UNIVERSITÀ DEGLI STUDI DI MILANO - BICOCCA

Dipartimento di Informatica Sistemistica e Comunicazione

Ph.D. in Computer Science
XXV. Cycle

Robotic Perception for Autonomous Navigation

Advisor: Prof. Domenico G. Sorrenti
Co-Advisor: Prof. Silvio Savarese

Doctoral dissertation by: *Axel Furlan*

Con un occhiolino da lassù
ed un sorriso da quaggiù
ai miei genitori.

Contents

Chapter 1

Introduction

Robots are artificial agents able to perform useful tasks. Such tasks span from very simple assignments, like automatically performing repetitive actions, to very complex ones, like autonomously driving in a crowded urban environment. Robotic agents are becoming more and more common in everyday life and people are getting used to their presence. Obtaining an autonomous robot for our home is nowadays as simple as going to the closest market and choose the one that better fits our needs, from home-cleaning to garden-keeping and so on. Yet, even to perform such simple tasks, the complexity of the underlying system is usually pretty high.

Autonomous robots

When we buy a robotic agent to help us with a task, we usually want it to operate in an environment it has never been in before. The robot must thus be able to adapt to completely new working conditions while keeping its ability to perform the requested task. It will need to somehow perceive the surrounding space, the objects within it and the potential obstacles on its way. It will also need to plan its actions in order to achieve the final goal of the assigned task. Finally, it will need to execute the planned actions and promptly react to unexpected modifications of the surrounding space. These three steps are the robotic primitives that compose the main cycle of an autonomous agent. The main cycle is endlessly iterated from the moment the robot is turned on and until its shutdown.

Robotic perception

Robotic perception is the research branch that copes with the problem of sensing the environment surrounding a robotic agent. It is of crucial importance to the success while performing a task, yet it covers the widest spectrum of unpredictability in autonomous robotic systems. It is delegated to collect useful and reliable informations about, among the most important, the static geometric structure of the surrounding environment, its higher level interpretation and the dynamics of the objects moving within it. Informations are collected by means of *sensors*, which are any type of hardware able to convert physical magnitudes into digital signals. Acquired digital signals must then be processed by robotic perception algorithms, in order to generate the desired final information that will be used by the robotic agent.

The work behind this thesis spans the following relevant problems of robotic perception, in the attempt of providing the informations a robotic agent needs to safely operate in an environment.

Mapping the environment and self-localization

The first step towards understanding how the environment the robotic agent is moving within looks like is the creation of a metric representation of the surrounding space. Typical sensors used to perform this kind of reconstructions are cameras and laser scanners, although different sensors can be used. Neither cameras nor laser scanners are in general able to gather enough information about the surrounding space in a single shot, thus, while the robot is moving around, integration of the informations over time is usually performed. Robot motion needs to be properly modeled and is often part of the estimation result. This is called the problem of *Simultaneous Localization and Mapping - SLAM*. This problem has been tackled in literature from different starting points and the resulting techniques can be roughly divided into *global* and *local* mapping, the former requiring global consistency among observations and robot poses, the latter tackling only a small, local subproblem. Among the global techniques, we can further distinguish between *on-line* and *full* approaches. On the one hand, on-line (sequential) approaches process a stream of consecutive measurements (*e.g.* laser scans or images), usually captured at high frame rates ($\geq 20\,fps$). From this, an accurate motion model of the observer can be estimated. These are prop-

erly called SLAM systems. On the other hand, full (batch) approaches process a set of sparse measurements, acquired from different points of view and not necessary in the correct temporal order, that do not need to be correlated by proper physical motion. These are called Structure from Motion approaches. The output of global mapping systems are typically a map, consisting in (more or less) sparse 3D points, and the estimated poses of the observer (the robotic agent) at the time steps the measurements were acquired. In some approaches the map consists of line segments or planes, instead of points.

Among local mapping techniques, the closest to the purpose of autonomous navigation is the so-called Odometry problem (Visual Odometry in the case the sensors are cameras). It does not require a global map to be created; instead, a local map is kept updated within a sliding temporal window and is used to estimate the relative robot motion.

Interpreting the scene

As the reader may observe, the reconstructed maps generated by mapping systems alone provide too sparse information to be used for safe action planning and execution. For complex action planning, higher level description of the surrounding environment may be needed. In the last decade, more and more sophisticated *scene understanding* approaches are arising, that are able to provide more dense and semantically meaningful models of the scene. Here “higher semantic” description means that elements of the maps are not only correlated by geometrical and physical constraints, but also by meaningful labels attached to each element and by descriptions that involve two or more elements of the map. As a simple example, it is enough to think about an indoor corridor. A mapping system would output a map in which hundreds of points would lie on the main surfaces of the corridor. Instead, a higher level interpretation of the scene would, for example, determine the existence of a ground plane, label it as ”floor” and as such as a surface over which the robot can navigate and plan its actions. Beside the geometric structure, a scene understanding algorithm would also try to estimate static objects in the scene and label them as obstacles or as elements of the scene with which to interact while performing a task.

Object detection and tracking

In previous sections we have been considering static components of the environment that surrounds a robotic agent. Yet, in most applications the robot will be required to operate in presence of moving objects, like people in an indoor environment or cars, pedestrians and bicycles in outdoor circumstances. An important branch of robotic perception regards the study of such dynamic parts of the scene, in the attempt to detect moving objects and to track their motions. Information about the dynamics of objects in the scene can be useful during the planning phase to predict where such objects may be found in the near future, *e.g.* in order to evaluate actions that avoid collisions. The act of detecting objects is very often accompanied by the effort of categorizing them. Object categorization may be useful, for example, because different object categories may implicate different dynamic models, or because the assigned task demands to interact with a certain category of objects in the scene. Tracking systems can gain a great boost in performance by integrating information about the static elements of the scene and, especially, about the self-estimated motion of the robotic agent itself. Mapping and scene understanding approaches can also improve the quality of their reconstructions by exploiting information about dynamic elements of the scene. Indeed, as will be shown later in this chapter, there exists a very strong interaction between these three branches of robotic perception.

Planning actions

Exploiting the informations provided by the perception systems, the robotic agent must be able to find its way of performing the requested task. In general, a task may involve mere navigation, thus planning the path the robot should follow to reach the destination point, or more complex interaction with elements of the scene, like manipulation of objects and interaction with humans. In any case, the output of an action planner is a set of intermediate configurations of the robot (*e.g.* poses with respect to the environment, conformation of joints, ecc.) that are needed to reach the final configuration. This research branch was not part of the work performed in this thesis.

Executing actions

Once the future actions have been planned, the robotic agent must be able to execute them as faithfully as possible. The quality of the execution will affect the performance of the global task. Action execution is the casting of a bridge between the algorithmic reasonings of the planning system and the real, imperfect, noisy hardware support of the robotic agent. The action execution sub-cycle is usually performed at a higher frequency with respect to the planning chain and is in charge, among other things, of conducting the robotic platform to the planned configuration, while respecting the physical constrains of the robot (and, in the case of an autonomous car, of its passengers). A small contribution was made by the author of this thesis in the field of action execution and is described in Appendix B.

About the work in this thesis

This thesis presents the research work the author carried on during his PhD on the topic of robotic perception for autonomous navigation. In particular, the efforts focus on the Self-Localization, Scene Understanding and Object Detection and Tracking problems, proposing for each of these three topics one or more approaches that present an improvement over the state-of-the-art. In some cases the proposed approaches mutually exploit the generated information to improve the quality of the final results. In particular, the Object Detection and Tracking approach presented in Chapter 4 exploits self-localization information to project object hypotheses into the surrounding 3D world and track them in the 3D space. Also, the Indoor Scene Understanding approach presented in Chapter 5 bases its initialization process on the same self-localization information, while the integration of object detections in the scene estimation process is currently under development and will therefor not be discussed in this thesis. Finally, the real-life implementation of an autonomously driving car presented in Chapter 6 exploits both the self-localization approaches presented in Chapter 3, while the adaptation of the approaches presented in Chapters 4 and 5 to the outdoor autonomous driving task are currently being implemented.

A brief introduction on all the ways approaches tackling the three mentioned problems may benefit from mutual interaction is presented in the following sections.

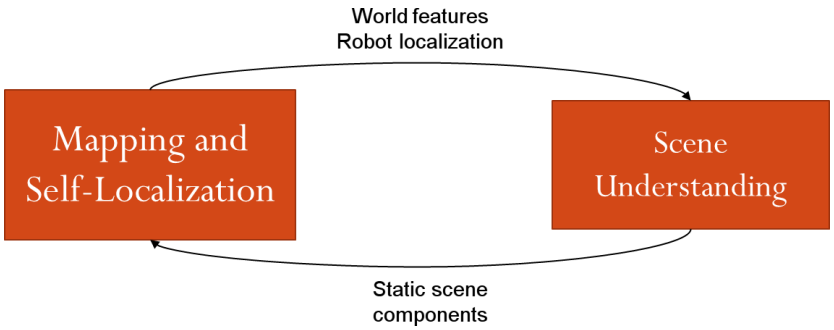


Figure 1.1: Diagram of the interaction between Mapping and Self-Localization and Scene Understanding systems. The left-to-right arrow shows that sparse 3D reconstruction and observer's pose estimation can be used as a prior to build higher level scene interpretation hypotheses. The righ-to-left arrow shows that higher level information can be exploited to introduce strong geometric constraints in the mapping process.

SLAM and Scene Understanding interaction

Both Mapping and Self-Localization and Scene Understanding systems aim at the creation of a map of the surrounding environment, possibly including its geometric structure and static objects. The main difference is in the level of semantic description of the estimated map. Despite this difference, the two problems are closely related and both systems can considerably improve their performances exploiting each other's informations. For example, a Scene Understanding system could start generating scene reconstruction hypotheses based on a sparse SLAM reconstruction, instead of hypothesizing them from scratch, which could make the difference in a realistic-time computation. On the other hand, a SLAM system could cluster the sparse 3D features it usually works on, according to the higher level scene interpretation, thus introducing strong geometric constraints in the mapping process. Figure 1.1 shows a diagram of this type of interaction.

SLAM and Object Detection and Tracking interaction

Mapping and Self-Localization systems process sensor data in order to generate a consistent map of the surrounding environment. Maps are commonly desired to represent the static components of the space, thus, if some prior on which parts of the observed scene are not static (*e.g.* moving objects) is given, the mapping algorithm could prune out those parts and be able to generate a better reconstruction. On the other side, as

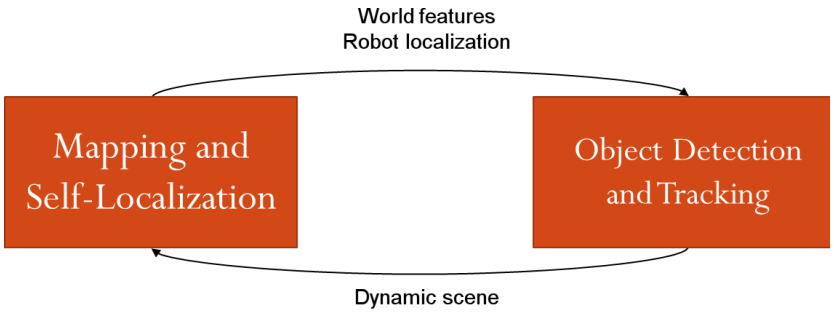


Figure 1.2: Diagram of the interaction between Mapping and Self-Localization and Object Detection and Tracking systems. The left-to-right arrow shows that sparse 3D reconstruction and observer's pose estimation can be used to prune out false positive detections and physically inconsistent motion estimations. The right-to-left arrow shows that dynamic elements of the observed scene can be removed from the mapping process to improve its performances.

shown in Figure 1.2, Object Detection and Tracking systems can exploit precious informations about the observer's motion and pose with respect to the environment to prune out false positive detections and physically inconsistent object motions. While this latter direction of interaction is pretty common, the former has rarely appeared in the literature.

Scene Understanding and Object Detection and Tracking interaction

This kind of interaction involves two higher semantic estimation problems. In fact, both Scene Understanding and Object Detection and Tracking aim at generating a meaningful interpretation of what is happening in the surrounding space, more than merely estimate anonymous and semantically uncorrelated sparse 3D features. Thus, as shown in Figure 1.3, higher level information about the scene geometric structure can allow to further prune out semantically wrong detections and motions (*e.g.* flying terrestrial vehicles, objects moving through walls, ecc.). On the other hand, information about objects moving in the scene can influence the interpretation of scene structure. An explicative example of this latter case would be the case where a scene structure hypothesis includes a wall between the observer and a reliably detected object. In this case, a strong prior on that wall can be applied to either consider it transparent or placed in the wrong position.

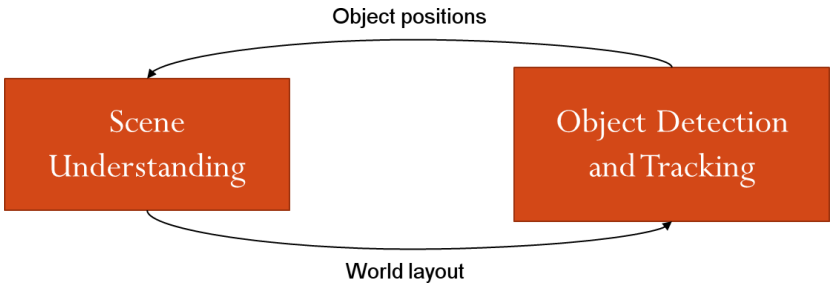


Figure 1.3: Diagram of the interaction between Scene Understanding and Object Detection and Tracking systems. The left-to-right arrow shows that scene geometric structure (layout) can be used to further prune out false positive detections and physically inconsistent motion estimations. The right-to-left arrow shows that reliable object detections in the observed scene can be exploited to introduce strong constraints on scene structure hypotheses.

