37

6 Experiments 38

6.1 Real-data Experiments 39

7 Conclusion 40

1 Introduction

This chapter describes the design and testing of a learning-mapping procedure to enable long-range localization and mapping using a pair of cameras in a "Bi-Camera" configuration as the only sensor. A system overview of our approach is shown in Fig. 2.1. During this phase, the robot is piloted along a route capturing monocular images. This visual information is processed into a Multi-Map Bi-camera SLAM to learn the path travelled and generate a global 3D map of topologically-connected local maps that may be used by a second robot to repeat autonomously the learned path. This second step of repeat and track a learned path, will be explained in next chapter.

The use of small local maps on one hand decouples the computational complexity of our algorithm, and on the other hand allows the system to operate in a multi-robot configuration to exploit a convoy task. We validate the overall approach with real data acquired in an urban environment.

It is worth noting that the algorithm presented in this chapter is now an important contribution to a research project called RINAVEC (**R**econnaissance d'**I**tinraires et **N**Avigation en convoi de **VE**hicules **C**ommunicants) sponsored by the French government and accompanied by Thales Optronique S. A (France) and Lasmea Laboratory in Clermont-Ferrand (France).

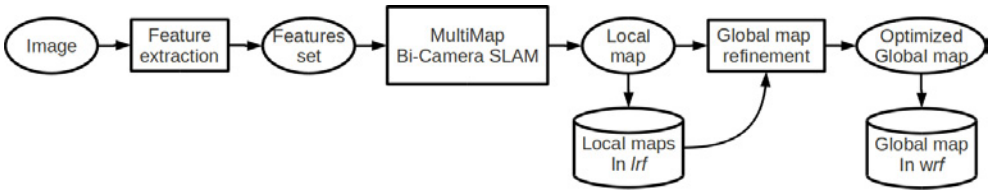


Figure 2.1: An overview of the learning-mapping approach in our proposed system.

The following chapter is organized as follow: In Section 2 Bi-Camera SLAM is presented. In Section 3 multi-map SLAM is analysed and a Multi-Map Bi-Camera SLAM is described. In Section 4, we present a

trajectory optimization method based in a submap joining algorithm. Section 5 present an analysis about bandwidth of the communication channel and how the local maps be transmitted to achieve convoy navigation. Finally, Section 6, gives some experimental results to validate the proposed system.

2 Bi-Camera SLAM

The fundamentals basis of Bi-Camera SLAM (BiCam SLAM) were originally proposed by Solà in [Solà 2007] and later extended to a Multi-Camera SLAM system in [Solà *et al.* 2008]. The idea behind Bi-Camera SLAM is to combine the advantages of monocular vision (bearing only, with infinity range but without instantaneous 3D information) and stereo-vision (3D information in a limited range) to build a system capable of instantly map "nearby objects" while still considering the information provided by the observation of "far objects".

Thus, Bi-Camera approach is based on the fusion of monocular and stereo information. In this way, we have the following advantages:

1. Instantaneous mapping of objects close to robot, thanks to stereo-like triangulation;
2. Bearing-only observations of far landmarks provides a good orientation localization;
3. Landmarks can be updated from any camera;
4. Images do not need to be rectified;
5. No need for precise calibration of extrinsic parameters of the camera.

It is know that the accumulation of angular errors due to uncertain robot motion makes simple SLAM algorithms become inconsistent and fail [Castellanos *et al.* 2004, Bailey *et al.* 2006]. Thus BiCam SLAM combining mono and stereo vision allows to dispose of long term absolute angle references, so, this long term observability will improve our localization and mapping performance.

2.1 System Overview

Bi-Cam SLAM has been described in depth by by Solà in their original publication [Solà 2007]. For the sake of completeness we summarize the important ideas and algorithmic steps but refer the readers to [Solà 2007] and [Solà *et al.* 2008] for the details.

The general equation of Bi-Cam SLAM system is described as follows. A central EKF-SLAM will hold the stochastic representation of the master camera C_L and slave camera C_R, plus the set of landmarks L_i, thus:

$$\mathbf{X} = \begin{bmatrix} C_L \\ C_R \\ L_1 \\ \vdots \\ L_N \end{bmatrix} \tag{2.1}$$

where the cameras states, is given by its position and orientation quaternion, $[C_T = (\mathbf{x}_T, \mathbf{q}_T) \in \mathbb{R}^7 \; with \; T = \{L, R\}$, and landmarks can be expressed either in inverse depth ($L_i = \mathbf{v}_i \in \mathbb{R}^6$) or in Euclidean coordinates ($L_i = \mathbf{p}_i \in \mathbb{R}^3$). As this representation must be incremented and updated properly at each frame, N_{obs} known landmarks are observed again by an active search procedure based on [Davison 2005], and N_{new} new ones are detected and added to the map. The algorithm's complexity increases linearly with the number of cameras and with the N_{obs} and N_{new} parameters, and quadratically with respect to the number of landmarks in the map.

Proprioceptive sensors (odometer, gyro) provide motion predictions $[\Delta x, \Delta y, \Delta \psi]$ in the robot's local plane. Gaussian uncertainties are added to the 6-DOF linear and angular components $[x, y, z, \phi, \theta, \psi]$ with a variance proportional to the measured forward motion Δx,

$$\{\sigma_x^2, \sigma_y^2, \sigma_z^2\} = k_L^2 \cdot \Delta x \tag{2.2}$$

$$\{\sigma_\phi^2, \sigma_\theta^2, \sigma_\psi^2\} = k_L^2 \cdot \Delta x \tag{2.3}$$

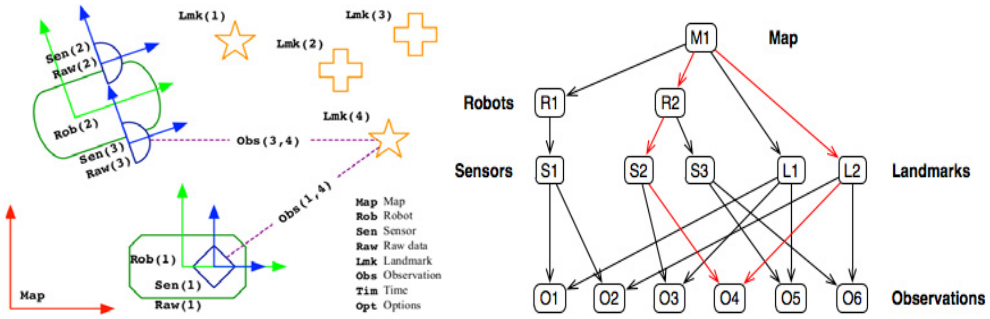


Figure 2.2: Main objects in our Bi-Camera SLAM context. Robots **Rob** have sensors **Sen**, sensors make observations **Obs** of landmarks **Lmk**. The stochastic map **Map** contains states of robots, sensors and landmarks. (Courtesy of the RT-SLAM project).

The variance in $[\phi, \theta, \psi]$ is mapped to the quaternion space using the corresponding Jacobians.

Events of camera motion, landmark initialization, and landmark observation are handled as in regular EKF-SLAM by just selecting the appropriate block elements from the SLAM state vector and covariance matrix, and applying the corresponding motion or observation models.

2.2 Overall Architecture

One of the main differences and contributions of our system with respect to the original BiCam SLAM approach presented in [Solà 2007] is that we have implemented the system taking into account "real-time" operation constraints to be implemented in a real robot platform. So, we address the problem from a "practical-implementation" point of view, initially thinking in the RINAVEC project but with enough generality to be used in different systems requiring a SLAM machinery. Thus, we try to establish intelligent strategies to simplify as much as possible the algorithm complexity and to speed up the process. This characteristic, usually neglected in many robotic research, allows practical solutions for live experiments on robots, for which localization and mapping require real-time execution and robustness. Thus, with this goal of *live experiment on robots* and *real-time* implementation, we define the main objects

in our Bi-Camera SLAM system following the basic concepts of the RT-SLAM library [1] [Roussillon *et al.* 2011]. In a typical SLAM problem, one or more robots navigate an environment, discovering and mapping landmarks by means of their on board sensors. The existence of robots of different kinds, carrying a different number of sensors of different kinds, which gather raw data and, by processing it, are capable of observing landmarks of different kinds.

Thus, the main objects in our Bi-Camera SLAM context are: the *world* that contains *maps*; maps that contains *robots* and *landmarks*; robots that have *sensors*; sensors that make *observations* of landmarks. Each of these objects are abstract and can have different implementations. They can also contain other objects that may themselves be generic. These objects are shown in Fig. 2.2.

Taking into account all the concepts presented before we present in Algorithm1 our approach to Bi-Camera SLAM; an efficient way to the visual localization and mapping problem.

3 Multi-Map Bi-Camera SLAM

The Extended Kalman Filter (EKF) is the implementation with the longest tradition to solve the SLAM estimation problem, but it has recently been challenged by global optimization methods that have shown superior precision for large scale SLAM. However EKF still has the advantage of (relative) simplicity implementation and faster processing for problems of limited size [Strasdat *et al.* 2010], and can be combined with global optimization methods [Estrada *et al.* 2005] where large maps are usually being built by means of small submaps, to take the best of both worlds, thus circumventing most of the EKF drawbacks: computational burden and filter consistency [Castellanos *et al.* 2004]. In this way, different scalable SLAM approaches to build multiple local maps have been proposed, mainly to reduce computational complexity and to delay linearisation er-

[1] RT-SLAM is a fast Slam library and test framework based on EKF. This is a collaborative project developed at LAAS-CNRS. See Appendix B for more detail.

Algorithm 1: Bi-Camera SLAM Algorithm

```
Data: Raw = (robot, sensor) with sensor ∈ robot
Result: Stochastic Map = (robots, sensors, landmarks)
begin
    for robot ⟵ 1 to N do
        // Process sensor observations
        for sensor ⟵ camera1 to cameraN ∈ robot do
            img ⟵ GetRawData(sensor)
            // observe and correct known landmarks
            foreach lmk ⟵ CorrectKnownLmk(img) do
                // update robot and sensor from Map
                robot ⟵ MapToRob(robot)
                sensor ⟵ MapToSen(sensor)
                // project all landmarks
                obs ⟵ ProjectLmk(robot, sensor, lmk)
                // select landmarks to correct
                foreach lmkToCorr ⟵ SelLmkToCorrect(obs)
                do
                    // try to match feature
                    obs ⟵ MatchFeature(sensor, obs, lmk)
                    if MatchFeature then
                        // compute innovations
                        robot ⟵ MapToRob(robot)
                        sensor ⟵ MapToSen(sensor)
                        obs ⟵ ObsInnovation(obs)
                        // check consistence
                        if ObsInnovation < threshold then
                            // landmark correction
                            CorrectLmk(robot, senor, obs, lmk)
            // Initialize new landmarks
            // update robot and sensor from Map
            robot ⟵ MapToRob(robot)
            sensor ⟵ MapToSen(sensor)
            // get index to first free IDP landmark
            existingProj ⟵ GetExistingProj(obs)
            if feat ⟵ DetectFeat(img, existingProj) then
                // feature detected - initialize it in IDP
                obs ⟵ GetNewObservation(feat)
                // retro-project feature onto 3D space
                retroProjLmk(robot, sensor, obs)
                // get new landmark covariance
                P ⟵ getNewLmkCov(robot, sensor, obs)
                // add landmark to Map
                lmk ⟵ AddToMap(P)
```

rors until the map merging [Tardós *et al.* 2002, Estrada *et al.* 2005, Paz *et al.* 2007, Blanco *et al.* 2009, Cadena & Neira 2010]. When the maps are merged into a single one, on the basis of either common landmarks between local maps, or simply the sequential constraint, a fully correlated map of the environment is obtained. Successful fast implementations exploiting the topology of the representation to systematically join and fuse local maps have been proposed, such as trees [Paz *et al.* 2007], or binary trees [Frese 2006].

Our application involves motions of robots in large environments, so the mapping problem is formulated using two steps:

1. Using multi-maps in a similar way to hierarchical SLAM in [Estrada *et al.* 2005] or hybrid metric-topological SLAM in [Blanco *et al.* 2009], where there are two levels: local level (sub-maps), and global level (adjacency graph);

2. A refinement of the global map is performed based on the local submap joining problem in a similar manner that [Gibson *et al.* 2009]. Thus, the proposed system is similar to CF SLAM [Cadena & Neira 2010], and can be seen as an optimal combination of Kalman and Information filter to perform highly efficient localization and mapping in large environments.

The general principle of our multi-map representation is shown in Fig. 2.3. Independent consecutive local maps are represented in their own reference frame *(lrf)*, and the upper global level *(wrf)* is a graph whose nodes correspond to local maps, and whose edges are annotate by relative transformations between *lrf*s.

3.1 System Overview

Our multi-map representation: local level and global level, is based on one hand, on Hierarchical SLAM described in depth in [Estrada *et al.* 2005], and on the other hand, on the hierarchical mapping approach described in [Vidal-Calleja *et al.* 2011]. For the sake of completeness we summa-

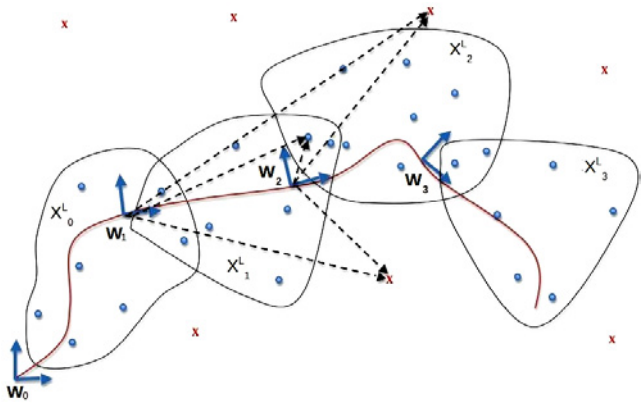


Figure 2.3: Multi-map representation; the map: a set of submaps with their own *lrf*s, and with common landmarks, known either by euclidean coordinates for close points (blue dots) or by IDP vectors generally for points at infinity (red X).

rize the important ideas and algorithmic steps but refer the readers to [Estrada *et al.* 2005] and [Vidal-Calleja *et al.* 2011] for the details.

3.1.1 Local Level

The local level contains the locally referred stochastic maps of landmarks, built with the Bi-Camera SLAM algorithm. Here a landmark is a 3D point, observed as an interest point in images. Each point is linked to an image patch which is defined as a small image region (traditionally 11 x 11 pixels).

The k-th local map is defined by

$$\mathbf{x}_k^L = \begin{bmatrix} \mathbf{x}_k^h \\ \mathbf{g}_k \end{bmatrix}, \tag{2.4}$$

where superscript "L" stands for local map, $\mathbf{x}_k^h$ is the current pose of the robot at instant h, and $\mathbf{g}_k = [\mathbf{l}_1^k \cdots \mathbf{l}_p^k]^\top$ is the set of p mapped landmarks, both with respect to the k-th *lrf*.

As shown in Section 2, cameras and robot positions are linked by a known rigid transformation, so that Bi-Camera SLAM keeps a Gaussian

estimate $\mathbf{x}_k^L \sim \mathcal{N}\{\hat{\mathbf{x}}_k^L, \mathbf{P}_k^L\}$ of this map, namely

$$\hat{\mathbf{x}}_k^L = \begin{bmatrix} \hat{\mathbf{x}}_k \\ \hat{\mathbf{g}}_k \end{bmatrix}, \quad \mathbf{P}_k^L = \begin{bmatrix} \mathbf{P}_{\mathbf{x}_k\mathbf{x}_k} & \mathbf{P}_{\mathbf{x}_k\mathbf{g}_k} \\ (\mathbf{P}_{\mathbf{x}_k\mathbf{g}_k})^\top & \mathbf{P}_{\mathbf{g}_k\mathbf{g}_k} \end{bmatrix} \tag{2.5}$$

Maps are built sequentially. Once a threshold is reached, either in number of landmarks or in robot uncertainty, the current map $\mathbf{x}_k^L$ is closed and a new map $\mathbf{x}_{k+1}^L$ is created, starting in a new *lrf* with robot pose $\mathbf{x}_{k+1}^{h+1}$ and error covariance equal to zero. Each landmark known in $\mathbf{x}_k^L$ reobserved when the new map $\mathbf{x}_{k+1}^L$ is created, keeps the same label.

3.1.2 Global Level.

The global level is represented as an adjacency graph in which local maps $\mathbf{x}_k^L$ in *wrf* are nodes, and the edges between them are annotated first by the relative transformations between successive *lrf*s, noted $\mathbf{w}_k^{k+1}$, and secondly, by the list of labels of common landmarks, between $\mathbf{x}_k^L$ and $\mathbf{x}_{k+1}^L$. Let us define the global level as the Gaussian state $\mathbf{w} \sim \mathcal{N}\{\hat{\mathbf{w}}; \mathbf{P}_w\}$ of relative transformations between local maps, namely:

$$\hat{\mathbf{w}} = \begin{bmatrix} \hat{\mathbf{w}}_0^1 \\ \vdots \\ \hat{\mathbf{w}}_k^{k+1} \end{bmatrix}, \quad \mathbf{P}_w = \begin{bmatrix} \mathbf{P}_{w_0^1} & 0 & 0 \\ 0 & \cdot & 0 \\ 0 & 0 & \mathbf{P}_{w_k^{k+1}} \end{bmatrix} \tag{2.6}$$

Let us note $\mathbf{W}_k$ the origin of the local map $\mathbf{x}_k^L$, expressed in the *wrf*; it can be computed by compounding relative transformations [2] from $\mathbf{w}_0^1$ to $\mathbf{w}_{k-1}^k$,

$$\mathbf{W}_{k+1} = \mathbf{W}_k \oplus \mathbf{w}_k^{k+1} \tag{2.7}$$

The current position of the robot in *wrf* can be computed as

$$\mathbf{X}_h = \mathbf{W}_k \oplus \mathbf{x}_k^h \tag{2.8}$$

[2] The compounding $\oplus$ and $\ominus$ operations are fully described in [Smith & Cheeseman 1987]

We can see that relative transformation between local maps can be considered as past robot poses, thus, the global level can be viewed as a sparse pose-SLAM as in [Eustice *et al.* 2006] or [Ila *et al.* 2011], where local maps are like landmarks hanging from robot poses in *wrf*. The main difference is that the global state in our case contains transformations $\mathbf{w}_k^{k+1}$, instead of absolute poses $\mathbf{W}_k$. Note that local maps are obtained sequentially, hence the relative transformation between the local maps is given by the last robot pose in the current *lrf*, $\mathbf{w}_k^{k+1} = \mathbf{x}_k$.

4 Trajectory Optimization based submap joining

In the case of a path recorded in order to be run later, the final model is memorized as a hierarchical map. For an edge between nodes k and $(k+1)$, a common landmark $\mathbf{l}$ has two representations $\mathbf{l}^{(k)}$ in the *lrf* k, and $\mathbf{l}^{(k+1)}$ in the *lrf* $(k+1)$. This model must be refined or transformed before to be exploited later for navigation.

4.1 Refinement of the global map

In our context, a path follows a route between two different areas, a robot can perform a mission to explore an area without having to close the loop, so that a "classical" Loop Closure is not possible; the global graph has no cycle. Nevertheless, the global map can be refined, by non linear optimisation. Successive submaps can be made consistent, computing optimal and unique representations for all common landmarks and optimizing the transforms between submaps.

Matches established between submaps (or landmarks within submaps) must be searched between submaps that are likely to match, which can either be selected on the basis of their position estimate, which implies that both local maps share the same *wrf*, or by applying loop closure detection approaches, e.g. using image indexing techniques like in [Cummins & Newman 2008]. The "map to map" kind of data association requires both local maps to be transformed to a common

frame, e.g. promoted to the global level. In other words, the matching process happens in a common frame, being the *wrf* or one of the *lrf*.

Once submaps or landmark are matched, an estimation method yields the relative transformation between the submaps *lrf* with the associated covariance, allowing us a refinement of the global map and an optimization of the trajectory.

Thus, we formulate the refinement of the global map as a local submap joining problem in a similar manner that in [Gibson *et al.* 2009]. The algorithm is generalized and formulated as a least squares optimization problem and solved by Extended Information Filter (EIF) together with smoothing and iterations.

From Section 3.1.1, we have that a local map is defined by

$$(\hat{\mathbf{x}}^L, P^L) \tag{2.9}$$

where $\hat{\mathbf{x}}^L$ is an estimate of the state vector and P^L is the associated covariance matrix. Since the local map provides a consistent estimate of the relative position from robot poses to local features, this map can be treated as an observation made from the robot start pose to all the features in the local map and a virtual robot located at the robot end pose. The observation value is the local map state estimate and the observation noise is a zero-mean Gaussian *pdf* with the covariance matrix equal to the local map covariance matrix. Thus, the refinement of the global map becomes a large-scale estimation problem with only local maps information.

Let us note h the function that computes the landmark representation $\mathbf{g}_i$ in *lrf* i (euclidean coordinates or IDP), from its representation in *lrf* j, knowing the transform $\mathbf{w}_i^j$,

$$\mathbf{g}_{i_j} = h(\mathbf{g}_j, \mathbf{w}_i^j) = \mathbf{g}_j \oplus \mathbf{w}_i^j \tag{2.10}$$

So, the transform between two successive *lrf* can be formulated as a least square problem

$$\hat{\mathbf{w}}_i^{i+1} = min_{\hat{\mathbf{w}}_i^{i+1}} \left(\sum_{l=1}^{L} |\hat{\mathbf{g}}_i^l - h(\mathbf{g}_{i+1}^l, \mathbf{w}_i^{i+1}) \right) \tag{2.11}$$

where L is the number of common landmarks.

The Extended Information Filter (EIF) is used to solve the estimation problem. A non zero off-diagonal element in the information matrix (a link between the two related objects) occurs only when the two objects are within the same local map. Since the size of each local map is limited, any object will only have links with its nearby objects no matter how many (overlapping) local maps are fused (Fig. 2.3). This results in an exactly sparse information matrix.

5 From local maps to convoy navigation

A multi-map approach in our system for the learning step, is very appropriate: each robot manages a set of submaps and a global graph between them. But the interest of multi-robots localization and mapping in a convoy configuration arise of course when the robots exchange mapping or position information. Approaches described here are focused primarily on non-isotopic communications, i.e. robots teams where there is an explicit "leader" that performs navigation information, and one or more "followers" that must act accordingly. This communication is affected by the communication channel bandwidth and by mechanisms used to communicate between the elements of the convoy.

The main advantage to exploit a multi-map structure for communicating vehicles, is the low communication bandwidth required among the robots; only submaps need to be communicated from the leader to the first follower, and then between consecutive vehicles of the convoy.

Let us note V_i as the data rate

$$V_i = \frac{1}{\tau} \log_2 m = V_b \log_2 m \quad \text{bps} \tag{2.12}$$

where τ is the duration of the bit or bit time, $V_b = \frac{1}{\tau}$ is the modulation rate expressed in baud and m is the number of symbols per transmission interval. In a binary system $m = 2$, so equation 2.12 is expressed as,

$$V_i = Vb \tag{2.13}$$

On the other hand, we can express the bandwidth of a digital communication channel as

$$B \geq \frac{1}{\tau} \tag{2.14}$$

Thus, from equations 2.12 and 2.14, we have

$$V_b = B \tag{2.15}$$

In this manner, equation 2.15, can be interpreted as follows:

Data with modulation rate equal to V_b, can be transmitted without loss of information by a digital communication channel with a minimum bandwidth B numerically equal to Vb.

It is clear that if $V_b < B$, where B is the actual bandwidth of the channel, there will be no problems in information retrieval. However, if $Vb > B$, information will be lost and would have to find other means to prevent that loss. In general, it must be verified that $Vb \geq B$ to avoid losing information.

6 Experiments

In this section we present preliminary results with the aim of validate the concepts presented before. The experiments presented are not intended to be particular challenging examples, they are simply used to take the reader through the functionality of the system. Further experiments will be presented in Chapter 5.

6.1 Real-data Experiments

For this experiment, we have tested our system in the urban environment surrounding the "Observatoire Midi-Pyrénées" in Toulouse-France, using a ground robot (Segway platform) equipped with a calibrated stereo-vision bench exploited in a Bi-Camera configuration, so the vision system is made of two Marlin 1280×960 cameras with a baseline of $0.40m$ (see Fig. 2.4).

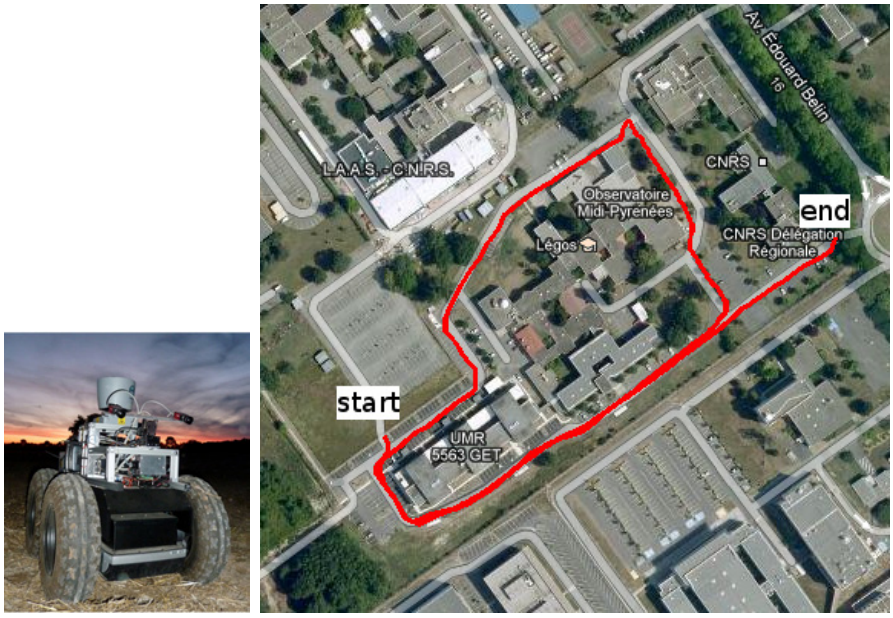


Figure 2.4: The Segway platform and an overhead view of the learned trajectory in the experiment.

The robot follows the trajectory shown in Fig. 2.4. The main issues of this experiment is to learn a safe path, computing the learning-mapping approach. So, learning-mapping procedure was used to map $3D$-points landmarks and build the submaps. New local maps are created when 80 landmarks are in the map. In each submap, the poses and their uncertainties are expressed in the associated *lrf*. All submaps were merged in a final global map, expressed in *wrf* using the refinement of the global map procedure (see Fig. 2.5).

As is shown in Fig. 2.4 and Fig. 2.5, the results obtained in this experiment, confirm the system performance in the learning-mapping

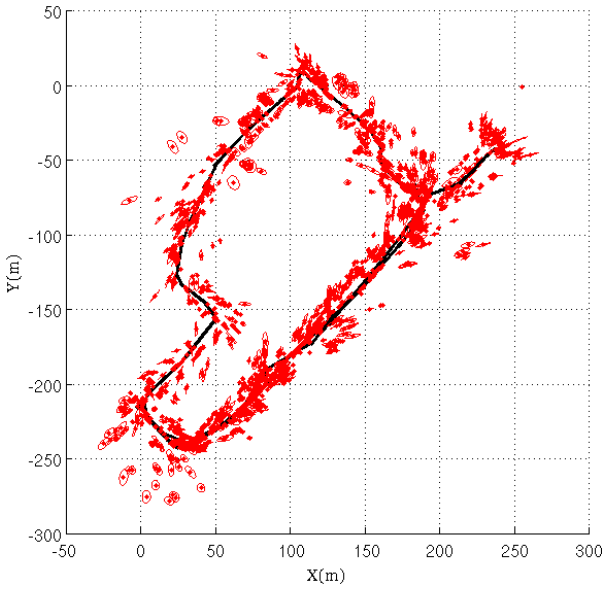


Figure 2.5: Global map obtained by joining the local submaps using our hierarchical/hybrid BiCam SLAM approach with refinement of the global map procedure; red: covariance ellipses of the features; black: the estimated robot trajectory.

procedure. A representative video of the experimental results can be seen at `http://homepages.laas.fr/dmarquez/bicam`

7 Conclusion

In this chapter, we have presented a complete learning-mapping procedure in order to generate the necessary information to implement convoy navigation by robots that must navigate in dangerous unknown routes. Only perception results have been exhibited, i.e. a 3D map from visual information and the estimated path are recorded and stored in a database to be exploited later by a second robot.

The genericity and performance of the algorithm make it both a useful experimentation tool and efficient solution for robot localization and mapping.

It has been proven that Bi-Camera SLAM converges faster than a classical monocular SLAM. So our Multi-Map Bi-Camera SLAM im-

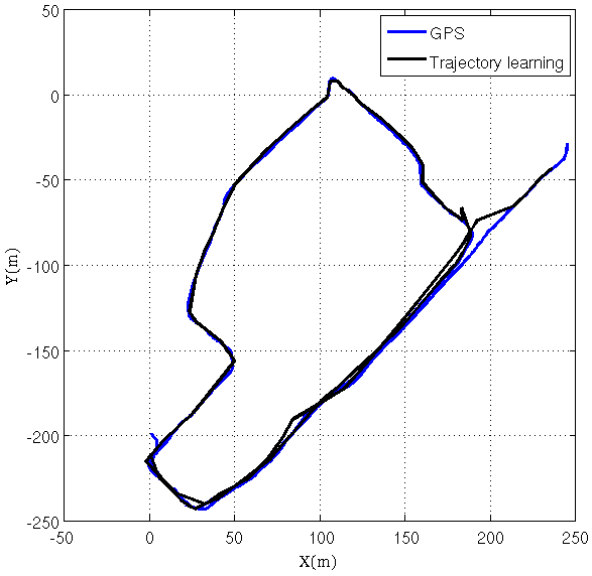


Figure 2.6: Robot trajectory estimate; black: the robot trajectory learned path; blue: the true robot trajectory (GPS).

proves landmarks observability and copes with large scale trajectories allowing the system to multi-robot collaborative localization and mapping.

Visual trajectory replay and map update

3

A pile of rocks ceases to be a rock when somebody contemplates it with the idea of a cathedral in mind.

Antoine de Saint-Exupery

Contents

1 Introduction 45

2 Replaying a learned trajectory 46

2.1 Patch search and data association 46

2.2 Matching points from 3D landmarks projections 47

2.3 Replacing points and update of the map . . . 48

3 Localization from the learned map 49

3.1 Initial position 49

3.2 On-line Localization 50

4 Path tracker: trajectory following 54

4.1 Trajectory generation: from robot position to reference path 56

5 Visual trajectory learning and replay: system overview 56

6 Experiments 57

6.1 Replaying the learnt trajectory 58
6.2 Tracking the reference path 60
7 Conclusion 61

1 Introduction

This chapter will present a description of our replay-localization approach. The outline is shown in Fig. 3.1. In this phase, the information generated in the learning step described in Chapter 2 the information is then loaded on the robot and used for localization during the autonomous replay of the learned trajectory

The use of a series of small local maps, is of vital importance in our approach because in addition to reducing the computational complexity allows us to implement a "convoy" of communicating vehicles able to accurately follow a path created by a leader vehicle.

Similar systems for teach-and-repeat have been proposed before, [Goedeme *et al.* 2007, P. Newman & Neira 2002] for indoor robot navigation, [Marshall *et al.* 2008, Royer *et al.* 2007, Segvic *et al.* , Zhang & Kleeman 2009, Furgale & Barfoot 2010] for outdoor navigation, however our system, in addition to learning-and-replay is able to run on multiple-robots in a "online-convoy" configuration. This application is appropriate to many scenarios requiring robot autonomy in which GPS is not available.

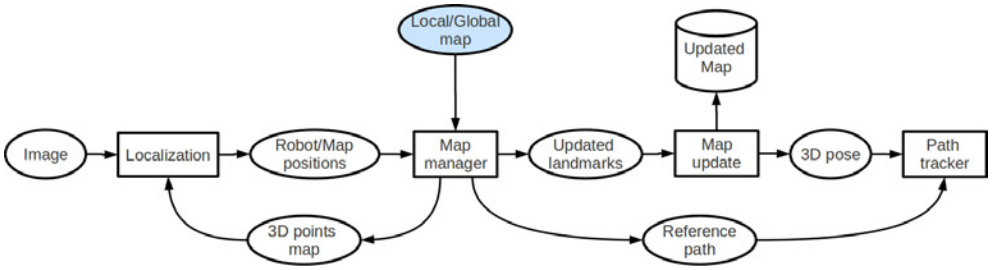


Figure 3.1: An overview of the replay-localization approach in our proposed system.

The following chapter is organized as follow: Section 2 describes how to use the 3D map information from the learning step to replay a learned trajectory. In Section 3 the localization method used in our approach is presented. Section4 describes the path tracker method implemented in our system. Finally, Section 6, gives some experimental results to validate the proposed system.

2 Replaying a learned trajectory

Using the optimized map database (local maps or global map) produced by the leader with the learning procedure described in the chapter 2, the followers robots are able to repeat a learned path any number of times.

In this way, two modes are considered:

1. Learning and replay : in the learning and replay mode the trajectory is globally stored in a database by the leader, and executed again long time after by another vehicle;

2. Convoy : in the convoy mode, submaps are successively transmitted from the leader to the followers. Each follower reexecutes the path attached to each submap, and possibly refines the landmark positions, overall if it can observe the previous vehicle of the convoy.

During the execution, every 3D-point landmark of the map is sent to a modified "*patch search*" procedure (as is explained in Section 2.1, where each image patch (i.e. small image region) associated to the 3D-point supposed to be observed is searched within an elliptical region in order to locate itself and verify that the environment has not been modified between the learning and the replay steps . If a landmark is not matched a number of times, then, it is removed from the map and replaced by another one initialized in the same image region.

2.1 Patch search and data association

The strategy currently implemented to deal with observations is an astute combination of active search [Davison 2005] and outliers rejection using one-point RANSAC [Civera *et al.* 2009].