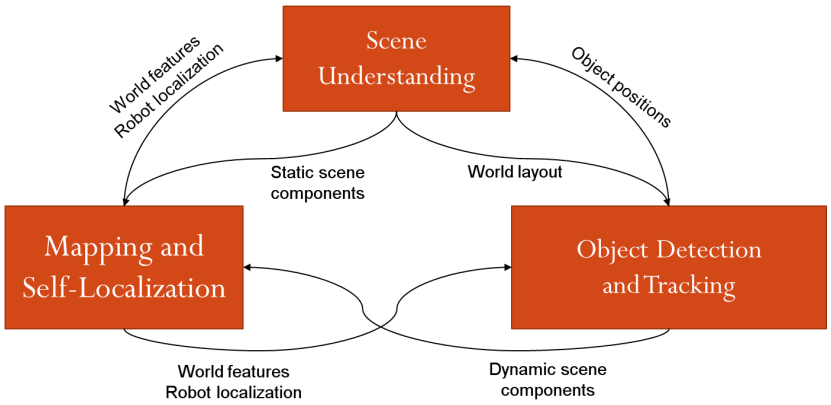


Figure 1.4: The global chart of interactions.

Global overview of the research work

Figure 1.4 shows the global diagram of possible interactions between the three problems mentioned so far. Throughout this thesis, innovations will be presented in the fields of Self-Localization (Chapter 3), Object Detection and Tracking (Chapter 4) and Scene Understanding (Chapter 5). When tackling these problems, mutual interactions were exploited in some cases, while in other cases they were collapsed in the attempt of solving the problems in a joint fashion.

Finally, a real world application is presented (Chapter 6), which reports all the work that has been carried on to robotize (high level software, mechanics and electronics) an autonomous vehicle (a golf cart)

and make it capable of autonomous driving in crowded environments. This example of application clearly shows the importance of high level algorithmic reasoning, including the above mentioned problems, to enable a robotic agent to safely operate in common, everyday environments. The presented autonomous vehicle has been involved in several live demos throughout these years, proving its effectiveness and capabilities to the wide public.

Chapter 2

Related work

This chapter presents an overview of the state-of-the-art approaches for the main topics discussed in this thesis.

2.1 Robot Self-Localization

It has been largely discussed the possibility of estimating the path of a robot, as it moves, only exploiting proprioceptive sensors. Although interesting results have been achieved on short distances and with accurate knowledge of the motion patterns [1], current localization approaches cannot, in general and over long distances, achieve sufficiently precise results by using purely dead reckoning, e.g., IMU data integration or wheel based odometry in the case of mobile robots [2, 3, 4]. It must be also noticed that in urban environments the GPS system, apparently an immediately available solution, has an absolutely not adequate reliability, with respect to the localization and navigation requirements, due to its precision and the frequent lack of signal [5] [6]. Global and local robot self-localization approaches have thus been proposed, that exploit exteroceptive sensors such as ranging devices (lasers, sonars) and cameras.

2.1.1 Laser-based Global Localization

Most of the state-of-the-art solutions for the 2D - 3DoF localization problem are primarily designed for indoor robotic environments or, more generally for those settings where the analysis of the motion in a 3D

space can be simplified, favoring an estimation of the robot pose, limited to a 3DoF pose in the 2D plane [7, 8, 9].

Full 3D approaches known in the literature [5, 6, 10] use methodologies that adapt 2D movements to a 3D space. This approaches adopt a 3DoF probabilistic motion model in 2D that do not allow accurate modeling of the uncertainty of a 6DoF movement in a 3D space. In [6] the robot poses are modeled only in 2D, i.e. the state vector includes only the components x, y and ϑ (the yaw angle). The other three components z, the roll angle φ, and the pitch angle ψ, are calculated from the 2D estimate and from the structure of the ground surface. Furthermore, the motion model does not consider the interactions between the different error source components, introducing uncertainty on the movement's single components independently. The independence between the single pose components is also assumed in [10].

In [5] a multilevel environment representation called multi-level surface maps is used. This technique is proposed as an extension of the elevation maps used in [11] and introduced in [12], and allows modeling vertical structures within a grid map used in localization with laser range finders. However, these structures do not allow the representation of a typical urban outdoor complex situation, like a bridge or a multilevel parking lot. Furthermore, in [5] the motion model, more sophisticated with respect to the one proposed in [6], uses as a starting basis an evolution of the model introduced in [13] and similar to that illustrated in [3], again a purely two dimensional motion model.

2.1.2 Vision-based Localization and Mapping

There are several state-of-the-art approaches, proposed in the last decade, that tackle the Localization and Mapping problem from different starting considerations. In particular, a big distinction should be made between real-time and non-real-time approaches.

Among the formers, the SLAM (Simultaneous Localization and Mapping) methods are primarily based on Bayesian filtering [14, 15, 16, 17, 18, 19, 20, 21, 22], although Strasdat *et al.* [23] recently questioned this historical approach. In the past few years, research in this field has focused on the problems of:

- Improving the faithfulness of the represented uncertainty [15, 18, 20], in particular by changing the way map points are parametrized in the filter state. The three most known parameterizations are the

Inverse Depth [15], the Inverse Scale [18] and the Anchored Homogeneous Points [20].

- Improving the mapping scalability by splitting the global map into sub-maps, parameterizing the relationships between sub-maps and performing loop closure [24, 25, 26, 27].
- Improving the computational efficiency by restating the SLAM problems as sparse non-linear problems and retrieving exact solutions [28, 25, 29].

A key feature of all SLAM approaches is to incorporate innovation in the filter from each image in a sequence, approach questioned by the PTAM approaches which, instead, rely on bundle adjustment optimization of a smaller number of key-frames [30, 31, 32, 33]. In the PTAM approach the map is generated in a separate, non-real-time thread performing structure from motion reconstruction, while the camera localization is performed in real-time exploiting Lie algebra based optimization.

The main drawbacks of all the so far mentioned techniques span scarce robustness to noisy initialization, high sensibility to the fidelity of the motion and noise models, bad scalability and the limited or absent ability of refining past estimations.

The group of non-real-time approaches is primarily focused on the global consistency of the map and of the camera poses. Global bundle adjustment techniques have been investigated to solve this problem. These methods usually process sparse sets of images from which partial geometry is recovered and optimized [34, 35, 36]. They are often used to model large scale environments such as cities [37, 38, 39]. The main drawback of these methods is the computation time needed to perform the matching among all the images. Recent implementations exploit GPU computing for feature extraction and matching [40], thus consistently speeding up the performances. Nevertheless, real-time computation for SfM on all frames of a video sequence is not yet possible on common computers.

2.2 Object detection and tracking

2.2.1 Object detection

An ideal object detection system should be able to detect its targets in the observed scene independently on their orientation, position, scale,

and also with some degree of occlusion.

An important challenge is to gain the capability of generalization, i.e., detecting the objects of a certain class and not only one instance, despite the differences in the object shape from one instance to the other in the class. An important thrust forward was introduced with corner detectors [41, 42, 43, 44, 45, 46, 47], blob detectors [48, 49, 50, 51, 52, 53], and region descriptors [54, 49, 55, 50, 52]. The combination of detectors and descriptors provide means to describe image patches in a way that is invariant not only with respect to orientation and position, but also to scale and partial occlusion. Basing on such descriptors, higher level object detectors can be developed.

Bag of words

Csurka *et al.* in [56] first introduced the term *bag of keypoints* to describe methods based on sets (the bags) of visual cues (the words, in some cases called keypoints or features), learned during a training phase and used to determine if, given an unknown image, there are instances of objects from the learned categories present in it. In particular, in [56] affine covariant regions described by means of the SIFT descriptors are used to collect the visual cues, which are then clustered using k-means to build the visual dictionary (the bag of words). At detection time, an SVM is used to decide whether an object is present in the image or not. In [57], the same authors expand the previous approach by using a boosted classifier instead of the SVM.

Please refer to the work presented by Zhang *et al.* in [58] for a comprehensive study of bag of words based methods, evaluated in conjunction with different feature detectors and descriptors.

Recognition with segmentation

While bag of words techniques are limited to indicate if and roughly where an object is located within an image, the problem of accurately segmenting its contours is very challenging, in particular in presence of partial occlusions.

When an instance of object is detected, segmentation can be performed back-projecting the object model into the image [49], by fitting a trained segmentation model to the scene [59, 60], or by pre-segmenting the image in regions and then matching them with the learned parts of the model [61, 62, 63, 64]. These latter approaches were conceived to

better tackle the problem of non-rigid objects, for example humans. In fact, in shape-based object recognition, see e.g., [65], the usage of statistics on the spatial frequencies of the descriptors is proposed. This is a quite effective approach whenever the objects to be detected have more or less constant shapes, and with regular textures; these approaches deal well with occlusions and position, orientation and scale changes. Such approaches do not perform well, on the other hand, on objects such as, e.g., people, that can appear in a wide set of different shapes and postures.

In 2003 Leibe and Schiele in [66] proposed a method that does not require an exact a priori knowledge of the forms and shapes of the objects to be detected. This feature makes the system more complex, as it allows large intra-class variations. In [67] the same authors, with Leonardis, formalized their approach in the so-called ISM (Implicit Shape Model) and, later [68], with Mikolajczyk, they extended it to include different descriptors on common probabilistic grounds. At training time, the ISM based approaches learn a set of local segmentation masks that will be used at recognition time to generate the final object segmentation.

Kumar and Hebert in [69] and He *et al.* in [70] approach the segmentation problem by assigning each pixel of the image a class label and solving for the whole image by energy minimization and Bayesian inference, respectively. A similar approach is the one presented in [71] and called *TextonBoost*.

2.2.2 Object tracking

The problem of object tracking identifies with the ability of a system of preserving the correct identity of an object through time despite detection errors, occlusions and presence of other (similar) objects. There are many state-of-the-art approaches for object tracking, covering substantially different application domains such as, for example, static surveillance camera systems, mobile robotic vision systems for obstacle avoidance, etc.

Background modeling object tracking

The first, intuitive way of discriminating objects and estimating their motion is to keep an updated model of the scene background. Any object can then be extrapolated as an outlier to this model. This intuition is straightforward in the case of a static observer, while it becomes very

complicate if the observer is moving. In the former case, the most relevant issues are related to the model ability to adapt to scene changes, *e.g.* slow and sudden changes in illumination, waving trees etc.

Among the first proposed approaches, Jain and Nagel [72] tackle the adaptation problem using frame-difference accumulation. Mixture of Gaussians have been proposed in [73] to model the per-pixel color distribution, extended in [74] to add robustness with respect to shadows, while texture-based background modeling is proposed in [75]. Object instances extrapolated from the background then need to be tracked within a spatial-temporal context [76]. An efficient GPU-based background modeling implementation is presented in [77].

When the observer is not static, the background can be modeled in 2D by, for example, compensating its motion exploiting the optical flow [78, 79, 80], or in 3D, exploiting for example Structure from Motion information [34].

Fixed model tracking

As their name says, these kind of methods build a model of the object being tracked off-line or at the first frame and keep it unmodified throughout the video sequence. At each frame the model can be searched for in the whole image [81, 82, 83, 84, 85, 86] or in a smaller subspace (exploiting motion prediction) [87, 88] to prevent drifting. Changes in model viewpoint can lead to mismatch the tracked target, thus several approaches proposed models that can deal with affine transformations [81, 82], that exploit region descriptors for template matching [83, 84] or that build a multi-view model [85] that can be used in conjunction with an SVM [86]. Techniques that rely on motion predictions face the problem of social interactions, i.e., for example, the fact that the trajectory of human walking in a crowded environment will be influenced by the trajectories and behaviors of other people. The work presented in [89] associates a motion model with social reasoning to account for these common phenomena.

Adaptive model tracking

As said in the previous section, changes in model viewpoint, illumination conditions and object non-rigidity can seriously degrade the quality of the tracking. In literature there are many approaches that tackle these

difficulties by keeping an updated model of the tracked object, dynamically changing throughout the tracking. Conceptually, the main drawback of such approaches is the risk of adapting too fast to the changes, leading the tracker to detach from the originally tracked object and attach to a different one or even to the background. We can roughly distinguish two main categories of adaptive model tracking techniques: generative and discriminative.

In the generative case, the tracker keeps an updated model of the object based on optical flow features [90, 91, 44, 92], color histograms and mean-shift [93, 94], object sub-parts [95], modeling partial occlusions [96] or combinations of two or more of the previous [97].

In the discriminative case, a (usually binary) classifier is trained and updated on-line and used to discriminate between background and object foreground at each frame. For example, in [98, 99] the idea of including the background in the training set is used to automatically discriminate the features to be used in a mean-shift tracker, while in [100, 101] a set of weak classifiers are combined together to generate a strong global one.

In the recent years, hybrid approaches that combine fixed and adaptive model tracking have also been proposed, such as [102, 103, 104].

2.3 Indoor scene reconstruction

The work presented in Chapter 5 is at the intersect of two major fields of Computer Vision: Indoor Scene Reconstruction, and Structure from Motion. While the end result is that of Indoor Scene Reconstruction, the proposed approach hinges on the success of sparse reconstruction methods, such as those studied in the Structure from Motion (SfM), Simultaneous Localization and Mapping (SLAM), or Parallel Tracking and Mapping (PTAM) literature.

All of the approaches mentioned in Section 2.1.2 aim at a camera motion estimation and a sparse reconstruction of the environment in terms of 3D point clouds. For a better interaction with the environment, robots and human-interacting applications need a semantically higher level representation of the environment. Such representation should allow to easily infer navigability maps and movement boundaries of the surrounding space.

Scene layout estimation

Many works have been proposed to solve the problem of indoor scene estimation, addressing it with techniques from different research areas like Bayesian filtering and Machine learning. Most of them face the problem of estimating the indoor layout from single images [105, 106, 107, 108, 109, 110, 111, 112, 113, 114]. Some other works use sparse images or image sequences [115, 116, 117, 118, 119].

Lee *et al.* [106] proposed a method to generate plausible interpretations of a scene from a collection of line segment. They estimate vanishing points and infer the three dimensional structure of the indoor scene in the image. A slightly different approach was used by Hedau *et al.* [107]. In this work they model 3D room hypotheses with a parametric 3D box (cuboid) and generate room hypotheses by sampling pairs of rays from two of the estimated vanishing points. They also propose a structured learning approach to rank sampled box layout hypotheses.

In [110] Lee *et al.* added volumetric reasoning to their previous work in order to incorporate physical constraints stemming from the 3D interaction between objects. They also use more geometrical evidence from the image in order to rank hypotheses (Orientation maps and Geometric context). In [111] Hedau *et al.* added an object detector to their previous work, which generates 3D object hypotheses by sliding cuboids in the room layout generated as in [107] and by scoring them with a linear function learned with an SVM approach.

Wang *et al.* [109] address the problem of jointly recovering the structural layouts of rooms and the furniture present in it. They tackle the problem of heavy clutter and intra-class appearance variation in indoor scenes by modeling them with latent variables and by training the detector to disambiguate furniture and room layout. Their approach does not need the clutter to be hand-labeled in the training set.

Gupta *et al.* [112] introduce physical representation of objects, accounting for their volume, mass and mechanical configurations between objects. This representation allows them to take into account global geometric constraints between volumes and the laws of statics. They build their representation starting from an empty scene reconstruction and iteratively adding consistent and physically plausible components. Their work focuses on outdoor scenes.

Gupta *et al.* [113] presented a human-centric approach for indoor scene understanding. Their work aims at jointly estimate the scene struc-

tural layout and model possible human actions within it. They explicitly use human physical models to validate hypotheses.

Schwing *et al.* [114] propose an efficient decomposition of higher order potentials used in [109] to pairwise potentials extending the concept of integral images to geometry. As a result both learning and inference can be performed orders of magnitude faster, which allows to avoid search space reductions and thus leading to better estimations.

All of the above mentioned approaches tackle the problem of inferring the indoor scene layout from single images. While being a challenging goal, this problem does not allow for exploration of the scene or for any interaction of robots or humans with the environment. Furthermore, except for [114], none aims at performing in near-real-time. Pursuing the need for reliable scene understanding while exploring a scene, which is typically the case, for example, of an autonomous robot, multiple-view approaches have been explored.

Sinha *et al.* [116] proposed a multi-view stereo method to generate piecewise planar depth maps from sparse images. They first estimate 3D point clouds with Structure from Motion and 3D line segments. To enforce plane hypotheses they then use 3D point and plane incidence and photo-consistency cues. Finally they generate the piecewise planar depth map solving a Multi-Label MRF minimization problem with graph-cuts. The minimization formulation exploits vanishing directions and free space violation constraints for visibility rays.

Tsai *et al.* [119, 120] presented a work where they focus on real-time indoor scene estimation tasks for mobile robot applications. Compared to other state-of-the-art approaches, this work is the most related to the work presented in Chapter 5. In fact it shares the same idea of exploiting video sequences to validate and support layout hypotheses and the decision of dropping the Manhattan world assumption. It also shares the decision of avoiding any kind of learning, which, due to the huge variability of indoor scenes, would require an extensive training dataset. In their work, however, they focus on indoor scene reconstruction for mobile robots navigation. This allows to greatly simplify the problem, since information such as the roto-translation between the camera and the floor plane can be supposed to be known. In the approach proposed in [120] they also assume the exact pose of the robot at each time step to be known. This information is often provided when the camera is mounted on a rigid mobile robot (no suspensions on the wheels) equipped with a laser-based self-localization system, but this is in general not the case

when the camera is either mounted on a flying drone or is represented by a hand device held by a human. Furthermore, they generate hypotheses by projecting on the 3D ground plane 2D floor-walls intersection lines observed in the images, which means that if that portion of the scene is not observed, the hypotheses will not be generated.

The work presented in Chapter 5 focuses on estimating the indoor layout from a fully 6DoF moving monocular observer (which movements are unknown), with no requirements on which part of the scene is being observed, aiming at performing in real-time.

Chapter 3

Robot Self-Localization

This chapter presents the research work [1] on the Self-Localization problem of a mobile robot. Two main aspects of the complex global problem are dealt with, i.e. the motion model used to represent the uncertainties over the physical dynamics of a wheeled robot and the localization of a visual observer within a known 3D map.

3.1 An effective 6DoF motion model for 3D-6DoF Monte Carlo Localization

This research activity deals with the probabilistic 6DoF motion model of a wheeled road vehicle. It allows to correctly model the typical errors introduced by mere dead reckoning motion propagation systems. To stress the fact that an appropriate motion model is of crucial importance, another model, which was previously developed, is shown not to allow a correct representation of the uncertainty, therefore misguiding 3D-6DoF Monte Carlo Localization. Experiments are also presented, testing the proposed approach on the real world autonomous vehicle presented in Chapter 6, to demonstrate that the proposed model improves the results of a Monte Carlo Localization system allowing a consistent determination of the 6DoF vehicle pose.

[1] Pleas refer to Section 3.3.1 for the acknowledgments.

3.1.1 Introduction

Mobile robot localization is a challenging task that has been intensively analyzed over the last few years, see e.g. [5] [6] [3] being itself at the base of any task that could be assigned to a mobile robot.

The accuracy a robot can achieve in localization assumes a key role in order to avoid collisions against the structural elements belonging to the surrounding environment, e.g. against static and dynamic objects placed within the scene. This becomes even more critical in the field of autonomous driving, where a proper and accurate localization is of primary relevance because on it depends the individuals safety.

The risk of collisions against the environmental static components such as sidewalks, walls, guardrails, or against the environmental dynamic components such as other vehicles, pedestrians and animals,m ust be reduced to a minimum, foreseeing a series of emergency procedures to be activated in case of unexpected situations.

In urban settings, autonomous driving closely relates to mobile robotics problems, in particular whene there is need to have a global localization of the vehicle. As discussed in Chapter 2 robus localization cannot be managed using purely dead reckoning, i.e. IMU and wheel based odometry [2, 3, 4]. Wheel sliding, e.g., due to contact with the ground surface, weather conditions, unexpected values of the wheels diameters, etc., require the use of external sensors and the corresponding algorithms, to determine the vehicle position [3]. It must be noticed that in urban environments the GPS system, apparently an immediately available solution, has an absolutely not adequate reliability, with respect to the localization and navigation requirements, due to the frequent lack of signal [5] [6].

While the state-of-the-art provides different solutions for the 2D - 3DoF localization problem, these solutions are primarily designed for indoor robotic environments, where the analysis of the motion in a 3D space can be simplified to a 3DoF pose in the 2D plane. The inadequacy of these simplifications in a urban outdoor situation has driven this research work to develop a different probabilistic motion model, based on the modeling of a spatial generic movement considering all the components of the 6DoF state vector. The proposed model, which is adaptable to different types of vehicle kinematics (differential, Ackerman), accommodates 6DoF movement predictions even without a complete “control” vector, i.e., when some component is missing; this can be the case of a

wheeled vehicle not equipped with an inertial measurement unit.

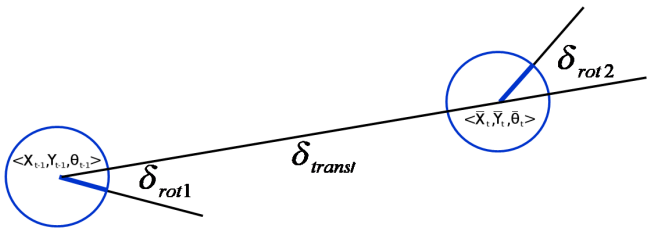
The next section introduces the proposed motion model, while Section 3.1.4 compares the proposed model with a previously developed model, used as baseline method for comparison, which made the 3D-6DoF Monte Carlo Localization fail, in order to clarify the relevance of an appropriate and accurate motion model. Section 3.1.5 concludes by presenting the experimental results.

3.1.2 Proposed motion model

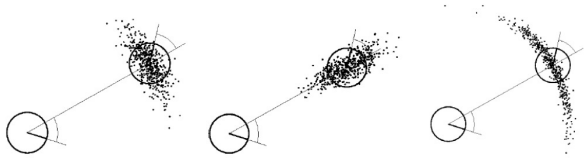
The proposed motion model bases on the 2D-3DoF formulation presented in [3, Sect. 5.4]. In that work a displacement between two robot poses is divided into a sequence of 3 steps: an on-site rotation δ_{rot1}, a translation δ_{transl}, and a last on-site rotation δ_{rot2}. A graphic representation of this well-known motion model is shown in Figure 3.1(a). This decomposition allows the introduction of the uncertainty on each step in the form of a normally distributed error. These errors are zero-mean and are parameterized by a standard deviation that is dimensioned according to the disturbances acting on each step. The composition of the applied uncertainties on each step gives a realistic uncertainty distribution affecting the pose after the application of the motion. Figure 3.1(b) shows how particles in a Monte Carlo Localization system spread accordingly to the motion model.

Similarly, the full 3D motion model is required to account for the motion in all the 6 degrees of freedom of a robotic agent and, at the same time, it should divide the whole displacement in a sequence of steps in order to apply a parameterized uncertainty on each of the components of the movement. In the same fashion as the above method, the composition of the steps with the applied uncertainty should result into a realistic uncertainty distribution on the final pose, i.e., after the application of the motion.

Common approaches for extended odometry (dead reckoning) are able to merge the estimates of displacements in all 6 DoFs, by integrating the odometry information, i.e., displacements estimated from the rotation of the wheels, with the inertial informations provided by IMU sensors. In particular, Δx, Δy, and $\Delta \vartheta$ (yaw) can be output by a wheel odometer, while the $\Delta \varphi$ (roll), $\Delta \psi$ (pitch), and Δz, can be output by an IMU. Unfortunately, often robotic platforms are not equipped with inertial measurement units. To overcome this problem, the proposed method



(a)



(b)

Figure 3.1: (a) The well-known 2D-3DoF motion model from [3, Sect. 5.4]. The motion between two consecutive robot poses is torn down into three steps, i.e. on-site rotation δ_{rot1}, translation δ_{transl}, and on-site rotation δ_{rot2}. (b) Particles of a Monte Carlo Localization system spreading accordingly to the motion model and generating the typical banana-shaped posterior distribution.

includes a set of additional parameters to be applied when the extended odometric readings lack some components, e.g. the IMUs.

The state vector has six components, to represent the pose of a rigid body in a 3D world $\mathbf{x}_t = |x, y, z, \varphi, \psi, \vartheta|'$. Let us group the first 3 components in $position_t$ and the last 3 in $orientation_t$, so that $\mathbf{x}_t = |position_t, orientation_t|'$.

The proposed decomposition of a displacement bases on six steps, which can be grouped in 2 different sets: 3 steps to define the new position $\overline{position}_t$ [2], and 3 steps to define the new orientation $\overline{orientation}_t$. On each step a normally distributed uncertainty will be added. Let us now review the first 3 steps, with reference to Figure 3.2, which give $\overline{position}_t$.

1. Rotation δ_{yaw_1}, which represents a rotation around the Z axis, is necessary to align the $orientation_{t-1}$ toward $\overline{position}_t$ in the XY plane; this step corresponds to 2D-3DoF rotation δ_{rot1}.

2. Rotation δ_{pitch_1}, which represents a rotation around the Y axis, and

[2] The notation ($\overline{abc}$) refers to the prediction of the state abc, obtained by the application of the motion model.

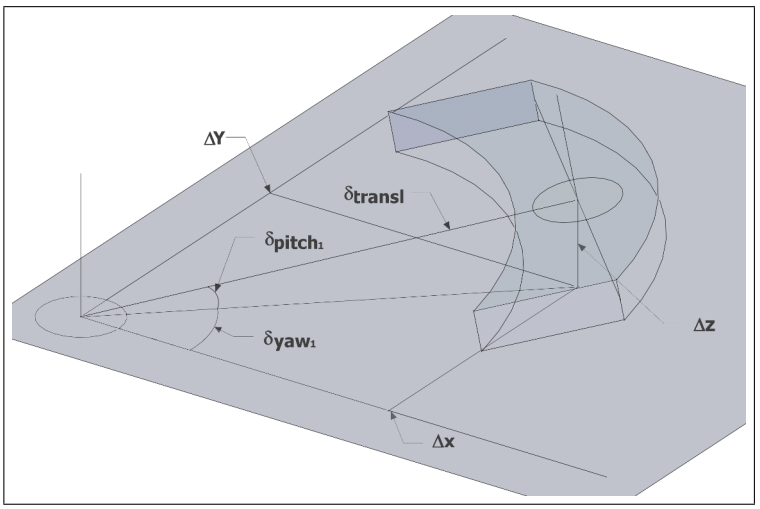


Figure 3.2: The proposed motion model decomposition and its parameters. A displacement is expressed in terms of a rotation δ_{yaw_1} about the Z axis, a rotation δ_{pitch_1} about the Y axis, a translation δ_{transl} along the new X axis, and a final generic 3D rotation (composition of three elementary rotations) to align the frame with the final robot position. The 3D banana-shaped box represents the 3σ volume where the particles of a Monte Carlo Localization system may spread as a result of applying Gaussian noise to each of the 6 basic movements that describe a displacement.

is also necessary to align $orientation_{t-1}$ toward $\overline{position}_t$, but in the XZ plane; this step introduces the possibility of a change in the value of the elevation.