The goal of active search is to minimize the quantity of raw data processing by constraining the search in the area where the landmarks are likely to be found. Observations outside of this 3σ observation uncertainty ellipse would be anyway considered incompatible with the filter and ignored by the gating process. In addition active search gives the possibility to decide anytime to stop matching and updating landmarks

with the current available data, thus enabling hard real-time constraints. We extended the active search strategy to projections of 3D-points from a known map of landmarks: each sensor strives to maintain projections in its whole field of view using a grid of fixed size, so feature projections is limited to empty cells of the grid. Furthermore the good repartition of features in the field of view ensures a better observability and tracking of points.

Outliers can come from matching errors in raw data or changes in the environment. Gating is not always discriminative enough to eliminate them, particularly right after the prediction step when the observation uncertainty ellipses can be quite large. To prevent faulty observations, outliers are rejected using a one-point RANSAC process. It is a modification of RANSAC, that uses the Kalman filter to obtain a whole model with less points than otherwise needed, and provides a set of strongly compatible observations that are then readily corrected. Contrary to [Civera *et al.* 2009] where data association is assumed given when applying the algorithm, we do the data association along with the one-point RANSAC process: this allows to look for features in the very small strongly compatible area rather than the whole observation uncertainty ellipse, and to save additional time for raw data processing.

2.2 Matching points from 3D landmarks projections

We implement an optimized point matching based on Zero-mean Normalized Cross Correlation (ZNCC). Integral images [Viola & Jones 2001] are used to compute means and variances, and a hierarchical search is made (two searches at half and full resolution are sufficient). We also implemented bounded partial correlation [Di Stefano *et al.* 2005] in order to interrupt the correlation score computation when there is no more hope to obtain a better score than the threshold or the best one up to now. To be robust to viewpoint changes and to track landmarks longer, tracking is made by comparing the initial appearance of the landmark with its current predicted appearance in a similar manner that in [Davison *et al.* 2007].



Figure 3.2: Two screen shots of a sequence in a parking lot, in the morning at 10 am (left) and in the afternoon at 6 pm (right). Note that despite being at the same place in both cases, the observations will be substantially different due to illumination and the changed number of parked cars.

2.3 Replacing points and update of the map

We consider that our robot operates in a dynamic environment: the world is constantly changing and a place can change between the learning and replay step as is shown in Fig. 3.2.

In our approach we are able to update the map that we are using (navigating). Depending on the matching process described before, if a landmark projected from the 3D map is not matched a number N of times, then it is removed from the map and replaced by another one initialized in the same image region. This process of actively replacing points, allows us to keep constantly updated the map, so an updated version of the map will be used for each follower.

In this way, a point extraction procedure based on Harris detector with several optimizations is implemented. Some of them are approximations: a minimal derivative mask $[1, 0, 1]$ is used, as well as a square and constant convolution mask, in order to minimize operations. This allows the use of integral images to efficiently compute the convolutions. Additional optimizations are related to to the fact that only one new feature is detected and initialized in a small region of interest, which eliminates the costly steps of thresholding and sub-maxima suppression.

3 Localization from the learned map

Before a vehicle replays a learnt path, it must be able to localize itself from an initial position close to that the leader initially had. Indeed, it is assumed that the follower vehicle starts from a position close to that of the leader but not necessarily identical. As mentioned above, in addition to the points of the path, a 3D points map collected by the leader is transmitted. Each point is linked to an image patch which is defined as a small image region (traditionally close to 11x11 pixels). Then the follower must compare the map information with its own perception of the environment and compute its localization.

In this manner, the first action to be executed by the robot, is to calculate an initial position from the first points of the received map, then from this starting position, the robot update its localization following the reference path.

3.1 Initial position

In our approach, the follower vehicle has to localize itself and navigate thanks to a $3D$ map with accuracy and in real time. The initialization step is a critical point because the vehicle location is not accurately known in the map. In order to focus on its real localization, it perceives its environment through its camera. So, each image patch associated to the $3D$ point supposed to be observed is actively searched within an elliptical region. The latter is based on the projection of the uncertainty of the considered point taking into account the vehicle position. A cross-correlation measure based on the matching points procedure presented in Section 2.2 is used. While this search is performed, the best score for each image patch is stored, which provides a first hypothesis of matching. For each pair of $3D - 2D$ points, an update step is calculated through a Kalman filter (details are explained in next Section). Once the update respects integrity, the difference between the estimated projection of each point of the map in the image and the result of the cross-correlation measure decreases. All these different criteria allow the selection of a

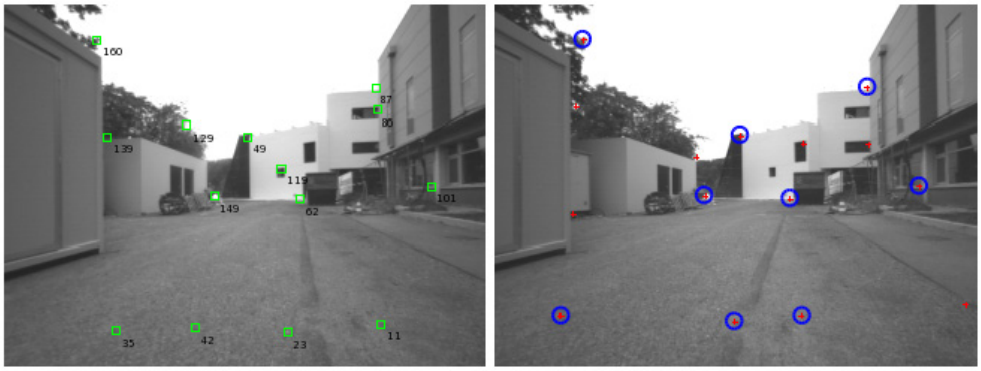


Figure 3.3: Results of initial position from two cameras mounted on an on board vision bench. (left) left camera, projections of 3D points. (right) right camera, result of research by cros-correlation of image patches.

first $2D - 3D$ couple. Then, it is possible to iterate the same procedure to refine the position through the remaining points.

To illustrate the initial position method, Fig. 3.3 presents results obtained from two cameras mounted on an on board vision bench. Fig. 3.3(left) represents the image captured by the left camera in which the image patches are acquired. Green rectangles with his respective label, represent the projections of 3D points. In this case we have 14 points projected. Fig. 3.3(right) represents the image captured by the right camera. The result of research by cross-correlation of image patches is displayed. Blue circles with red crosses represent a correct matched.We can observe, that after the first iteration 9 points of 14 have found their counterpart. We have found a set of pairs 3D-2D points that allows us to continue with an online localization.

3.2 On-line Localization

At this stage a good estimate of the starting point is available. So from this initial position we want to achieve real time localization along the path. Based in the previous work of [Féraud 2011] in the RINAVEC project context, our approach exploits a Kalman filter to estimate the position with proprioceptive sensors and refine it by using observations collected during the motion. The main difference of our approach and

contrary to [Féraud 2011] where all the localization process is based in monocular information from one camera, is that we take advantage of the Bi-camera configuration to allow better observations of the projected landmarks.

3.2.1 Prediction step

The first step of the Kalman filter consists in the prediction through proprioceptive data. It establishes a model of evolution of the vehicle. For practical purposes, we use the well known non-holonomic equations for a rear wheel driven tricycle. This model, so called Ackermann-steering, means that no lateral forces occur at the front wheels during driving, thus, a "virtual wheel" is introduced, comprising the effective steering angles of each front wheel and simplifies the real car to a tricycle model.

$$X_{k+1|k} = \mathbf{f}\left(X_{k|k}, u_{k-1}\right) \tag{3.1}$$

where $X_{k+1|k}$ is the state vector and u_k is the control vector. Thus, the equation model is

$$X_{k+1} = X_k + ds\cos(\psi_k + \alpha/2) + \eta_X \tag{3.2}$$

$$Y_{k+1} = Y_k + ds\sin(\psi_k + \alpha/2) + \eta_Y \tag{3.3}$$

$$Z_{k+1} = Z_k + ds\sin(\theta) + \eta_Z \tag{3.4}$$

$$\psi_{k+1} = \psi_k + \eta_\alpha \tag{3.5}$$

$$\theta_{k+1} = \psi_k + \eta_\beta \tag{3.6}$$

$$\varphi_{k+1} = \psi_k + \eta_\gamma \tag{3.7}$$

where ψ, θ, φ are respectively the *yaw*, *pitch* and *roll* angles. The length ds is a function of a priori traveled distance and parameters which are

specific to the vehicle. It is supplied by odometry measurements. The angle α reflects the constant steering angle between times k and $k+1$ and the parameters of the vehicle. The uncertainty in this estimate is provided by the Jacobians of the evolution model.

The variance-covariance matrix is then expressed as,

$$\mathbf{P}_{k+1|k} = \mathbf{F}_X \mathbf{P}_{k|k} \mathbf{F}_X^\top + \mathbf{Q}_{k+1} \tag{3.8}$$

where $\mathbf{Q}$ is the covariance of process noise and $\mathbf{F}_X$ is the Jacobian of the evolution model $\mathbf{f}\left(X_{k|k}, u_{k-1}\right)$, with respect to X.

3.2.2 Data association

The search for correspondence is limited by the projection of the uncertainty associated with the movement of the vehicle and the uncertainty of the position of the $3D$ point. Thus, the method presented in Sections 2.1 and 2.2, reject absurd tracking results and find the correct matchings.

3.2.3 Estimation step

This prediction is then refined with the 2D observations of the $3D$-world points. The association provides information on data matching between the points $(u_{obs}, \nu_{obs})^T$ of the 2D image and $3D$ points P_{3D} of the map.

Let us define the rigid transformation between the world and the reference frame of the vehicle as,

$$\mathbf{R} = R_z(\psi_k) R_y(\theta_k) R_x(\varphi_k) \tag{3.9}$$

and T is defined as,

$$T = (X_k, Y_k, Z_k)^\top \tag{3.10}$$

The $3D$ points of the map are projected in the image with the linear relationship,

$$P_{2D} = \mathbf{K}\mathbf{R}^\top (P_{3D} - T) = (ku, k\nu, k)^\top \tag{3.11}$$

where $\mathbf{K}$ is the intrinsic parameter matrix of the camera.

The estimated coordinates are divided by the scale factor, $u_{est} = ku/k$ and $\nu_{est} = k\nu/k$

$$u_{est} = h_u(P_{3D}) = \frac{K_1\mathbf{R}^\top(P_{3D} - T)}{K_3\mathbf{R}^\top(P_{3D} - T)} \tag{3.12}$$

$$\nu_{est} = h_\nu(P_{3D}) = \frac{K_2\mathbf{R}^\top(P_{3D} - T)}{K_3\mathbf{R}^\top(P_{3D} - T)} \tag{3.13}$$

where $\mathbf{K}_i$ is the i-th line from the intrinsic parameter matrix and h_u, h_ν are the part of the observation function related to the u and ν coordinates.

The covariance of the innovation of the Kalman filter ($\mathbf{H}_X$), is obtained using the Jacobians of the observation model. The Jacobians are calculated, using equation (3.12) and (3.13).

The Kalman gain associated with a pair can be calculated by,

$$\mathbf{G}_{k+1} = \mathbf{P}_{k+1|k}{\mathbf{H}_X}^\top(\mathbf{H}_X\mathbf{P}_{k+1|k}{\mathbf{H}_X}^\top + \mathbf{R}_{obs})^{-1} \tag{3.14}$$

where $\mathbf{R}_{obs}$ denotes the covariance of the noise associated with the observation, in pixels.

Finally, the data is updated from observation by:

$$X_{k+1|k+1} = X_{k+1|k} + \mathbf{G}_{k+1}\left(\begin{pmatrix} u_{obs} \\ \nu_{obs} \end{pmatrix} - \begin{pmatrix} u_{est} \\ \nu_{est} \end{pmatrix}\right) \tag{3.15}$$

$$\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - \mathbf{G}_{k+1}\mathbf{H}_X\mathbf{P}_{k+1|k} \tag{3.16}$$

This step is for each pair of $3D - 2D$ points. Localization accuracy depends on the number of points, the accuracy of the initial positioning and of the learnt map.

4 Path tracker: trajectory following

Let us consider D a robotic system operating in a frame $W \in \mathbb{R}^3$, its dynamics can be described by,

$$\dot{g} = f(g, \tilde{u}) \tag{3.17}$$

where $g \in \mathbf{G}$ is the state of D, $\dot{g}$ its time derivative and $u \in \mathbf{U}$ a control. $\mathbf{G}$ and $\mathbf{U}$ respectively denote the state space and the control space of D.

Let $\tilde{u} : [0, tf] \rightarrow \mathbf{U}$ denote a control trajectory, ie a time-sequence of controls. Starting from an initial state g_0 (at time 0) and under the action of a control trajectory $\tilde{u}$, the state of D at time t is denoted by,

$$\tilde{u}(g_0, t) \tag{3.18}$$

Thus, a couple

$$\tilde{g} = (g_0, \tilde{u}) \tag{3.19}$$

defines a state trajectory for D, i.e. a curve in $\mathbf{G} \times \mathbf{T}$, where $\mathbf{T}$ denotes the dimension.

Trajectory tracking (see [De Luca *et al.* 1998, Yang *et al.* 1998, Aguiar & Hespanha 2007]) is the problem of reaching and following a trajectory of the state-time space $\mathbf{G} \times \mathbf{T}$, (i.e. a geometric path with an associated timing law) starting from a given initial configuration. It is aimed to compensate for the possible derivation from an initial trajectory to follow due to sensor and actuator errors. It consists then in trying to determine at each time instant the input control to apply on the robotic system, given the current robots state perception.

Path tracker module have been developed in a similar manner to *Tiji* [Delsart & Fraichard 2010], to integrate the replay step algorithm into our proposed system. Fig. 3.4, shows the path tracker processing block.

Path tracker is an algorithm that computes a feasible trajectory between a start and a goal state. The method proposed, which relies upon a parametric trajectory representation, is variational in nature. The tra-

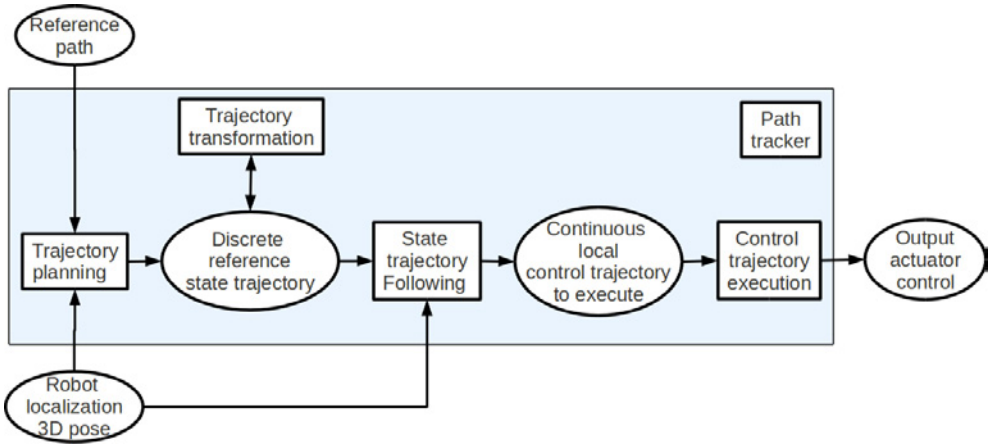


Figure 3.4: An overview of the path tracker processing block in our proposed system.

jectory parameters are incrementally updated in order to optimize a cost function involving the distance between the end of the trajectory computed and the desired goal. Should the goal state be unreachable (eg. if the final time is ill-chosen), path tracker returns a trajectory that ends as close as possible to the desired goal. One of the major advantages over other trajectory generation schemes is that the implemented path tracker is able to compute a trajectory that reaches the goal state at a specified time instant or within a given time interval. This feature is useful in order to compute a trajectory that avoids collision with moving objects (see [Delsart & Fraichard 2010]).

Thus, the implemented approach, which relies upon a parametric trajectory representation, is variational in nature. The trajectory parameters are incrementally updated in order to optimize a cost function involving the distance between the end of the trajectory computed and the desired goal. Should the goal state be unreachable (eg. if the final time is ill-chosen), thus the *path tracker* returns a trajectory that ends as close as possible to the desired goal.

4.1 Trajectory generation: from robot position to reference path

At this point, we are able to obtain the robot localization at instant t, applying the proposed approach presented in Section 3. On the other hand, from the information transmitted by the leader, a map of 3D-points together with a reference path is available to be exploited. The reference path is a set of robot-positions stored during the learning process.

Let us note two states, g_0 and g_r, corresponding to robot position (from localization) and reference position (from reference path) respectively. Trajectory generation consists in finding the control trajectory $\tilde{u}$ to apply from state g_0 in order to reach the goal state g_r.

Thus, this problem can be stated as a constraint optimization process expressed as follows,

$$\begin{aligned} &\text{minimize:} && \mathbf{J}(g_0, \tilde{u}, t_f) = \textstyle\sum_{k=1}^{n} \lambda^k (g_r^k - g_0^k)^2 \\ &\text{subject to:} && \mathbf{h}(g_0, \tilde{u}) \leq 0 \\ & && tmin \leq t_f \leq tmax \end{aligned} \tag{3.20}$$

where g_r^k, represents the k-th feature of the reference state (reference path) and λ^k its associated weight.$\mathbf{J}(g_0, \tilde{u}, t_f)$ its a cost function that represents a Mahalanobis distance between the current and the reference states. $\mathbf{h}(g_0, \tilde{u})$ is a set of motion constraints i.e. bounds over its control and state parameters.

5 Visual trajectory learning and replay: system overview

Based on the information presented in Chapters 2 and 3, we are able to summarize our proposed approach to learning and replay trajectories based on visual information. The major processing blocks of our system are depicted in Fig. 3.5. The system consists of two steps:

1. **learning-mapping** step, where a leader vehicle, builds several $3D$-points maps and learn a safe path. The initial path learning is based

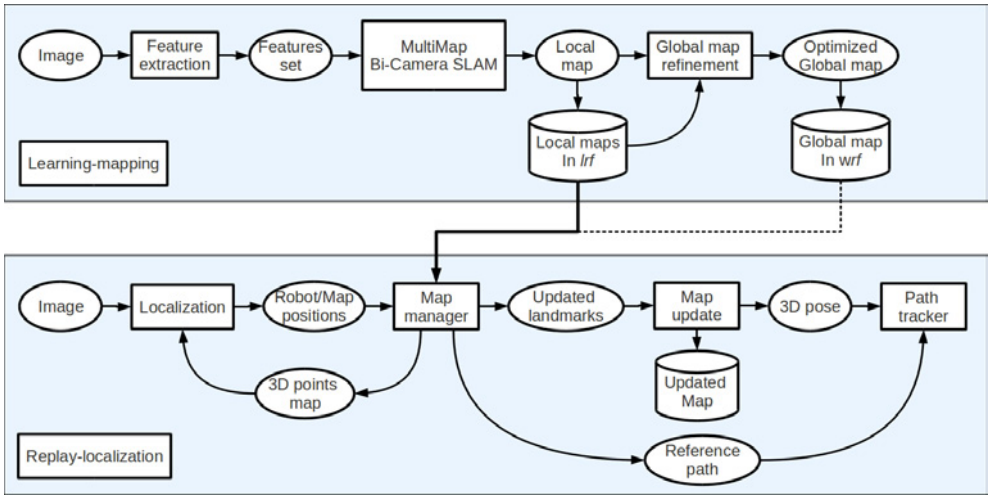


Figure 3.5: Visual trajectory learning and replay system.

on a proposed Multi-map SLAM using only vision in a Bi-Camera configuration (as explained in Chapter 2).

2. **replay-localization** step, where the information learned and transmitted by the leader, allows other vehicles of the fleet, to follow the learned safe path. The replay procedure exploits a localization method based on an active search procedure to localize and control a vehicle so that it is maintained on the same path than during the learning step (as explained in Chapter 3).

6 Experiments

In this section we present preliminary results with the aim of validate the concepts presented before. The experiments presented are not intended to be particular challenging examples, they are simply used to take the reader through the functionality of the system. Further experiments will be presented in Chapter 5.

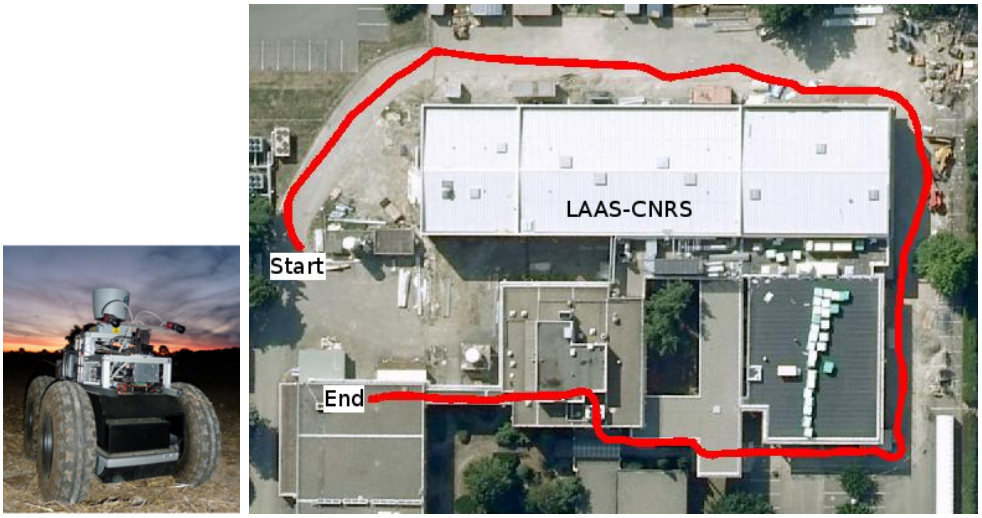


Figure 3.6: The Segway platform and an overhead view of the learned and replayed trajectory in the experiment..

6.1 Replaying the learnt trajectory

For this experiment, we have tested our system in the urban environment surrounding the LAAS-CNRS using a ground robot (Segway platform) equipped with a calibrated stereo-vision bench made of two Marlin 1280× 960 cameras with a baseline of 0.40m (see Fig. 3.6)

The experiment was done with the same robot but with different cameras between the learning (with camera left) and replay (with camera right) steps. For this experiment the robot follows the trajectory shown in Fig. 3.6. Thus, the main issues are, first replay a learned trajectory with our replay-localization approach and second, validate the map update procedure and observe the behaviour of the system in a changing environment. So, we have made the experiment with path learning and replay done on different days, so that the environment has changed between the two steps.

The first step is to obtain a global map that will be exploited in the replay-localization algorithm. The learning-mapping procedure was used to map $3D$-points landmarks and build an optimized global map, expressed in *wrf*, as is shown in Fig. 3.7(left). The 3D global map an the global reference path are loaded on the robot. In this point, we have a good knowledge of the $3D$ map of the environment, thus, the robot starts

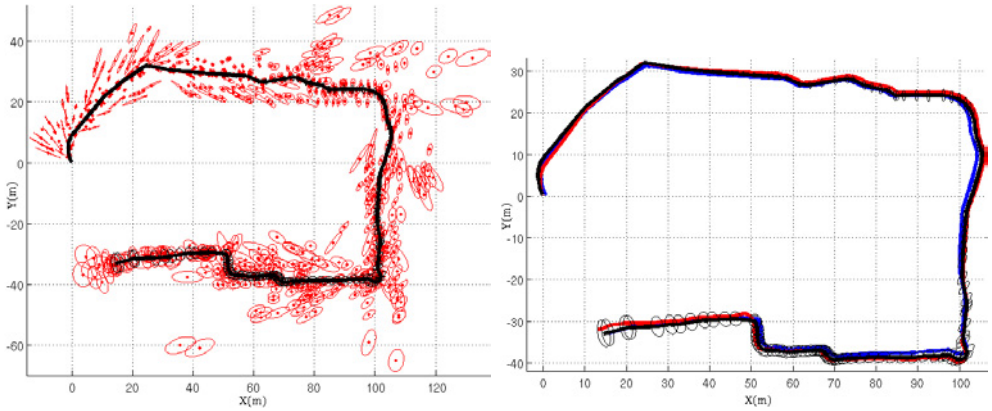


Figure 3.7: LAAS experiment:(left) global map expressed in *wrf*; global map obtained using the learning-mapping approach with *refinement of the global map* procedure; red: covariance ellipses of the features; black: the estimated robot trajectory; (right) robot trajectory estimate; black: the robot trajectory learned path; blue: the robot trajectory replayed path; red: the true robot trajectory (GPS).

with a huge uncertainty and applies the algorithm of replay-localization which replays the trajectory. This algorithm is able to localize independently either from the left or the right cameras.

As is shown in Fig. 3.8, the replay-localization approach, works in spite of bad matchings on occluded landmarks. Fig. 3.8, shows representative images acquired on the same area during the learning step and the replay one (a single camera was used); landmarks are presented as red crosses; green squares show matchings, magenta dots are landmarks used for localization in the replay step, red dots are not matched landmarks, while false matchings due to the environment changes, are pointed by arrows.

The results obtained in this experiment confirm the system performance in the learning-mapping procedure as well as the accuracy of the replay-localization along the path (see Fig. 3.7). A representative video of the experimental results can be seen at `http://homepages.laas.fr/dmarquez/maplaas`

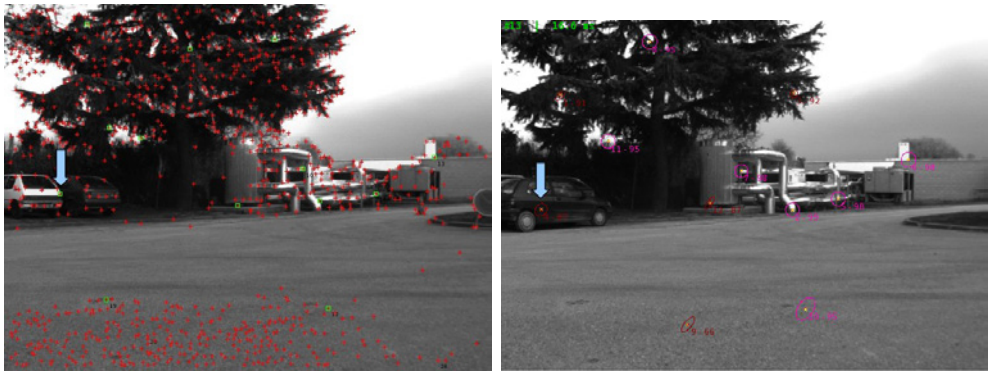


Figure 3.8: An experiment at LAAS: (left) 3D-points maps are built and a safe path is learned using the learning-mapping approach; (right) the learned path is executed again, but environment has changed. Replay-localization works in spite of bad matchings on occluded landmarks.

6.2 Tracking the reference path

A second experiment using MORSE[1] was undertaken with learning and replay done on different time and with different -simulated- robots. Morse is a domain independent simulator, where virtual robots can interact with a 3D environment, using sensors and actuators that behave in the same way as their counterparts in the real world.

For this experiment the robot (follower) must replay a path learned during the learning step. The main issues for the robot are, on the one hand compute the replay-localization to localize itself, and on the other hand, to track and follow (online) the reference path.

The 3D global map an the global reference path are loaded on the robot. Fig. 3.10 shows the experiment where we can see that the initial position of the follower robot it is not the same that the initial position of the leader robot in the learning step. The learned path to follow is presented as a red line. As is shown in Fig. 3.9, the results obtained in this experiment, confirm the system performance in the learning-mapping procedure as well as the accuracy of the replay-localization along the path.

[1] `http://morse.openrobots.org`

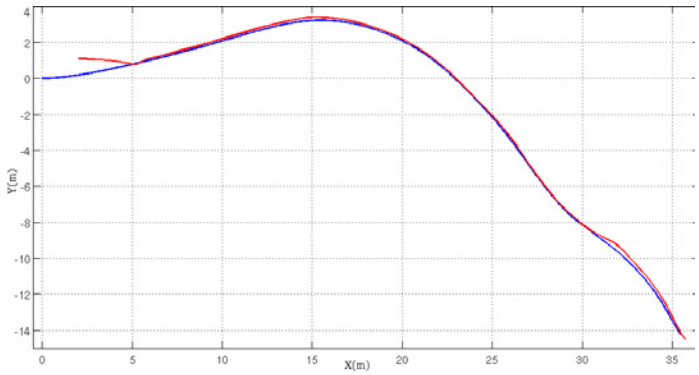


Figure 3.9: MORSE experiment: Robot trajectory estimate; blue: the robot trajectory learned path; red: the robot trajectory replayed path.

7 Conclusion

In this chapter we have presented a complete algorithm for performing replay-localization along a path, using cameras and low cost proprioceptive sensors (odometer, gyro).

The system can be configured to be executed in two ways: (1) learning and replay mode, that takes as input a global 3D optimized map or (2) Convoy mode that takes as input a sequence of small overlapping local maps.

Only perception results have been exhibited, a general method for trajectory control has been used to maintain the follower on the learnt trajectory.

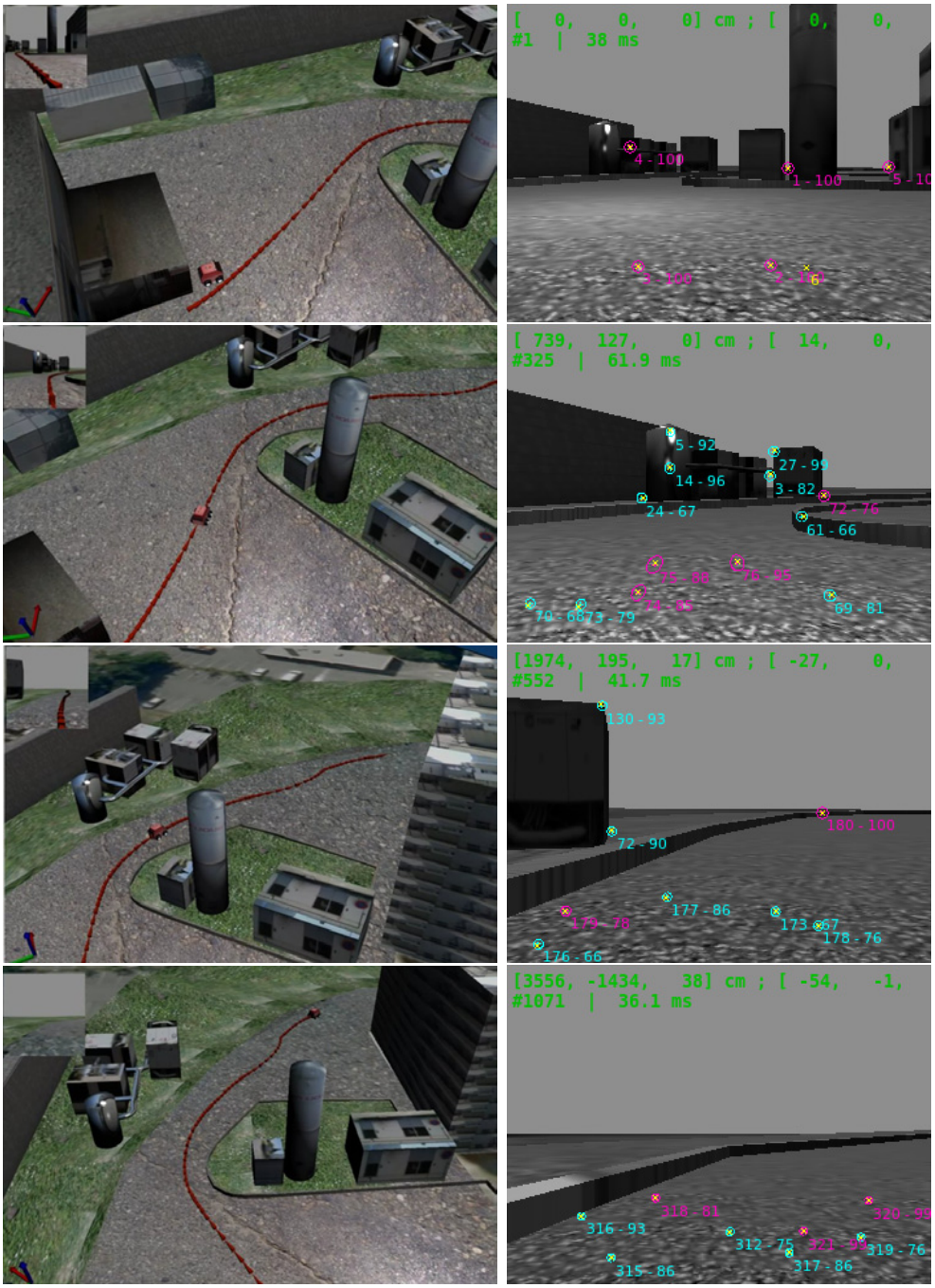


Figure 3.10: MORSE experiment. Experimental results: (left) global view of the experiment scenario; red line is the path to follow; we can see that the initial position of the follower robot it is not the same that the initial position of the path to follow; (rigth) replay-localization step; magenta: initial projections of $3D$ points from the learned $3D$ map; cyan: projected and matched points using for localization.

Visual exploration in dynamic environments

4

The search for truth is in one way hard and in another way easy, for it is evident that no one of us can master it fully, nor miss it wholly. Each one of us adds a little to our knowledge of nature, and from all the facts assembled there arises a certain grandeur.

Aristotle

Contents

1 Introduction 65

2 Related Works 66

3 Moving Objects: Model and Representation 68

3.1 Moving Objects Model 69

3.2 Moving Objects Representation 69

4 System Description 70

4.1 Feature Selection 70

4.2 Feature classification 71

4.3 Motion Grid 72

4.4 Classification approach 76

4.5 Moving Objects detection 81

4.6 Moving Objects tracking 84

5 Experiments 88

6 Conclusion 90

1 Introduction

Moving objects detection and tracking has motivated many works in the robotics community using mainly 3D sensors like laser range finders or radars [Wang 2004] [Vu & Aycard 2009]. In order to make these methods more robust, multi-sensory methods [Gate *et al.* 2009a] have been proposed, combining detection from 3D sensors and obstacle identification from vision, using a model-based method, i.e. the system has a knowledge data base with appearance-based models of objects it could find in the environment, typically pedestrians, bikes or vehicles. The company MobilEye [MobilEye Vision Technologies LTD] has designed an integrated camera using a "Tracking from Detection" approach: on every image pedestrians or vehicles are detected from a model-based classification method.