3. Translation δ_{transl}, which represents a translation along the X axis; this translation moves the reference system, after the rotation by δ_{yaw_1} and δ_{pitch_1}, to $\overline{position}_t$; this step corresponds to 2D-3DoF translation δ_{transl}.

The three parameters δ_{yaw_1}, δ_{pitch_1}, and δ_{transl} can be seen as the coordinates, in a spherical coordinate system, of the origin of the new pose $\mathbf{x}_t$. To compute the motion parameters from the extended odometer readings, equations 3.1 to 3.3 can be used.

$$\delta_{yaw_1} = arctan(\frac{\Delta y}{\Delta x}) \tag{3.1}$$

$$\delta_{pitch_1} = arctan(\frac{\Delta z}{\sqrt{\Delta x^2 + \Delta y^2}}) \tag{3.2}$$

$$\delta_{transl} = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} \tag{3.3}$$

For the computation of $\overline{orientation_t}$, the proposal is to compose the $orientation_{(t-1)}$ with a generic rotation, which is in turn the composition of 3 last rotation steps. The parameters of these steps, i.e., δ_{roll}, δ_{pitch_2}, δ_{yaw_2}, are sensed directly by the extended odometer.

$$\delta_{roll} = \Delta\varphi \tag{3.4}$$

$$\delta_{pitch_2} = \Delta\psi \tag{3.5}$$

$$\delta_{yaw_2} = \Delta\vartheta \tag{3.6}$$

In order for the motion model to generate realistic motion uncertainty, it is necessary to add randomness to the components of the state vector $\overline{x}_t$, by acting on the parameters of the motion model. This randomness will be normally distributed, with zero mean, and the standard deviation of the components can be calculated according to the following considerations, which are specific to each single step.

1. Rotation δ_{yaw_1}, as in [3], is influenced by:

   - how much the vehicle has rotated, as measured by the wheel odometer;
   - how much space the vehicle has traveled, as measured by the wheel odometer.

   For both factors, the larger the factor, i.e., the change of orientation and/or the traveled distance, the larger the potential mismatch between the odometric measure and real pose.

2. Rotation δ_{pitch_1}, is influenced by:

   - how much the z coordinate has changed, i.e., by Δz, as measured by the extended odometer, from the IMU.

3. Translation δ_{transl} is influenced by:

   - how much space the vehicle has traveled, as measured by the extended odometer; the longer the traveled distance, the larger the potential mismatch between the odometric measure and real pose;

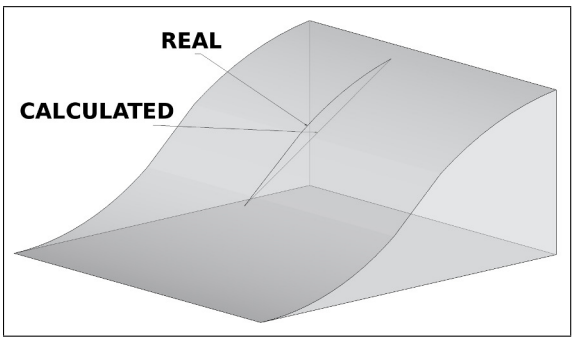


Figure 3.3: REAL and CALCULATED (basing on odometry) trajectories, which impact on the uncertainty on δ_{transl}, as due to a change in pitch ($\Delta\psi$).

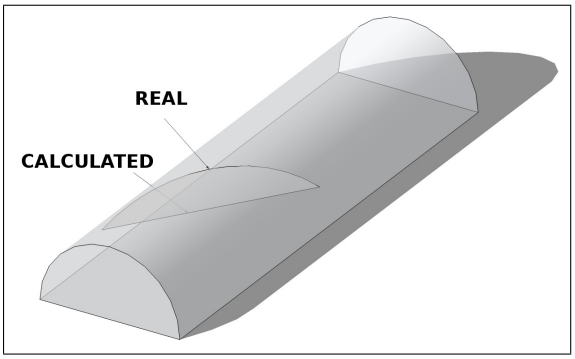


Figure 3.4: REAL and CALCULATED (basing on odometry) trajectories, which impact on the uncertainty on δ_{transl}, as due to a change in roll ($\Delta\varphi$).

- how much the vehicle has rotated about the Y axis, i.e., the variations $\Delta\psi$, as measured by the extended odometer. A change of pitch while performing a translation, represents a situation where the motion is taking place over a non planar surface. Therefore the traveled distance is larger and the uncertainty is also larger. Figure 3.3 illustrates the translation resulting from integration of odometry, and the real translation.

- how much the vehicle has rotated about the X axis, i.e., the variation in roll $\Delta\varphi$, as measured by the extended odometer. Figure 3.4 illustrates the translation resulting from integration of odometry, and the real translation.
- how much the vehicle has rotated about the Z axis, i.e., the variation $\Delta\vartheta$, as measured by the extended odometer. Figure 3.5 illustrates the translation resulting from integration of odo-

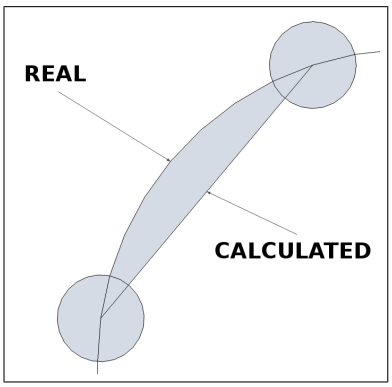


Figure 3.5: REAL and CALCULATED (basing on odometry) trajectories, which impacts on the uncertainty on δ_{transl}, as due to a change in yaw ($\Delta\vartheta$).

metry, and the real translation.

4. Rotation δ_{roll} is influenced by:
   - how much the vehicle has rotated around its X axis, i.e., variation $\Delta\varphi$, as measured by the extended odometer, from the IMU.
5. Rotation δ_{pitch_2} is influenced by:
   - how much the vehicle has rotated around the Y axis, i.e., the variation $\Delta\psi$, as measured by the extended odometer, from the IMU.
6. Rotation δ_{yaw_2} is influenced by:
   - how much the vehicle has rotated around the Z axis, i.e., the variation $\Delta\vartheta$, as measured by the extended odometer, from the wheel odometer.
   - how much space the vehicle has traveled: the longer the traveled distance, the larger the potential mismatch between the odometric measure and reality, as measured by the wheel odometer.

Basing on the above mentioned influences, we can define the standard deviations of the noise representing the uncertainty affecting the 6 steps. Finally, in order to gain a better control on the model behavior and similarly to what has been done in [3], we introduce a weight α, for each step.

$$\sigma_{yaw_1} = \alpha_1 \cdot \delta_{yaw_1} + \alpha_2 \cdot \delta_{transl} \tag{3.7}$$

$$\sigma_{pitch_1} = \alpha_3 \cdot \Delta z \tag{3.8}$$

$$\sigma_{transl} = \alpha_4 \cdot \delta_{transl} + \alpha_5 \cdot \delta_{yaw_2} + \alpha_6 \cdot (\delta_{roll} + \delta_{pitch_2}) \tag{3.9}$$

$$\sigma_{roll} = \alpha_7 \cdot \delta_{roll} \tag{3.10}$$

$$\sigma_{pitch_2} = \alpha_8 \cdot \delta_{pitch_2} \tag{3.11}$$

$$\sigma_{yaw_2} = \alpha_9 \cdot \delta_{yaw_2} + \alpha_{10} \cdot \delta_{transl} \tag{3.12}$$

The IMU uncertainty is assumed not correlated with the wheel odometer uncertainty. Moreover, notice that σ_{roll}, σ_{pitch_1}, and σ_{pitch_2} are influenced only by the IMU part of the extended odometer, while σ_{transl} is influenced both by the wheel odometer and the IMU, see Figures 3.3, 3.4, and 3.5.

The sampling motion model will be the following:

$$\begin{aligned}\hat{\delta}_{yaw_1} =& \delta_{yaw_1} + \\ & SAMPLE \underbrace{\{\alpha_1 \cdot \delta_{yaw_1} + \alpha_2 \cdot \delta_{transl}\}}_{\sigma_{yaw_1}}\end{aligned} \tag{3.13}$$

$$\hat{\delta}_{pitch_1} = \delta_{pitch_1} + SAMPLE \underbrace{\{\alpha_3 \cdot \Delta z\}}_{\sigma_{pitch_1}} \tag{3.14}$$

$$\begin{aligned}\hat{\delta}_{transl} =& \delta_{transl} + \\ & SAMPLE \left(\underbrace{\begin{array}{l}\alpha_4 \cdot \delta_{transl} + \alpha_5 \cdot \delta_{yaw_2} + \\ \alpha_6 \cdot (\delta_{roll} + \delta_{pitch_2})\end{array}}_{\sigma_{transl}} \right)\end{aligned} \tag{3.15}$$

$$\hat{\delta}_{roll} = \delta_{roll} + SAMPLE \underbrace{(\alpha_7 \cdot \delta_{roll})}_{\sigma_{roll}} \tag{3.16}$$

$$\hat{\delta}_{pitch_2} = \delta_{pitch_2} + SAMPLE \underbrace{(\alpha_8 \cdot \delta_{pitch_2})}_{\sigma_{pitch_2}} \tag{3.17}$$

$$\begin{aligned}\hat{\delta}_{yaw_2} =& \delta_{yaw_2} + \\ & SAMPLE \underbrace{(\alpha_9 \cdot \delta_{yaw_2} + \alpha_{10} \cdot \delta_{transl})}_{\sigma_{yaw_2}}\end{aligned} \tag{3.18}$$

In case an extended odometer is not available, the expected value will of course be null, and we can use an a priori standard deviation value for each parameter, determined on the basis of the expectations on the change that the terrain can induce in each degree of freedom. Of course this option implies a larger uncertainty, which in turn requires a larger computational effort.

3.1.3 Model thresholds

The model exploits a few parameters, i.e., thresholds, in order to handle some situations.

Minimum thresholds

As it can be noticed in the relationships above, and similarly to what is done in [3, Sect 5.4], the standard deviations of the uncertainties are proportional to the amount of motion involved into each step. Whenever the motion is too small, the standard deviation gets underestimated. These thresholds are used in such cases; they guarantee a minimum dispersion of the sampled data, which is necessary to correctly represent the real uncertainty.

Maximum thresholds

These thresholds have been introduced in order to handle situations where the extended odometer does not give out values in 6DoF, i.e., when there is no IMU on the vehicle. Maximum thresholds represent the maximum a priori uncertainty, which should always lead to a larger estimate of robot movements with respect to the more concentrated one obtained when using inertial information. The σ_{max} value that is associated to every model parameter needs to be suitably large so to ensure that samples can be generated with enough dispersion about the mean value, in order to represent all possible changes on the given degree of freedom. In the experiments presented in Section 3.1.5 the values of these thresholds were chosen considering a maximum vehicle speed of 7 m/s and a 20Hz sampling frequency for the odometer.

3.1.4 Comparison with a baseline motion model

In order to clarify the relevance of a careful design of the motion model, a comparison between the proposed model and a baseline method is pre-

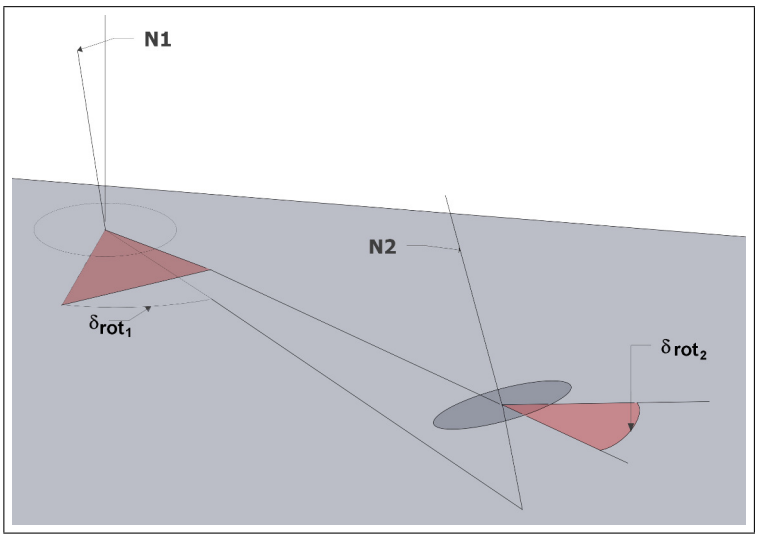


Figure 3.6: The naïve motion model decomposition and its parameters. A displacement is expressed in terms of an initial rotation about the axis N_1, a translation along the new X axis, and a final rotation about the axis N_2.

sented. The latter stems from a direct derivation of the 2D-3DoF model presented in [3, Sect 5.4]. This model is based on dividing the displacement into 3 steps only, as shown in Figure 3.6:

1. Rotation δ_{rot_1}, a rotation about an axis N_1. To obtain N_1, let us call D the vector $(\overline{position}_t - position_{(t-1)})$. N_1 is the vector product of the X axis of frame $pose_{(t-1)}$ and D.

$$N_1 = (\overline{position}_t - position_{(t-1)}) \times X_{pose_{(t-1)}} \tag{3.19}$$

2. Translation δ_{transl}, a translation along the X axis of the frame obtained after the previous step, i.e., after the rotation of $pose_{t-1}$ by δ_{rot_1}. At the end the origin will reach $\overline{position}_t$.

3. Rotation δ_{rot_2}, a rotation about an axis N_2. N_2 is the vector product of the X axis of frame $pose_t$ and D.

$$N_2 = (\overline{position}_t - position_{(t-1)}) \times X_{pose_t} \tag{3.20}$$

   This rotation aligns the reference frame, which has been obtained rotating $pose_{(t-1)}$ by δ_{rot_1} and then translating it by δ_{transl}, to finally obtain $\overline{orientation}_t$.

The uncertainty on the components of the motion model is sampled from normal distributions, for each of the 3 parameters δ_{rot_1}, δ_{transl},

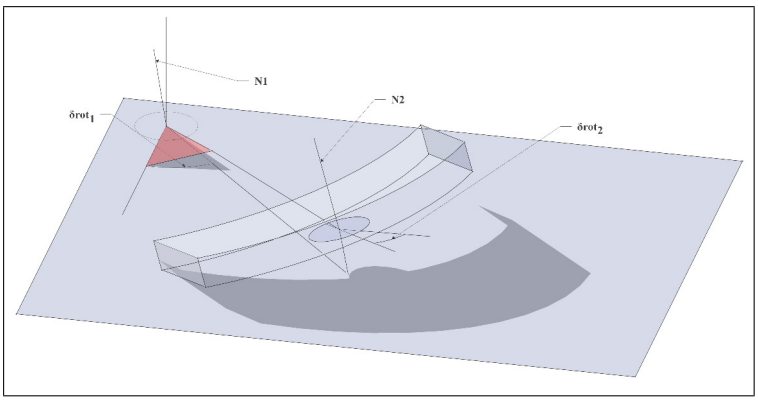


Figure 3.7: The 3D banana-shaped box representing the 3σ volume where the particles of a Monte Carlo Localization system may spread as a result of applying Gaussian noise to each of the 3 basic movements that describe a displacement with the naïve motion model.

and δ_{rot_2}. Such distributions have zero-mean and standard deviations computed similarly to what has been done for the motion model in [3, Sect 5.4]. It is just a similarity, because of the need to introduce other degrees of freedom to the uncertainty affecting $\overline{position}_t$, which would be just 2 (δ_{rot_1}, and δ_{transl}) for a 3D point. From here the decision of adding noise to the vector N_1 since, if it would have been added to the vector D, it would have led to the model proposed above. Notice that the 2 parameters of N_1 are not independent w.r.t. the uncertainty of rotating about N_1, so the DoF count for $\overline{position}_t$ is correct.

This naïve model, which turned out not being well performing, demonstrates how heavily the decomposition of the overall displacement, i.e., the motion model, affects the capability to produce realistic poses. Actually, the poses generated by this model are not realistically distributed about the real pose, see Figure 3.7 and Figure 3.8, where it can be observed that the uncertainty is badly rotated along the X axis; the larger Δz, the more rotated the particle cloud. Figure 3.9 shows the corresponding uncertainty for the proposed motion model.

3.1.5 Experimental Results

Tests of the software implementation of the proposed motion model have been performed on the real world application presented in Chapter 6. The experimental settings include the parking area of the U5 building of the University of Milano - Bicocca. Figures 3.10 and 3.11 show a 3D

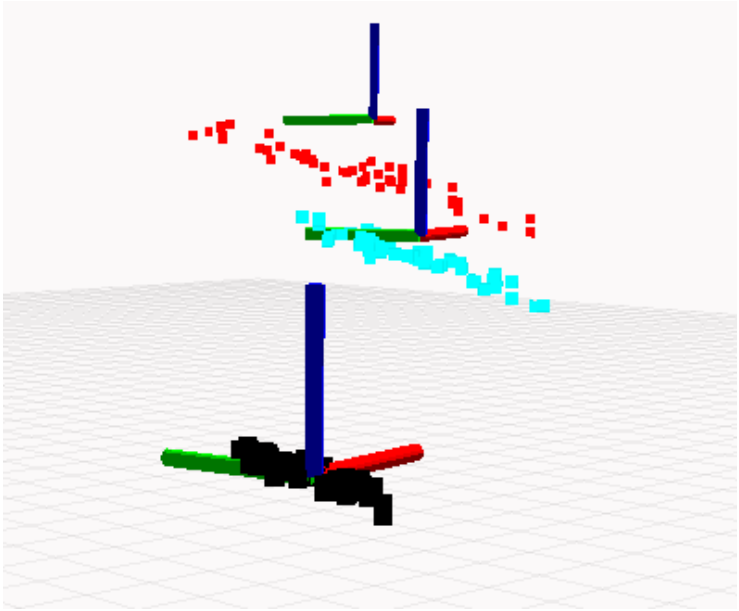


Figure 3.8: 3D view of the particle set for the naïve motion model. Notice that the larger the overall Δ_{yaw} and Δz, the larger the distortion of the particle set.

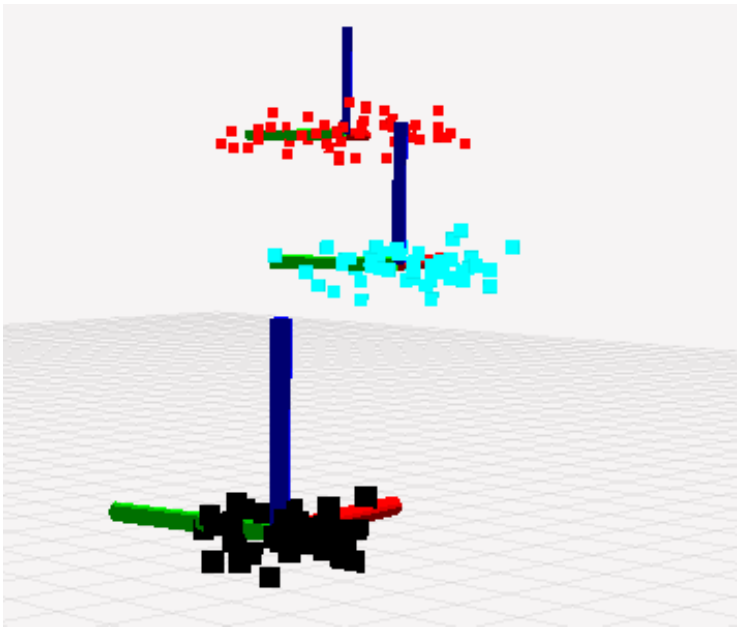


Figure 3.9: 3D view of the particle set for the proposed motion model. Notice the absence of the distortion that is affecting the particle set presented in Figure 3.8

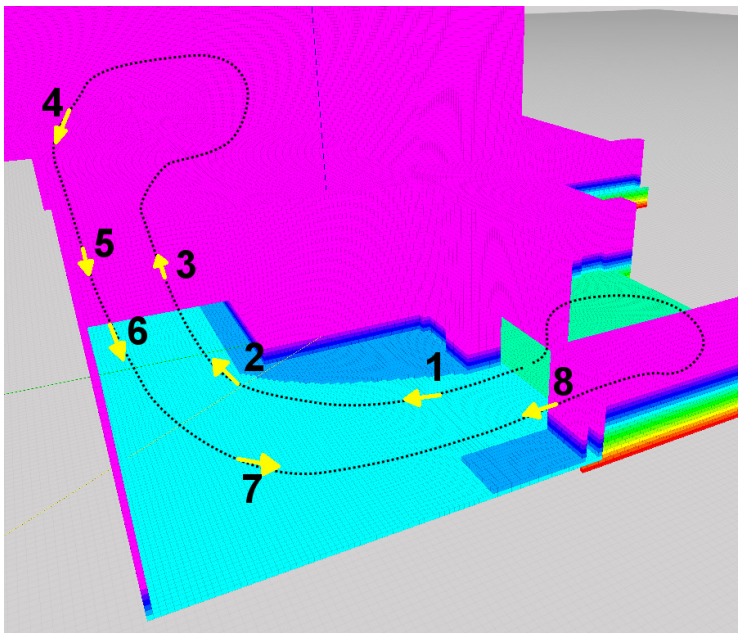


Figure 3.10: Voxel representation of the U5 building underground parking. Here is shown the part that includes the ramp leading to the outdoor parking lot: pose n. 8 is at the gate of the underground parking, pose n. 4 is in the outdoor parking nearby where the autonomous car (golf cart) is depicted in Figure 3.12, poses n. 3 and 5 are on the ramp, poses n. 1, 2, 6, 7 are in the planar road leading to the ramp.

voxel map and an aerial picture of the parking lot, respectively.

The motion model has been plugged into a state of the art Monte Carlo Localization software [121]. The tests were performed in this order: first a verification that the localization was correct was performed when moving on an almost planar surface, i.e., the model was performing at least as the state of the art 2D-3DoF model. This has been done in the underground parking of the building, where the floor appears to be reasonably planar. Indeed, the localization results were perfectly comparable to the ones obtained with the 2D-3DoF state of the art software [121]. Secondly, the cart was driven along a path including the ramp leading to the outdoor parking area, as shown in Figures 3.13. Also in these cases the localization was successful, as demonstrates the example shown in Figure 3.14. As there was no possibility of acquiring the ground truth of the robot poses along the path, the correctness of the localization was proved by checking that at the end of the path, at about pose n. 8 in Figure 3.10, the estimated pose was matching the real one.

In the same experiments, the naïve motion model proved its limita-

Figure 3.11: U5 building - Aerial view, the ramp from the underground garage can be noticed on the left of the largest tree.

Figure 3.12: The ramp from the underground garage to the outdoor park, picture taken from the outdoor park.

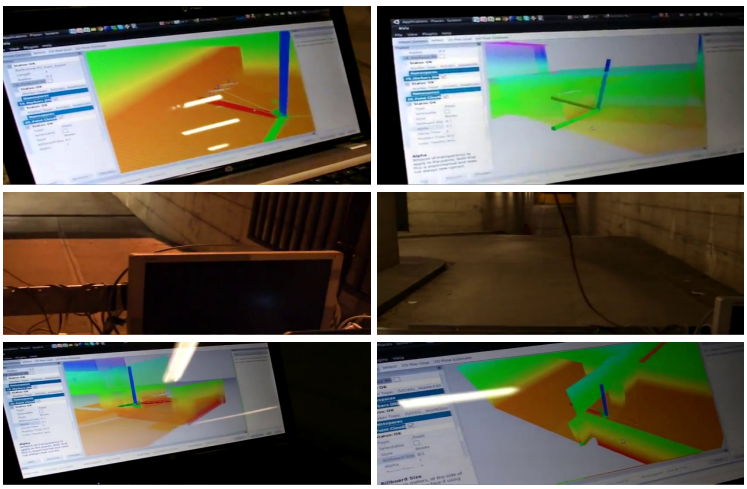


Figure 3.13: Snapshots along the path shown in Figure 3.10 from the underground garage, through the ramp, to the outdoor parking lot and back into the underground garage; 1) top to 8) bottom. Notice in 1), 3), 7) and 8) the cart reference frame, shown in the same map used by the software. In 3) notice the frame pitching up along the ramp.

tions and failed in high curvature curves, as it might be expected from observing, in Figure 3.7, the unrealistic uncertainty generated by this model; an example of failure of the naïve motion model is shown in Figure 3.15.

Despite roll and pitch data were available, thanks to an MTi X-sens IMU sensor, they proved to be pretty noisy and, as a result, the experimental activity verified that using only the available LIDAR sensors, altogether with appropriate minimum and maximum thresholds, sufficed for a correct localization. This statement holds when both types of LIDAR models available on the golf cart (Sick LMS111 and LDMRS4001) were used for localization. This proved that measuring the surrounding environment on different scanning planes, together with a motion model capable of modeling the full 6DoF dynamics of a rigid body, allows for robust localization within a 3D map.

Conclusions

This work led to the publication in [122] and presents a motion model for full 3D-6DoF localization, showing that a careful design is required to obtain a realistic representation of the involved uncertainties and of their influence on the posterior uncertainty distribution of the robot pose. The

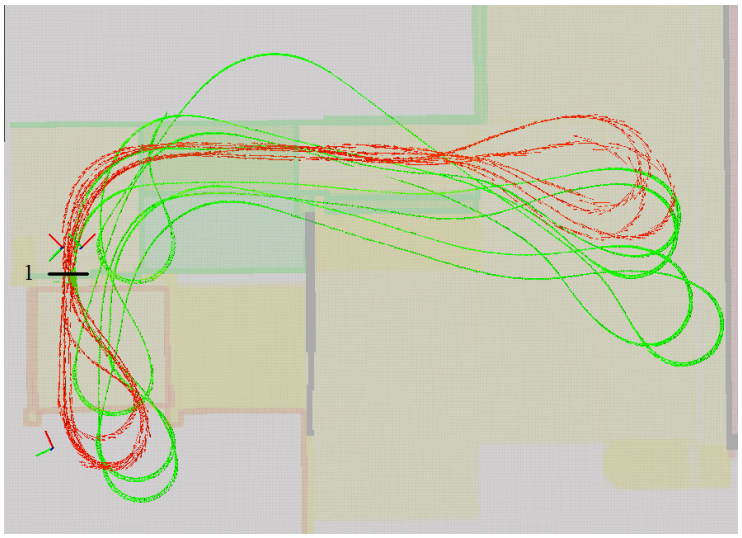


Figure 3.14: The green path represents the odometric path; the red path represents the localization obtained using the proposed motion model. Notice at 1) i.e., nearby pose n. 8, the correctness of the localization.

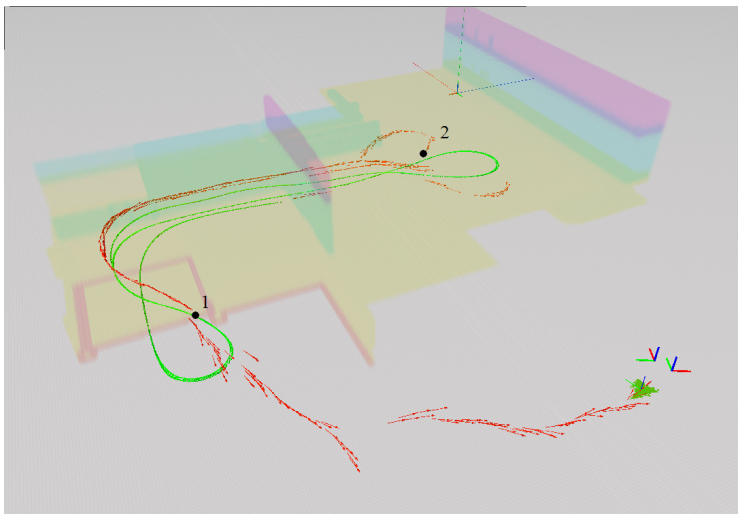


Figure 3.15: An experiment with the naïve motion model: in 2) the localization system could luckily recover from a localization error, while in 1) it failed and could not recover.