This Chapter copes with the detection and the tracking of moving objects from a camera embedded on a mobile robot, navigating in an unknown and dynamic environment. The goal is to provide a convenient description of the world, and especially of the moving objects: position, speed and, when possible, type (pedestrian, vehicle, bike...). Indeed in order to build such a model for moving objects, it is required in the same time, to characterize the robot position and speed with respect to static components of the environment.

This last problem is solved by a SLAM algorithm, for Simultaneous Localization and Mapping; thanks to the convergence between Structure From Motion in Vision, and SLAM in Robotics it exists now efficient methods like PTAM [Klein & Murray 2007], iSAM2, [Kaess *et al.* 2008], among others, able of estimating both a vehicle trajectory and the world structure. But all SLAM algorithms assume that the world is static, so that landmark position estimates can be iteratively refined, and can be used as external references for robot localization.

If the robot moves in a dynamic world, the SLAM algorithm must avoid landmarks to be selected on moving objects. This Chapter proposes a complete system to solve the SLAM with Detection and Tracking of Moving Objects (*SLAM+DATMO*) using only vision. Here visual SLAM

based on point landmarks is supposed to be solved, using the EKF-SLAM algorithm developed in Chapter 2.

This Chapter is focused on the Moving Objects Detection (MOD) problem, i.e. how to initially *detect* moving objects or features. Taking into account robustness, safety and real time constraints, one difficult issue arises from the impossibility to exhaustively analyse the whole image surface at every single frame. It is proposed first an active method that exploits knowledge extracted from previous images in order to anticipate where potential moving objects may appear in the image, thus focusing the detecting procedure to those areas. Second a framework is proposed in order to combine several detection algorithms, on the basis of a probabilistic Motion Grid, in which every detection result is recorded. Once the robot position is known and moving features are detected, the Moving Objects Tracking (MOT) problem is solved by filtering techniques.

The following chapter is organised as follows: previous works are detailed in section 2. Initial moving object model definition and representation is presented in section 3. The proposed *SLAM+DATMO* system is described in section 4. Finally experimental results and a brief discussion are presented in section 5

2 Related Works

Simultaneous localization, mapping and moving object tracking (SLAMMOT) involves both simultaneous localization and mapping (SLAM) in dynamic environments and detecting and tracking these dynamic objects.

The first remarkable work on SLAMMOT is due to Wang [Wang 2004]. The system takes 2D laser-based SLAM and incorporates tracking of moving objects. These objects are detected by isolating portions of the current laser scan that do not show sufficiently good match with respect to the current SLAM map. The whole system relies on high speed, long range, laser range finders. The most impressive aspect resides in its ability to perform large area SLAM at high speeds with long loop-closing in urban, dense-traffic conditions.

An alternative, non-SLAM based interesting approach is due to Agrawal et al. [Agrawal *et al.* 2005]. Dense disparity images are produced by means of a calibrated stereo rig. A robust RANSAC method on triplets of matched points between two disparity images is used to determine the main camera-to-scene rigid transformation which is assigned to the camera motion. A homography is computed for each pixel based in this motion. Then the appearance of each pixel is compared to the appearance it should have based on the previous image and the predicted homography. Pixels that show sufficiently large appearance variation are classified as candidates to have suffered independent rigid motion, hence belonging to moving objects. Some blob grouping and spurious rejection is performed and finally a Kalman filter in 3D space is set for each moving object, with a constant-speed model, to obtain trajectory and motion estimation.

Solà [Solà 2007], did an observability analysis in order to isolate moving objects from static ones. Some contextual rules allow to make simpler the detection problem. Tracking was done separately and individually for each moving object in a robocentric representation. In particular, Solà tries to solve the SLAMMOT problem estimating the position of the robot, the static map and tracking of moving objects at the same time, separating the SLAM algorithm from the tracking one using a different EKF filter for each moving object.

In [Wangsiripitak & Murray 2009], a 3D object tracker runs in parallel with the monocular camera SLAM for tracking a predefined moving object. This prevents the visual SLAM framework to incorporating moving features from the moving object. But the proposed approach does not perform moving object detection; so moving features apart from those from the tracked moving object can still corrupt the SLAM estimation. Also, they used a model based tracker, which can only track a previously modeled object with manual initialization.

Migliore et al. [Migliore *et al.* 2009], proposed monocular SLAM and moving object tracking in which moving objects are detected by a simple statistic test and tracked by separated bearing only trackers, the main disadvantage of their system is the inability to obtain an accurate es-

timate of the moving objects in the scene. This is due to the fact that they maintain separate filters for tracking each individual moving feature, without any analysis of the structure of the scene.

Lin et al. [Lin & Wang 2010], recently proposed an augmented state approach to Stereo-based SLAMMOT in which the states of moving objects are augmented into the SLAM state vector, and demonstrated that moving object tracking could improve the SLAM performance. Like in BiCam [Solà *et al.* 2008], the two cameras are treated as two observers, and measurement updates are performed for each instance in EKF.

Gate et al [Gate *et al.* 2009b] proposes a multisensor system based on a laser-based hypothesis generation, and in a vision-based classification approach to make the Object Detection function more robust.

All the presented approaches, propose original methods to compute SLAM with MOT. Indeed, these systems can arguably and mostly be seen as a juxtaposition of a SLAM algorithm and a MOT algorithm that share limited information. There are still important boundaries between *localization and mapping* on the first hand and *detection and tracking* on the other hand, besides, only a small number of possible interactions between SLAM and MOT is exploited.

3 Moving Objects: Model and Representation

In this section, following the concepts developed and presented by Solà in [Solà 2007], we described the initial model definition and representation of moving objects for understanding the proposed *Bi-Camera SLAM-DATMO* approach described in next section. The main difference of our definition of a *moving object* with respect to the definition in [Solà 2007], is that we consider an object as a "fixed structure" defined by a set of moving points. However, the filter side (estimation) and the stochastic map representation is made in a similar manner that in the work of Solà.

3.1 Moving Objects Model

A moving points, $\mathbf{o}_j$, is defined by his position (3D) and linear velocity. So a moving object, $\mathcal{O}_i$, is defined as a set of moving points with similar velocities defining an structure.

The initial representations of *moving points* state is defined by

$$\mathbf{o}_j = \begin{bmatrix} \mathbf{p}^{\mathbf{o}_j} \\ \mathbf{v}^{\mathbf{o}_j} \end{bmatrix} \tag{4.1}$$

where $\mathbf{o}_j$ with $j = \{1, 2, \ldots, n\}$, is the state of *moving point* j, containing their position, $\mathbf{p}_j^{\mathbf{o}_j}$, and linear velocity $\mathbf{v}_j^{\mathbf{o}_j}$, respectively.

The set of moving points ($\mathbf{o}_j$), defines the *moving object* i ($\mathcal{O}_i$). Thus, the initial definition of *moving points* state is defined by:

$$\mathcal{O}_i = \begin{bmatrix} \mathbf{o}_1^{\mathcal{O}_i} \\ \vdots \\ \mathbf{o}_n^{\mathcal{O}_i} \end{bmatrix} \tag{4.2}$$

where $\mathcal{O}_i$ with $i = \{1, 2, \ldots, N\}$, is the state of *moving object* i and $\mathbf{o}_j^{\mathcal{O}_i} \in \mathcal{O}_i$, are the moving points $\mathbf{o}_j$ that belong to object $\mathcal{O}_i$.

A moving object can be expressed in different frames (world, robot, camera, ...) using the usual frame transformation functions

3.2 Moving Objects Representation

The stochastic map of our system, consists of the main Bi-Camera SLAM state, described in Chapter 2 and a set of stochastic vectors, updated by an EKF filter, one per moving object, thus:

$$\mathbf{M} = \left(\mathbf{X}, \mathcal{O}_{i=\{1,\ldots,N\}}\right) \tag{4.3}$$

with,

$$\mathbf{X} = \begin{bmatrix} C_L \\ C_R \\ L_1 \\ \vdots \\ L_Z \end{bmatrix}, \quad \mathcal{O}_i = \begin{bmatrix} \mathbf{o}_1^{\mathcal{O}_i} \\ \vdots \\ \mathbf{o}_n^{\mathcal{O}_i} \end{bmatrix} \tag{4.4}$$

where $C_{L/R} = [\mathbf{r}_{L/R}, \mathbf{q}_{L/R}] \in \mathbb{R}^7$, is the camera state Left or Right respectively, given by its position and orientation quaternion, $[L_1, \ldots, L_Z]$ are the set of landmarks with $L_j = d_j \in \mathbb{R}^6$ (inverse depth) or $L_j = c_j \in \mathbb{R}^3$ (euclidean coordinates) and $[\mathbf{o}_1^{\mathcal{O}_i}, \ldots, \mathbf{o}_n^{\mathcal{O}_i}]$, are the set of moving point states that characterize the moving object $\mathcal{O}_i$, given by its position and motion model (velocity), $\mathbf{o}_j^{\mathcal{O}_i} = (\mathbf{p}^{\mathbf{o}_j}, \mathbf{v}^{\mathbf{o}_j}) \in \mathcal{O}_i$. Here, for practical purposes, motion model is considered as a constant-velocity model.

4 System Description

We have developed a complete system for SLAM with DATMO, using a Bi-Camera-vision bench as the only sensor. The major processing blocks of our system are depicted in Figure 4.1. Hereafter functions related to the two detection approaches, are described; the SLAM function, executes a vision-based SLAM either from a monocular system or from several cameras using the Bi-Camera strategy described in Chapter 2

4.1 Feature Selection

We take advantage of the KLT (Kanade-Lucas-Tomasi) tracker where the feature selection is made by analysing the values of the image intensity gradient. Thus, two gradient images are obtained in two orthogonal direction from the original image. Then, we build a matrix of gradients around a search window for every point of the initial image using those two gradient images.

Let α_1 and α_2 be the eigenvalues of the gradient matrix in each point, a point of the image is considered as a characteristic feature if

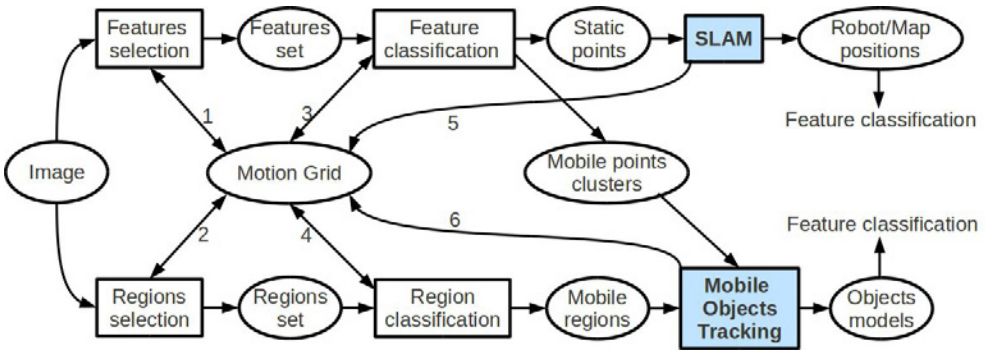


Figure 4.1: The major processing blocks in our system and their connections, executed for each frame of a video sequence.

$min(\alpha_1, \alpha_2) > \alpha$, where α is a threshold taking into account the characteristic of the image, in this manner, each point of the image is analysed and ranked. Then, the first N_p points are selected as the more significant characteristic features.

At this point, N_p initial feature points are selected, thus, the *Feature selection* function can take advantage of all knowledge from previous images, or from contextual or application-dependant knowledge. So every time new points have to be selected, it exploits a proposed *Motion Grid*, built from a discretization of the image space; this *Motion Grid* gives the probability on every image zone (cell), to find a dynamic point. Initially this probability is uniform; then it is updated according to several criteria discussed in section 4.3.

New points are detected in zones which have the higher probability; initially these points are uniformly distributed in the image. Then as one goes along the method, points are replaced once they have been either lost by the tracker, or identified as static, while dynamic points are kept in the points set and tracked continuously.

4.2 Feature classification

We need to classify new features as dynamic or static before using them to estimate the robot pose, the moving objects pose and the map.

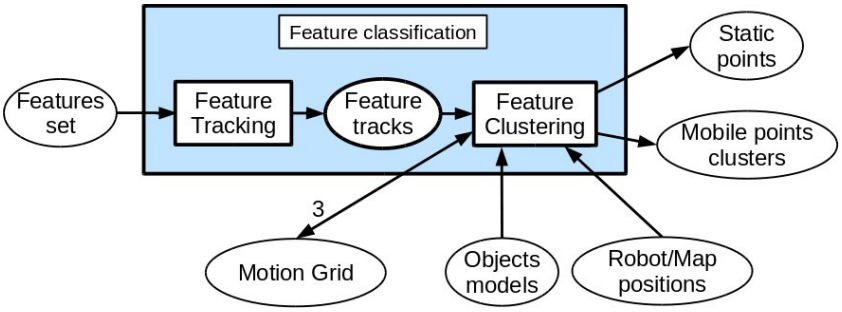


Figure 4.2: The *Feature classification* process.

Based on [Almanza-Ojeda *et al.* 2011], a solution to tracking and clustering features is proposed, we call this procedure *Feature classification*. Fig. 4.2 shows the different modules that are involved in this procedure

In this point, N_p initial feature points are selected. Feature locations in next image are searched by a KLT tracker, based on correlation and optimization functions. This process loops on N_{Im} successive images. This set of N_{Im} processed images, are the number of images used to accumulate positions and apparent velocities of tracked features. We have found that 5 images are sufficient in order to accumulate enough information of positions and velocities to form a vector $V(x, y, v_x, v_y)$ containing the accumulate optical flow for each feature tracked.

Only moving KLT features are selected for being grouped by the *a contrario clustering method* [Veit *et al.* 2007]. Thus, this function generates a list of static points and a list of B_i clusters or potential moving objects, each one defined by a list of points and by characteristics: the cluster barycenter and the mean points apparent velocity. This function requires the robot position or the motion estimation between the last acquisitions, in order to remove false positives by an ego-motion compensation. It needs also the current object models, in order to try to fuse new clusters to existing objects, before creating new ones.

4.3 Motion Grid

In our system, it is important to add new points that make the model of the detected moving object denser, and at the same time detect new

incoming moving objects, thus, it is required to actively select points or regions to be classified as static or as mobile; especially features tracks will be searched only in regions where the occurrence of a moving object is the more probable.

To describe the behaviour of this occurrence, we introduce the concept of *Motion Grid* in a similar manner that in [Almanza-Ojeda *et al.* 2011]. A Motion Grid acts as an occupancy grid, in the sense that higher probabilities represent occupied (moving) zones and lower probabilities free (static) ones, where to monitor the arrival of new moving points.

4.3.1 Motion grid initialization

At this point, the image is divided in a grid of $N \times M$ zones. This grid define the zones where landmarks (static features) and moving objects (mobile features) will be searched. Based on a probabilistic framework, the *Motion Grid*, will activate only the zones with sufficient interest for monitoring.

We use the following Gaussian distribution to assign a probability value in each zone,

$$\mathcal{N}\{z_{iu}, z_{iv}|\mu_u, \mu_v; \sigma_u, \sigma_v\} = \mathcal{N}\{z_i|\boldsymbol{\mu}; \boldsymbol{\sigma}\} \tag{4.5}$$

$$\mathcal{N}\{z_i|\boldsymbol{\mu}; \boldsymbol{\sigma}\} = \frac{1}{2\pi\sigma_u\sigma_v} \cdot e^{-\frac{1}{2}\left(\left(\frac{z_{iu}-\mu_u}{\sigma_u}\right)^2+\left(\frac{z_{iv}-\mu_v}{\sigma_v}\right)^2\right)} \tag{4.6}$$

where z_{iu} and z_{iv} are the coordinates of the centre of the image zone z_i. The parameters μ_u and μ_v are the means of the motion probability distribution in a zone z_i. The parameters σ_u and σ_v are the variances of the motion probability distribution.

Given that the grid that divides the image into $N \times M$ zones, is the same for all the images of the sequence, the variance values are equals and fixed to $\frac{n}{3}$. Thus, the distribution for a grid of 14×14 image zones, will have the form shown in Fig. 4.3(c), where the highest probability value will be in the centre of each zone (z_i).

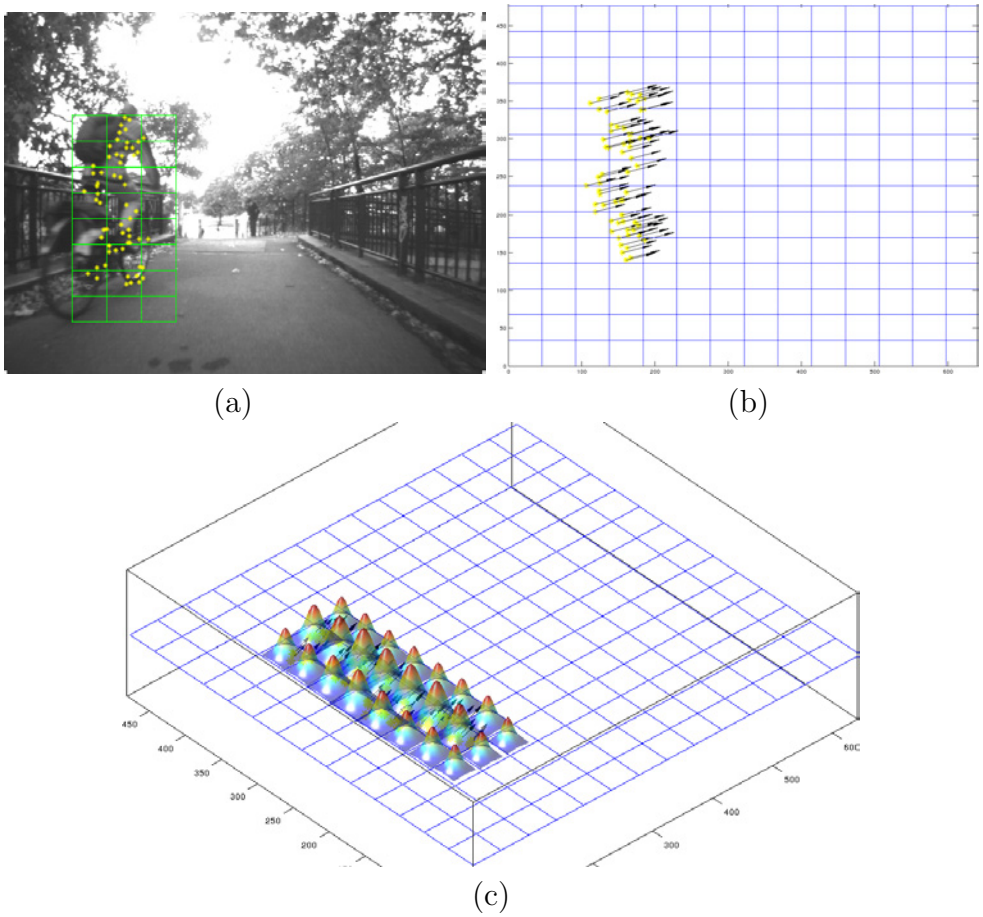


Figure 4.3: *Motion Grid* principle.

Fig. 4.3 shows the *Motion Grid* principle for a grid of 14×14 image zones. Thus, Fig. 4.3(a) shows the clustering of mobile points resulting from the *feature selection, tracking and classification* process and the interesting zones (in green) where mobile features reside, Fig. 4.3(b) shows the vector $V(x, y, v_x, v_y)$ for each mobile feature in the cluster that gives us an idea of the motion evolution of the moving object. Before beginning a new cycle of N_{Im} images the algorithm need to update the *Motion Grid* in order to know where to look for new static and mobile features, Fig. 4.3(c) shows the evolution of the probability in the *Motion Grid*, the update procedure will be discussed below (section 4.3.2). We note that the probabilistic grid has the same size as the image and that white areas are free (static) zones.

4.3.2 Motion grid update

The motion grid is modified or exploited by a function on every labelled edge on figure 4.1. On the edges 1 and 2, the *Features selection* and *Regions selection* functions, select features or regions on the more probable areas where to detect moving objects. It will allow to confirm or not a previous detection. It has been found that the feature-based method to detect moving objects, could generate false alarms on the ground in regions close to the camera, because here, the apparent motion is important; the region-based method could remove these false alarms. But it is possible that it exist really here a moving object, the class of which has not been learnt[1] (for example an animal); so one detector can be efficient while the other one fails. For example, the region-based approach is able to detect "static objects" belonging to the learnt classes; even if static, these objects could move, so that, on one hand, features selected on these regions must be monitored to detect when these objects move, and on the other hand, SLAM must avoid to select and exploits features on these areas.

On the edge 3 and 4, the *Feature classification* and *Region classification* functions update the motion grid. As explained in previous section, a set of features tracked during N_{im} successive images, are grouped in a *mobile feature cluster*, so the *Motion Grid* must be updated.

The probability values in the map are modified by a bidimensionnal Gaussian function centred on every mobile point (u, v), with σ_u and σ_v, parameters to be tuned.

We can express the the spatio temporal evolution of probability into moving and static zones at each current image as,

$$p(u, v, t) = \Theta_i^{pm} = \sum_{u,v \in zone} (\alpha(u, v)_{t-1} + G(\mu_u, \mu_v, \sigma_u, \sigma_v)) \qquad (4.7)$$

[1] This detection based learning approach will be explained in section 4.4

where $\alpha(u, v)$ describes the previous probability in the zone in function of their previous state, that is:

$$\alpha(u,v)_t = \begin{cases} 1 - p(u,v)_t & \text{if } \alpha(u,v)_{t-1} = moving \\ p_0 & \text{if } \alpha(u,v)_{t-1} = static \end{cases} \tag{4.8}$$

Thus, the probability that a pixel belongs to a moving object, is inversely proportional to its distance to the closer detected moving point.

The same method is applied around a static point, using the inverse of a bidimensionnal Gaussian function, so that the probability that a point is static is also inversely proportional to its distance to the closer detected static point. So every pixel in the Motion Grid receives the influence of closer points labelled static or dynamic.

For a region, the uncertainty given by the classification method is directly transferred to every pixel of the region, whatever the label (details are explained in next Section).

Finally, for the edge 5 and 6, the *SLAM* and *MOT* functions update the motion grid. For SLAM, if a landmark is not observed, it could be interpreted that it is occluded by a new obstacle. So probabilities around the predicted position of its observation, are updated using the Gaussian function. Reciprocally, if a landmark is observed, it means that the surrounding area is static, so probabilities are updated using the inverse Gaussian function.

For MOT, probabilities are updated according to the object tracking results: features will be selected inside and around regions already detected as mobile in the previous images. It will allow to densify the object models, to select features behind an object track (perhaps the tracked object occludes another one).

4.4 Classification approach

Our system takes advantage of a classification discriminant function in combination with the cluster information to choose in an intelligent way where to look for possible moving objects. This approach to detection and creation of moving objects is described

in next section. The system we propose is independent of a particular classifier choice, so we have tested with two state-of-the-art classifiers [Dalal & Triggs 2005], [Viola & Jones 2002] implemented as follow:

4.4.1 Rectangular filters or Haar-like features

Rectangular filters or Haar-like features provide information about the grey-level distribution of two adjacent regions in an image.

Figure 4.4, shows the set of Haar filters used in our algorithm. These filters consist of two, three or four rectangles. To compute the output of a filter on a certain region of image, the sum of all pixels values in the grey region is subtracted from the sum of all pixels values in the white one (and normalized by a coefficient in case of a filter with three or four rectangles).

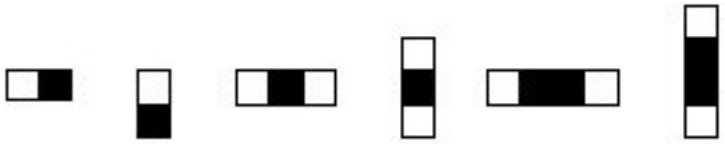
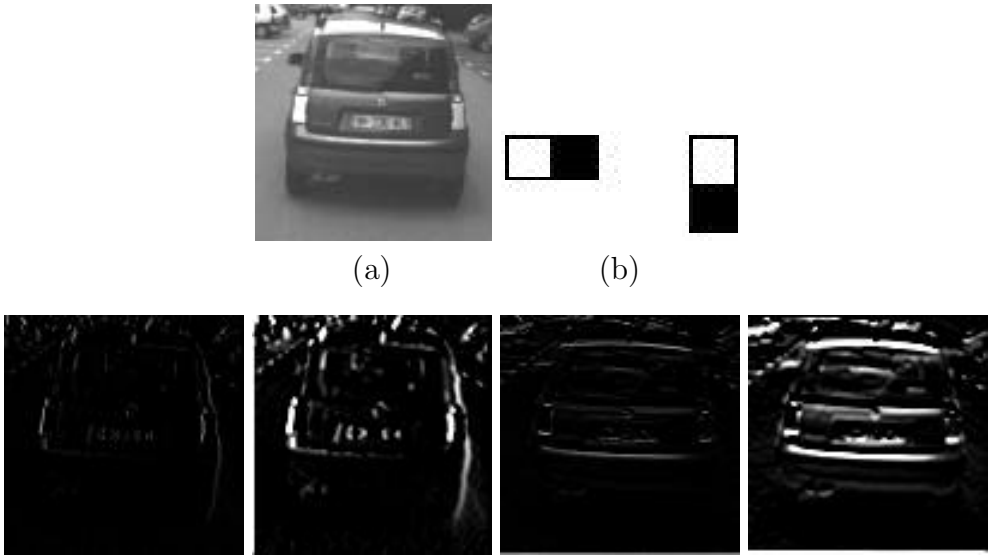


Figure 4.4: Set of Haar-like features. Black and white areas have negative and positive weights respectively.

Viola and Jones [Viola & Jones 2004] introduced the integral image which is an intermediate representation of an input image and reduces the computation time for the filters. Sum of the rectangular regions can be calculated by using only four references in the integral image. As a result, the difference of two adjacent rectangular regions can be computed by using only six references in the integral image. For a filter with three rectangular regions, only eight references are needed. At the same time, integral image allows to perform fast variance normalization, necessary to reduce the effect of different lighting conditions.

Every feature f is defined as

$$g_f = (x_f, y_f, s_f, r_f) \tag{4.9}$$



(a) (b)

(c) (d) (e) (f)

Figure 4.5: Original image and the output obtained by applying the vertical and horizontal Haar filters: (a) original image; (b) vertical and horizontal Haar filters used; (c) output for a vertical Haar filter with scale 1×2 pixels; (d) output for a horizontal Haar filter with scale 1×2 pixels; (e) output for a vertical Haar filter with scale 2×4 pixels; (f) output for a horizontal Haar filter with scale 2×4 pixels.

where r_f the type of rectangular filter (Figure 4.4), s_f is the scale, and (x_f, y_f) is its position in the window. Five scales are used for the two rectangles filters: 1×2, 2×4, 4×8, 8×16, 16×32, similar scales are used for the three and four rectangles filters.

Figure 4.5 illustrates the filtering of an image using two types of rectangular filters on two different scales: 1×2 and 2×4 pixels. These pictures show that the chosen filters emphasize the horizontal and vertical edges in the image. We can also observe that when the filter size is doubled, details in the image and the most important edges are conserved.

4.4.2 Histogram of oriented gradient

The other feature space used in this work is HoG. This local feature uses gradient magnitude and orientation around a point of interest or in a region of the image to construct the histograms. To calculate

the input image (grey-level) gradient, we follow the construction in [Dalal & Triggs 2005] to define a dense representation of an image at a particular resolution. The image is first divided into 8×8 non-overlapping pixel regions, or cells. For each cell we accumulate a $1D$ histogram of gradient orientations over pixels in that cell. These histograms capture local shape properties but are also somewhat invariant to small deformations.

The gradient at each pixel is discretized into one of nine orientation bins, and each pixel "votes" for the orientation of its gradient, with a strength that depends on the gradient magnitude at that pixel. Finally, the histogram of each cell is normalized with respect to the gradient energy in a neighborhood around it. We look at the four 2×2 blocks of cells that contain a particular cell and normalize the histogram of the given cell with respect to the total energy in each of these blocks. This leads to a 9×4 dimensional vector representing the local gradient information inside a cell.

4.4.3 AdaBOOST

Adaboost algorithm [Freund & Schapire 1996] has shown its capability to improve the performance of various classification and detection systems. It finds precise hypotheses by combining several weak classification functions which, in general, have moderate precision. Adaboost is an iterative algorithm that finds, from a feature set, some *weak* but discriminative classification functions and combines them in a *strong* classification function:

$$C = \begin{cases} 1, & \text{if } \sum_{d=1}^{D} \alpha_d c_d \geq \frac{1}{2} \sum_{d=1}^{D} \alpha_d = S \\ 0, & \text{otherwise} \end{cases} \tag{4.10}$$

where C and c are the *strong* and *weak* classification functions, respectively, and α is a weight coefficient for each c. S is the threshold of strong classifier C.

Different variants of boosting algorithm are developed like discrete AdaBoost [Viola & Jones 2004], real AdaBoost [Friedman *et al.* 2000],

gentle AdaBoost, and so forth. However, we use the discrete AdaBoost, this algorithm had shown to be an appropriate method for fast and reliable object detection.

Thus, we have to define the weak classifiers for two different types of features: Haar and HoG.

4.4.4 Weak classifier-Haar

We define the weak classifier for a feature f as a binary response c_{Haar}:

$$c_{Haar} = \begin{cases} 1, & \text{if } p_f g_f < p_f \lambda_f \\ 0, & \text{otherwise} \end{cases} \tag{4.11}$$

where g_f is the absolute value of the feature f, λ_f is the threshold, and p_f is the parity. For each feature f, AdaBoost determines an optimal threshold λ_f for which the classification error on training database (with positive and negative examples) is minimised.

4.4.5 Weak classifier-HoG

In this case, we construct a classifier based on the class (vehicle) model. The median of histograms of positive examples from training database is used as our model, defined as

$$m_f = median\{h_f^i\}_{i=1,\ldots,k} \tag{4.12}$$

where h_f is the histogram computed before (see section 4.4.2) and k is the number of positive examples in the training database.

The classifier computes the distance between a histogram h_f of the input image and a model histogram m_f. Thus, we defined the weak classifier c_{HoG} as follows:

$$c_{HoG} = \begin{cases} 1, & \text{if } B(h_f, m_f) < \lambda_f \\ 0, & \text{otherwise} \end{cases} \tag{4.13}$$

where $B(h_f(x), m_f)$ is the Bhattacharya distance between the histogram h_f and the model histogram m_f, and λ_f is the optimal threshold on the distance for this feature.

Bhattacharya distance is defined as

$$B(h_f, m_f) = \sqrt{1 - h_f \odot m_f} \tag{4.14}$$

where $\odot$, is the scalar product. This distance is a similarity measure between two histograms, that is, values close to 0 for similar histograms. The output values are bounded between 0 and 1.

4.5 Moving Objects detection

Before to create a "moving object", we need to detect it. We have developed an intelligent procedure to take advantage, in a probabilistic way, of the cluster information combined with the output of a classification discriminant function.

4.5.1 Object Hypotheses: classification based approach

Object hypotheses, are created from the output of a detector based classification. As mentioned above, the system we propose is independent of a particular detector choice. Nevertheless in our experiments for practical purposes we use the AdaBoost algorithm implemented as described previously in the Section 4.4

The algorithm described here called *Object Hypotheses*, is specifically and only intended to classify a region as being a pedestrian or a car.

Each cluster or group candidate noted B_i sent by the *Feature classification* function, produces a region of interest in the corresponding image, in the same way, a mobile region, D_i, is sent by the *Region classification* function.

We can define a "region of hypothesis" as the union of the region of interest from the *Feature classification* function and the mobile region from the *Region classification* function,

$$M_i = B_i \cup D_i \tag{4.15}$$

Only these regions of hypothesis will be processed by the *Object Hypotheses* algorithm to accelerate the search-detection of the classification algorithm inside the image.

Thus, the region of hypothesis M_i is decomposed into n different regions, $(M_i^k)_{1\leq k\leq n}$, each one of these n regions is processed by the classification algorithm and is given a value λ_i^k corresponding to the sum of the weighted combination of the N weak classifiers, $(w_l)_{1\leq l\leq N}$

$$\lambda_i^k = \sum_{1\leq l\leq N} \alpha_l w_l(M_i^k) \tag{4.16}$$

Finally, for each cluster M_i a global classification score is computed as follow:

$$\Theta_i^c = \max_{1\leq k\leq n} \lambda_i^k \tag{4.17}$$

As the algorithm described above is only trained to compute a classification probability for a given region to be a pedestrian or a car, no detection or tracking information can be deduced from such result. However, as explained above every cluster region is given a new classification score Θ_i^c.

For any given cluster-object i at time k, this classification score is independent and different from the uncertainty score already computed by the *Motion Grid*. The fusion rule to combine those different scores is detailed below.