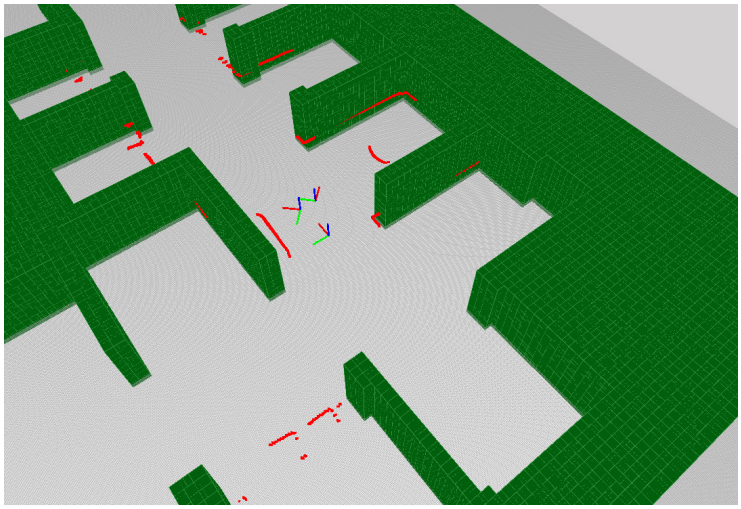


Figure 3.16: An example of the localization system running in the U5 underground parking lot. Please note the presence of many non-map elements observed by the laser scans, in this case parked cars and bikes.

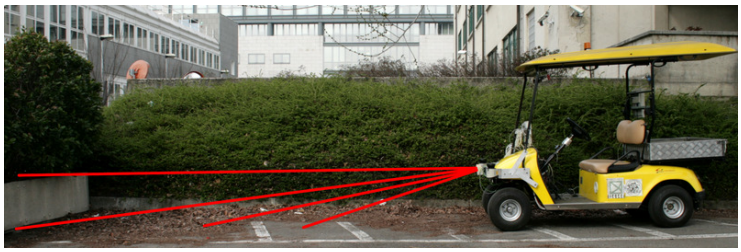


Figure 3.17: The scanning planes of the LDMRS-4001 Sick laser scanner.

presented model demonstrated its suitability in different real and simulated experiments and is currently in use on the robotic agents involved in the research activities for urban autonomous driving.

3.2 Lie algebra camera localization

This projects tackles the problem of single camera localization for a monocular 6DoF-moving observer, given a 3D visual map. While not innovative from the approach point of view, this project aims at the developing of an efficient implementation exploiting Lie algebra based camera tracking, to be deployed in compliance with the standards of the Robot Operating System (ROS). Such a work is in fact missing from the wide palette of ROS implementations of state-of-the-art approaches and it is the author's belief that its availability could bring advantage to the robotic community. For the mapping part, publicly available software is used.

3.2.1 Proposed approach

This project follows the PTAM [30] intuition of keeping the mapping and the camera tracking tasks separate. A non-real-time thread is in charge of performing global bundle adjustment to generate a globally consistent map, while the camera tracking is performed in real-time by locally refining its position, with respect to the map, exploiting Lie algebra optimization.

For the mapping part, the effective implementation in VisualSfM [123] was chosen, which combines Multicore Bundle Adjustment [36] and SiftGPU [40]. As in [30], VisualSfM is fed with a small subset of key-frames in order to keep the computation time within the real-time boundaries. The resulting map is thus constant between key-frames and is exploited for camera localization. In fact, this choice of exploiting a global SfM approach for mapping introduces an improvement over the approach proposed in [30], since it allows to overcome the delicate initialization complications present in [30].

For the camera tracking part, the key idea is to refine the estimate of the camera pose with respect to the 3D visual map by minimizing an error function based on the reprojection error of the map features in the image plane. The authors of [30] suggest to exploit the nice local properties of the Lie algebra $\mathfrak{se}(3)$ associated with the SE(3) manifold representing the group of rigid body motions. In particular, the fact that the $\mathfrak{se}(3)$ algebra represents the tangent space at the identity for SE(3) is leveraged to efficiently compute partial derivatives of the error function to be minimized with respect to the camera pose parameters. While

only holding for small camera motions, due to its nature of first order approximation of SE(3), the $\mathfrak{se}(3)$ algebra based minimization proved its effectiveness in the experimental activity, in both code simplicity and speed-up of the computation time.

The Special Euclidean Group SE(3) and the associated $\mathfrak{se}(3)$ algebra

The Special Euclidean Group SE(3) is the group of rigid transformations on $\mathbb{R}^3$, defined as the set of mappings $g : \mathbb{R}^3 \to \mathbb{R}^3$ of the form $g(x) = Rx + T$ and can be identified with the space of 4×4 matrices of the form

$$g = \begin{bmatrix} R & T \\ 0_{1\times 3} & 1 \end{bmatrix}$$

It is in fact a Lie group, i.e. a group that is also a differentiable manifold and it is six dimensional. Its tangent space $\mathfrak{se}(3)$ is closed under the Lie bracket, making it a proper algebraic structure. $\mathfrak{se}(3)$ is known as the Lie algebra of SE(3) and represents its linearized version, i.e. its infinitesimal group. An elements $x \in \mathfrak{se}(3)$ can be expressed in the form

$$x = \begin{bmatrix} [\omega]_\times & v \\ 0_{1\times 3} & 0 \end{bmatrix}$$

where the operator $[\omega]_\times$ maps a 3-vector ω into a 3×3 matrix of the form

$$[\omega]_\times = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

To map elements of SE(3) into $\mathfrak{se}(3)$ and vice versa, the *matrix logarithm* and the *exponential map* are used, respectively.

Please refer to Appendix A for a thorough tractation of Lie groups and Lie algebras.

3.2.2 Camera tracking

The main goal of a camera tracking system is to estimate the camera poses with respect to a known visual map. In absence of any proprioceptive sensor, the only information that can be measured is the location discrepancy between the map elements projected into the image planes (according to the potentially wrong camera pose estimations) and the

visual features extracted from the images themselves. Therefore, minimizing the specific error function that describes this discrepancy, with respect to the camera pose parameters, allows to refine the estimation of where the observer was located within the visual map at each time step.

Let

$$\begin{pmatrix} u \\ v \end{pmatrix} = \mathbf{Pr}(\mathbf{Rt}_w^c P^w)$$

be the projection into a pinhole camera of a point P^w (expressed in world, i.e. map, coordinates), after being roto-translated to camera coordinates by means of the roto-translation matrix $\mathbf{Rt}_w^c$, thus becoming P^c. Now, let us decompose $\mathbf{Rt}_w^c$ into a fixed $\mathbf{Rt}_w'^{c_0}$ (representing the roto-translation between world coordinates and camera coordinates at time 0) and a variable $\mathbf{M}$, representing the camera motion with respect to the camera at time 0. As stated in Section 3.2.1, $\mathbf{M}$ is part of the SE(3) group and can be expressed in terms of a 6-vector $\mu = (\omega_{1\times 3}, v_{1\times 3})^T$ by means of the *exponential map*, so as to obtain

$$P^c = \mathbf{Rt}_w^c P^w = \mathbf{M}\mathbf{Rt}_w'^{c_0} P^w = \exp(\mu)\mathbf{Rt}_w'^{c_0} P^w$$

and

$$\begin{pmatrix} u \\ v \end{pmatrix} = \mathbf{Pr}(\exp(\mu)\mathbf{Rt}_w'^{c_0} P^w) \tag{3.21}$$

Please refer to Appendix A for the explanation of how to compute partial derivatives of $\exp(\mu)$ with respect to μ. It is in fact a trivial differentiation and can be efficiently written in closed form.

Now, let us properly define the error function. As stated above, there will be the pure measurements z directly extracted from the image by means of a feature detector, and there will be the expected measurements $\hat{z}$, generated by the map 3D points projected onto the image plane. After performing proper data association between z and $\hat{z}$, a set of matched couples $(z_k, \hat{z}_k)$ will be available, where $\hat{z}_k$ can be written in terms of Equation 3.21.

The reprojection error function for a single point takes the form

$$\begin{aligned} E_k(\mu) &= z_k - \hat{z}_k \\ &= z_k - \mathbf{Pr}(\exp(\mu)\mathbf{Rt}_w'^{c_0} P_k^w) \end{aligned} \tag{3.22}$$

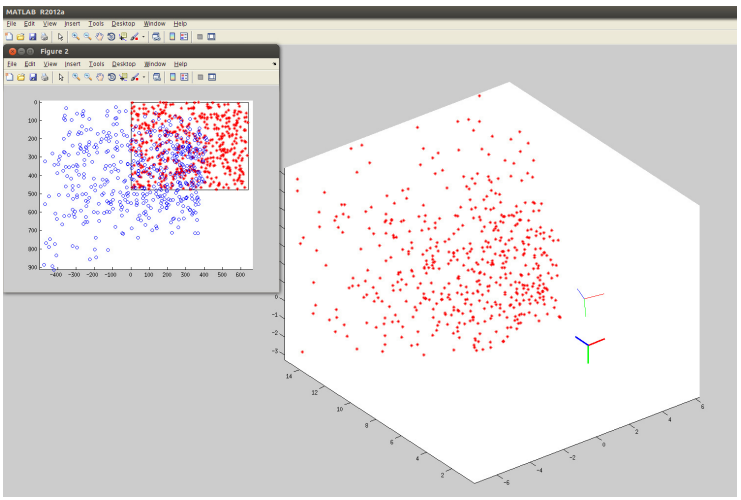


Figure 3.18: Convergence process of the camera tracking system in simulated environment without noise: starting condition. The 3D viewport on the right shows the 3D space with 500 randomly generated map points, the starting estimate of the camera pose (thin reference frame) and the correct camera pose (thick reference frame). The smaller, foreground window in the top left corner shows the image plane (thin black frame) and the reprojected 3D points according to the estimated (blue circles) and correct (red stars) camera pose. The starting pose is evidently misplaced as the estimated reprojected map points not only do not stay nearby the corresponding ones, but also stay outside the boundaries of the image itself.

The complete error function we want to minimize is squared sum of the single points errors

$$\begin{aligned} E(\mu) &= \sum_k (z_k - \hat{z}_k)^2 \\ &= \sum_k (z_k - \mathbf{Pr}(\exp(\mu)\mathbf{Rt}_w^{'c_0} P_k^w)) \end{aligned} \tag{3.23}$$

Differentiating Equation 3.23 with respect to the parameter vector μ allows us to iteratively refine the estimation of μ by means of a standard function minimization method, such as, *e.g.* Gauss-Newton or Levenberg–Marquardt.

3.2.3 Experimental results

The experimental activity was performed in a simulated environment, for which a Matlab prototype was implemented and synthetic data were

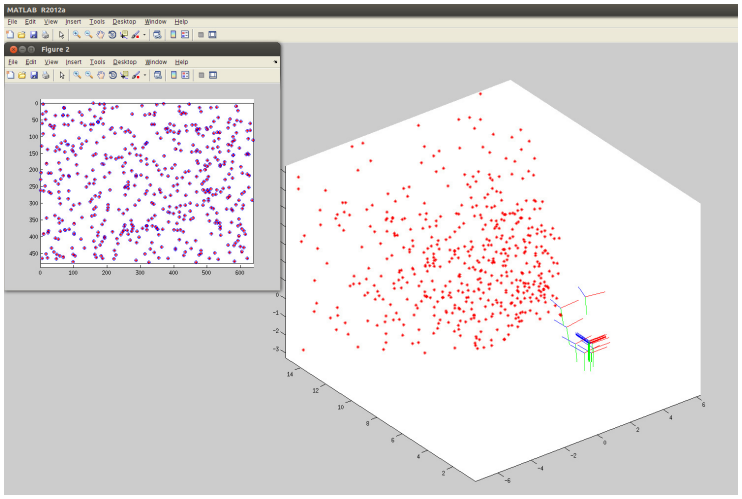


Figure 3.19: Convergence process of the camera tracking system in simulated environment without noise: evolution of the minimization. The 3D viewport on the right shows the 3D space with 500 randomly generated map points, the estimated camera poses (thin reference frames) at each minimization step and the correct camera pose (thick reference frame). The smaller, foreground window in the top left corner shows the image plane (thin black frame) and the reprojected 3D points according to the estimated (blue circles) and correct (red stars) camera pose. Note how the minimization process converged to the true camera pose estimate, despite the substantially wrong initial guess.

elaborated. The implementation consists of three different approaches that differ in the way the jacobians are calculated:

- Lie algebra based derivatives, i.e. elements of the $\mathfrak{se}(3)$ algebraic structure. Exploiting local properties of smooth manifolds, partial derivatives of the reprojection error function can be easily written in a very short form, with 16 terms per each of the two rows of the jacobian (see Equation A.8).
- Lie algebra symbolic derivatives. The parameter vectors to be minimized are still elements of the $\mathfrak{se}(3)$ algebraic structure, but the partial derivatives of Equation 3.23 are calculated with the Matlab symbolic toolbox. The resulting code is huge, about 8K terms per each of the two rows of the jacobian, which must in turn be evaluated for each 3D point of the map.
- Standard camera pose minimization, using symbolic partial derivatives of SE(3) roto-translation matrices. The resulting code is pretty long, about 200 terms per each of the two rows of the jacobian.