4.5.2 Final Object Hypotheses fusion rule

In order to finally detect correctly moving objects, the estimation of the *Motion Grid* has to be combined with the estimation of the *Object Hypotheses classification based approach*. Two different estimates of the same probabilities, $P(\mathcal{O}_i|A_j)$, the probability of the event $\mathcal{O}_i$ (to be a moving object) given A_j. These estimates have been computed by algorithms that are subject to uncertainties. Thus, each estimate can be rewritten as follows:

$$P(\mathcal{O}_i|A_1) = \Theta_i^{pm} \tag{4.18}$$

$$P(\mathcal{O}_i|A_2) = \Theta_i^c \tag{4.19}$$

where $P(\mathcal{O}_i|A_1)$ is the estimation of the *Motion Grid* and $P(\mathcal{O}_i|A_2)$ is the estimation of the *classification based approach*. Assuming that A_1 and A_2 are independent, the following fusion rule can be derived from basic Probability rules:

$$\begin{aligned} P(\mathcal{O}_i) &= P(\mathcal{O}_i \cap (A_1 \cup A_2)) + P(\mathcal{O}_i \cap (\overline{A_1 \cup A_2})) \\ &= P(\mathcal{O}_i|A_1)P(A_1)P(\overline{A_2}) + P(\mathcal{O}_i|A_2)P(\overline{A_1})P(A_2) \\ &+ P(\mathcal{O}_i|A_1 \cap A_2)P(A_1)P(A_2) + P(\mathcal{O}_i|\overline{A_1} \cap \overline{A_2})P(\overline{A_1})P(\overline{A_2}) \end{aligned} \tag{4.20}$$

where $P(A_j)$ refers to the probability for the subsystem j (motion grid or classification based approach) to return wrong estimation. $P(\mathcal{O}_i|\overline{A_1} \cap \overline{A_2})$ can be approximated to ξ, a priory classification probabilities. Thus, the fusion rule is approximated as a weighted sum:

$$\begin{aligned} \Phi_{\mathcal{O}_i} &= \Theta_i^{pm} P(A_1)P(\overline{A_2}) + \Theta_i^c P(\overline{A_1})P(A_2) \\ &+ \frac{\Theta_i^{pm} P(A_1) + \Theta_i^c P(A_2)}{P(A_1) + P(A_2)} P(A_1)P(A_2) + \xi P(\overline{A_1})P(\overline{A_2}) \end{aligned} \tag{4.21}$$

At this point, final estimate for the detection probability is available. The final Moving Object detection is done based on a threshold. Thus, for each region of interest M_i,

$$M_i = \begin{cases} \textit{Moving Object} & \text{if } \Phi_{\mathcal{O}_i} \geq \text{threshold} \\ \\ \textit{NOT Moving Object} & \text{if } \Phi_{\mathcal{O}_i} < \text{threshold} \end{cases}$$

Our fusion strategy is based on the combination of the estimates of two algorithms of different nature that are contributing, the *Motion Grid* and the *Object Hypotheses classification based approach* algorithms.

4.6 Moving Objects tracking

4.6.1 Moving points and Objects creation

Upon each moving object detection an EKF is created to host its state's *pdf*. The mobile state vector is hence the Gaussian $\mathcal{O} \sim \mathcal{N}\{\hat{\mathcal{O}}; \mathbf{P}_{\mathcal{O}}\}$. For its creation we will follow the IDP initialization method like in [Civera *et al.* 2008]. Initial mean and covariances matrix for moving points positions $\mathbf{p}^{\mathbf{o}_j}$, belonging to moving point $\mathbf{o}_j^{\mathcal{O}_i} \in \mathcal{O}_i$, are determined from the inverse depth parametrization from the left camera

$$\mathbf{p}^{\mathbf{o}_j} = \mathbf{d}_L(C_L, \mathbf{h}_L, \rho) \tag{4.22}$$

where $\mathbf{d}_L(\cdot)$ is a function of $C_L = [\mathbf{x}_L, \mathbf{q}_L]$ the left camera state containing current position and orientation, $\mathbf{h}_L \sim \mathcal{N}\{\mathbf{y}_L; \mathbf{R}\}$ the Gaussian observation from the left camera and $\rho \sim \mathcal{N}\{\hat{\rho}; \sigma_\rho^2\}$ the inverse of the Euclidean distance from C_L to the moving point position. After the first observation with the left camera, all parameters of $\mathbf{p}^{\mathbf{o}_j}$ except ρ are immediately observable, and their values and covariances are obtained by proper inversion and linearization of the observation functions.

Thus, moving point position's, $\mathbf{p}^{\mathbf{o}_j}$, initial mean and covariances matrix are then

$$\hat{\mathbf{p}}^{\mathbf{o}_j} = \mathbf{d}_L(C_L, \hat{\mathbf{h}}_L, \hat{\rho}) \tag{4.23}$$

$$\mathbf{P}_{\mathbf{pp}} = \mathbf{D}_{L\mathbf{h}}\, \mathbf{R}\, \mathbf{D}_{L\mathbf{h}}^T + \mathbf{D}_{L\rho}\, \sigma_\rho^2\, \mathbf{D}_{L\rho}^T \tag{4.24}$$

where $\mathbf{R}$ is the rotation matrix and the Jacobian matrices are

$$\mathbf{D}_{L\mathbf{h}} = \left.\frac{\partial \mathbf{d}_L}{\partial \mathbf{h}^T}\right|_{(\hat{C}_L, \hat{\mathbf{h}}, \hat{\rho})} \tag{4.25}$$

$$\mathbf{D}_{L\rho} = \left.\frac{\partial \mathbf{d}_L}{\partial \rho^T}\right|_{(\hat{C}_L, \hat{\mathbf{h}}, \hat{\rho})} \tag{4.26}$$

Initial mean and covariances matrix for the moving points's velocity $\mathbf{v}^{\mathbf{o}_j} \sim \mathcal{N}\{\hat{\mathbf{v}}^{\mathbf{o}_j}; \mathbf{P}_{\mathbf{vv}}\}$ are heuristically determined like

in [Lin & Wang 2010].

Finally, the full moving object's Gaussian *pdf* is the specified by the couple

$$\hat{\mathcal{O}}_i = \begin{bmatrix} \hat{\mathbf{p}}^{\mathbf{o}_j} \\ \hat{\mathbf{v}}^{\mathbf{o}_j} \end{bmatrix}, \quad \mathbf{P}_{\mathcal{O}_i} = \begin{bmatrix} \mathbf{P_{pp}} & 0 \\ 0 & \mathbf{P_{vv}} \end{bmatrix}, \tag{4.27}$$

Immediately after creation, this EKF is updated with the observation from the right camera. The observation function from the right camera is the simple pin-hole camera projection function expressed in right-frame, generically written as

$$\mathbf{y}_R = \mathbf{h}_R(C_R, \mathbf{p}^{\mathbf{o}_j}) + v_R \tag{4.28}$$

where $v_R \sim \mathcal{N}\{0; \mathbf{R}\}$ is independent white Gaussian observation noise.

Thus, upon observation from right camera, with uncertainty pose $C_R \sim \mathcal{N}\{\hat{C}_R; \mathbf{P}_{C_R}\}$, the moving object's *pdf* is update following the EKF equations as follow,

$$\mathbf{z}_R = \mathbf{y}_R - \mathbf{h}_R(\hat{C}_R, \hat{\mathbf{p}^{\mathbf{o}_j}}) \tag{4.29}$$

$$\mathbf{Z}_R = \mathbf{H}_{R\mathcal{O}_i}\, \mathbf{P}_{\mathcal{O}_i}\, \mathbf{H}_{R\mathcal{O}_i}^T + \mathbf{H}_{RC}\, \mathbf{P}_{CR}\, \mathbf{H}_{RC}^T + \mathbf{R} \tag{4.30}$$

$$\mathbf{K} = \mathbf{P}_{\mathcal{O}_i}\, \mathbf{H}_{R\mathcal{O}_i}^T\, \mathbf{Z}_R^{-1} \tag{4.31}$$

$$\hat{\mathcal{O}}_i^{\,+} = \hat{\mathcal{O}}_i + \mathbf{K}\, \mathbf{z}_R \tag{4.32}$$

$$\mathbf{P}_{\mathcal{O}_i}^{+} = \mathbf{P}_{\mathcal{O}_i} - \mathbf{K}\, \mathbf{Z}_R\, \mathbf{K}^T \tag{4.33}$$

where the innovation $\{\mathbf{z}_R; \mathbf{Z}_R\}$ is obtained for practical purposes from the expectation $\mathbf{e}_R \sim \mathcal{N}\{\hat{\mathbf{e}}_R; \mathbf{E}_R\}$ defined by

$$\hat{\mathbf{e}}_R = \mathbf{h}_R(\hat{C}_R, \hat{\mathbf{p}^{\mathbf{o}_j}}) \tag{4.34}$$

$$\mathbf{E}_R = \mathbf{H}_{R\mathcal{O}_i}\,\mathbf{P}_{\mathcal{O}_i}\,\mathbf{H}^T_{R\mathcal{O}_i} + \mathbf{H}_{RC}\,\mathbf{P}_{CR}\,\mathbf{H}^T_{RC} + \mathbf{R} \tag{4.35}$$

The linearity test in [Civera *et al.* 2007] is regularly evaluated. If passed, the object position can be safely transformed into a 3-D Euclidean parametrization.

4.6.2 Moving Objects and Robot Motion compensation

The robot time-evolution model is considered an odometry model, this, can be generically written as $R^+ = \mathbf{f}_R(R, \mathbf{u})$ where $\mathbf{u}$ is a vector of robots controls or odometry data

$$\mathbf{u} = \begin{bmatrix} \delta\mathbf{x} \\ \delta\mathbf{e} \end{bmatrix} = [\delta x, \delta y, \delta z, \delta\phi, \delta\theta, \delta\psi]^T \in \mathbb{R}^6 \tag{4.36}$$

with Gaussian *pdf* $\mathbf{u} \sim \mathcal{N}\{\hat{\mathbf{u}}; \mathbf{U}\}$. Upon robot motion, moving objects must change their coordinates from the old robot frame-position to the new one, thus

$$\mathcal{O}_i^+ = \mathbf{j}_{\mathcal{O}}(\mathcal{O}_i, \mathbf{u}) \tag{4.37}$$

which is a frame transformation that is specified by the odometry parameters $\mathbf{u}^T = [\delta\mathbf{x}^T, \delta\mathbf{e}^T]$, we can then write the moving objects as

$$\mathbf{p}^{\mathbf{o}_j +} = \mathbf{R}^T\delta(e) \cdot (\mathbf{p}^{\mathbf{o}_j} - \delta x) \tag{4.38}$$

$$\mathbf{v}^{\mathbf{o}_j +} = \mathbf{R}^T\delta(e) \cdot \mathbf{v}^{\mathbf{O}_j} \tag{4.39}$$

where $\mathbf{R}^T\delta(e)$ is the rotation matrix from the Euler angles increment $\delta(e)$.

Thus, with the new (compensated) positions and velocities, the moving object's *pdf* is update following the EKF equations.

4.6.3 Moving Objects motion

Constant velocity motion models produce more or less smooth trajectories depending on the amount of noise introduced in the system. For example, the motion of a pedestrian can be modeled with a constant velocity model in the 2D plane. If we consider it cannot stop abruptly, a single, relatively small Gaussian will do. A larger Gaussian could be used to accommodate for sudden stops and goes, but that would reduce the motion's smoothness and eliminate the difference between the states go and stop. The better suited model is a two hypotheses model with associated transition probabilities, for example, following Solà analysis [Solà 2007], we have:

$$\text{At any instant : } \begin{cases} \textit{if } \mathcal{O}_i \textit{ is moving} & \text{probably keeps moving,} \\ & \text{but may eventually stop.} \\ \textit{if } \mathcal{O}_i \textit{ is stopped} & \text{probably stays,} \\ & \text{but may eventually go.} \end{cases} \tag{4.40}$$

In this point, different Gaussian noises could be assigned and incorporated to the system. Nevertheless, in this work, for practical purposes, motion model is considered constant-velocity model.

Therefore, moving object's time-evolution model is written as

$$\mathcal{O}_i^+ = \mathbf{f}_{\mathcal{O}}(\mathcal{O}_i, \omega) \tag{4.41}$$

which responds to the constant-velocity model with no rotational part defined

$$\mathbf{p}^{\mathbf{o}_j+} = \mathbf{p}^{\mathbf{o}_j} + T_s\, \mathbf{v}^{\mathbf{o}_j} \tag{4.42}$$

$$\mathbf{v}^{\mathbf{o}_j+} = \mathbf{v}^{\mathbf{o}_j} + \omega \tag{4.43}$$

where T_s is the filter's sampling time and $\omega = [\omega_x, \omega_y, \omega_z]^T \in \mathbb{R}^3$ is a white Gaussian velocity perturbation $\omega \sim \mathcal{N}\{0, \mathbf{Q}\}$.

Thus, the moving object *pdf* is updated following the EKF equations as follows

$$\hat{\mathcal{O}}_i^{+} = \mathbf{f}_{\mathcal{O}}(\hat{\mathcal{O}}_i, 0) \tag{4.44}$$

$$\mathbf{P}^{+}_{\mathcal{O}_i} = \mathbf{F}_{\mathcal{O}_i\mathcal{O}_i}\,\mathbf{P}_{\mathcal{O}_i}\,\mathbf{F}^{T}_{\mathcal{O}_i\mathcal{O}_i} + \mathbf{F}_{\mathcal{O}_i\omega}\,\mathbf{Q}\,\mathbf{F}^{T}_{\mathcal{O}_i\omega} \tag{4.45}$$

5 Experiments

In this section we present preliminary results with the aim of validate the concepts presented before. This is not intended to be a particular challenging example, it is simply used to take the reader through the functionality of the system. Further experiments will be presented in Chapter 5.

We present preliminary results with a ground robot: a Segway platform equipped with a calibrated stereo-vision bench made of t two Marlin 1280 x 960 cameras with a baseline of 0.40m. We demonstrate our Bi-Camera SLAM-DATMO system operation in a real data experiment.

We tested our system in outdoor environment. For this experiment, the robot with the two cameras looking forward is run in the *Canal du Midi* Toulouse-France. A simple 2D odometry model is used for robot-motion predictions.

A single moving object is presented in the scene. A person riding a bicycle traversing the scene and went toward the positive z direction.

For a quantitative evaluation, we measure bounding box overlap in each frame and plot recall over false positives per image. The results of this evaluation are shown in Fig. 4.6. Fig. 4.6(a) and Fig. 4.6(b) shows results of the detection and tracking process, *cyan* and *blue* features are update and predicted landmarks respectively, used by the SLAM algorithm; *yellow* features are updated moving points; *red squares* are moving objects. The plot compares the raw detector output (Fig. 4.6(c)), the proposed approach using a classification discriminant based on Vi-

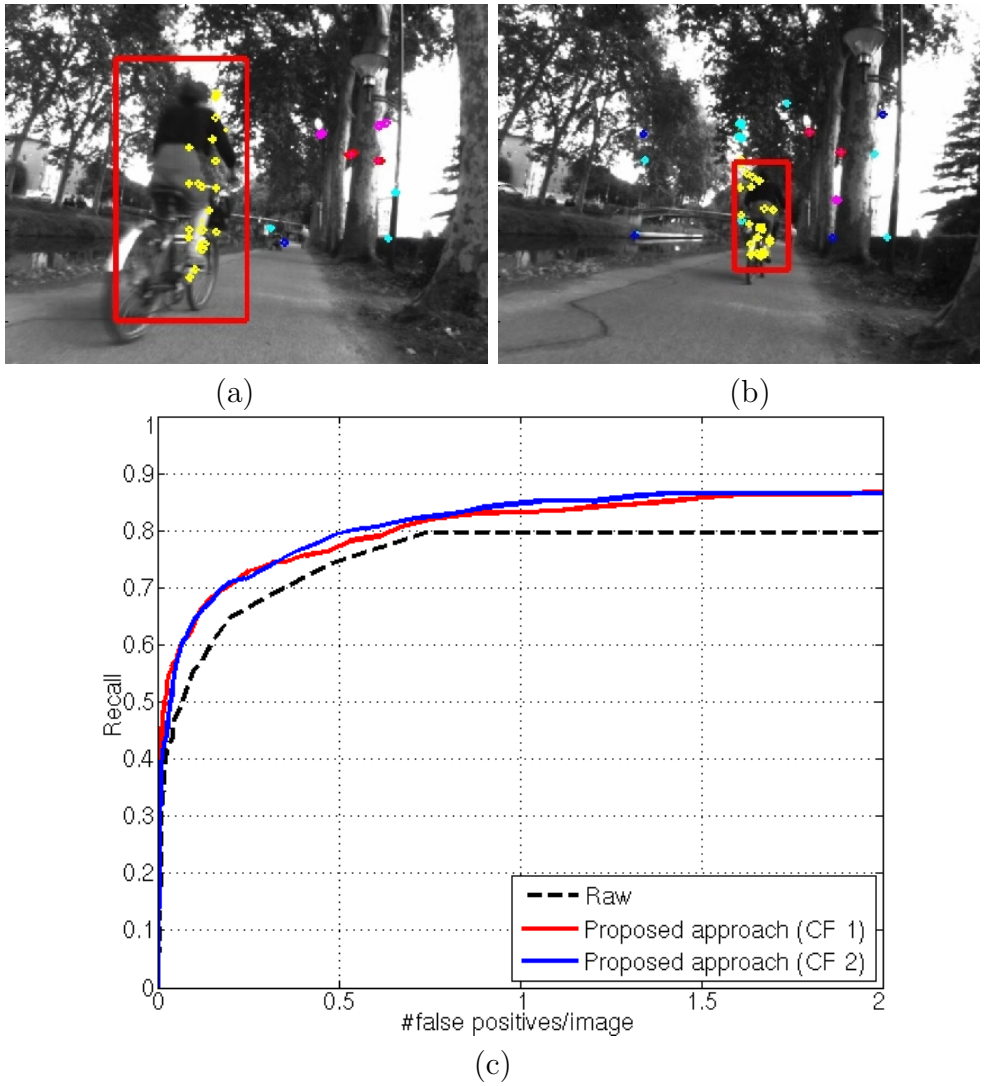


Figure 4.6: Example tracking results and performance plot of our system for the first sequence. We plot the overall recognition performance.

ola & Jones (CF 1) and the proposed approach using a classification discriminant based on Dalal & Triggs (CF 2).

Figure 4.7, shows the BiCam SLAM-DATMO results for this experiment. It is showed that moving objects were successfully detected and tracked in 3D space by the proposed approach.

The results shown in the experiments, verifies the efficacy of the system. The performance of robot localization, mapping and tracking of moving objects verify that our proposed approach is feasible in dynamic

environments.

6 Conclusion

This chapter presents a system that enables a practical solution to the SLAM problem with Detection and Tracking of Moving Objects that is based only on visual exteroceptive perception. The detection of moving objects is performed by a method that allows selection of interesting regions in the image in order to anticipate eventual moving objects apparition in the next few frames.

The different functions in the system were integrated to perform visual SLAM and detection and tracking of moving objects, and we presented, how each function interact with other. Experiments on various real indoor and outdoor sequences shows the efficacy of the system.

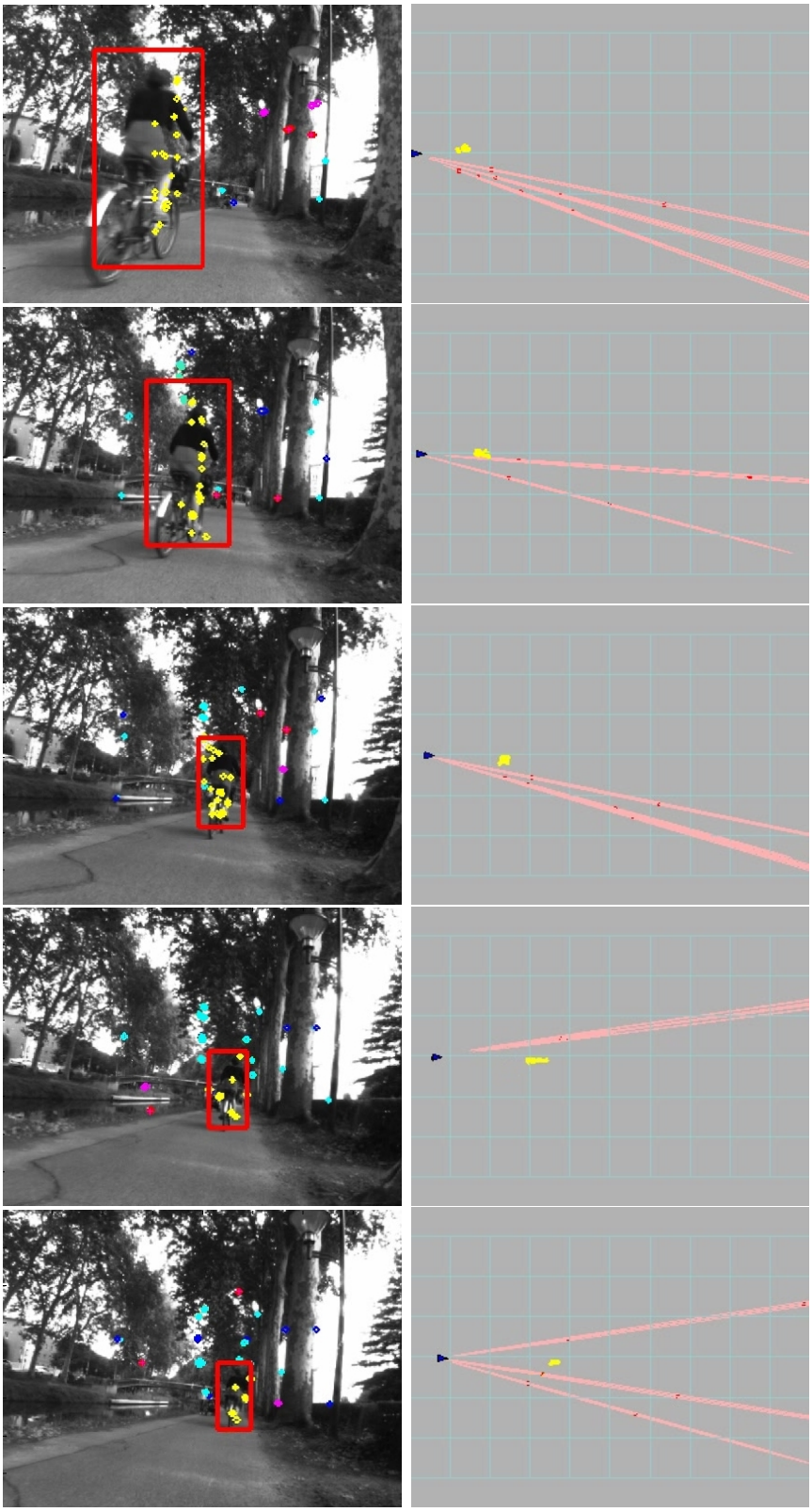


Figure 4.7: Experiment results: image plane and top views of the produced 3D map. In the image view, *cyan* features are update landmarks; *blue* features are predicted landmarks; *yellow* features are updated moving points; *red squares* are moving objects.

5 Applications

The fundamental laws necessary for the mathematical treatment of a large part of physics and the whole of chemistry are thus completely known, and the difficulty lies only in the fact that application of these laws leads to equations that are too complex to be solved.

Paul Dirac

Contents

1 Preliminary 94

2 The MORSE Simulator 94

3 Learning and Replay: a real robot experiment 95

3.1 Learning step 96

3.2 replay step 97

4 Convoy Navigation 101

5 Active Visual-based Detection and Tracking of Moving Objects 104

5.1 Indoor Experiment: 2D detection and tracking task 104

5.2 Outdoor Experiment: Bi-Camera SLAM-DATMO 105

1 Preliminary

In this chapter we present the application context of our research. Several experiments using real robotics platforms have been performed. Before implementing an algorithm in the robot, this was validated using the MORSE simulator.

The convoy task was only implemented in MORSE because currently the laboratory has only a single unmanned robot. However, actually, we are trying to implement our algorithm in the convoy task using an urban vehicle as a leader (using a Bi-Camera Inertial SLAM system) and the laboratory robot as a follower.

2 The MORSE Simulator

The Modular OpenRobots Simulation Engine[1] (MORSE) is a open-source tool developed for robotics research. It is a domain independent simulator, where virtual robots can interact with a 3D environment, using sensors and actuators that behave in the same way as their counterparts in the real world.

MORSE relies on the advanced 3D (OpenGL shaders) and physics simulation (Bullet physics engine) capabilities of the Blender Game Engine, a real-time 3D runtime integrated to the open-source Blender modeling toolkit. This allows for semi-realistic simulation of complex environments.

The MORSE components (sensors and actuators) exchange data with the robotics software via middleware bindings, using a Software In The Loop (SAIL) architecture. Middleware supported in the current version include ROS and YARP, as well as a socket-based raw protocol. This design allows in principle to use the same software in both the real robots and the simulator. Instructions given to the robot are interpreted in the simulator to provide the control of actuators, such as the motion of the robot. The data from simulated sensors is sent back through the middlewares, e.g. exporting the images from cameras, or the positions of

[1] `http://morse.openrobots.org`

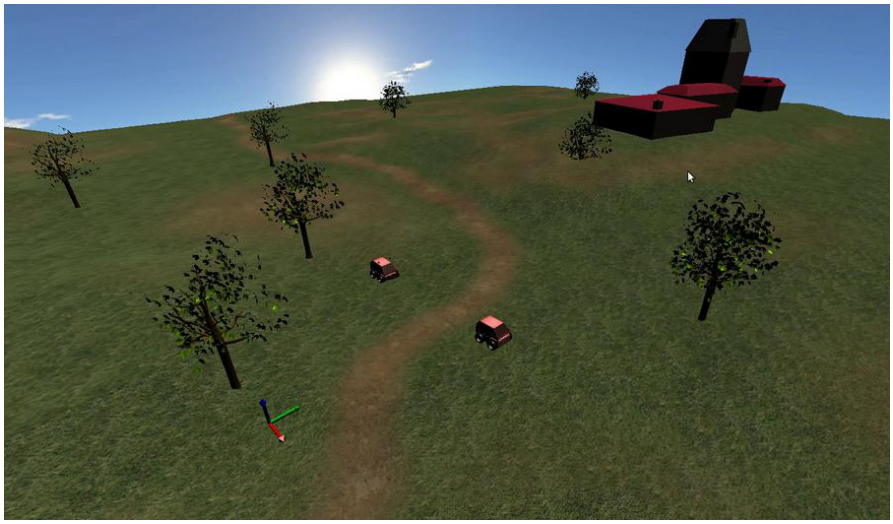


Figure 5.1: The MORSE Simulator: a domain independent simulator, where virtual robots can interact with a $3D$ environment, using sensors and actuators that behave in the same way as their counterparts in the real world.

the robot. MORSE provides support for several classes of robots out of the box, and allows for easy customization of those, either by composing individual sensors and actuators with empty robot structures directly in the MORSE interface, or through a Python-based script language that permits to conveniently describe robots and simulation scenario (see Fig. 5.1).

3 Learning and Replay: a real robot experiment

Let us begin remembering our *Learning-mapping* and *Replay-localization* system [2], as can be seen in Figure 5.2.

For this experiment, we have tested our system in the parking lot of the LAAS-CNRS (see Fig. 5.3) using a ground robot (Segway platform) equipped with two independent stereo-vision benches. The first one used in the *Learning* step is made of two Marlin 1280 × 960 cameras. The second one used in the *Replay* step is made of two Flea2 1280 × 960 cameras (see Fig. 5.4).

[2]The concepts of *Learning mapping* and *Replay-localization* were presented in Chapters 2 and 3 respectively

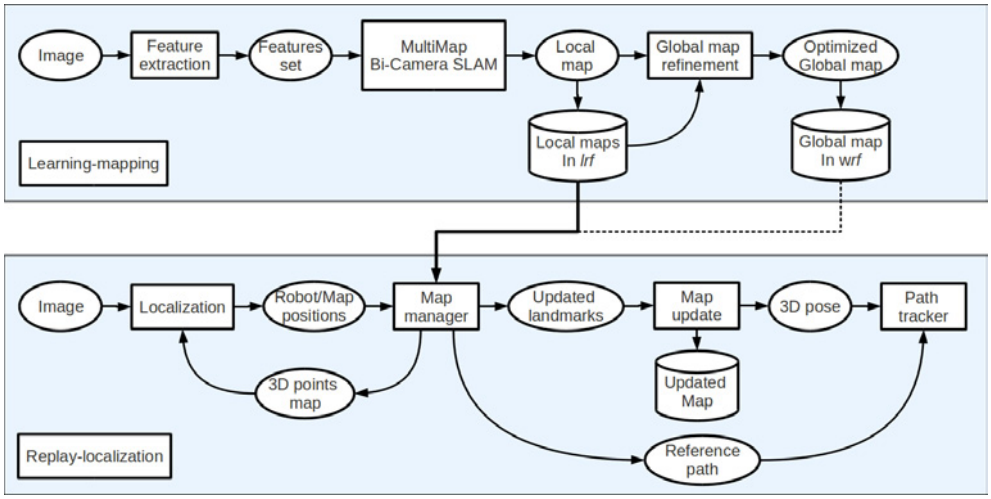


Figure 5.2: Learning and Replay: an overview of the major processing blocks in our proposed system.

The experiment was undertaken with the same robot but with different cameras between the learning (with Marlin cameras) and replay (with Flea2 cameras) steps, exploited in a Bi-Camera configuration. Moreover the experiment was undertaken with path learning and replay, done on different time, thus, the environment has changed between the two steps.

The robot follows the trajectory shown in Fig. 5.3. The main issues of this experiment are:

1. learn a safe path computing the learning-mapping approach,
2. replay the path autonomously using the replay-localization and the path tracker method.

3.1 Learning step

During the first step, the learning-mapping procedure was used to map $3D$-points landmarks and build the submaps. New local maps are created when 80 landmarks are in the map. In each submap, the poses and their uncertainties are expressed in the associated *lrf*. All submaps were merged in a final global map, expressed in *wrf* using the refinement of the global map procedure (see Fig. 5.5).

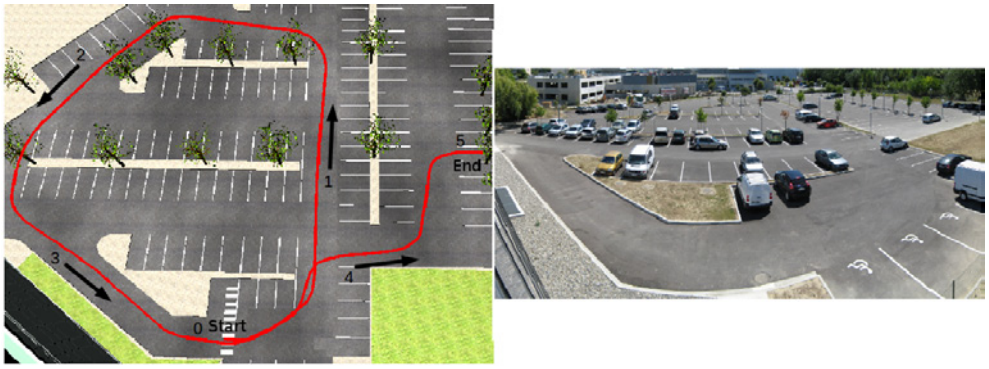


Figure 5.3: View of the learned and replayed trajectory in the experiment. The robot follows the trajectory described by the sequence 0-1-2-3-4-5.

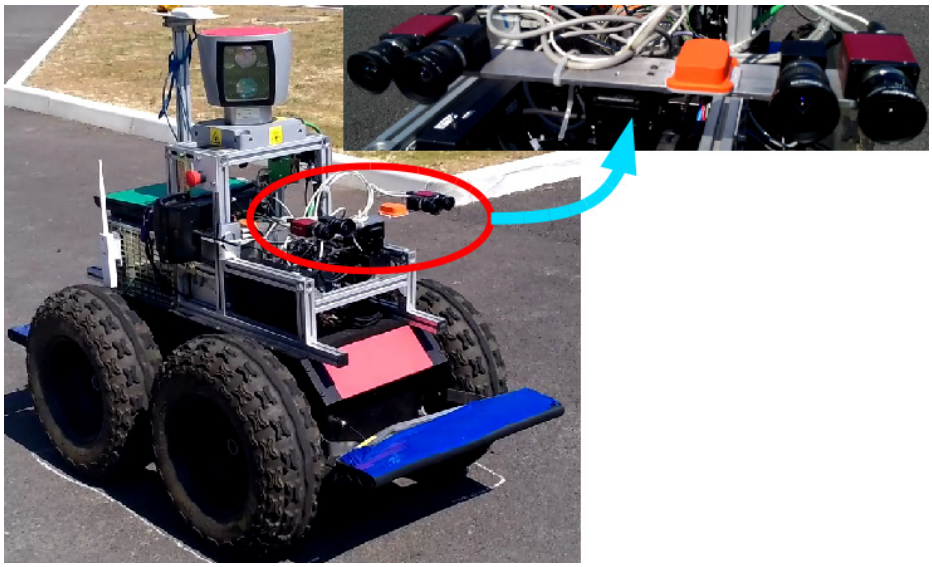


Figure 5.4: The Segway platform used in the experiment. Two independent stereo-vision benches. The first one made of two Marlin cameras (red cameras) with a baseline of 0.40 m. The second one made of two Flea2 cameras (black cameras) with a baseline of 0.30 m.

3.2 replay step

As a second step to test the replay-localization approach, the replay-robot must replay the path learned during the learning-mapping procedure. We can note that the initial position of the robot in the replay step is not exactly the same that the initial position in the learning step.

The $3D$ global map and the global path are loaded on the robot. In

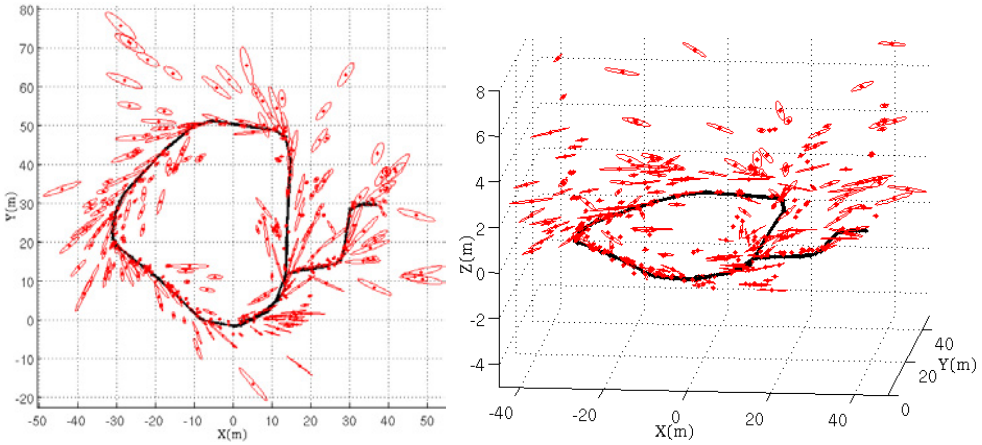


Figure 5.5: Learning and Replay experiment: $2D$ and $3D$ plots of the global map obtained by joining 67 local submaps using our hierarchical/hybrid BiCam SLAM approach with *refinement of the global map* procedure; red: covariance ellipses of the features; black: the estimated robot trajectory.

this point, the replay-robot, has a good knowledge of the $3D$ map of the environment, thus, the robot starts with a huge uncertainty and applies the algorithm of replay-localization.

The results obtained in this experiment[3], confirm the system performance in the learning-mapping procedure as well as the accuracy of the replay-localization along the path (Fig. 5.6). We can see that the replay step works in spite of bad matchings on landmarks occluded or lost, due to the environments changes (Fig. 5.7).

[3]A representative video of the experimental results can be seen at `http://homepages.laas.fr/dmarquez/maplaas`

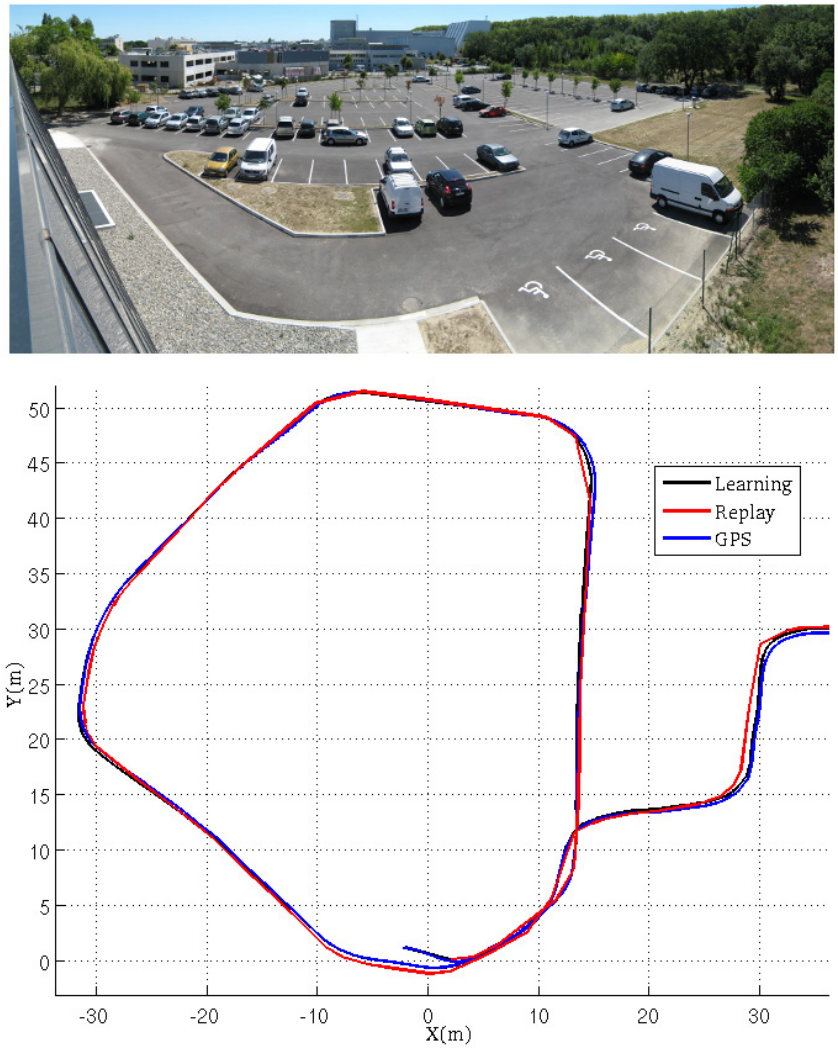


Figure 5.6: Learning and Replay experiment: (top) the real scenario: LAAS-CNRS parking lot, (down) robot trajectory estimate; black: the robot trajectory learned path; red: the robot trajectory replayed path; blue: the true robot trajectory (GPS).



Figure 5.7: Learning and Replay experiment: images acquired on the same area. (left) learning-mapping step; *magenta*: IDP landmarks; *dark red*: IDP landmarks not observed in the frame; *cyan*: euclidean landmarks; *dark blue*: euclidean landmarks not observed in the frame. (right) replay-localization step; *magenta*: initial projections of $3D$ points from the learned $3D$ map; *cyan*: projected and matched points using for localization.

4 Convoy Navigation

A second experiment using MORSE[4] was undertaken. The main issues of this experiment is to make a "Convoy" with two robots (leader and follower) where the leader applies the learning-mapping approach to build successive submaps, and to transmit online each submap once it is built, defining a path that the follower must stay on. The follower, waits for the first submap to start and applies the replay-localization approach to follow the trajectory, processing the information received (online) from the leader.

Thus the leader-robot start to explore the area and build the first local map, this first submap is the origin of the world. New local maps are created when 80 landmarks are in the map, but before starting a new local map, the submap and the local trajectory are expressed in *wrf* and transmitted to the follower-robot. Immediately after, the follower-robot pose is the new relative transformation in the global graph.

As results, 20 submaps were generated and sent from the leader-robot to the follower-robot. As is shown in Fig. 5.9 and Fig. 5.8, the follower was able to process each submap and apply the replay-localization approach to track and follow (online) the path defined by the leader.

The execution of the experiment is shown in Fig. 5.8. We can see the "convoy experiment" in progress where in left is showing the global view of the experiment scenario; Right-top shown the leader view, the learning-mapping approach is performed and the points in the map are expressed as *magenta* for IDP landmarks or *cyan* for euclidean landmarks. Right-down shown the follower view, the replay-localization step is executed, in this case, initial projections of $3D$ points from the learned $3D$ map represented in *magenta* and projected and matched points using for localization represented in *cyan* [5].

The results obtained in this experiment, confirm the system performance in the convoy task. We can see in Fig. 5.9, how the learned path is is effectively reexecuted.

[4] `http://morse.openrobots.org`

[5] A representative video of the experimental results in the convoy task can be seen at `http://homepages.laas.fr/dmarquez/maplaas`

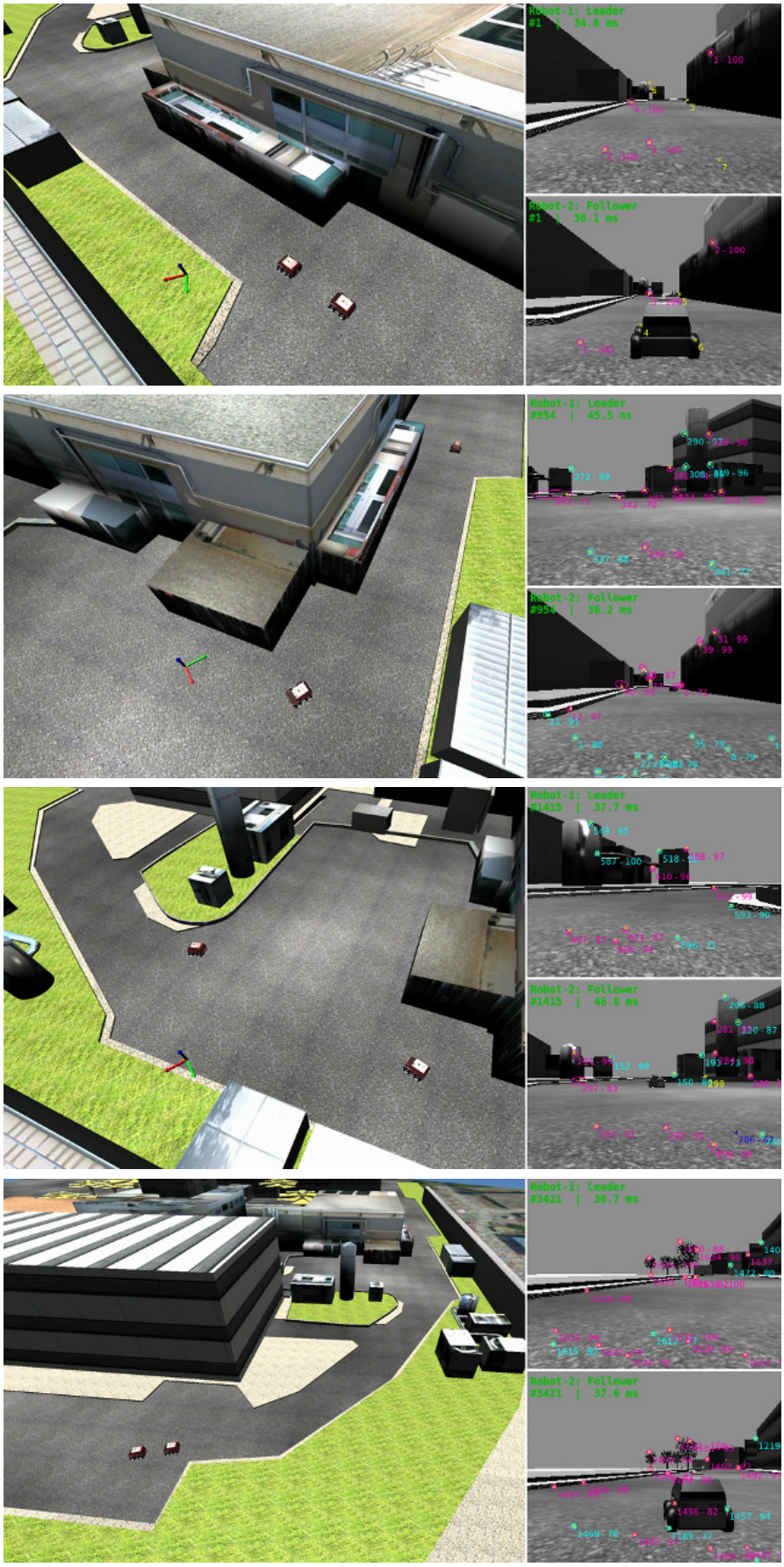


Figure 5.8: Convoy experiment.

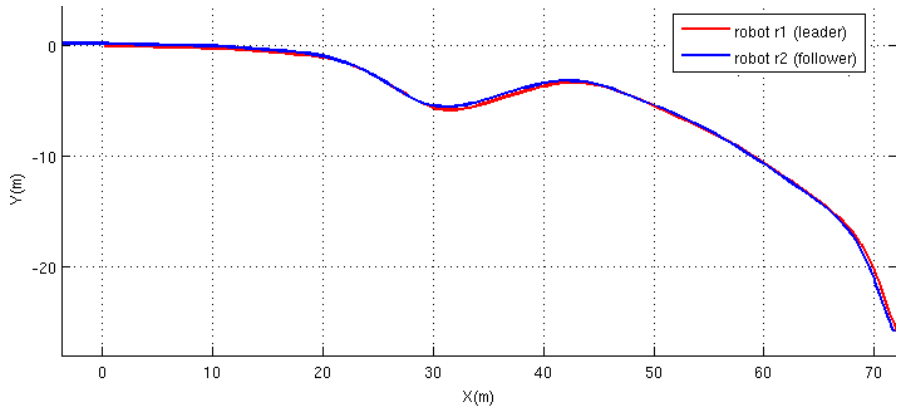


Figure 5.9: Convoy experiment: Robot trajectory estimate; blue: the robot trajectory learned path; red: the robot trajectory replayed path.

5 Active Visual-based Detection and Tracking of Moving Objects

Let us begin remembering our *Bi-Camera SLAM-DATMO* system [6], as can be seen in Figure 5.10.

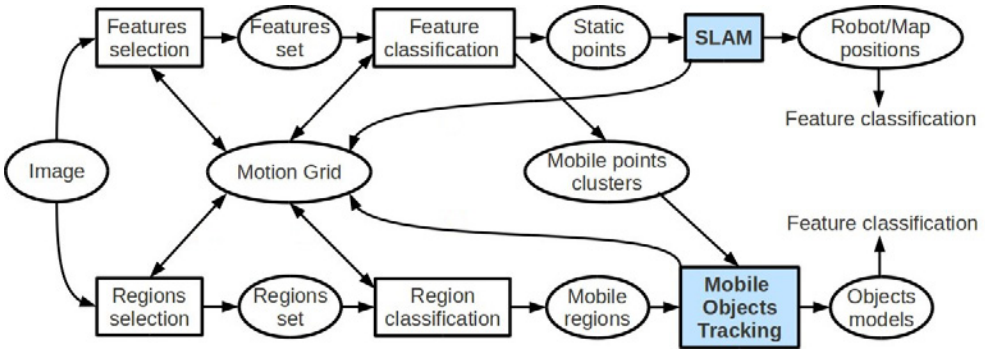


Figure 5.10: The major processing blocks in our SLAM-DATMO system and their connections, executed for each frame of a video sequence.

5.1 Indoor Experiment: 2D detection and tracking task

The robot with the two cameras looking forward is run in straight line inside the robotic lab at LAAS. A simple 2D odometry model is used for robot-motion predictions. Two moving objects are presented in the scene, first, a single pedestrian traversing from left to right appears and then a second pedestrian with a caddy traversing from right to left.

Figure 5.11, shows results for this experiment. The result showed that effectively moving objects are successfully detected clustered and tracked by the proposed approach. The left image of Figure 5.11 shows the bounding box on two moving objects, $\mathcal{O}_1$ which enters from the left side of the image and $\mathcal{O}_2$ on the right side; all detected points inside this box are used to initialize the object model. Object region could grow at each time when new clusters are detected, thanks to the cluster fusion mechanism. An example of this behavior is shown in the center image

[6] The concepts of exploration in dynamic environments were presented in Chapter 4.

of Figure 5.11, where we clearly see that the bounding box of $\mathcal{O}_2$ was enlarged.

5.2 Outdoor Experiment: Bi-Camera SLAM-DATMO

We tested our system in outdoor environment. For this experiment, the robot with the two cameras looking forward is run in the *Canal du Midi* in Toulouse-France. A simple 2D odometry model is used for robot-motion predictions.

Three moving objects can be observed ($\mathcal{O}_1$, $\mathcal{O}_2$, $\mathcal{O}_3$): three persons riding bicycle traversing the scene, moving in all directions and occluding each other.

Results fot this experiment are shown in Fig. 5.12 and Fig. 5.13. Example results for the *Motion Grid* performance are shown in Fig. 5.12. Note that the *Motion Grid* is in constant evolution. The highest probability will be in the centre of each moving zone belonging to a moving object.

Figure 5.13, shows the results for the complete system SLAM with MOT. As can be seen, the system is able to detect and track moving objects and to accurately predict their future motion in 3D space. The bounding boxes are colored to show objects identities. Below each image, we show the map in a overhead view, static landmarks used by the SLAM process are marked in blue.

In the first row, the three first images show the evolution of a moving object $\mathcal{O}_1$ (with bounding box yellow) which enters from the left side; in the first image of the third row, a second moving object $\mathcal{O}_2$ (with bounding box green) which enters from the right side is detected. The system ability to track through occlusion is demonstrated in the third row: note how the person entering from the left with bounding box magenta ($\mathcal{O}_3$), temporarily occludes the person entering from the right with bounding box green ($\mathcal{O}_2$). Still, the system manages to pick up the trajectory of the person on the right again.

The performance of robot localization, mapping and tracking of moving objects verify that the proposed Bi-Camera SLAM-DATMO approach

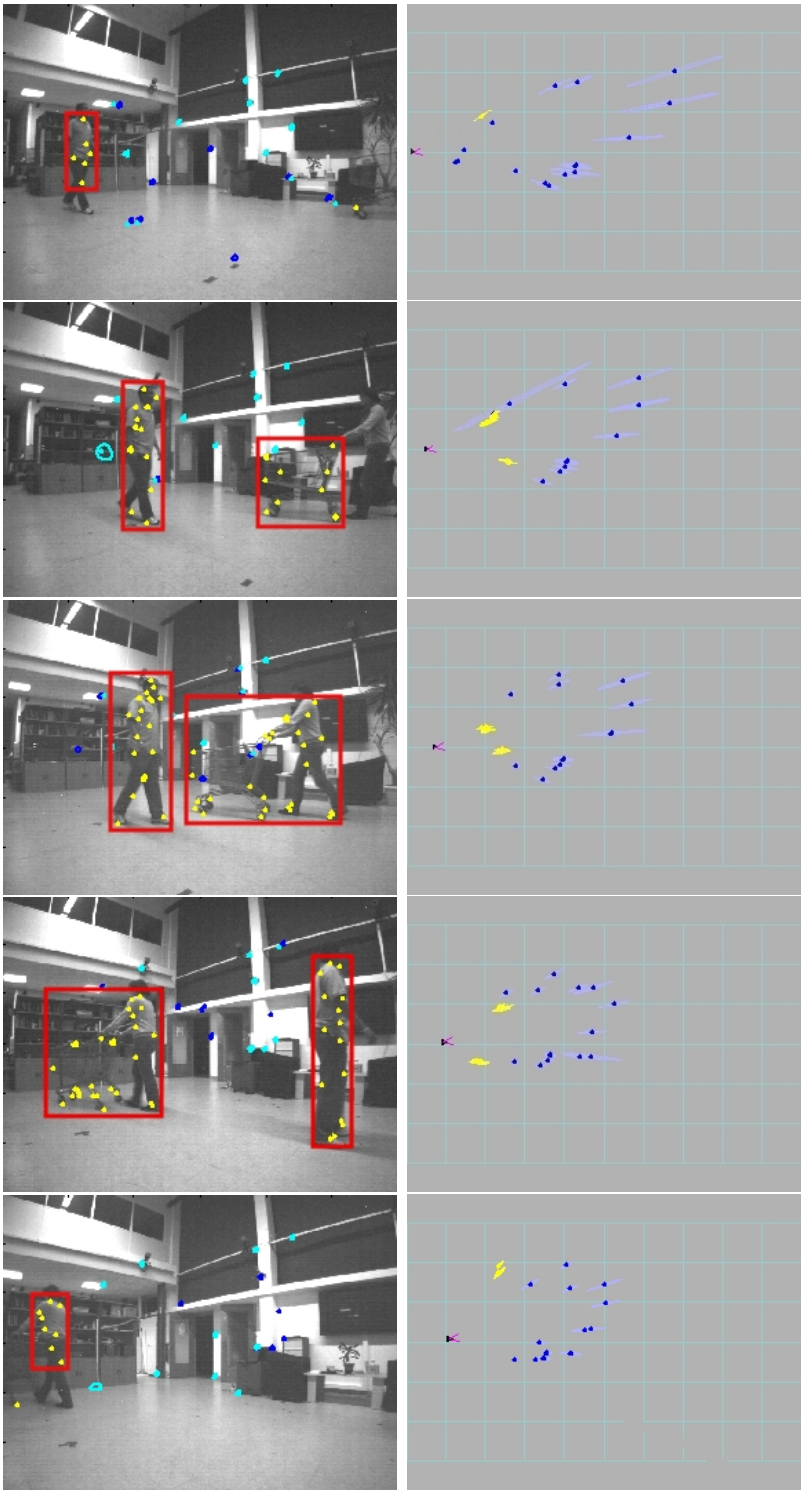


Figure 5.11: Detection clustering and tracking objects. In the image view, *cyan* and *blue* features are static points to be used by the SLAM process; *yellow* features are updated moving points; *red squares* are moving objects. The result show that two moving objects are detected and tracked simultaneously in 2D space.

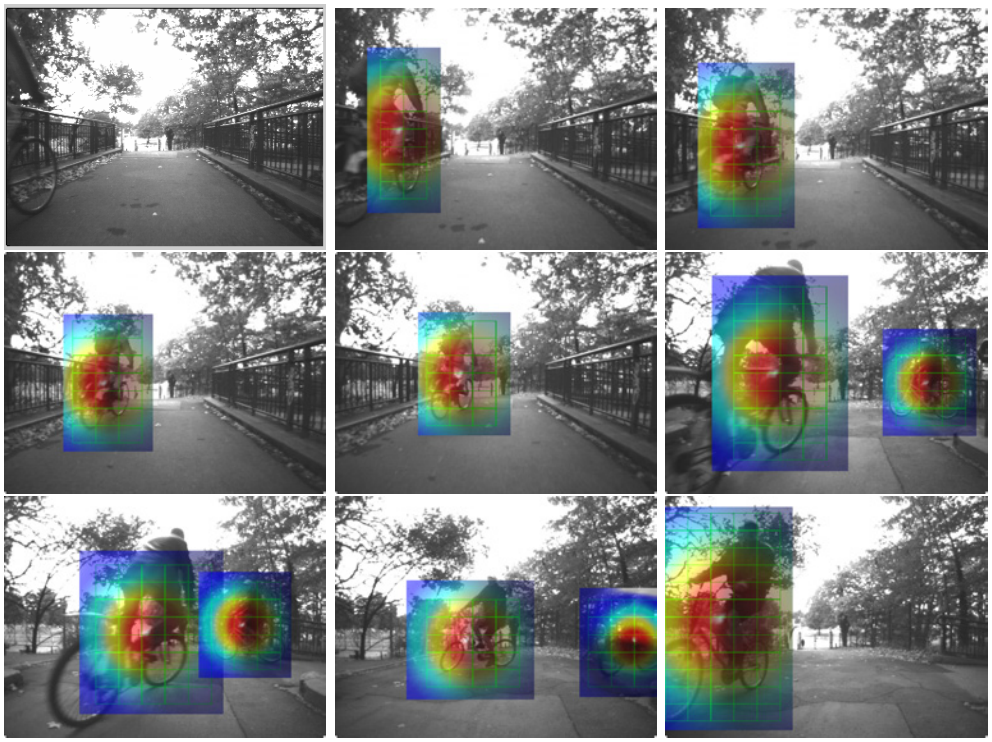


Figure 5.12: *Motion Grid* evolution for the sequence in the Outdoor Experiment.

is feasible in dynamic environments. Videos illustrating the results, can be can be seen at:

`http://homepages.laas.fr/dmarquez/slammot`

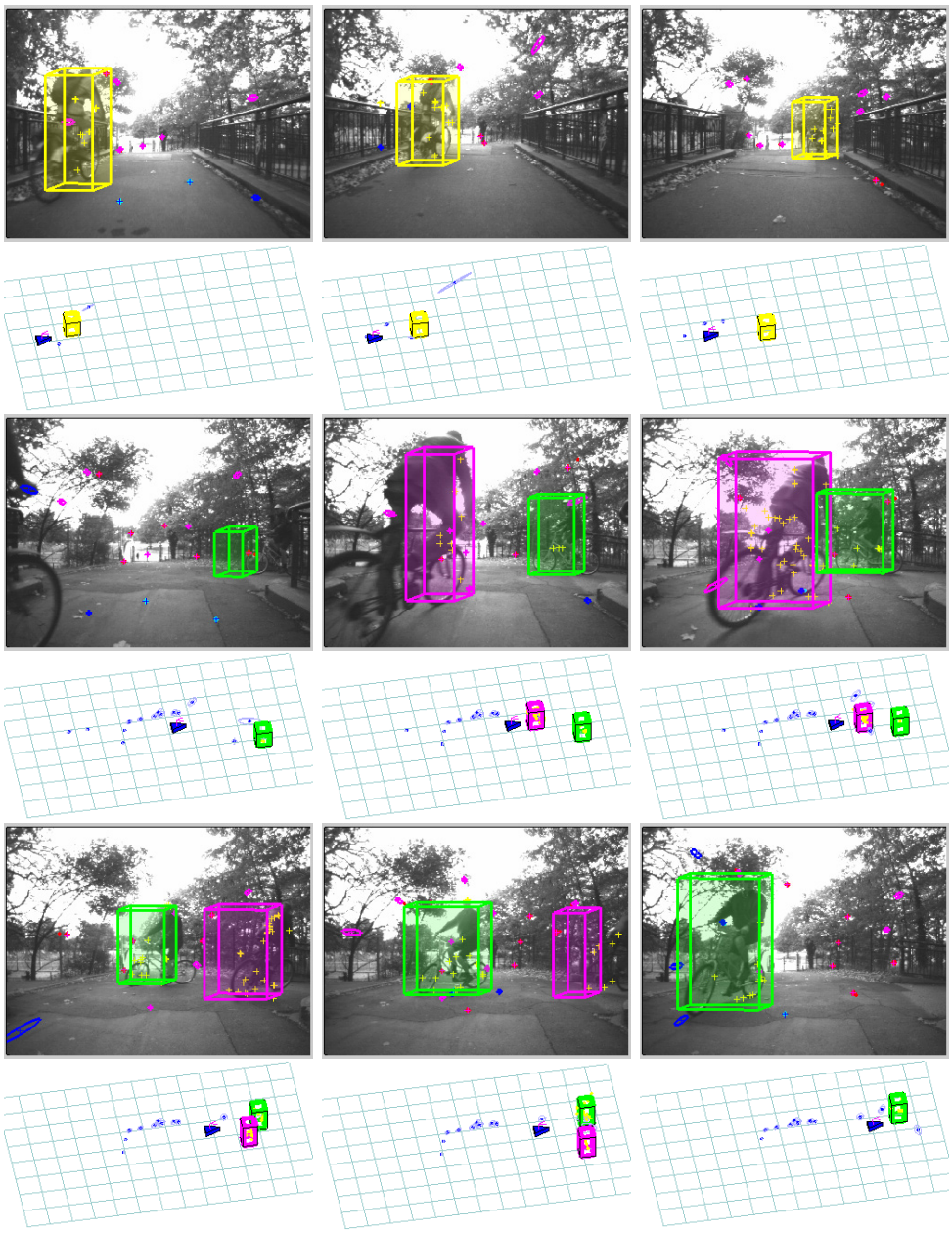


Figure 5.13: Outdoor experiment results: image plane and top views of the produced 3D map. In the image view, *red* features are updated IDP landmarks; *magenta* features are predicted IDP landmarks; *cyan* features are update euclidean landmarks; *blue* features are predicted euclidean landmarks; *yellow* features are updated moving points; *3D bounding box* are moving objects (yellow: $\mathcal{O}_1$; green: $\mathcal{O}_2$; magenta: $\mathcal{O}_3$). The moving objects has been successfully detected and tracked in 3D space as can be seen in the views.

Conclusions and Future Directions

6

An idea is a point of departure and no more. As soon as you elaborate it, it becomes transformed by thought.

Pablo Picasso

Contents

1 Conclusions 109

2 Future Directions 112

1 Conclusions

Research in the field of mobile robotics raises, at this time, a large number of major challenges. The development of intelligent vehicles that can interact and navigate autonomously in any environment indoor or outdoor, involves solving a number of open problems.

One of these fundamental problems, consists in the construction of reliable maps that allow the robot to navigate in the environment. If the robot has no external service, like RTK GPS, that provides information about its position, then the map construction poses a major challenge: The robot must be able to build a map while simultaneously use that

map to locate itself. Errors made by the robot in the map construction would cause an error in the estimation of the position and vice-versa.

The work in this thesis can be viewed from two different angles:

On one hand, we have been carried out a lot of work to obtain experimental data using real robotic platforms. The tuning of the robots (software/hardware), the installation of the cameras and other sensors, the calibration process, the capturing of images sequences, the creation of a communication system, among other tasks, have raised many practical integration problems. This "dirty" work and, in many cases, boring one, has been made by the author of this thesis. It is difficult to appreciate the number of hours spent on these tasks only reading the thesis manuscript.

On the other hand, in this thesis, we develop a knowledge base in the field of mobile robotics and computer vision. So we present a set of functional algorithms and software tools that allow autonomous navigation of robots based only on visual information. This can be summarized as follows:

First a static world is considered: we have designed and tested a navigation system in order to implement convoy navigation by robots that must navigate in dangerous unknown routes. So a system for learning-mapping of a long-range trajectory executed by a robot in an unknown outdoor environment has been proposed. The successive robot poses are computed fusing proprioceptive sensory data (odometer and gyro) with visual data acquired from two cameras mounted on the robot in a Bi-Camera configuration. A multi-map SLAM approach allows to express the robot poses with respect to local reference frames successively initialized along the trajectory; observations of visual landmarks are detected and matched in the image sequence. Robot and landmarks poses are computed and updated by a Bi-Camera SLAM based method.

When the robot reaches the end of the trajectory, an optimization step allows to refine both the trajectory and the landmarks map. All poses could be expressed either in the successive local reference frames,

or in a single one created at the first point of the trajectory.

The proposed approach facilitates the path planning strategies and the generalization of the system to a multi-vehicles SLAM. The trajectory is built for two possible applications. Either it will be executed latter by the same vehicle, typically by a transportation system executing autonomously this path, or it will be sent in real time to follower vehicles to cope with the navigation of a convoy along a path.

From the information provided by a leader robot after applying the learning-mapping procedure, other robots are able to navigate with this base of knowledge, so a replay-localization approach has been proposed for this purpose. This second step in our system allows to a follower robot: first compute its localization in the environment based in a known 3D-point map and second, apply a trajectory control method to maintain the follower on the learnt trajectory, in this way we obtain autonomous navigation. The 3D-point map previously created by the leader robot can be either a set of local maps or an optimized global map that allows learning-and-replay or a Convoy task.

The different mechanisms developed are illustrated and validated by experimental results, the algorithms have been integrated and executed on real robots.

Breaking with the assumption of a static world, we have presented a system to solve the SLAM with Detection and Tracking of Moving Objects problem (SLAM-DATMO) with the aid of vision. With this approach we are able to provide a description of the visually perceived dynamic scene: how the static world is, where the robot is in this world, and how other existing bodies move. A Clustering-Classification method in a probabilistic framework is proposed and described to detect static points, to be used by the SLAM algorithm and moreover to detect and cluster the moving ones in order to detect and track moving objects.

This system that implement interaction between SLAM and DATMO is able to cope reasonably well with dynamic environments, however this approach only deals with a tiny part of a bigger problem.

2 Future Directions

With the aim of developing intelligent systems for autonomous navigation this thesis raises several questions that are not directly addressed, numerous possible extensions of this work can be foreseen to achieve a more mature system.

Optimization-based methods, are in vogue in the SLAM community, an interesting extension, may be the implementation of one of these optimization methods, e.g. bundle adjustmen (BA), however it should be note that loop closing is not mandatory in our system.

The algorithm derived in Chapter 2, is explicitly designed to generate maps based points-landmarks. The mapping-learning procedure can be improved with the use of heterogeneous landmarks, e.g. segments-based landmarks, that will provide a much richer representation of the real world, moreover a representation that a human could understand.

Another extension will consist in extending the analysis of visual simultaneous localization and mapping in dynamic environments. How the robot can interact with a dynamic world in constant evolution? How the robot can use in an optimal way the visual information it receives? How the robot can share this information with other robots? maintaining real-time constraints.

As a general remark, I would like to mention that I am a strong believer that robotics research must be integrated in real applications. I am convinced that computer science is for people who want to change the world. Thus, one of my main goals is translate research into "real world" applications and advance the technology and knowledge to a new level. The end of this thesis, marks the beginning of a new stage in which I will continue studying and deepening in this fascinating world of robotics and computer vision. There are still many promising directions to explore.

Appendix A



The RINAVEC project

The project is motivated by development and evaluation of advanced Environment Modelling and Perception functions, for vehicles moving in convoy on an unknown route a priori, in an open environment (peri-urban or natural). The application context concerns humanitarian convoys navigation in war zones or rough terrains. For various reasons (mining, other vehicles or people), progression of such convoys is very difficult and dangerous. The aim is to assess recent progress in Perception for communicating vehicles navigation, in this very demanding context.

Each vehicle in the convoy is equipped with low-cost proprioceptive sensors (GPS, inertial) and cameras. The lead vehicle (leader) models the trajectory followed by the convoy to prevent any danger, other vehicles (followers) should take this path, with minimal deviation (see Fig. A.1).

The guidance of the leader vehicle is done manually (by teleoperation, for example), a real-time map and its trajectory reconstruction are extracted by SLAM-type approaches using data from two onboard cameras. The leader, records its trajectory in the form of successive positions, relative to fixed environment 3D landmarks, detected and tracked in the in motion acquired images. The leader also detects moving objects close to the route, and assesses their positions

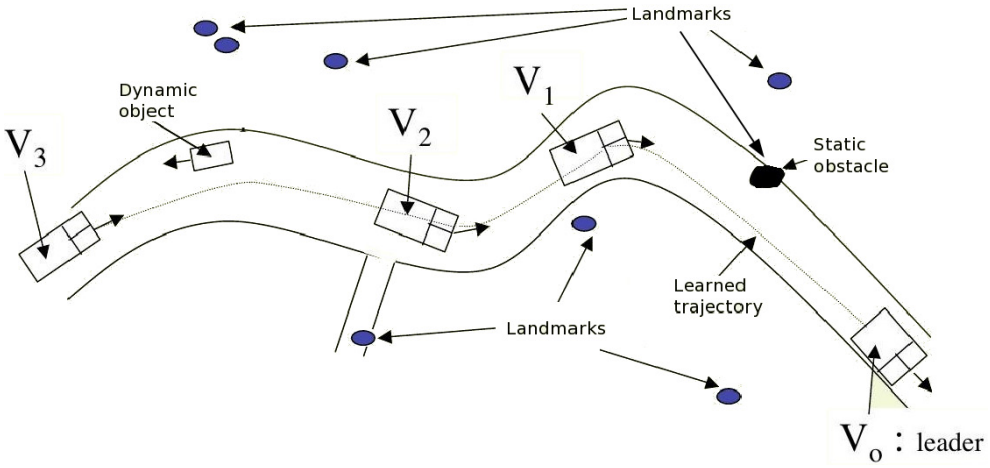


Figure A.1: The RINAVEC project. V_0 leader robot, V_1, V_2, V_3 followers robots.

and velocities states.

The map and the path thus constructed can be online transmitted (with a reduced bandwidth constraint) to following vehicles (in the case of a convoy for example) so they follow the exact same way. This information can also be stored and later exploited by a vehicle or a convoy, in a situation of delayed replay.