Minimization is performed in all the three cases by means of the classical Gauss-Newton algorithm (Levenberg–Marquardt was also tested but the modest gain with respect to Gauss-Newton does not justify the slightly more complicated formulation). Figures 3.18 and 3.19 show examples of the running code. Tables 3.1, 3.2 and 3.3 and Figures 3.20, 3.22 and 3.21 show the differences in computation time per minimization step, computation time for convergence and number of iterations, as the number of map elements grows, of the minimization process when using the three approaches mentioned above. As it can be seen, the compactness of the Lie algebra based derivatives leads to a substantial gain in computation time per each iteration (almost one order of magnitude). The main drawback is that the Lie algebra based minimization approach needs a higher number of iterations to achieve the same final accuracy. This leads to a smaller gain in the total computation time for convergence, although approaches like [30] overcome this problem by assuming that the initial guess is not far from the correct one and thus limit the number of iteration to a lower, fixed number.

3.2.4 Conclusions

The work presented in this section demonstrates that, by exploiting local properties of the SE(3) manifold and its associated $\mathfrak{se}(3)$ Lie alge-

Map	Iter. Time	Time Mean	Time Std. Dev.	Iter. Mean	Iter. Std. Dev.
10	0.000161	0.003980	0.000216	24.73	0.47
20	0.000166	0.004104	0.000109	24.79	0.41
50	0.000176	0.004390	0.000111	24.93	0.26
100	0.000190	0.004743	0.000102	24.99	0.1
200	0.000220	0.005491	0.000114	25	0
500	0.000310	0.007751	0.000168	25	0
1000	0.000674	0.016846	0.003625	25	0

Table 3.1: Lie algebra based derivatives, i.e. elements of the $\mathfrak{se}(3)$ algebraic structure. The columns represent, from left to right: the number of 3D features in the map, the mean time in seconds per each iteration, the mean and standard deviation of both the computation time (seconds) and number of iterations necessary for convergence, computed over 100 randomized runs.

Map	Iter. Time	Time Mean	Time Std. Dev.	Iter. Mean	Iter. Std. Dev.
10	0.004872	0.068059	0.001196	13.97	0.22
20	0.004902	0.068727	0.001053	14.02	0.14
50	0.004953	0.069342	0.001215	14	0
100	0.004914	0.068789	0.000464	14	0
200	0.005012	0.070163	0.000521	14	0
500	0.005212	0.072967	0.000797	14	0
1000	0.006332	0.088653	0.005679	14	0

Table 3.2: Lie algebra symbolic derivatives. The columns represent, from left to right: the number of 3D features in the map, the mean time in seconds per each iteration, the mean and standard deviation of both the computation time (seconds) and number of iterations necessary for convergence, computed over 100 randomized runs.

Map	Iter. Time	Time Mean	Time Std. Dev.	Iter. Mean	Iter. Std. Dev.
10	0.001065	0.008040	0.000559	7.55	0.5
20	0.001085	0.008226	0.000570	7.58	0.5
50	0.001102	0.008440	0.000526	7.66	0.48
100	0.001113	0.008661	0.000478	7.78	0.42
200	0.001184	0.009369	0.000367	7.91	0.29
500	0.001338	0.010701	0.000241	8	0
1000	0.002270	0.018160	0.003498	8	0

Table 3.3: Standard symbolic partial derivatives of SE(3) roto-translation matrices. The columns represent, from left to right: the number of 3D features in the map, the mean time in seconds per each iteration, the mean and standard deviation of both the computation time (seconds) and number of iterations necessary for convergence, computed over 100 randomized runs.

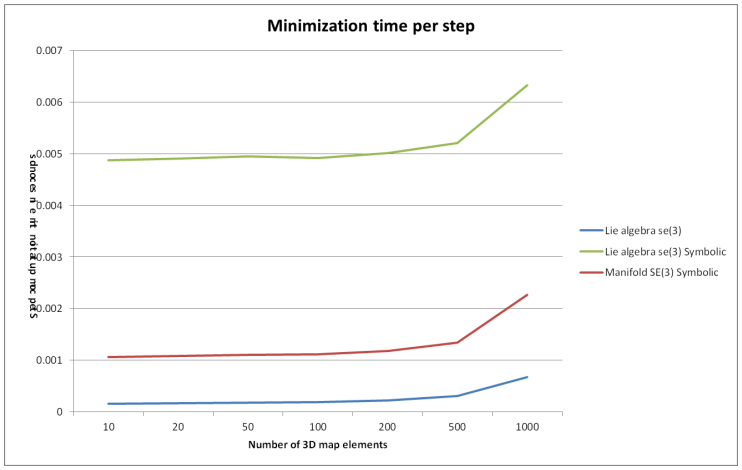


Figure 3.20: Graphical comparison of the minimization process execution time, as the number of map elements grows, using $\mathfrak{se}(3)$ fashion derivatives and SE(3) standard symbolic derivatives. The mean and standard deviation are computed over 100 runs.

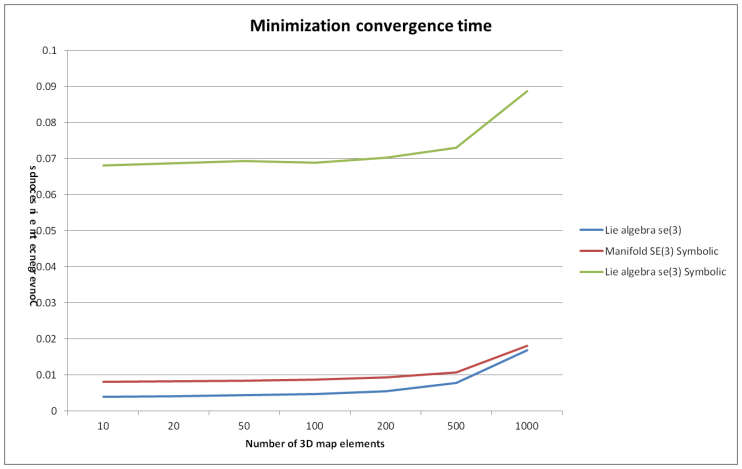


Figure 3.21: Graphical comparison of the minimization process execution time, as the number of map elements grows, using $\mathfrak{se}(3)$ fashion derivatives and SE(3) standard symbolic derivatives. The mean and standard deviation are computed over 100 runs.

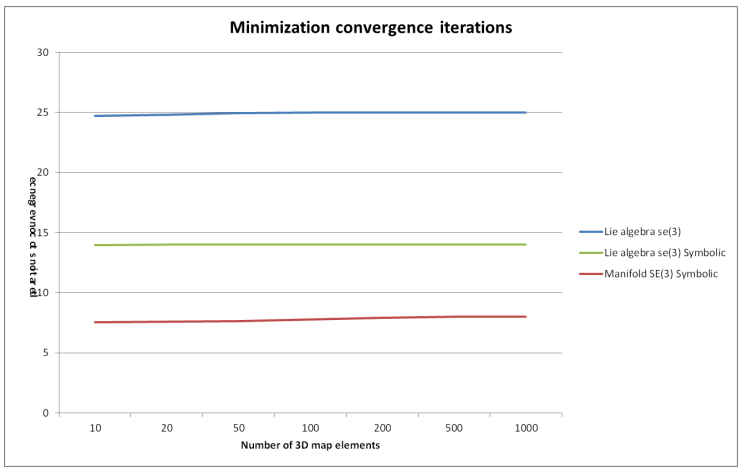


Figure 3.22: Graphical comparison of the minimization process execution time, as the number of map elements grows, using $\mathfrak{se}(3)$ fashion derivatives and SE(3) standard symbolic derivatives. The mean and standard deviation are computed over 100 runs.

bra, a very efficient estimation of the camera pose with respect to a 3D visual map can be performed. In particular, the speed-up in computation time and code simplicity is shown when comparing the Lie algebra implementation and the traditional way of calculating symbolic partial derivatives of the reprojection error with respect to the roto-translation parameters.

3.3 Summary

The topics presented in this chapter can be summarized as follows:

- Autonomous robots require a reliable Self-Localization module in order to efficiently perform assigned tasks.
- Self-Localization can be performed using pure proprioceptive sensors (dead reckoning), although the quality of the localization quickly degrades, because of the integration of small errors over time.
- Self-Localization can be performed very efficiently processing informations generated by exteroceptive sensors like cameras and laser scanners.
- When poorly or ambiguously featured informations are used (typical laser scanner based approaches), an accurate motion model of

the robot's dynamics is of crucial importance for the success of the localization process.

- When a 3D visual map is available, extremely efficient camera localization can be performed by exploiting the local properties of the SE(3) manifold and its associated $\mathfrak{se}(3)$ Lie algebra.

3.3.1 Acknowledgments

For the work presented in Section 3.1, special thanks are owed to the coauthors of [122], Augusto L. Ballardini, Andrea Galbiati, Matteo Matteucci, Francesco Sacchi, and Domenico G. Sorrenti.

For the work presented in Section 3.2, special thanks to Nicola Donadoni for the implementation work under the ROS framework.

Chapter 4

Object detection and tracking

In this chapter I will present the work on object detection and tracking that led to the publication of Scale-Independent Object Detection with an Implicit Shape Model (SIODISM) [124] and the follow up work carried on by the master students Prisciantelli and Ventimiglia, BISMD [125] and BMTT [126], under my supervision.

4.1 Scale-Independent Object Detection with an Implicit Shape Model

4.1.1 Introduction

In the last years object detection and tracking could afford dealing with realistic, i.e., challenging, conditions thanks to the advancements in computer vision research. The object-detection task consists of the process of localizing and determining the class of specific objects in an image. Humans can perform this task easily, even if the objects appear rotated, scaled or partially occluded. To reach a human level ability is still an unsolved issue for state-of-the-art computer vision systems.

Usually, object detection algorithms (and humans too) cannot guarantee that the resulting interpretation of the image is the correct one. Therefore it is advisable to accumulate evidence over time, which implies coupled classification and tracking. We believe to be preferable to generate different interpretation hypotheses and then to choose those that better explain the actual observations of the world. In our thinking, the different interpretation hypotheses should be characterized and compared in a probabilistic framework. The idea of probabilistic mul-

tiple hypotheses classification and tracking dates back to the early 90' and has been further developed over the years. The approach that best matches our objectives is the one in [127, 128].

In [124] we propose an improvement to the Implicit Shape Model (ISM) (see Section 2.2) based robust object detection system proposed by Leibe *et al*. [129]. Object detection with ISM allows to approach the classification and tracking problem in a probabilistic way with multiple hypotheses. Unlike the original approach, our method is independent from object scale in the training sets, and this allows to work with much smaller training sets and also to avoid to supply information about scale to the trainer. This is done while maintaining the robustness of the original approach. Leibe *et al.* mentioned a potential solution to overcome the scale problem in the training set, i.e., the usage of the scale produced by the local descriptor. Our proposal is different: since the scale measure generated by local descriptors is in general subject to noise, we try to walk around this noise by estimating the scale measure only from the evidence collected in the image.

In the view of integrating the output of an object detector in a recognition and tracking system, we need to express the detector output in the form of a probability distribution. The approach proposed by Liebe and Schiele has the relevant peculiarity that the detection output is a set of hypotheses on the size, pose, and class of one or more objects. Their approach can also be easily modified to accommodate a pixel level segmentation. This means that the detector will be able to output, for each object, a pixel map where, for each pixel, the probability of that pixel being part of the object (or being part of the background) is available. The usage of such method in a probabilistic framework for classification and tracking allows to generate 3D hypotheses (by means of camera-to-world projection) and to select the best interpretation of the image, provided some constraints are satisfied like, e.g., a pixel not being associated to two objects, etc. This in turn allows the classifier and tracking system to handle realistic partial object occlusions.

In Section 4.1.2 the original formalization of the Implicit Shape Model is presented. In Section 4.1.3 we focus on the codebook creation, and in Section 4.1.4 we review the original object detection system. In Section 4.1.5 the proposed method for scale-independence is presented, while in Section 4.1.6 we present some results of our system.

4.1.2 The Implicit Shape Model formalization

The approach proposed in [66], developed in [67] and extended in [68] rely on an implicit description of the objects. Thanks to this characteristic there is no more need for an explicit description which, as seen in the previous section, requires large training sets and complicated description models. This approach, on the other hand, builds its knowledge base, called codebook, by collecting description and spatial evidence from the training images.

This is performed by computing some local descriptors like, e.g. SIFT or Shape Context, on interest regions automatically extracted from the images. This information, in conjunction with the spatial coordinates of the interest regions, is stored in the codebook and used later at detection time. Thus we collect information about the object in areas which provide more details to the object description.

Formally, an Implicit Shape Model $ISM(C) = (I_C, P_{I,C})$ for a given object category C consists of a knowledge base I_C (*codebook*), built-up with local descriptors discriminative for the object class, and of a spatial distribution $P_{I,C}$ that indicates where each codebook entry may be found on the training object [67]. There are two requirements for the spatial distribution $P_{I,C}$. First, such distribution should be defined independently for each codebook entry, thus making the approach flexible and capable to merge, at detection time, parts of objects observed on different training images. Second, this distribution should be computed in a non-parametric way, thus emulating the real distribution as in detail as the training objects permit, or, in other words, exploiting as much training informations as possible. Furthermore this requirement frees us from making Gaussian assumption on the spatial distribution.