Trajectory, fixed landmarks positions, moving objects states are stored in a stochastic map, periodically sent to followers, in this way:

- Each uses it for self-localization relative to already received landmarks by the previous, and to correct its trajectory to meet the maximum deviation constraint.

- Each enriches the map according to its own observations. However this feature is available to assist in the relative localization, the reference map is not modified to avoid deviations.

The project first operates punctual landmarks reconstructed from points of interest (SIFT, Harris ...), although theoretical developments were made on the feasibility to use structured landmarks (lines,surfaces) corresponding to environment semantic entities (trees, curbs, panels,

building frontages) with the aim of reducing the amount of stored and transmitted information.

Several scenario variants are considered:

- spaced convoy (vehicles do not see each other): the trajectory is received and replayed by the followers
- tightened convoy (mutual localization), in that case, the followers improve their localization by taking into account the visual detection which they can make of the leader
- delayed replay of a route: the map is stored within every follower

The issues addressed in this project are numerous. The two principals are:

- Navigation should not depend on expensive proprioceptive sensors such as RTK GPS or high precision Inertial Measurement Unit. Replay of a trajectory learned by exploiting all such devices being in the state of the art, RINAVEC project suggests an approach based primarily on the exteroceptive perception.
- Vision is the only exteroceptive sensor operated, whether to build the map or to detect unexpected events, while in most existing projects, the vehicles are equipped with laser range finders or radar.

Experimental results

The site on which we conducted our experiments it is in the Yvelines department[1]. We can not disclose the name of the site or the GPS coordinates associated. However, we can provide satellite images of the place where we conduct the experiments and trajectories executed (see Fig. A.2.

Figure A.3 shows R-trooper and VELAC, the robotic vehicles used in the project. The demonstrator R-trooper, is an unmanned vehicle property of Thales Optronique and VELAC, is an experimental vehicle, property of LASMEA laboratory for the Driving Assistance.

[1]French department in the region of Ile-de-France



Figure A.2: Satellite view of the experiment site.

Experiment 1: Learning and replay

Figure A.4 shows the trajectory executed in this experiment

The Learning step is performed by a human operator at a speed of $1.5m/s$. On the RINAVEC web page[2], you will find a video called 7.1 representing the learning procedure.

Figure A.5 shows the trajectory estimate results of the learning-mapping procedure. Blue corresponds to the GPS RTK, red is the trajectory estimated by our Bi-Camera SLAM, green corresponds to the odometry. Black dots are euclidean points on the map, red dots are IDP points.

A single map has been generated. The trajectory of the slam is converted into readable file path in the management system trajectory R-trooper. The robot is repositioned on the starting point and the replay

[2]http://www.lasmea.univ-bpclermont.fr/Rinavec/SiteWebRinavec/VIDEOS.html



Figure A.3: R-trooper (left) and VELAC (rigth); the robotics vehicles used in the project.



Figure A.4: Trajectory executed in the learning and replay experiment.

algorithm is enabled. The robot starts the replay when the operator gives hand.

On the RINAVEC web page[3], you will find a video called 7.2 representing the replay procedure.

Figure A.6 shows the trajectory estimate results of the replay procedure. Blue corresponds to the GPS RTK, black is the trajectory estimated by the Bi-Camera SLAM in the learning step. It has become the reference path to replay. Green trajectory corresponds to odometry (it is not used during the replay, except in case of loss of localization estimated by the replay algorithm). Red is the trajectory of the replay. We can

[3]http://www.lasmea.univ-bpclermont.fr/Rinavec/SiteWebRinavec/VIDEOS.html

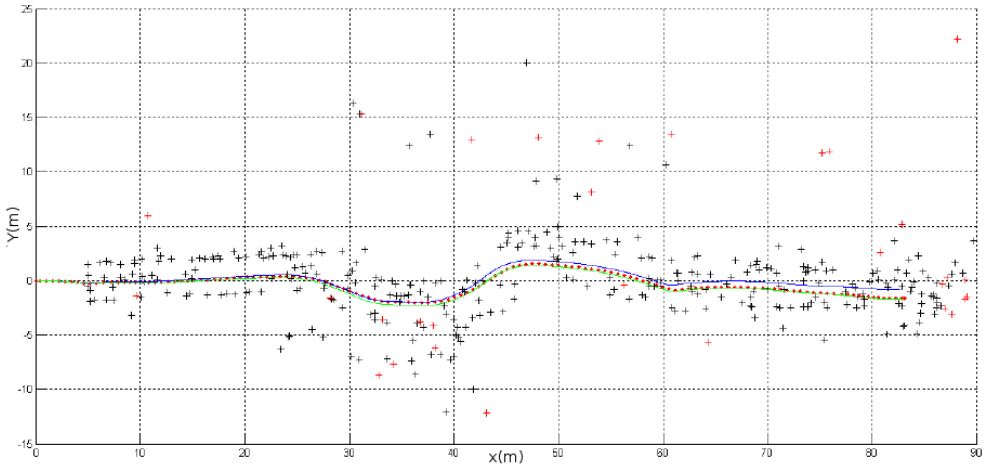


Figure A.5: Trajectory estimation in the learning step and 3D-points map.

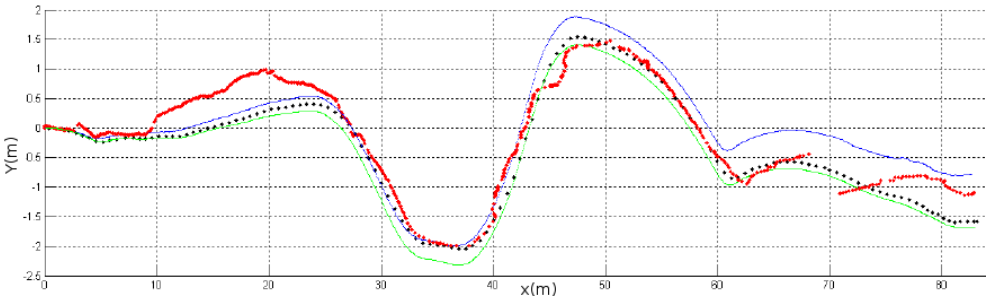


Figure A.6: Trajectory estimate results of the replay procedure.

notice a brief loss of localization 70 meters on the axis X. So the robot is switched to the position provided by the odometry until find and project points to used with the replay algorithm.

Experiment 2: Updating the learn map

The use of maps (global or local maps) generated by the leader allows the replay as many times as desired. However, an additional process allows update a map whose environment has changed (replay after a long period).

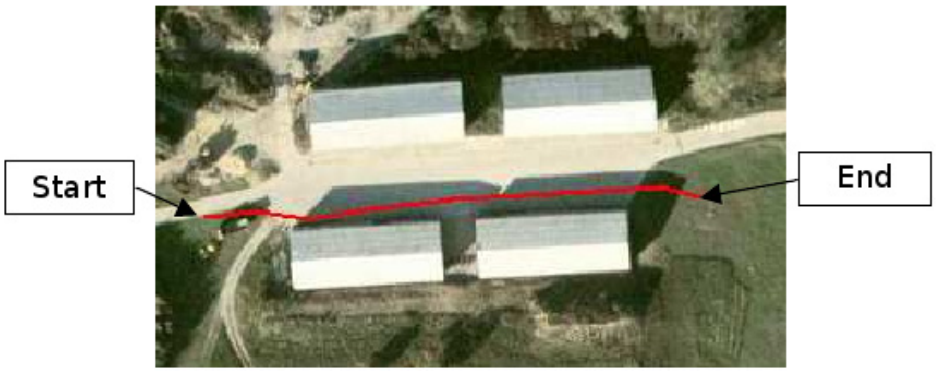


Figure A.7: Trajectory executed in the replay step.

The Updating map process is described as follows: During replay step, the patch associated with each 3D point assumed observable is actively search. The location of this patch ensures that the environment has not changed. If a patch is not localized with sufficient certainty, it is removed from the map giving way to a new pair point/patch in the same region of interest. The map is then updated.

Figure A.7, shows the trajectory executed in the experiment.

Figure A.8, shows a view of the terrain travelled in the learning step.



Figure A.8: View of the terrain travelled in the learning step.

Figure A.9, shows the map and the trajectory estimated results of the learning step. Red is the trajectory estimated and black dots are the points of the map.

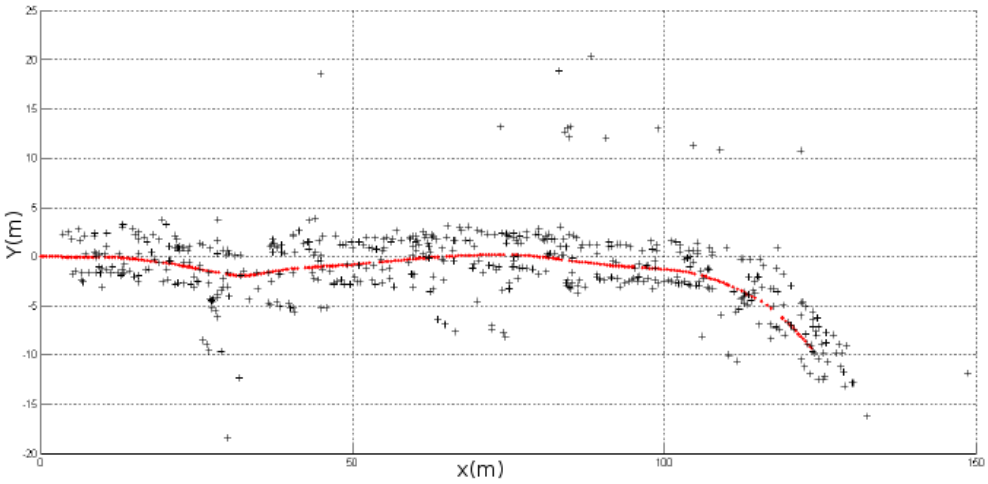


Figure A.9: Map and trajectory estimated in the learning step.

Figure A.10, shows the terrain travelled in the relay step. Note how the environment has changed compared with learning step (Figure A.8)



Figure A.10: The terrain travelled in the replay step.

The resulting map and trajectory in the replay step is shown in Fig-

ure A.11. We can see as new landmarks are added to the initial map. Red is the estimated trajectory; black dots are the points of the map.

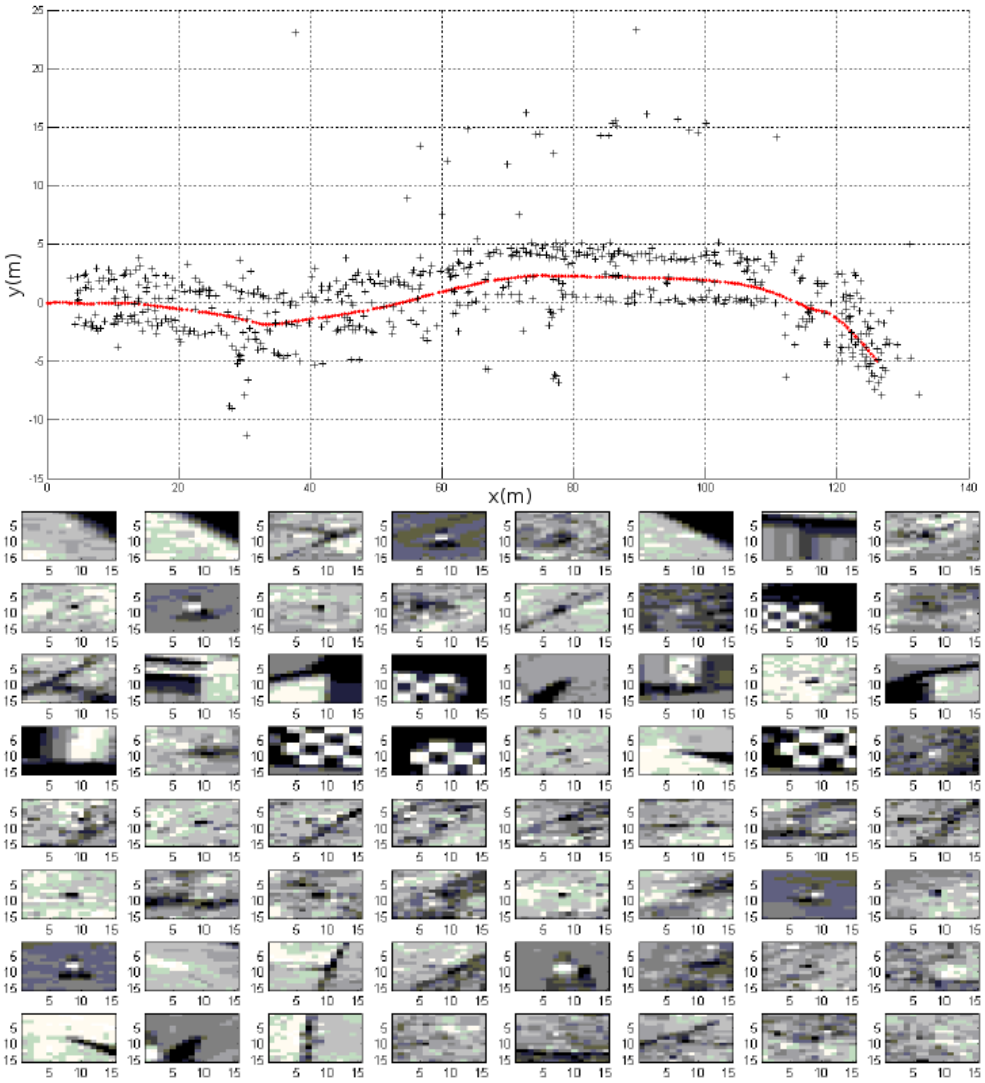


Figure A.11: (top) Map updated in the learning step; (bottom) A sample of the new patchs added to the map.

Experiment 3: Detecting the leader vehicle

The leader vehicle detection is not necessary or required for the operation of the convoy. However, this information can be useful to refine the localization of followers vehicles.

In the framework of the project, we have implemented an approach based on the AdaBOOST algorithm. A template-based tracking combined with a light detection procedure, allows us to detect and track the leader vehicle and calculate its position. The principle is shown in Figure A.12

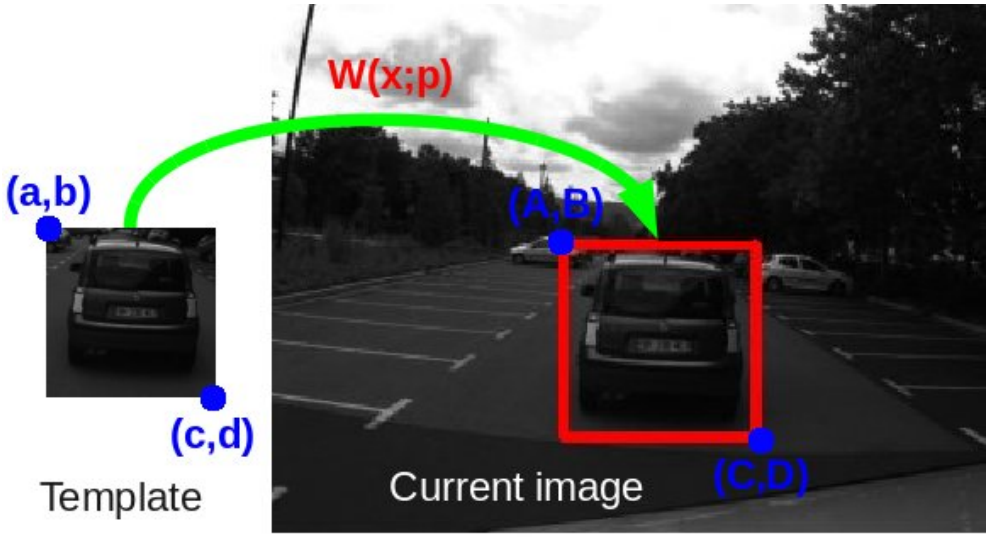


Figure A.12: Template-Based Tracking.

Figure A.13, shows preliminary results in a standard situation for the leader detection task. Note that only the leader vehicle is detected and their relative position is estimated based in the light detection. Special care has been taken to make the system robust, by minimizing the false negatives and filtering out the false positives using temporal consistency. Finally the system runs in real-time on ordinary hardware using as input a standard video camera.

Discussions

The experimental stage, allowed us to feel the difficulty of obtaining a replay with an accuracy of $10cm$. Moreover, despite the available processing power, we had to reduce our speed to adjust the ratio processed image/distance between two images in order to not to degrade the local-



Figure A.13: Representative snapshots of the sequence. In the image view, *red square* is the leader vehicle detection; *blue circles* are pairs of features-light detection. The leader vehicle has been successfully detected and tracked. The pose estimation is based in the pairs of light detection.

ization. In the development of experiments, we have confronted with a lot of problems related to software integration, as it was the first time that the algorithms were tested in real conditions. Note that when we mention software integration we mean the integration of the learning algorithm developed by LAAS, the replay algoritm developed by LASMEA to be integrated into the RTmaps[4] platform of Thales.

[4]Rtmaps (Real Time Multisensor Advanced Prototyping Software) by Intempora

Appendix B

B

Software Development

In the course of this thesis, I had the opportunity to participate together with other members of the LAAS robotic group in the following software development. This software was used to validate the algorithms presented in the previous chapters.

An EKF-SLAM toolbox for MATLAB

This toolbox performs 6DOF SLAM (Simultaneous Localization And Mapping) using the classical EKF implementation. It is conceived as an "active-search" SLAM. It is provided for free under the GPL license.

It can be downloaded from:

`https://github.com/damarquezg/SLAMTB/`

`http://www.joansola.eu/JoanSola/eng/toolbox.html`

SLAM toolbox presentation

In a typical SLAM problem, one or more robots navigate an environment, discovering and mapping landmarks on the way by means of their on-board sensors. Observe in Fig. 1 the existence of robots of different kinds,

carrying a different number of sensors of different kinds, which gather raw data and, by processing it, are capable of observing landmarks of different kinds. All this variety of data is handled by the present toolbox in a way that is quite transparent.

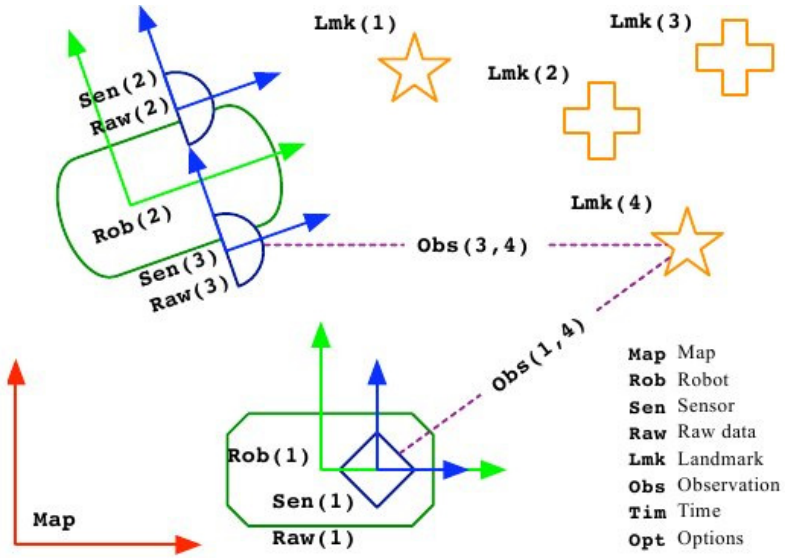


Figure B.1: Overview of the SLAM problem with the principal data structures..

In this toolbox, we organized the data into three main groups: SLAM, Simulator and Graphics. The first group contains the objects of the SLAM problem itself, as they appear in Figure B.1. A second group contains objects for simulation. A third group is designated for graphics output.

Apart from the data, we have the functions. Functions are organized in three levels, from most abstract and generic to the basic manipulations. The highest level, called High Level, deals exclusively with the structured data we mentioned just above, and calls functions of an intermediate level called the Interface Level. The interface level functions split the data structures into more mathematically meaningful elements, check objects types to decide on the applicable methods, and call the basic functions that constitute the basic level, called the Low Level Library.

RT-SLAM: Real-time EKF-SLAM in C++

RT-SLAM is a fast Slam (Simultaneous Localization And Mapping) library and test framework based on EKF. It is provided for free under the GPL license.

It can be downloaded from:

`http://www.openrobots.org/wiki/rtslam/`

`https://github.com/damarquezg/rtslam/`

RT-SLAM presentation

This library implements real-time single-map EKF-SLAM based on an active-search architecture. It supports the coexistence of multiple robots, multiple sensors, and multiple landmark types. It is initially conceived for vision sensors (cameras), although this is not a constraint.

Its main qualities are:

- Genericity: for sensor models, landmark types, landmark models, landmarks reparametrization, biases estimation;
- Speed: real time at 60 fps (VGA, gray level) on a decent machine (at least one Core2 core at 2.2GHz)
- Flexibility: different estimation/image processing sequencing strategies (active search), independent base brick for a hierarchical multimap and multirobots architecture;
- Robustness: near-optimal repartition of landmarks, data association errors detection (gating, ransac);
- Developer-friendly: visualization tools (2D and 3D), offline replay step by step, logs, simulation.
- Open source: in C++ for Linux and MacOS

For now it provides:

- Landmarks: Anchored Homogeneous Points (Inverse Depth) that can be reparametrized into Euclidean Points;

- Sensors: Pinhole cameras;
- Prediction: Constant velocity model, inertial sensor, odometry;
- Data association: Active search, 1-point Ransac, and mixed strategies.

Bibliography

[Agrawal *et al.* 2005] M Agrawal, K Konolige and L Iocchi. *Real-time detection of independent motion using stereo.* In Proceedings IEEE workshop on visual motion, 2005.

[Aguiar & Hespanha 2007] A. Pedro Aguiar and Joao P. Hespanha. *Trajectory-tracking and pathfollowing of underactuated autonomous vehicles with parametric modeling uncertainty.* IEEE Transactions on Automatic Control, vol. 52, no. 8, August 2007.

[Almanza-Ojeda *et al.* 2011] Dora Luz Almanza-Ojeda, Michel Devy and Ariane Herbulot. *Active Method for Mobile Object Detection from an Embedded Camera, Based on a Contrario Clustering.* In Informatics in Control, Automation and Robotics, volume 89, pages 267–280. 2011.

[Ayache & Faugeras 1988] N. Ayache and O.D. Faugeras. *Building, registrating, and fusing noisy visual maps.* Int. J. Rob. Res., vol. 7, no. 6, pages 45–65, dec 1988.

[Bailey *et al.* 2006] T. Bailey, J. Nieto, J. Guivant, M. Stevens and E. Nebot. *Consistency of the EKF-SLAM Algorithm.* In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pages 3562 –3568, oct. 2006.

[Balch & Arkin 1998] T. Balch and R.C. Arkin. *Behavior-based formation control for multirobot teams.* Robotics and Automation, IEEE Transactions on, vol. 14, no. 6, pages 926 –939, dec. 1998.

[Blanco *et al.* 2009] JL Blanco, J. González and J.A. Fernández-Madrigal. *Subjective local maps for hybrid metric-topological SLAM.* Robotics and Autonomous Systems, vol. 57, no. 1, pages 64–74, 2009.

[Cadena & Neira 2010] C. Cadena and J. Neira. *SLAM in O(log n) with the Combined Kalman-Information Filter.* Robotics and Autonomous Systems, vol. 58, no. 11, pages 1207–1219, 2010.

[Castellanos *et al.* 2004] J. A. Castellanos, J. Neira and J. D. Tardós. *Limits to the consistency of EKF-based SLAM.* In 5th IFAC Symp. on Intelligent Autonomous Vehicles, IAV'04, Lisbon, Portugal, 2004.

[Chatila & Laumond 1985] R. Chatila and J.P. Laumond. *Position referencing and consistent world modeling for mobile robots.* In IEEE Int. Conf. on Robotics and Automation (ICRA'85), pages 138–145, March 1985.

[Civera *et al.* 2007] Javier Civera, Andrew J. Davison and J. M. M. Montiel. *Inverse Depth to Depth Conversion for Monocular SLAM.* In International Conference on Robotics and Automation, pages 2778–2783, 2007.

[Civera *et al.* 2008] J. Civera, A. Davison and J. Montiel. *Inverse depth parametrization for monocular SLAM.* IEEE Trans. on Robotics, vol. 24, 2008.

[Civera *et al.* 2009] Javier Civera, Oscar G. Grasa, Andrew J. Davison and J. M. M. Montiel. *1-point RANSAC for EKF-based structure from motion.* In Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems, pages 3498–3504, 2009.

[Crowley & Crowley 1989] James L. Crowley and James L. Crowley. *World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging.* In IEEE Int. Conf. on Robotics and Automation (ICRA'89), pages 674–680, 1989.

[Cummins & Newman 2008] M. Cummins and P. Newman. *FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance.* Int. Journal of Robotics Research, vol. 27(6), pages 647–665, 2008.

[Dalal & Triggs 2005] Navneet Dalal and Bill Triggs. *Histograms of Oriented Gradients for Human Detection.* In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), pages 886–893, Washington, DC, USA, 2005.

[Davison *et al.* 2007] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton and Olivier Stasse. *MonoSLAM: Real-Time Single Camera SLAM.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 6, pages 1052–1067, jun 2007.

[Davison 2005] A.J. Davison. *Active search for real-time vision.* In Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on, volume 1, pages 66 – 73 Vol. 1, oct. 2005.

[De Luca *et al.* 1998] A. De Luca, G. Oriolo and C. Samson. *Feedback control of a nonholonomic car-like robot.* In J.-P. Laumond, editeur, Robot motion, planning and control, Lecture Notes in Control and Information Sciences 229. Springer, 1998.

[Delsart & Fraichard 2010] V. Delsart and T. Fraichard. *Tiji, a generic trajectory generation tool for motion planning and control.* In IEEE-RSJ Int. Conf. on Intelligent Robots and Systems, pages 1439–1444, Oct 2010.

[Di Stefano *et al.* 2005] Luigi Di Stefano, Stefano Mattoccia and Federico Tombari. *ZNCC-based template matching using bounded*

partial correlation. Pattern Recogn. Lett., vol. 26, no. 14, pages 2129–2134, oct 2005.

[Durrant-Whyte 1988] H. F. Durrant-Whyte. *Uncertain geometry in robotics.* Robotics and Automation, IEEE Journal of, vol. 4, no. 1, pages 23–31, feb 1988.

[Estrada *et al.* 2005] C. Estrada, J. Neira and J.D. Tardós. *Hierarchical SLAM: Real-time accurate mapping of large environments.* IEEE Trans. on Robotics, vol. 21(4), pages 588–596, 2005.

[Eustice *et al.* 2006] R.M. Eustice, H. Singh and J.J. Leonard. *Exactly sparse delayed-state lters for view-based SLAM.* IEEE Trans. on Robotics, vol. 22(6), pages 1100–1114, 2006.

[Féraud 2011] Thomas Féraud. *Rejeu de chemin et localisation monoculaire : application du Visual SLAM sur carte peu dense en environnement extérieur contraint.* PhD thesis, Université Blaise Pascal - Clermont II, 2011.

[Franchi *et al.* 2009] A. Franchi, L. Freda, G. Oriolo and M. Vendittelli. *The Sensor-based Random Graph Method for Cooperative Robot Exploration.* Mechatronics, IEEE/ASME Transactions on, vol. 14, no. 2, pages 163 –175, april 2009.

[Frese 2006] Udo Frese. *Treemap: An O(log n) algorithm for indoor simultaneous localization and mapping.* Autonomous Robots, vol. 21, pages 103–122, 2006.

[Freund & Schapire 1996] Yoav Freund and Robert E. Schapire. *Experiments with a New Boosting Algorithm.* In International Conference on Machine Learning, pages 148–156, 1996.

[Friedman *et al.* 2000] Jerome Friedman, Trevor Hastie and Robert Tibshirani. *Additive Logistic Regression: a Statistical View of Boosting.* Annals of Statistics, vol. 28, no. 2, pages 337–407, 2000.

[Furgale & Barfoot 2010] Paul Furgale and Timothy D. Barfoot. *Visual teach and repeat for long-range rover autonomy.* J. Field Robot., vol. 27, no. 5, pages 534–560, Septembre 2010.

[Gate *et al.* 2009a] G. Gate, A. Breheret and F. Nashashibi. *Centralised Fusion for Fast People Detection in Dense Environments.* In IEEE Int. Conf. on Robotics Automation (ICRA), Kobe, Japan, 2009.

[Gate *et al.* 2009b] G. Gate, A. Breheret and F. Nashashibi. *Fast Pedestrian Detection in Dense Environment with a Laser Scanner and a Camera.* In Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th, pages 1 –6, april 2009.

[Gibson *et al.* 2009] H. Gibson, H. Shoudong and D. Gamini. *3D I-SLSJF: A consistent sparse local submap joining algorithm for building large-scale 3D Map.* In Proceedings of the 48th IEEE Conference on Decision and Control, pages 6040–6045, Dec 2009.

[Goedeme *et al.* 2007] Toon Goedeme, Marnix Nuttin, Tinne Tuytelaars and Luc Van Gool. *Omnidirectional Vision Based Topological Navigation.* International Journal of Computer Vision, vol. 74, pages 219–236, 2007.

[Ila *et al.* 2011] V. Ila, J.M. Porta and J. Andrade-Cetto. *Amortized constant time state estimation in Pose SLAM and hierarchical SLAM using a mixed Kalman-information filter.* Robotics and Autonomous Systems, vol. 59, no. 5, pages 310–318, 2011.

[Kaess *et al.* 2008] M. Kaess, A. Ranganathan and F. Dellaert. *iSAM: Incremental Smoothing and Mapping.* IEEE Trans. on Robotics (TRO), vol. 24, no. 6, pages 1365–1378, Dcembre 2008.

[Klein & Murray 2007] Georg Klein and David Murray. *Parallel Tracking and Mapping for Small AR Workspaces.* In Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07), Nara, Japan, November 2007.

[Leonard & Durrant-Whyte 1991] J.J. Leonard and H.F. Durrant-Whyte. *Simultaneous map building and localization for an autonomous mobile robot.* In Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on, pages 1442 –1447 vol.3, nov 1991.

[Lin & Wang 2010] Kuen-Han Lin and Chieh-Chih Wang. *Stereo-based Simultaneous Localization, Mapping and Moving Object Tracking.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan, October 2010.

[Madhavan *et al.* 2004] Raj Madhavan, Kingsley Fregene and Lynne E. Parker. *Distributed Cooperative Outdoor Multirobot Localization and Mapping.* Autonomous Robots, vol. 17, pages 23–39, 2004.

[Mariottini *et al.* 2005] G.L. Mariottini, G. Pappas, D. Prattichizzo and K. Daniilidis. *Vision-based Localization of Leader-Follower Formations.* In Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on, pages 635 – 640, dec. 2005.

[Márquez-Gámez & Devy 2010] David Márquez-Gámez and Michel Devy. *Simultaneous trajectory learning and world mapping from a mobile robot in outdoor environment, using heterogeneous landmarks.* In Congrès des doctorants EDSYS 2010, Toulouse, France, 2010.

[Márquez-Gámez & Devy 2011a] David Márquez-Gámez and Michel Devy. *Visual modeling from a mobile robot: from image sequences to hierarchical maps and robot trajectories in outdoor environments.* In Inter-American Congress of Computing Applied to the Process Industry (CAIP 2011), Girona, Spain, 30 May - 3 June 2011.

[Márquez-Gámez & Devy 2011b] David Márquez-Gámez and Michel Devy. *Visual Navigation of Communicating Vehicles in Unknown*

and Changing Environment. In IEEE International Conference on Electronics, Circuits, and Systems (ICECS 2011), Beirut, Lebanon, 11-14 December 2011.

[Márquez-Gámez & Devy 2012a] David Márquez-Gámez and Michel Devy. *Active Visual-Based Detection and Tracking of Moving Objects from Clustering and Classification Methods*. In Advanced Concepts for Intelligent Vision Systems, volume 7517 of *Lecture Notes in Computer Science*, pages 361–373. Springer Berlin / Heidelberg, 2012.

[Márquez-Gámez & Devy 2012b] David Márquez-Gámez and Michel Devy. *SLAM visuel avec dtection et suivi d'objets mobiles par une approche de segmentation/classification*. In Reconnaissance des Formes et Intelligence Artificielle (RFIA 2012), Lyon, France, January 2012.

[Márquez-Gámez & Devy 2012c] David Márquez-Gámez and Michel Devy. *Visual trajectory learning and following in unknown routes for autonomous navigation*. In IEEE/RSJ IROS Workshop on Planning, Perception and Navigation for Intelligent Vehicles, Vilamoura, Algarve, Portugal, October 2012. (to appear).

[Marshall *et al.* 2008] Joshua Marshall, Timothy Barfoot and Johan Larsson. *Autonomous underground tramming for center-articulated vehicles*. Journal of Field Robotics, vol. 25, no. 6-7, pages 400–421, 2008.

[Migliore *et al.* 2009] Davide Migliore, Roberto Rigamonti, Daniele Marzorati, Matteo Matteucci and Domenico Sorrenti. *Use a Single Camera for Simultaneous Localization And Mapping with Mobile Object Tracking in dynamic environments*. In In proceedings of International workshop on Safe navigation in open and dynamic environments application to autonomous vehicles, May 2009.

[MobilEye Vision Technologies LTD] MobilEye Vision Technologies LTD. Jerusalem, Israel. `http://www.mobileye.com/`.

[P. Newman & Neira 2002] J. Tardós P. Newman J. Leonard and J. Neira. *"Explore and Return: Experimental Validation of Real-Time Concurrent Mapping and Localization"*. In IEEE Int. Conf. on Robotics and Automation, pages pp. 1802–1809., Washington DC, USA, novi-15 May 2002. ICRA'2002.

[Paz *et al.* 2007] L.M. Paz, P. Jensfelt, J.D. Tardós and J. Neira. *EKF SLAM updates in O(n) with Divide and Conquer SLAM*. In Proc. IEEE Int. Conf. on Robotics and Automation, Rome, Italy, 2007.

[Roussillon *et al.* 2011] Cyril Roussillon, Aurélien Gonzalez, Joan Solà, Jean Marie Codol, Nicolas Mansard, Simon Lacroix and Michel Devy. *RT-SLAM: a generic and real-time visual SLAM implementation*. In Int. Conf. on Computer Vision Systems (ICVS), Sophia Antipolis, France, Sept. 2011.

[Royer *et al.* 2007] Eric Royer, Maxime Lhuillier, Michel Dhome and Jean-Marc Lavest. *Monocular Vision for Mobile Robot Localization and Autonomous Navigation*. Int. J. Comput. Vision, vol. 74, no. 3, pages 237–260, sep 2007.

[Segvic *et al.*] Sinisa Segvic, Anthony Remazeilles, Albert Diosi and Franois Chaumette. *A mapping and localization framework for scalable appearance-based navigation*. Computer Vision and Image Understanding, vol. 113, no. 2.

[Smith & Cheeseman 1987] R.C. Smith and P. Cheeseman. *On the representation and estimation of spatial uncertainly*. Int. Journal of Robotics Research, vol. 5(4), pages 56–68, 1987.

[Smith *et al.* 1990] R. Smith, M. Self and P. Cheeseman. *Autonomous robot vehicles*. chapitre Estimating uncertain spatial relationships in robotics, pages 167–193. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[Solà *et al.* 2008] J. Solà, A. Monin, M. Devy and T. Vidal-Calleja. *Fusing monocular information in multi-camera SLAM*. IEEE Trans. on Robotics, vol. 24, no. 5, pages 958–968, 2008.

[Solà 2007] J. Solà. *Towards Visual Localization, Mapping and Moving Objects Tracking by a Mobile Robot: a Geometric and Probabilistic Approach.* PhD thesis, Institut National Polytechnique de Toulouse, 2007.

[Strasdat *et al.* 2010] H. Strasdat, J.M.M. Montiel and A.J. Davison. *Real-time Monocular SLAM: Why Filter?* In IEEE Int. Conf. on Robotics and Automation, Anchorage, USA, 2010.

[Tardós *et al.* 2002] J.D. Tardós, J. Neira, P.M. Newman and J.J. Leonard. *Robust mapping and localization in indoor environments using sonar data.* Int. J. Robotics Research, vol. 21, pages 311–330, 2002.

[Veit *et al.* 2007] T. Veit, F. Cao and P. Bouthemy. *Space-time A Contrario Clustering for Detecting Coherent Motions.* In Robotics and Automation, 2007 IEEE International Conference on, pages 33 –39, april 2007.

[Vidal-Calleja *et al.* 2011] Teresa A. Vidal-Calleja, Cyrille Berger, Joan Solí and Simon Lacroix. *Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain.* Robot. Auton. Syst., vol. 59, no. 9, pages 654–674, Septembre 2011.

[Vig & Adams 2006] L. Vig and J.A. Adams. *Multi-robot coalition formation.* Robotics, IEEE Transactions on, vol. 22, no. 4, pages 637 –649, aug 2006.

[Viola & Jones 2001] Paul Viola and Michael Jones. *Rapid object detection using a boosted cascade of simple features.* In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), pages 511–518, 2001.

[Viola & Jones 2002] Paul Viola and Michael Jones. *Robust Real-time Object Detection.* International Journal of Computer Vision, vol. 57, no. 2, pages 137–154, 2002.

[Viola & Jones 2004] Paul Viola and Michael J. Jones. *Robust Real-Time Face Detection*. International Journal of Computer Vision, vol. 57, pages 137–154, May 2004.

[Vu & Aycard 2009] Trung-Dung Vu and Olivier Aycard. *Laser-based detection and tracking moving objects using data-driven Markov chain Monte Carlo*. In Proceedings of the 2009 IEEE international conference on Robotics and Automation, ICRA'09, pages 3952–3958, Piscataway, NJ, USA, 2009. IEEE Press.

[Wang 2004] Chieh-Chih Wang. *Simultaneous Localization, Mapping and Moving Object Tracking*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2004.

[Wangsiripitak & Murray 2009] S. Wangsiripitak and D.W. Murray. *Avoiding moving outliers in visual SLAM by tracking moving objects*. In Robotics and Automation, 2009. ICRA '09. IEEE International Conference on, pages 375 –380, may 2009.

[Yamashita *et al.* 2003] A. Yamashita, T. Arai, Jun Ota and H. Asama. *Motion planning of multiple mobile robots for Cooperative manipulation and transportation*. Robotics and Automation, IEEE Transactions on, vol. 19, no. 2, pages 223 – 237, apr 2003.

[Yang *et al.* 1998] Jung-Min Yang, In-Hwan Choi and Jong-Hwan Kim. *Sliding Mode Control of a Nonholonomic Wheeled Mobile Robot for Trajectory Tracking*. In ICRA, pages 2983–2988, 1998.

[Zhang & Kleeman 2009] Alan M. Zhang and Lindsay Kleeman. *Robust Appearance Based Visual Route Following for Navigation in Large-scale Outdoor Environments*. Int. J. Rob. Res., vol. 28, no. 3, pages 331–356, Mars 2009.

Vers une navigation visuelle en environnement dynamique inconnu : apprentissage et exécution de trajectoire avec détection et suivi d'objets mobiles

---

Ecole doctorale : Systèmes (EDSYS)
Unité de recherche : Laboratoire d'Analyse et d'Architecture des Systèmes - LAAS-CNRS
Directeur de Thèse : Michel DEVY, Directeur de Recherche CNRS

Résumé :

L'objectif de ces travaux porte sur la navigation de robots autonomes sur de grandes distances dans des environnements extérieurs dynamiques, plus précisément sur le développement et l'évaluation de fonctions avancées de perception, embarquées sur des véhicules se déplaçant en convoi sur un itinéraire inconnu a priori, dans un environnement urbain ou naturel.

Nous avons abordé trois problématiques : d'abord nous avons exploité plusieurs méthodes de l'état de l'art, pour qu'un véhicule A, équipé d'un capteur stéréoscopique, apprenne à la fois une trajectoire et un modèle de l'environnement supposé d'abord statique. Puis nous avons proposé deux modes pour l'exécution de cette trajectoire par un véhicule B équipé d'une simple caméra : soit un mode différé, dans lequel B charge toute la trajectoire apprise par A, puis l'exécute seul, soit un mode convoi, dans lequel B suit A, qui lui envoie par une communication HF, les tronçons de la trajectoire au fur et à mesure qu'ils sont appris.

Enfin nous avons considéré le cas des environnements évolutifs et dynamiques, en traitant de la détection d'événements depuis les images acquises depuis un véhicule mobile : détection des changements (disparition ou apparition d'objets statiques, typiquement des véhicules garés dans un milieu urbain), ou de la détection d'objets mobiles (autres véhicules ou piétons).

Cette thèse a été préparée dans le contexte du projet RINAVEC (Reconnaissance d'Itinéraires et NAvigation en convoi de VEhicules Communicants), en collaboration avec Thales Optronique S. A et le laboratoire LASMEA : de nombreuses expérimentations et simulations préparées pour ce projet avec les démonstrateurs de Thales ou du LAAS, ont permis de valider les approches présentées dans cette thèse.

1 SLAM et flotte communicante de véhicules

L'acheminement d'approvisionnements par des véhicules en convoi dans des territoires hostiles est une préoccupation forte de tous les acteurs du monde de l'humanitaire. Pour diverses raisons (mines, autres véhicules ou personnes), la progression de tels convois est très difficile et dangereuse. Afin d'atténuer une part du danger, la stratégie qui peut être imaginée est qu'un véhicule de tête (leader), téléopéré par un opérateur depuis une autre position dans le convoi, définisse une trajectoire que tous les autres véhicules du convoi (suiveurs) doivent emprunter.

Le travail présenté dans cette thèse s'inscrit donc dans le contexte applicatif de la navigation de convois humanitaires dans des zones de conflit ou accidentées. L'objectif visé est de spécifier, concevoir, développer et valider une méthode permettant à un robot terrestre de se localiser précisément dans un environnement ouvert quelconque (urbain ou naturel, terrain plat ou accidenté). Les capteurs constituant ce système sont une caméra et des senseurs proprioceptifs bas coût (odométrie, gyromètre).

La problématique traitée rejoint en réalité le problème évoqué sous l'acronyme SLAM (Simultaneous Localization And Mapping) qui consiste à cartographier l'environnement en même temps qu'on se localise dans celui-ci. Depuis le milieu des années 80, de nombreuses contributions ont été apportées à ce verrou scientifique abordé originellement dans [Chatila & Laumond 1985] et aujourd'hui solutionné par des approches telles que celles décrites dans [Eustice *et al.* 2005, Mahon *et al.* 2008, Newman *et al.* 2009]. Dans le domaine du SLAM monoculaire, [Davison 2003], [Davison *et al.* 2007], [Mouragnon *et al.* 2009] et [Dissanayake *et al.* 2001] proposent une avancée significative dans la dernière décennie et plus récemment [Civera *et al.* 2009] qui affine encore l'association de données.

Souvent, le filtre de Kalman est exploité, parfois même en relation avec d'autres filtres (particulaire ré-échantillonné ou non), comme c'est le cas dans [Thrun *et al.* 2001]. Si d'autres approches seraient à signaler et à évaluer, notamment celles s'appuyant sur le filtre d'information [Mahon *et al.* 2008], c'est pourtant l'approche traditionnelle exploitant le filtrage de Kalman étendu qui est ici mise en oeuvre.

2 La problématique : Localisation et Cartographie Simultanées

Le problème connu dans la littérature sous le sigle SLAM (Simultaneous Localization And Mapping) ou CML (Concurrent Localization and Mapping)

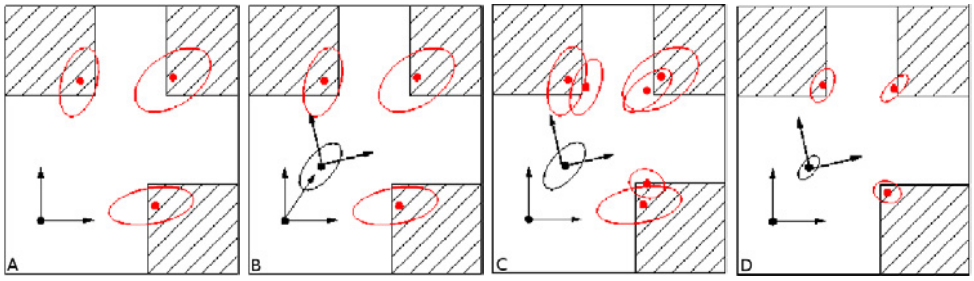


FIGURE 1: Les étapes du SLAM.

consiste à estimer conjointement la position du robot et les positions d'un ensemble d'objets remarquables de l'environnement, ces objets sont appelés amers et sont supposés immobiles.

La figure 1 présente les différentes étapes du SLAM : (A) Le robot observe une première fois les trois amers (les angles des carrés), (B) le robot se déplace, il connait sa position avec incertitude, (C) le robot réobserve les amers, (D) la fusion de cette observation avec la carte construite précédemment permet de diminuer à la fois les incertitudes sur les positions des amers et sur la position du robot.

Le SLAM est très lié au problème SFM (Structure From Motion) étudié dans la communauté vision. Il consiste à reconstruire, à partir d'une série d'images, un modèle tridimensionnel d'un objet. Pour cela on doit aussi reconstituer la trajectoire de la caméra. Les différences entre les deux approches proviennent du domaine d'application : la SFM n'a pas de contrainte temps-réel, et peut se contenter d'algorithmes de type batch processing alors que le SLAM est destiné à être implanté sur une plate-forme temps réel. De plus le robot a besoin de construire de manière incrémentale son modèle du monde car il utilise ce modèle à tout instant.

2.1 Axes principaux de l'algoritme SLAM

Deux axes principaux caractérisent un algorithme SLAM :

1. La technique d'estimation : nous aborderons particulièrement l'approche académique basée sur le filtrage de Kalman, mais nous parlerons aussi de filtre d'information et d'approche particulaire.
2. La nature de la représentation de la carte : l'approche classique consiste à positionner les amers dans un unique repère global, d'autres approches utilisent un ensemble de sous-cartes, ou des cartes relatives.

Ensuite nous aborderons le problème SLAM et nous présenterons des techniques de filtrage qui permettent d'estimer la position d'amers pour lesquels on ne dispose que d'observations partielles.

3 Le SLAM-EKF

Nous décrivons ici un algorithme basé sur un filtre de Kalman étendu (EKF, Extended Kalman Filter) qui produit une estimée de la position du robot ainsi que des amers et les incertitudes associées (on parle souvent de carte stochastique). Cette approche a été introduite dans [Smith & Cheeseman 1990] et reprise de nombreuses fois, par exemple [Thrun 2002, Leonard *et al.* 2002]. L'approche générale pose le problème sous la forme de l'estimation des distributions de probabilité d'un ensemble de variables aléatoires x à un instant t à partir de l'ensemble des observations effectuées par le robot jusqu'à t $z^t = z_1, z_2, \ldots, z_t$, et de l'ensemble des commandes envoyées au robot $u^t = u_1, u_2, \ldots, u_t$, on cherche donc à estimer $p(x_t|u^t, z^t)$. Nous rappelons ici la règle de Bayes :

$$p(x|d) = \alpha \cdot p(d|x) \cdot p(x) \tag{1}$$

qui permet d'écrire :

$$p(x_t|u^t, z^t) = \alpha \cdot p(z_t|x_t) \int p(x_t|u_t, x_{t-1}) \cdot p(x_{t-1}|u^{t-1}, z^{t-1}) dx_{t-1} \tag{2}$$

Deux éléments se détachent de cette équation et correspondent à deux évènements principaux de la "vie" du robot :

1. Le terme $p(z_t|x_t)$ modélise le processus d'observation (Quelle est la probabilité d'effectuer l'observation z_t sachant que "je suis à x_t").
2. Le terme $p(x_t|u_t, x_{t-1})$ modélise l'évolution du robot suite à une commande u_t (Quelle est la probabilité que "je me trouve en x_t" sachant que "j'étais en x_{t-1}" et que j'ai éxécuté la commande u_t), u_t est une donnée proprioceptive.

Cette relation de récurrence peut être simplifiée en sortant de l'intégrale les termes du vecteur d'état représentant la carte (qui sont supposés statiques).

Le filtre de Kalman est un filtre récursif qui implémente l'équation 2. Il est démontré qu'il est l'estimateur optimal dans le cas d'un système linéaire (commande et observation) avec des incertitudes gaussiennes sur les commandes et les observations. Dans un cas réel où le système n'est pas linéaire, on utilise un filtre de Kalman étendu (EKF) pour lequel on doit linéariser la fonction d'observation notamment, en calculant sa jacobienne. Le Kalman étendu n'est plus alors l'estimateur optimal. De manière plus pragmatique, le filtre de Kalman tronque cette estimation car le filtre produit les valeurs des moyennes et des variances/covariances des variables, c'est à dire les distributions gaussiennes de ces variables.

En deux dimensions, avec des amers ponctuels $A^i_{x,y}$, le vecteur d'état X à estimer est donc constitué de :

$$X = [x, y, \theta, A^1_x, A^1_y, \ldots, A^k_x, A^k_y]^T \tag{3}$$

et la matrice de variances/covariances de X est notée P . Un troisième élément qui n'apparait pas dans l'equation du filtre de Bayes (Eq. 2) est l'observation d'un nouvel amer. Cette approche necessite donc de définir trois fonctions et leur jacobienne associée qui sont :

1. $f(\cdot)$ qui permet de passer de l'$E_{commande}$ à l'E_{etat},
2. $h(\cdot)$ la fonction d'observation qui permet de passer de l'E_{etat} à l'$E_{observation}$,
3. $g(\cdot)$ la fonction d'initialisation d'un amer qui permet de passer de l'$E_{observation}$ à l'E_{etat}.

Dissanayake et al. [Dissanayake *et al.* 2001] démontrent des résultats théoriques, entre autres que, au bout d'un temps infini, les estimées des amers sont complètement corrélées (les positions relatives des amers sont connues de manière certaine), et la borne inférieure de l'incertitude sur la position des amers est déterminée par l'incertitude sur la position du robot lors de l'observation du premier amer. La complexité algorithmique de cette approche est en (N^3) où N est la taille du vecteur d'état, on peut se ramener en $O(N^2)$ car chaque observation porte sur un nombre limité (borné) d'amers. Dans des cas pratiques cette complexité est un réel problème qui va limiter la taille de l'environnement que l'on pourra explorer. Ce problème peut être abordé selon deux axes :

1. la méthode d'estimation peut être modifiée, on abandonne le filtrage de Kalman pour d'autres filtres,
2. la complexité vient de la matrice $N \times N$ de variances/covariances, on peut adopter d'autres types de représentation pour la carte stochastique.

3.1 Le problème d'estimation

Différents auteurs présentent des solutions au SLAM basées sur des techniques d'estimation différentes. [Knight *et al.* 2001] propose de retarder la mise à jour complète de la matrice de variances/covariances du filtre de Kalman. Dans [Thrun *et al.* 2004] c'est un filtre d'information (SEIF Sparse Extended Infomation Filter) qui est utilisé (dual du filtre de Kalman), une approximation justifiée est introduite afin de rendre la matrice d'information creuse, et de diminuer la complexité $(O(N))$.

3.2 Différentes techniques de représentation

Nous mettons ici l'accent sur la structure interne de la carte qui définit les variables qui seront estimées. Une grosse partie des calculs de l'approche Kalman provient de la mise à jour de la matrice de variances/covariances,

cette matrice est absolument indispensable. Sans les corrélations croisées, on obtient une carte incohérente lorsque l'on parcourt un boucle dans l'environnement par exemple. Ces corrélations croisées apparaissent car les amers sont exprimés dans un même repère global.

Les approches suivantes utilisent des représentations pour lesquelles des corrélations croisées entre les variables sont théoriquement nulles ou sont négligeables, ce qui diminue d'autant la charge de calcul lors de la mise à jours des estimées.

3.2.1 La carte relative

Une carte relative, à l'opposé d'une carte globale, représente les coordonnées des transformations géométriques entre amers voisins et non les transformations entre une référence globale et les amers. Cette représentation est beaucoup plus proche des données issues des capteurs : une perception d'un robot permet de mesurer les transformations géométriques entre le robot et les amers (mais la position du robot est connu avec incertitude), mais aussi les tranformations géométriques entre deux amers (ici, seule l'incertitude de mesure intervient). Les principaux inconvénients des cartes relatives viennent de la représentation elle-même qui est rarement exploitable directement, mais aussi de la cohérence géométrique de la carte : rien n'assure que les transformations mesurées entre les amers A et B, B et C et C et A permettent de reconstituer un triangle cohérent. Ce problème est résolu par [Newman 1999] avec un dérive d'un filtre de Kalman appelé Geometric Projection Filter (GPF). Les contraintes linéaires exprimées sous la forme $Cx = b$ (x est l'état du filtre) sont intégrées dans le filtre sous la forme d'observations de bruit nul (cette opération est effectuée uniquement lorsque l'utilisateur demande la carte), le cas des contraintes non-linéaires est aussi traité.

3.2.2 Les sous-cartes

Au lieu de faire "vivre" le robot dans une carte de taille importante, seulement une carte locale avec un nombre limité d'amers est maintenue en temps réel. Un filtrage de Kalman est appliqué dans cette carte, sa complexité est $O(N_L^2)$, avec N_L le nombre d'amers dans cette carte. On peut raisonnablement borner la densité d'amers dans l'environnement, alors la complexité devient $O(1)$. Ensuite deux approches distinctes existent.

La solution proposée par Williams [Williams 2002] maintient à un instant donné une carte globale complète et une carte locale. Ce filtre est connu sous le nom de CLSF (Constrained Local Submap Filter). Lorsque le robot "sort" de cette carte locale une nouvelle carte locale vierge est initialisée et le robot (re)découvre son environnement. Un processus en tâche de fond intègre les données de la carte locale précédente dans la carte globale courante.

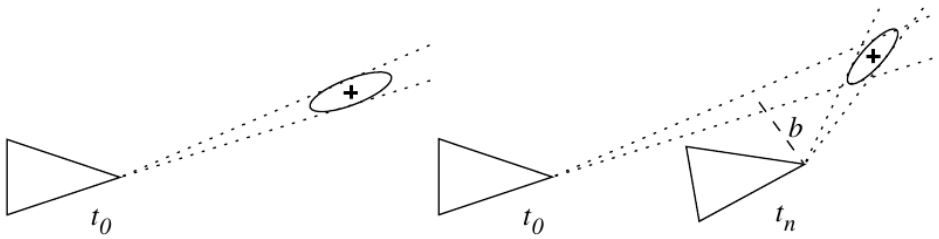


FIGURE 2: Initialisation d'amers par triangulation. La base est marquée b.

L'autre approche proposée par Newman [Newman & Leonard 2003] n'estime pas une carte globale mais un ensemble de cartes locales qui se recouvrent, le robot change de carte locale en fonction de ses déplacements. Des algorithmes sont mis en oeuvre pour assurer la cohérence des cartes entre elles.

3.3 Observation partielle de l'état des amers

Une des opérations les plus délicates dans le SLAM-EKF est l'initialisation des amers et leur introduction dans la carte. Le EKF a besoin des représentations Gaussiennes de toutes les variables aléatoires de la carte (la position et l'orientation du robot et la position de chacun des amers). De plus, leurs variances doivent être petites pour bien approcher les différentes fonctions non linéaires impliquées dans le système par des morceaux linéaires. Si le capteur est une seule caméra, à partir d'une seule observation d'un amer il n'est pas possible de définir un estimé de sa position qui respecte cette règle. Cette estimation est seulement possible via des mesures successives depuis différents points de vue, jusqu'au moment où l'on aura une base suffisante pour une bonne triangulation (figure 2).

Ce raisonnement conduit naturellement à des systèmes qui doivent attendre jusqu'à ce que cette base soit disponible. Davison [Davison 2003] utilise un Filtre Particulaire pour estimer la distance, qui n'est pas corrélée avec le reste de la carte. L'initialisation est reportée au moment où la variance sur cette distance est suffisamment petite pour permettre une représentation gaussienne. Dans [Bailey 2003], des positions passées du robot sont stockées dans la carte, avec ses observations associées, et un test de gaussiennité est évalué sur la Fonction de Distribution de Probabilité (FDP) de la position de l'amer résultante de combiner la première et la dernière mesure. L'initialisation est aussi retardée au moment où ce test est validé, et toutes les autres mesures sont alors utilisées pour raffiner et corriger la carte entière. Ces méthodes souffrent de deux inconvénients : elles ont besoin d'un critère pour décider si la base est suffisante, et elles introduisent un retard à l'initialisation.

L'élimination du critère et du retard s'avère un sujet intéressant. Le critère est très souvent lourd à calculer, et le système devient plus robuste sans lui car on évite des décisions binaires irréversibles. Sans le retard, nous aurons à disposition des informations sur l'amer, même partielles, qui pourront être utilisées comme des références angulaires. Nous aurons aussi la possibilité d'inclure des amers qui se trouvent très proches de l'axe d'avancement du robot, pour lesquels la base mettrait trop de temps à grossir. Ce dernier point est très important pour des applications en extérieur dans lesquelles les véhicules suivent des trajectoires rectilignes et les caméras regardent naturellement en avant.

Solà [Solà 2007] propose une initialisation non retardée avec un grossissement seulement additif de la taille du problème. Il définit un ensemble d'hypothèses sur la position de l'amer le long du rayon optique sur lequel il a été observé pour la première fois, et il les inclut toutes dans la carte comme s'il s'agissait d'amers différents. Dans les observations postérieures, l'hypothèse la plus vraisemblable va être utilisée pour la correction.

4 SLAM visuel avec détection et suivi d'objets mobiles

Les méthodes SLAM permettent d'estimer à la fois la trajectoire d'un véhicule et la structure du monde à partir de données acquises depuis un capteur embarqué. Une hypothèse forte souvent considérée dans ces approches, est que le monde est statique, donc que les éléments modélisés dans la carte SLAM sont statiques. Cela a deux avantages : les positions de ces éléments peuvent être estimées de manière incrémentale, et ils peuvent être exploités comme des références fixes (d'où le nom *amers*) pour estimer la position du robot dans le monde.

Dans cette thèse, nous considérons qu'il peut exister des objets mobiles dans le monde perçu par le robot ; malgré cela, nous voulons résoudre le problème du SLAM, et dans le même temps, estimer l'état de chaque objet mobile. Cela nécessite de traiter aussi de la détection et du suivi de ces objets mobiles, tout en conservant la capacité de détecter une nombre suffisant d'amers fixes afin de pouvoir localiser le robot. Nous avons développé un système complet qui traite des deux fonctions SLAM d'une part, et Détection et Suivi d'objets mobiles (souvent appelé DATMO, pour *Detection And Tracking of Mobile Objects*) d'autre part, en exploitant uniquement la vision : nous appelons ce système *BiCam SLAM with DATMO* car nous exploitons deux caméras montées sur notre robot.

La fonction DATMO nécessite de résoudre deux problèmes de nature très différentes : la détection *Moving Objects Detection* (MOD) et le suivi, ou

Moving Objects Tracking (MOT). Le problème MOT est résolu par des approches d'estimation. Mais, afin de rester dans la zone d'observabilité complète du monde 3D par un système BiCam [Solà *et al.* 2008], les objets mobiles seront uniquement suivis par le MOT dans la zone proche du robot, alors que les amers statiques pourront être extraits pour le SLAM dans tout le champ de vue (donc jusqu'à l'infini). La fonction MOD traite de la détection *initiale* des objets mobiles : la difficulté vient de l'impossibilité d'analyser de manière exhaustive le contenu de chaque image à une fréquence élevée (typiquement 30Hz). Nous proposons une méthode active, qui exploite les connaissances disponibles afin de focaliser l'analyse sur chaque image, sur les zones dans lesquelles un objet mobile a le plus de probabilites d'apparaitre. Cette approche rapide et efficace permet non seulement de détecter et suivre des objets mobiles, mais aussi d'en identifier la nature.

5 Etat de l'art sur le SLAM-MOT

La fonction désignée par "cartographie et localisation simultanées, avec suivi d'objets mobiles" (SLAM-MOT) combine un module de SLAM, prévu normalement pour s'exécuter en milieu statique, avec un module DATMO en charge de détecter et suivre des objets mobiles.

Le travail fondateur sur cette problématique du SLAM dans un environnement dynamique, a été présenté par B.Wang [Wang 2004]. C'est une méthode classique de SLAM 2D fondée sur des données télémétriques laser, mais qui prend en compte la présence d'objets mobiles. Ces objets sont détectés en segmentant les parties de chaque coupe-laser qui ne s'apparient pas avec la carte courante. Comme ils ne peuvent servir pour estimer la position du robot dans l'environnement, les objets mobiles détectés sont laissés en dehors de la carte. Le système complet, fondé sur des télémètres laser 2D longue portée et grande vitesse, a permis de réaliser du SLAM sur de grandes surfaces à des vitesses élevées avec des fermetures de boucle de grande taille, le tout en milieu urbain avec un trafic dense.

D'autres approches intéressantes ont été proposées. Citons les travaux de Agrawal et al.[Agrawal *et al.* 2005] via des cartes denses de disparité produites par un banc stéréo calibré, une méthode RANSAC robuste appliquée pour déterminer la transformation rigide entre deux positions successives du capteur stéréo, et une approche de type *Inverse Perspective mapping* exploitée pour trouver ensuite les zones de l'image ayant des mouvements propres indépendants du mouvement du capteur. Dans [Wangsiripitak & Murray 2009], un objet mobile est prédéfini et suivi en 3D, en parallèle avec une méthode de SLAM monoculaire ; cette approche ne propose pas de solution pour la détection des objets mobiles et exploite pour suivre l'objet, une méthode basée modèle uniquement adaptée pour suivre une classe d'objets (véhicules).

Migliore et al. [Migliore *et al.* 2009] ont proposé une approche intégrant SLAM et MOT, dans laquelle les objets mobiles sont détectés par un simple test statistique, puis suivis par des modules MOT monoculaires séparés. G.Gaté et al [Gate *et al.* 2009] ont proposé un système multi-capteurs permettant de rendre plus robuste l'étape de détection : l'objet mobile est d'abord détecté dans les données laser, puis sa présence est confirmée par classification, dans les données visuelles.

Solà [Solà 2007] a proposé une analyse d'observabilité pour la détection et le suivi d'objets mobiles depuis un capteur visuel embarqué. Il conclut que seul un système de type BiCam (deux caméras) permet de discriminer mouvements des objets et mouvements propres des caméras. Il propose une méthode pour détecter des points mobiles ; un suivi est lancé séparément pour chaque point ainsi détecté, représenté dans un repère robot-centré.

Lin et al. [Lin & Wang 2010] ont proposé récemment une approche de SLAM-MOT par stéréovision, dans laquelle le vecteur d'état mis à jour par le SLAM est "augmenté" avec les états des objets mobiles qui ont été détectés. Ils ont montré que le MOT ainsi couplé au SLAM, permettait d'améliorer les performances du SLAM. Ils exploitent une stratégie identique à l'approche BiCam [Solà *et al.* 2008] ; les deux caméras sont traitées comme des capteurs indépendants, dont les observations permettent de mettre à jour le vecteur d'état par EKF.

Tous ces auteurs ont proposé des approches originales pour traiter du SLAM en présence d'objets mobiles. Mais les approches proposées apparaissent comme la juxtaposition des fonctions SLAM et MOT, avec partage minimal d'informations. Il existe généralement peu d'interactions entre *localization and mapping* d'un côté, et *detection and tracking* de l'autre.

5.1 Modèle des objets mobiles

Un simple point est représenté en $3D$ dans l'espace euclidien ; un tel point, s'il est dynamique, devra être suivi dans les images acquises depuis le robot. Ces points $\mathbf{o}_j^{\mathcal{O}_i}$,sont définis par leurs positions et leurs vitesses linéaires.

Un objet $\mathcal{O}_i$, est défini par un ensemble de points connexes qui ont des mouvements cohérents en $3D$ (vitesses très proches). Ces points sont rigidement liés, ce qui définit la structure de l'objet. Cette structure, caractéristique de l'objet, peut être décrite dans un repère propre à l'objet, par l'ensemble des points acquis sur la surface de l'objet. Un objet mobile conserve sa structure rigide, mais la position de son repère évolue dans le temps, selon $\mathbf{o}_j^{\mathcal{O}_i} = \mathbf{o}_j^{\mathcal{O}_i}(t)$ et $\mathcal{O}_i = \mathcal{O}_i(t)$. Par la suite, nous omettons de préciser le temps (t) dans nos

équations. Nous écrivons en conséquence les représentations initiales des états des *points mobiles* and *objets mobiles* comme suit :

$$\mathbf{o}_j^{\mathcal{O}_i} = \begin{bmatrix} \mathbf{m}_j^{\mathcal{O}_i} \\ \mathbf{v}_j^{\mathcal{O}_i} \end{bmatrix}, \quad \mathcal{O}_i = \begin{bmatrix} \mathbf{o}_1^{\mathcal{O}_i} \\ \vdots \\ \mathbf{o}_n^{\mathcal{O}_i} \end{bmatrix}, \tag{4}$$

où $\mathbf{o}_j^{\mathcal{O}_i}$ est l'état d'un *point mobile* j, défini par sa position $\mathbf{m}_j^{\mathcal{O}_i}$, et sa vitesse linéaire $\mathbf{v}_j^{\mathcal{O}_i}$, respectivement. Un ensemble de points mobiles ($\mathbf{o}_j^{\mathcal{O}_i}$), définit un *objet mobile* i ($\mathcal{O}_i$). Chaque point mobile peut être exprimé dans le repère monde W ($\mathbf{o}_j^{\mathcal{O}_i W}$), le repère robot R ($\mathbf{o}_j^{\mathcal{O}_i R}$) ou encore le repère caméra C ($\mathbf{o}_j^{\mathcal{O}_i C}$), avec les transformations de repères usuelles.

5.2 Représentation des objets mobiles.

Dans une fonction SLAM classique, le robot est localisé grâce à des observations sur des amers statiques, cela afin de garantir une bonne localisation. Les objets mobiles sont ignorés dans ce processus SLAM dédié à la localisation du robot. Les observations que le robot peut avoir sur ces objets mobiles ne doivent pas perturber la fonction SLAM, et donc le modèle du robot lui-même.

Cette approche conduit à construire des représentations des objets mobiles totalement indépendantes de la carte SLAM : on ne considére pas les corrélations entre les états des objets mobiles, du robot et de la carte SLAM. La seule relation entre les états du robot et d'un objet mobile sont décrites par les observations courantes de cet objet depuis le robot, et par un modèle dynamique choisi pour cet objet. Finalement la carte construite par notre système *BiCam SLAM with DATMO* est faite de la carte SLAM construite par l'approche BiCam et par un ensemble de vecteurs stochastiques, mis à jour par EKF, un vecteur pour chaque objet détecté. Donc :

$$X^T = \begin{bmatrix} C_L^T & C_R^T & L_1^T & \dots & L_M^T \end{bmatrix} \tag{5}$$

$$\mathcal{O}_i^T = \begin{bmatrix} \mathbf{o}_1^{\mathcal{O}_i T} \dots \mathbf{o}_n^{\mathcal{O}_i T} \end{bmatrix} \tag{6}$$

où dans l'équation 5, $C_{L/R}^T = [\mathbf{r}_{L/R}^T, \mathbf{q}_{L/R}^T] \in \mathbb{R}^7$, est l'état de la caméra, respectivement gauche et droite, donné par sa position et un quaternion pour l'orientation, plus l'ensemble des amers $L_j = i_j \in \mathbb{R}^6$ (en IDP) ou $L_j = p_j \in \mathbb{R}^3$ (en euclidien) et, dans l'équation 6, $\mathbf{o}_j^{\mathcal{O}_i T} = (\mathbf{m}_j^{\mathcal{O}_i}, \mathbf{v}_j^{\mathcal{O}_i})$ est l'état d'un point mobile défini par sa position et sa vitesse. Le modèle de mouvement d'un tel point est un modèle à vitesse constante.

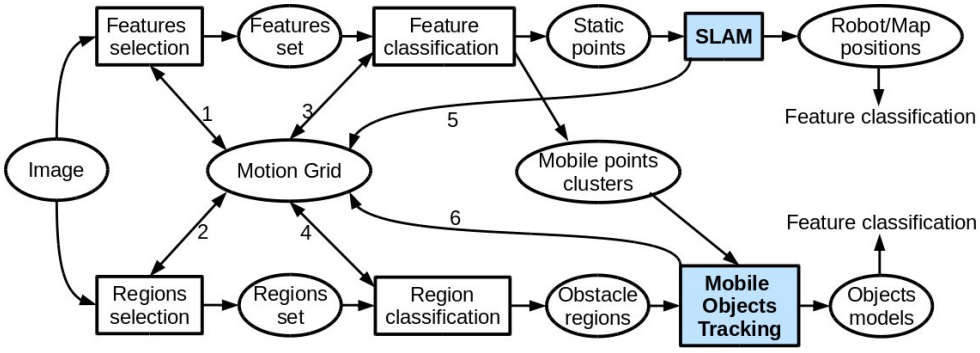


FIGURE 3: Les fonctions de notre système *BiCam SLAM with DATMO*.

6 Description du système complet

Nous proposons un système complet pour traiter les fonctions SLAM et MOT sur un robot équipé d'un banc de stéréovision (ici traité comme un système BiCam, deux caméras indépendantes), évoluant dans un environnement dynamique. Les principales fonctions considérées dans ce système sont présentées en Figure 3.

6.1 Sélection des points d'intérêt

La fonction *Détection des Points d'intérêt* doit tirer profit de toutes les connaissances déjà issues des images précédentes, ou d'informations contextuelles liées à l'application. Aussi à chaque itération de l'algorithme de détection, des points d'intérêt doivent être d'abord extraits : cette extraction est guidée par une *Grille de Mobilité*, construite comme une grille d'occupation, sur la base d'une discrétisation de l'espace image en cellules (typiquement un pavé 10x10 de l'image). Cette grille donne la probabilité de trouver en une telle cellule, un point d'intérêt mobile.

Initialement, en l'absence d'informations contextuelles, cette probabilité est uniforme ; ensuite la grille est mise à jour selon différents critères qui sont présentés en section 6.2. Les nouveaux points sont donc extraits dans les cellules qui ont la plus forte probabilité de présence d'un objet mobile. Vu l'initialisation de la grille, ces points sont d'abord uniformément répartis dans l'image. Ensuite au fil des déplacements de notre robot et de l'exécution des différentes fonctions de notre système, les points sont remplacés soit parce qu'ils ont été perdus par la fonction de suivi des points, soit parce qu'ils ont été classés statique ; les points qui sont classés dynamique sont par contre conservés, associés à des groupes de points qui sont suivis de manière continue dans la suite des images.

6.2 Grille de Mobilité

Comment instancier et mettre à jour la *Grille de Mobilité* ? A chaque cycle de la fonction de détection, il est important (1) de détecter de nouveaux points sur un objet mobile qui a déjà été détecté dans les images précédentes, cela pour segmenter toute la zone image qui lui correspond, et (2) de détecter de nouveaux objets mobiles qui sont entrés dans le champ de vue de la caméra, donc d'extraire de nouveaux points à classer statique ou dynamique dans les zones de l'image où la probabilité de présence d'un objet mobile est la plus élevée.

Pour sélectionner de manière active ces points d'intérêt qui vont être suivis avant d'évaluer la probabilité qu'ils correspondent à des points statiques ou dynamiques de la scène, nous proposons cette *Grille de Mobilité* introduite dans [Almanza-Ojeda *et al.* 2011]. Cette grille se comporte comme une grille d'occupation en robotique (*occupancy grid*), au sens où les plus fortes probabilités correspondent potentiellement aux zones mobiles (occupées) et les plus basses aux zones statiques (libres) ; il convient de focaliser la détection de nouveaux objets mobiles dans les zones de la grille qui ont les plus fortes probabilités.

Cette grille est donc donnée en entrée de la fonction *Sélection de Points d'intérêt* qui extrait de nouveaux points d'intérêt ; ces points seront ensuite caractérisés comme statique ou dynamique, après les avoir suivis pendant N_{im} images successives. Une fois ces points caractérisés, les zones dans lesquelles ils se trouvent après ces N_{im} images, peuvent hériter de leur label statique ou dynamique : il faut donc mettre à jour les probabilités de ces zones dans la *Grille de Mobilité*.

Ces probabilités sont modifiées selon des fonctions gaussiennes bidimensionnelles centrées sur les points qui ont été caractérisés à cette étape de la fonction de détection. Pour un point en (u,v), cette gaussienne a des variances σ_u et σ_v qui sont fonctions de r, r étant la taille d'une cellule de la grille (ici 10 pixels). L'équation 7 montre l'évolution spatio-temporelle des probabilités pour des cellules qui sont mobiles, statiques ou inconnues dans la grille. La probabilité qu'une cellule recouvre un objet mobile dans l'image courante, est inversement proportionnelle à sa distance au point classé dynamique le plus proche ; la décroissance est obtenue grâce à la fonction gaussienne centrée sur ce point. Et donc, dans le temps, cette probabilité évolue comme l'inverse de la valeur établie dans le premier cas de l'équation 8 :

$$p(u,v,t) = \Theta_i^{pm} = \sum_{u,v \in cell} (\alpha(u,v)_{t-1} + G(\mu_u, \mu_v, \sigma_u, \sigma_v)) \qquad (7)$$

où $\alpha(u,v)$ désigne la probabilité dans une cellule centrée en (u,v). L'évolution

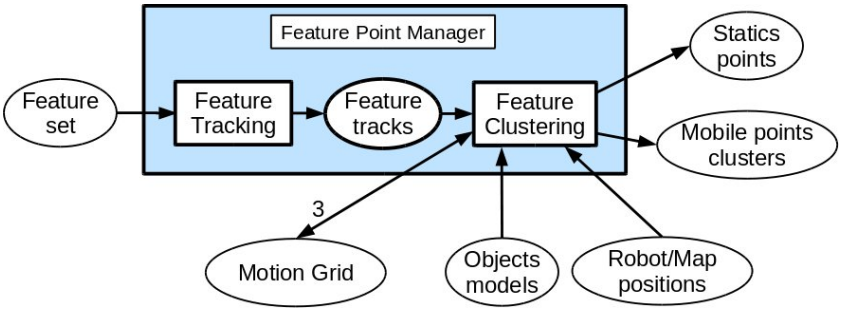


FIGURE 4: La fonction *Caractérisation des points d'intérêt*.

dans le temps de cette probabilité est donnée par :

$$\alpha(u,v)_t = \begin{cases} 1 - p(u,v)_t & \text{if } \alpha(u,v)_{t-1} = moving \\ p_0 & \text{if } \alpha(u,v)_{t-1} = static \end{cases} \quad (8)$$

6.3 Caractérisation des Points d'intérêt

Nous devons classifier chaque primitive visuelle comme dynamique ou statique, avant de l'exploiter soit comme amer qui sera rajouté dans la carte SLAM pour localiser le robot, soit comme observation qui permettra au module MOT de caractériser un objet mobile. Nous exploitons la méthode proposée dans [Almanza-Ojeda *et al.* 2011] afin de détecter un objet mobile : nous appelons cette procédure *Caractérisation des Points d'intérêt*. Le but de cette méthode est de discriminer les points comme statique ou dynamique et de regrouper les points dynamiques qui pourraient appartenir aux mêmes objets mobiles. Le processus de *Caractérisation des Points d'intérêt* est présenté en Figure 4.

A chaque itération de la fonction de détection, N_p points d'intérêt ont été sélectionnés de manière active en exploitant la *Grille de Mobilité* : $N_p = 150$ dans nos tests. Ces points sont suivis par la fonction KLT fondée sur la corrélation et l'optimisation des déformations des fenêtres de corrélation. Supposons d'abord que l'environnement est statique ; la fonction KLT boucle sur N_{Im} images successives. Ce temps de suivi est appelé temps de pistage : $N_{im} = 6$ dans nos évaluations ; c'est le temps nécessaire pour construire une *piste* pour chaque point suivi. Une piste donne les positions successives de ce point, et permet de calculer sa vitesse apparente. Certains points sont perdus durant ces N_{im} images ; ils ne sont pas remplacés de suite. L'ensemble des points suivis sera reconstruit uniquement après ces N_{im} images, surtout après mise à jour de la *Grille de Mobilité*. La fonction KLT s'exécute donc en continu, mais en remplaçant l'ensemble de points suivis après chaque période de pistage.

Seuls les points suivis par la fonction KLT sont sélectionnés pour être groupés par la méthode de clustering a contrario inspirée des travaux présentés dans [Veit *et al.* 2007]. Cette fonction génère une liste de points statiques et une liste de M_i clusters ou objets mobiles potentiels. Chaque cluster est défini par une liste de points classés dynamique, par le barycentre de ces points, la vitesse apparente (moyenne des vitesses apparentes de ces points) et une boîte (rectangle) englobant ces points. La fonction de clustering prend en entrée des informations issues de la fonction SLAM, afin de compenser le mouvement de la caméra quand cela est possible (uniquement sur le plan du sol). Elle prend aussi en entrée des informations venant de la fonction MOT, requises pour mettre à jour la *Grille de Mobilité*. En effet, les caractéristiques (positions, vitesses et tailles apparentes dans l'image courante) de chaque objet suivi par la suite, sont exploitées pour évaluer la probabilité de trouver des points dynamique dans chaque cellule de la grille.

6.4 Génération des hypothéses : approche *Classification*

Des hypothèses sur la présence d'objets mobiles dans la scène, sont issues des résultats d'un détecteur fondé sur une approche *Classification*. Le système que nous proposons est indépendant du choix de la méthode de classification [Dalal & Triggs 2005], [Viola & Jones 2002], que ce soit les attributs attachés à une région à classifier, ou le choix du classifieur lui-même. Dans nos expérimentations, pour des raisons pratiques, nous exploitons un classifieur AdaBoost [Viola & Jones 2002], fondé sur l'idée qu'une combinaison de classifieurs faibles permet d'obtenir un classifieur fort et efficace, cela après une phase d'apprentissage supervisée, donc sur une base d'exemples positifs et négatifs. Ici le classifieur AdaBoost a été entraîné de manière spécifique pour classer une région, soit comme *Personne*, soit comme *Véhicule*.

Chaque groupe noté M_i fourni par la fonction *Gestion des Points d'Intérêt*, est associée à une région d'intérêt dans l'image ; une telle région est un *Objet Potentiel*. Le classifieur sera uniquement appliqué sur ces régions d'intérêt afin d'associer une probabilité à l'hypothèse *Objet Potentiel*, cela pour éviter la phase de parcours spatial multi-échelles de l'image, étape lourde des approches de détection par classification.

Donc, la région d'intérêt M_i est décomposée en n régions différentes, $(M_i^k)_{1\leq k\leq n}$; chacune de ces n régions est donnée à l'algorithme de classification, qui lui associe une valeur λ_i^k obtenue par une somme pondérée des résultats des N classifieurs faibles, $(w_l)_{1\leq l\leq N}$

$$\lambda_i^k = \sum_{1\leq l\leq N} \alpha_l w_l(M_i^k) \tag{9}$$

Finalement, pour chaque groupe M_i, un score de classification global est calculé de la manière suivante :

$$\Theta_i^c = \max_{1 \leq k \leq n} \lambda_i^k \tag{10}$$

Cette méthode de classification est seulement entraînée pour calculer la probabilité qu'une région donnée corresponde à un objet de type *Personne* ou *Véhicule* ; cela n'est pas suffisant pour conclure sur la détection ou le suivi d'un objet mobile dans la scène. Cependant, chaque groupe M_i se voit attribuer un score de classification Θ_i^c.

Pour chaque groupe ou objet potentiel i à l'instant k, ce score de clasisfication est indépendant de la probabilité que la zone de l'image dans laquelle se trouve ce groupe, contienne un objet mobile, probabilité déjà disponible par l'approche *Grille de Mobilité*. Il reste à traiter de la fusion de ces deux informations.

6.5 Evaluation finale des hypothèses sur la détection des objets mobiles

Finalement, afin de détecter correctement les objets mobiles, l'estimée fournie par l'approche *Grille de Mobilité* doit être fusionnée avec l'estimation issue de l'approche *Classification*. Nous disposons de deux estimées différentes de la même probabilité pour une région, $P(\mathcal{O}_i|A_j)$, la probabilité de l'événement $\mathcal{O}_i$ (cette région correspond à un objet mobile) connaissant A_j. Ces deux estimées ont été obtenues par des méthodes qui sont elles-même incertaines. Aussi, chacune de ces estimées peut être réécrite de la manière suivante :

$$P(\mathcal{O}_i|A_1) = \Theta_i^{pm} \tag{11}$$

$$P(\mathcal{O}_i|A_2) = \Theta_i^c \tag{12}$$

où $P(\mathcal{O}_i|A_1)$ est l'estimation donnée par l'approche *Grille de Mobilité* et $P(\mathcal{O}_i|A_2)$ est l'estimation fournie par l'approche *Classification*. En faisant l'hypothèse que A_1 and A_2 sont des événements indépendants, la règle de fusion suivante est obtenue des opérations de base sur les probabilités :

$$\begin{aligned} P(\mathcal{O}_i) &= P(\mathcal{O}_i \cap (A_1 \cup A_2)) + P(\mathcal{O}_i \cap (\overline{A_1 \cup A_2})) \\ &= P(\mathcal{O}_i|A_1)P(A_1)P(\overline{A_2}) + P(\mathcal{O}_i|A_2)P(\overline{A_1})P(A_2) \\ &+ P(\mathcal{O}_i|A_1 \cap A_2)P(A_1)P(A_2) + P(\mathcal{O}_i|\overline{A_1} \cap \overline{A_2})P(\overline{A_1})P(\overline{A_2}) \end{aligned} \tag{13}$$

où $P(A_j)$ désigne la probabilité fournie par le sous-système j (ici approches Grille de Mobilité ou Classification) de retourner une mauvaise estimation.

$P(\mathcal{O}_i|\overline{A_1} \cap \overline{A_2})$ peut être approximée par ξ, la probabilité de classification a priori de la région dans une classe donnée. Alors la règle de fusion entre les deux approches est approximée par une somme pondérée :

$$\begin{aligned}\Phi_{\mathcal{O}_i} &= \Theta_i^{pm} P(A_1) P(\overline{A_2}) + \Theta_i^{c} P(\overline{A_1}) P(A_2) \\ &+ \frac{\Theta_i^{pm} P(A_1) + \Theta_i^{c} P(A_2)}{P(A_1) + P(A_2)} P(A_1) P(A_2) + \xi P(\overline{A_1}) P(\overline{A_2})\end{aligned} \tag{14}$$

A ce niveau, l'estimée finale de la probabilité de détection est disponible. Le résultat final de la fonction MOD est donné par un seuillage sur cette probabilité. Donc, pour chaque région d'intérêt M_i,

$$M_i = \begin{cases} \textit{Moving Object} & \text{if } \Phi_{\mathcal{O}_i} \geq \text{threshold} \\ \\ \textit{NOT Moving Object} & \text{if } \Phi_{\mathcal{O}_i} < \text{threshold} \end{cases}$$

Notre stratégie de détection est donc fondée sur la fusion de deux algorithmes de natures différentes, d'une part la *Grille de Mobilité* et d'autre part l'*approche fondée sur la classification*. Comme le second algorithme dépend du processus d'apprentissage pour calculer une probabilité de classification sur certains objets (piétons, voitures...), on ne peut dire que cette méthode de détection d'objets mobiles est "générique".

6.6 Fonction MOT : suivi des objets mobiles

6.6.1 Création des objets mobiles depuis les groupes de points

Dès qu'un objet est détecté via un groupe de points d'intérêt dynamiques, une vecteur d'état lui est associé sous la forme d'une pdf qui sera mise à jour par EKF. Ce vecteur d'état suit donc une loi gaussienne $\mathcal{O} \sim \mathcal{N}\{\bar{\mathcal{O}}; \mathbf{P}_{\mathcal{O}}\}$. Pour sa création nous exploitons la paramétrisation par IDP, décrite dans [Civera *et al.* 2008]. L'estimée et la matrice de variances initiales des points dynamiques $\mathbf{m}_j^{\mathcal{O}_i}$, associées à cet objet $\mathcal{O}_i$, sont produitess par la paramétrisation IDP définie dans la caméra gauche :

$$\mathbf{m}_j^{\mathcal{O}_i} = \mathbf{d}_L(C_L, \mathbf{h}_L, \rho) \tag{15}$$

où $\mathbf{d}_L(\cdot)$ est fonction de , $C_L^T = [\mathbf{x}_L^T, \mathbf{q}_L^T]$ l'état de la caméra gauche, défini par sa position et son orientation, $\mathbf{h}_L \sim \mathcal{N}\{\mathbf{y}_L; \mathbf{R}\}$ l'observation du point dans la caméra gauche représentée par un vecteur gaussien, et $\rho \sim \mathcal{N}\{\bar{\rho}; \sigma_\rho^2\}$ l'inverse de la distance euclidienne entre C_L et le point. Après sa détection initiale dans l'image gauche, tous les paramètres de $\mathbf{m}_j^{\mathcal{O}_i}$ sauf ρ sont immédiatement observables, et leurs estimées et matrices de variances sont obtenues par inversion et linéarisation des fonctions d'observation.

Donc pour la position d'un point mobile $\mathbf{m}_j^{\mathcal{O}_i}$, l'estimée et la matrice de variances initiales sont donc :

$$\bar{\mathbf{m}}_j^{\mathcal{O}_i} = \mathbf{d}_L(C_L, \bar{\mathbf{h}}_L, \bar{\rho}) \tag{16}$$

$$\mathbf{P_{mm}} = \mathbf{D}_{L\mathbf{h}}\, \mathbf{R}\, \mathbf{D}_{L\mathbf{h}}^T + \mathbf{D}_{L\rho}\, \sigma_\rho^2\, \mathbf{D}_{L\rho}^T \tag{17}$$

où $\mathbf{R}$ est la matrice de rotation, tandis que les jacobiennes sont :

$$\mathbf{D}_{L\mathbf{h}} = \left.\frac{\partial \mathbf{d}_L}{\partial \mathbf{h}^T}\right|_{(\bar{C}_L, \bar{\mathbf{h}}, \bar{\rho})}, \quad \mathbf{D}_{L\rho} = \left.\frac{\partial \mathbf{d}_L}{\partial \rho^T}\right|_{(\bar{C}_L, \bar{\mathbf{h}}, \bar{\rho})} \tag{18}$$

L'estimée et la matrice de variances initiales pour la vitesse d'un point mobile $\mathbf{v}_j^{\mathcal{O}_i} \sim \mathcal{N}\{\bar{\mathbf{v}}_j^{\mathcal{O}_i}; \mathbf{P_{vv}}\}$ sont déterminées de manière heuristique comme dans [Lin & Wang 2010].

Finalement, le vecteur d'état d'un objet mobile est donné par la pdf définie par le couple :

$$\bar{\mathcal{O}}_i = \begin{bmatrix} \bar{\mathbf{m}}_j^{\mathcal{O}_i} \\ \bar{\mathbf{v}}_j^{\mathcal{O}_i} \end{bmatrix}, \quad \mathbf{P}_{\mathcal{O}_i} = \begin{bmatrix} \mathbf{P_{mm}} & 0 \\ 0 & \mathbf{P_{vv}} \end{bmatrix}, \tag{19}$$

Dès sa création, cet état est mis à jour par EKF avec l'observation issue de la caméra droite. La fonction d'observation pour la caméra droite est donnée par la fonction de projection, en fait le modèle pin-hole de la caméra, exprimée dans son propre repère par :

$$\mathbf{y}_R = \mathbf{h}_R(C_R, \mathbf{m}_j^{\mathcal{O}_i}) + v_R \tag{20}$$

où $v_R \sim \mathcal{N}\{0; \mathbf{R}\}$ est le bruit d'observation, supposé être un bruit gaussien blanc.

Donc, à partir de son observation dans la caméra droite, considérant la pose incertaine de cette caméra $C_R \sim \mathcal{N}\{\bar{C}_R; \mathbf{P}_{C_R}\}$, la pdf donnant l'état de l'objet mobile est mise à jour par EKF.

Le test de linéarité introduit par [Civera *et al.* 2007] est évalué après chaque exécution du filtre. Si ce test est positif, la position de l'objet peut être reparamétrée de IDP en coordonnées euclidiennes 3D.

6.6.2 Mise à jour des objets mobiles par compensation du mouvement du robot

Le modèle dynamique du robot exploite des mesures de déplacement données par odométrie. Ce modèle dynamique exploite donc le modèle de l'odométrie, pour le modèle unicycle de robot qui correspond à notre

plateforme $R^+ = \mathbf{f}_R(R, \mathbf{u})$, où $\mathbf{u}$ est le vecteur de contrôle du robot ou les mesures odométriques :

$$\mathbf{u} = \begin{bmatrix} \delta\mathbf{x} \\ \delta\mathbf{e} \end{bmatrix} = [\delta x, \delta y, \delta z, \delta\phi, \delta\theta, \delta\psi]^T \in \mathbb{R}^6 \tag{21}$$

avec la *pdf* gaussienne $\mathbf{u} \sim \mathcal{N}\{\bar{\mathbf{u}}; \mathbf{U}\}$. Dès que la position du robot a été mise à jour, les positions des objets mobiles doivent être transformées, pour être exprimées vis-à-vis de la nouvelle position du repère du robot. Donc l'équation $\mathcal{O}_i^+ = \mathbf{j}_{\mathcal{O}}(\mathcal{O}_i, \mathbf{u})$, exprime simplement un changement de coordonnées qui est spécifié à partir des mesures odométriques $\mathbf{u}^T = [\delta\mathbf{x}^T, \delta\mathbf{e}^T]$. Nous pouvons donc représenter les objets mobiles par :

$$\mathbf{m}_j^{\mathcal{O}_i +} = \mathbf{R}^T\delta(e) \cdot (\mathbf{m}_j^{\mathcal{O}_i} - \delta x) \tag{22}$$

$$\mathbf{v}_j^{\mathcal{O}_i +} = \mathbf{R}^T\delta(e) \cdot \mathbf{v}_j^{\mathcal{O}_i} \tag{23}$$

où $\mathbf{R}^T\delta(e)$ est une matrice de rotation calculée à partir des incréments de rotation exprimés en angles d'Euler $\delta(e)$. Donc en considérant ces équations de compensation appliquées aux positions et vitesses, la pdf représentant chaque objet mobile peut être mise à jour par EKF.

6.6.3 Estimation du mouvement propre des objets mobiles

Nous n'avons aucune observation directe sur le mouvement des objets mobiles. Des modèles de mouvement à vitesse constante fournissent des trajectoires plus ou moins lisses en fonction de la quantité de bruit pris en compte dans le système. Par exemple, le mouvement d'un objet de type *Personne* peut être considéré avec une vitesse constante dans le plan 2D du sol. S'il est supposé qu'il ne peut pas s'arrêter brutalement, il suffit de considérer un simple bruit gaussien assez faible. Mais un modèle de bruit plus fort devra être considéré si on veut prendre en compte des changements de dynamique brusques.

Pour l'instant dans notre système, pour des raisons pratiques, nos objets ont tous des modèles à vitesse constante. Donc le modèle dynamique de nos objets est donné par $\mathcal{O}_i^+ = \mathbf{f}_{\mathcal{O}}(\mathcal{O}_i, \boldsymbol{\omega})$, qui est un modèle à vitesse constante sans rotation donné par

$$\mathbf{m}_j^{\mathcal{O}_i +} = \mathbf{m}_j^{\mathcal{O}_i} + T_s\,\mathbf{v}_j^{\mathcal{O}_i} \tag{24}$$

$$\mathbf{v}_j^{\mathcal{O}_i +} = \mathbf{v}_j^{\mathcal{O}_i} + \boldsymbol{\omega} \tag{25}$$

où T_s est le temps d'échantillonnage du filtre et $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T \in \mathbb{R}^3$ est un bruit blanc gaussien, qui est une perturbation sur la vitesse $\boldsymbol{\omega} \sim \mathcal{N}\{0, \mathbf{Q}\}$.

6.7 BiCam SLAM

BiCam SLAM [Solà *et al.* 2008] vient du constat initial que les systèmes SLAM stéréo sont très myopes, puisqu'ils ne prennent en compte que les amers détectés dans le champ d'observabilité 3D du capteur stéréovision ; l'idée est de combiner SLAM stéréo et SLAM mono qui permet de prendre en compte des amers lointains, éventuellement sur la ligne d'horizon. Donc le SLAM stéréo est traité comme un SLAM coopératif centralisé avec deux caméras indépendantes (mais synchronisées) : la caméra maître est responsable de la détection et de l'initialisation des amers, tandis que la seconde caméra, dite esclave, permet de réobserver la carte SLAM construite par la caméra maître. Se faisant, elle fournit un deuxième point de vue sur les amers. Lorsque ceux-ci se trouvent dans la zone d'observabilité 3D des deux caméras (possible triangulation), alors cet amer peut être directement initialisé avec une représentation euclidienne.

La fonction *BiCam SLAM* réalise donc un SLAM coopératif, qui peut exploiter plus de deux caméras éventuellement non rigidement liées. Le coeur est la fonction SLAM-EKF exploitée pour mettre à jour une carte stochastique qui contient les états de toutes les caméras C_i et de tous les amers détectés jusque là L_j,

$$X^T = [C_1^T \quad \dots \quad C_N^T \quad L_1^T \quad \dots \quad L_M^T] \tag{26}$$

Le vecteur d'état d'une caméra C_i contient sa position et son orientation décrite par un quaternion $[C_i = (\mathbf{r}_i, \mathbf{q}_i) \in \mathbb{R}^7]$, tandis que l'état d'un amer L_j peut être paramétré soit en *inverse depth* $(L_j = i_j \in \mathbb{R}^6)$ soit en Euclidien $(L_j = p_j \in \mathbb{R}^3)$.

Cette représentation doit être incrémentée et mise à jour à chaque acquisition d'images ; N_{obs} amers déjà connus sont traités par une fonction de recherche active, tandis que N_{new} nouveaux amers sont détectés et rajoutés à la carte. La complexité de cet algorithme augmente de manière linéaire en fonction du nombre de caméras et des paramètres N_{obs} and N_{new}, et de manière quadratique en fonction du nombre d'amers dans la carte.

Des capteurs proprioceptifs (odométre, gyro) fournissent les estimées des déplacements faits par les caméras. Les événements "mouvement caméra", "initialisation d'amers" et "mise-à-jour d'amers" sont traités comme dans le SLAM-EKF en sélectionnant le bloc adéquat dans la carte stochastique et dans la matrice de variance, puis en appliquant les modèles de mouvement du robot et d'observation de la caméra.

Bibliographie

[Agrawal *et al.* 2005] M Agrawal, K Konolige and L Iocchi. *Real-time detection of independent motion using stereo*. In Proceedings IEEE workshop on visual motion, 2005.

[Almanza-Ojeda *et al.* 2011] Dora Luz Almanza-Ojeda, Michel Devy and Ariane Herbulot. *Active Method for Mobile Object Detection from an Embedded Camera, Based on a Contrario Clustering*. In Informatics in Control, Automation and Robotics, volume 89, pages 267–280. 2011.

[Bailey 2003] T. Bailey. *Constrained Initialisation for Bearing-Only SLAM*. In IEEE International Conference on Robotics and Automation, 2003.

[Chatila & Laumond 1985] R. Chatila and J.P. Laumond. *Position Referencing and Consistent World Modeling for Mobile Robots*. In 2nd IEEE International Conf. on Robotics and Automation, St. Louis, March 1985.

[Civera *et al.* 2007] Javier Civera, Andrew J. Davison and J. M. M. Montiel. *Inverse Depth to Depth Conversion for Monocular SLAM*. In International Conference on Robotics and Automation, pages 2778–2783, 2007.

[Civera *et al.* 2008] J. Civera, A. Davison and J. Montiel. *Inverse depth parametrization for monocular SLAM*. IEEE Trans. on Robotics, vol. 24, 2008.

[Civera *et al.* 2009] Javier Civera, Oscar G. Grasa, Andrew J. Davison and Montiel J.M.M. *1-Point RANSAC for EKF-Based Structure from Motion*. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2009.

[Dalal & Triggs 2005] Navneet Dalal and Bill Triggs. *Histograms of Oriented Gradients for Human Detection*. In Proceedings of the 2005

IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), pages 886–893, Washington, DC, USA, 2005.

[Davison *et al.* 2007] A. Davison, I. Reid, Molton N. and Stasse O. *MonoSLAM : Real-Time Single Camera SLAM*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007.

[Davison 2003] A. Davison. *Real-time simultaneous localisation and mapping with a single camera*. In International Conference on Computer Vision (ICCV), Nice,France, October 2003.

[Dissanayake *et al.* 2001] G. Dissanayake, P. Newman, Durrant-Whyte H.F., Clark S. and Csobra M. *A solution to the simultaneous localisation and map building (SLAM) problem*. IEEE Trans. on Robotics and Automation, 2001.

[Eustice *et al.* 2005] R. Eustice, H. Singh, J. Leonard, Walter M. and Ballard R. *Visually Navigating the RMS Titanic with SLAM Information Filters*. In Robotics : Science and Systems (RSS), Cambridge,MA,USA, June 2005.

[Gate *et al.* 2009] G. Gate, A. Breheret and F. Nashashibi. *Fast Pedestrian Detection in Dense Environment with a Laser Scanner and a Camera*. In Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th, pages 1 –6, april 2009.

[Knight *et al.* 2001] J. Knight, A. Davison and I. Reid. *Towards constant time SLAM using postponement*. In IEEE/RSJ Conf. on Intelligent Robots and Systems, Maui, Oct 2001.

[Leonard *et al.* 2002] J. Leonard, R. Rikoski and M. Bosse. *Mapping partially observable features from multiple uncertain vantage points*. International Journal of Robotics Research, Jan 2002.

[Lin & Wang 2010] Kuen-Han Lin and Chieh-Chih Wang. *Stereo-based Simultaneous Localization, Mapping and Moving Object Tracking*. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan, October 2010.

[Mahon *et al.* 2008] I. Mahon, S.B. Williams, Pizarro O. and Johnson-Roberson M. *Efficient View-Based SLAM Using Visual Loop Closures*. IEEE Trans. Robotics, Oct 2008.

[Migliore *et al.* 2009] Davide Migliore, Roberto Rigamonti, Daniele Marzorati, Matteo Matteucci and Domenico Sorrenti. *Use a Single Camera for Simultaneous Localization And Mapping with Mobile Object Tracking in dynamic environments*. In In proceedings of International workshop on Safe navigation in open and dynamic environments application to autonomous vehicles, May 2009.

[Mouragnon *et al.* 2009] E. Mouragnon, M. Lhuillier, Dhome M., Dekeyser F. and Sayd P. *Generic and Real-Time Structure from Motion using Local Bundle Adjustment*. Image and Vision Computing, 2009.

[Newman & Leonard 2003] P. Newman and J. Leonard. *Consistent convergent constant time slam*. In International Joint Conference on Artificial Intelligence, Acapulco Mexico, August 2003.

[Newman *et al.* 2009] P. Newman, G. Sibley, Smith M., Cummins M., Harrison A., Mei C., Posner I., Shade R., Schröte D., Cole D. and Reid I. *Navigating, Recognising and Describing Urban Spaces With Vision and Laser*. International Journal of Robotics Research, To Appear 2009.

[Newman 1999] P. Newman. *On the structure and solution of the simultaneous localisation and map building problem*. In Ph.D. dissertation, Australian Centre for Field Robotics - The University of Sydney, March 1999.

[Smith & Cheeseman 1990] R. Smith and P. Cheeseman. *Estimating uncertain spatial relationships in robotics*. Autonomous Robot Vehicles, 1990.

[Solà *et al.* 2008] J. Solà, A. Monin, M. Devy and T. Vidal-Calleja. *Fusing monocular information in multi-camera SLAM*. IEEE Trans. on Robotics, vol. 24, no. 5, pages 958–968, 2008.

[Solà 2007] J. Solà. *Towards visual localization, mapping and moving objects tracking by a mobile robot : a geometric and probabilistic approach*. In Doctorat, Institut National Polytechnique, Toulouse, Février 2007.

[Thrun *et al.* 2001] S. Thrun, D. Fox, W. Burgard and F. Dellaert. *Robust Monte Carlo Localization for Mobile Robots*. Artificial Intelligence, 2001.

[Thrun *et al.* 2004] S. Thrun, Y. Liu, D. Koller, A. Ng, Ghahramani Z. and Durrant-Whyte H. *Simultaneous localization and mapping with sparse extended information filters*. The International Journal of Robotics Research, July-August 2004.

[Thrun 2002] S. Thrun. *Robotic mapping : A survey*. Exploring Artificial Intelligence in the New Millenium, 2002.

[Veit *et al.* 2007] T. Veit, F. Cao and P. Bouthemy. *Space-time A Contrario Clustering for Detecting Coherent Motions*. In Robotics and Automation, 2007 IEEE International Conference on, pages 33 –39, april 2007.

[Viola & Jones 2002] Paul Viola and Michael Jones. *Robust Real-time Object Detection*. International Journal of Computer Vision, vol. 57, no. 2, pages 137–154, 2002.

[Wang 2004] Chieh-Chih Wang. *Simultaneous Localization, Mapping and Moving Object Tracking*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2004.

[Wangsiripitak & Murray 2009] S. Wangsiripitak and D.W. Murray. *Avoiding moving outliers in visual SLAM by tracking moving objects*. In Robotics and Automation, 2009. ICRA '09. IEEE International Conference on, pages 375 –380, may 2009.

[Williams 2002] D.G. Williams. *An efficient approach to the simultaneous localisation and mapping*. In IEEE International Conference on Robotics and Automation, Washington, DC, May 2002.