4.1.3 Codebook creation

In the approach presented in [68] the codebook is filled with N entries, each one representing some evidence extracted from the training images. For each type of descriptor (named *cue*) we build a separate codebook. Each entry of each codebook should contain informations about its own relative position with respect to the object center, the values of the associated local descriptor calculated on the image patch that generated that entry, a segmentation mask *figure/background*, the parameters of the ellipsis that includes the image patch, the object scale and the list of possible object centers for which this entry may cast votes. As will be

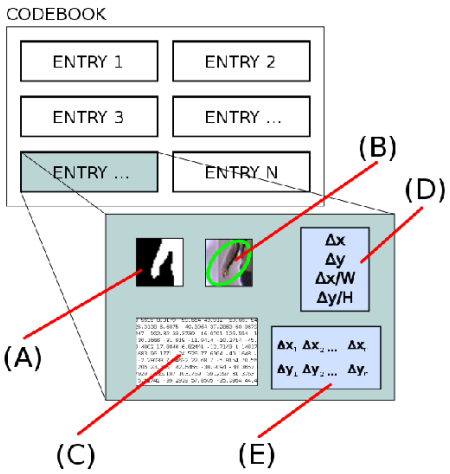


Figure 4.1: Structure of the codebook and its entries. (A) represents the segmentation mask, (B) contains the parameters of the ellipsis which includes the image patch, (C) the values of the associated local descriptor calculated on the image patch, (D) the informations about the relative patch position with respect to object center, (E) lists all possible image centers for which this patch may cast votes.

shown in Section 4.1.5, the approach proposed in this chapter is independent from the object scale, so that this information will be omitted. In Fig. 4.1 the structure of the codebook and its entries are shown.

The procedure of codebook creation is divided in two parts. First we analyze the training images and populate the codebook with entries containing information about interest regions in the image. Second we compare, calculating the Euclidean distance in the space of descriptor values, all codebook entries with each other. In each couple that results sufficiently similar, its entries mutually enrich each other's spatial information. This makes it possible for a codebook entry to vote, at detection time, for more than one object center.

The first phase, i.e., the analysis of the training image, is performed in the following steps:

- We extract the most interesting image areas using one or more automatic interest region detectors. These algorithms ground on functions such as Harris function, Hessian determinant, etc. in order to extract scale invariant regions of interest from an image. An interest region is a small image portion whose content is highly descriptive (high edgeness, cornerness, etc.), that can be used for discriminative purposes, in conjunction with some local descriptors.

- We calculate the values of local descriptors on interest regions. Each descriptor is composed by a different number of values, thus the codebook structure need to be sufficiently flexible in order to contain different descriptors. Local descriptors extract some kind of "fingerprint" of an image region, which are typically scale and rotation invariant. Some example of local descriptors are SIFT, PCA-SIFT, Shape Context, etc [130]. In our implementation we tested both SIFT and Shape Context descriptors.

- Finally we save in the codebook, for each interest region, the parameters of the ellipse, the spatial information (i.e. the distance of the interest region to the training object's center), the values of the local descriptor and the local segmentation mask (extracted from the global segmentation mask supplied in the training set.

In the second part of the codebook creation we compare all codebook entries with each other. The aim of this procedure is to enforce the generation of object hypotheses and their segmentation during detection and to enable single image patches to vote for more than one object center. The similitude measurement is calculated as the Euclidean distance in the M-dimensional space of descriptor values:

$$d(P_i, P_j) = \sqrt{\sum_{k=1}^{M} \left(P_i^k - P_j^k\right)^2} \tag{4.1}$$

If the distance between two entries is under an acceptance threshold, we say that their domain of discrimination is similar, thus we enable both entries to cast votes, at detection time, for their own center and for the center of the other entry.

This step concludes the codebook creation procedure.

4.1.4 Object detection

Image analysis

The image analysis procedure is similar to the one described in the codebook creation section. Applying several region detection algorithms on the image we are able to extract some interesting areas (in the following these areas will be called patches). For each patch we compute different values obtained by computing different region descriptors. These values will be stored for the following operations.

Next, the set of extracted image patches will be compared with all the codebook entries. For each pair patch-entry an Euclidean distance in the space of descriptor values is computed. If the distance is less than an a-priori threshold (the same used in the codebook creation procedure) then the patch will be associated to the codebook entry.

The original approach for voting

The original approach proposed by Leibe et al. in [68] bases on the equation that describes the probability that an object is located in a particular position with a specific scale, given the evidence, the position of the patch, and its descriptor.

$$p(o_n, \lambda|\mathbf{e}, l, q) = \sum_i p(o_n, \lambda|C_i^q, l, q)p(C_i^q|\mathbf{e}) \tag{4.2}$$

The purpose of this section is to define a practical meaning for the components of this equation.

- The probability $p(o_n, \lambda|C_i^q, l, q)$ represents the strength with which the patch votes for the object center (λ). It is inversely proportional to the number of possible interpretations of the patch. This formulation comes from the intuitive idea that the more interpretations a patch has, the more its vote will be unreliable. For example, if a patch representing a car wheel matches in the codebook with both car wheel entries and hood entries, then it will be not discriminative to univocally identify the car center. For this reason the center will have a low probability $p(o_n, \lambda|C_i^q, l, q)$.

- $p(C_i^q|\mathbf{e})$ represents the probability for the patch being correctly explained by a codebook entry C_i^q. The intuitive meaning is the following. Given a pair patch-entry, the perfect explanation of the patch generates a vote for the exact center of the object. Since an entry generally votes for more than one center, the probability of obtaining the correct explanation of the patch decreases when this number increases. Therefore, we can represent the probability $p(C_i^q|\mathbf{e})$ as the inverse of the number of centers voted by the C_i^q entry.

Finally, the $p(o_n, \lambda|C_i^q, l, q)p(C_i^q|\mathbf{e})$ term represents the strength of the vote of each patch.

4.1.5 Proposed method

In the approach described above each image patch can cast votes for some object centers **as point coordinates**. This implies that a single vote represents a possible object center at that location and **at that scale**, thus requiring the scale information generated by local descriptors. Since this information is often subject to noise, we try to walk around it by estimating the object scale only from the spatial distribution of the image patches detected. Furthermore, we want the voting procedure to be independent from the object scale. Such a result would allow to operate with a much smaller training sets, since the many images of the same objects at different scales would not be necessary any more. Smaller training sets and, thus, smaller codebooks, noticeably reduce computational costs when comparing image patches to codebook entries, since each patch needs to be compared with all entries.

In our approach each patch will no longer vote for a point as the center of the object. Instead, a half straight line will be drawn in the voting space, starting from patch center and going towards the hypothesized object center. Since the region descriptors are scale-invariant, the same object detail should generate similar descriptor values even if scaled or rotated. Thus we save, at training time, the information about the direction of the object center with respect to the patch position in the training image. This information is used at detection time to specify the direction where the relative position of the trained object center, with respect to the matched patch, can be found. In Fig. 4.2 we show the difference between the original method and ours. The strength of each vote, as described above, is given by the product $p(o_n, \lambda|C_i^q, l, q)p(C_i^q|\mathbf{e})$. As suggested in [68], the procedure for extracting local maxima is performed by the mean shift mode estimator technique.

4.1.6 Results

In order to show that our system is capable to detect objects at different scales, regardless of object scale in the training sets, we trained a codebook with only one training image and run the detection system both on the original image, and a set of scaled copies of the same. In Fig. 4.3 we can see one example of this experiment, where a scaled copy of the original training object is detected without any loss of precision. Please note how the interest regions, extracted by the automatic interest region finders, are quite different in the two images.

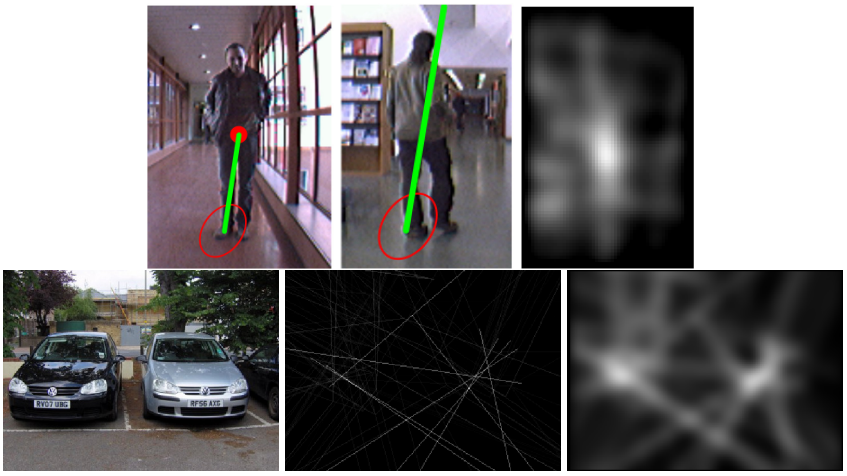


Figure 4.2: In this figure we show the difference between the original (top-left) and our (top-center) approach. Note that in the former each image patch casts a vote for the object center; in the latter each patch casts a votes for a line on which the object center may be found. The result of the proposed method is a voting space like the one shown in the top-right image. The lower three images show another example with the original image (bottom-left), the rough voting space (bottom-left) and the same space after some image processing (mainly Gaussian blurring) for the enhancement of the local maxima (bottom-right).

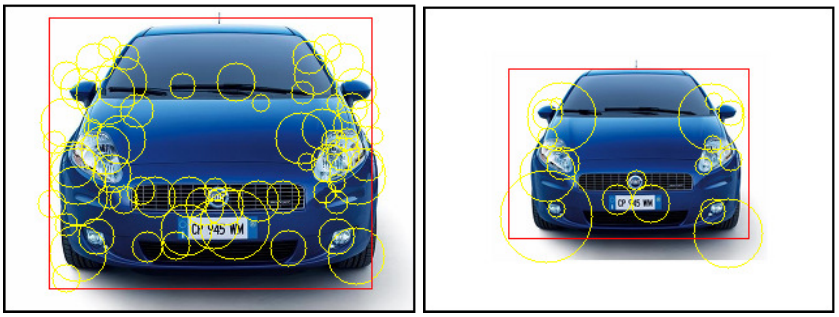


Figure 4.3: In this figure we show an example of the independency of our approach from the scale of the training objects. Please note that the interest regions, extracted by the interest region finders, are quite different in the two images.

Figure 4.4: The images used as training set.

We need also to demonstrate that our system is capable to detect objects in cluttered scenes and under difficult lighting, at a level at least comparable with the one of the original proposal. Out of the many experiments performed, we selected the ones concerning people. The codebooks for these experiments were generated from the 5 images presented in Fig. 4.4. The results are from images from other datasets. Fig. 4.5 presents some results for images from the RAWSEEDS datasets, which have been collected by a mobile robot in both indoor and outdoor conditions; they are freely available on the web, see [131]. The 5 training images were also taken from RAWSEEDS project ones. The results in Fig. 4.6 are from datasets from the VISOR repository, also freely available on the web, see [132]. The same codebook, trained without any particular attention on the images in Fig. 4.4, was used for the images from VISOR.

Note that the number of images in the training set is much smaller

Figure 4.5: Some results achieved by the detection system on images from the RAWSEEDS project.

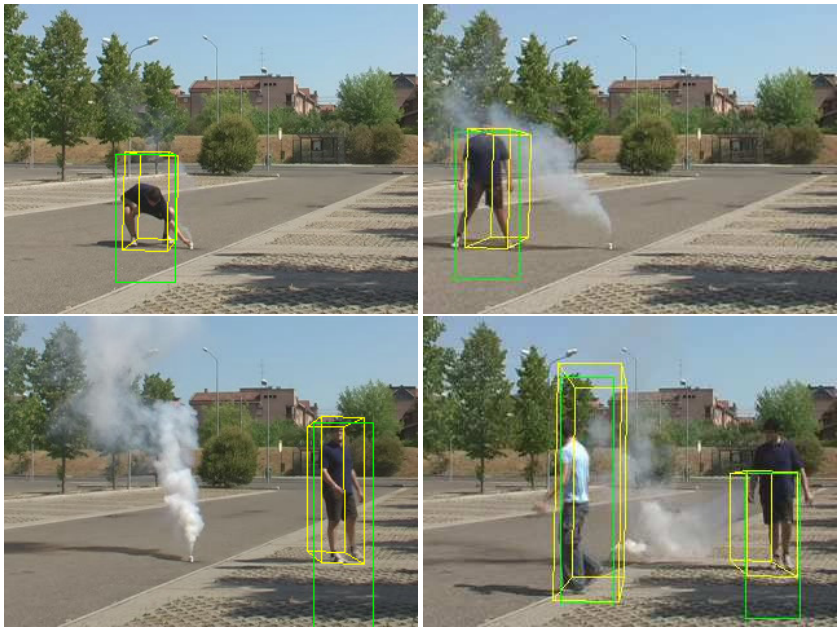


Figure 4.6: Some results achieved by the detection system on images from the VISOR datasets.

Figure 4.7: Some mistakes of the detection system: (left) a false positive; (right) a false negative, notice that the person detected is seating, a situation missing from the training set.

than in the training sets used by Leibe et al. (typically more than 200), see e.g., [128]. Furthermore, the training sets only consist of training images and their segmentation masks, without any information about object scale.

In Fig. 4.7 we show some mistake of the system. We also collected statistics about one VISOR dataset [1]. The dataset includes 1901 images, we processed one image every 20, but many of them did not include any person. In total we had 58 images including at least one person. The processed images were then annotated for groundtruth in order to collect the number of false positives and negatives. The results on a total number of 67 person observed in the 58 processed images are: 37 correct detections of the person; 4 "pure" false positives, i.e., the detection of a non-existent person when another person, correctly detected, was in the image; 11 "pure" false negatives, i.e., missed detection; 19 false positives with a false negative beside, i.e., an error in the localization of the person. Some explanation is in order for these mistakes: the camera parameters were not available and we used the parameters of the camera used in the RAWSEEDS dataset, see e.g., Fig. 4.8. Therefore these 19 errors are more the consequence of the un-accuracy of the camera calibration than of the un-accuracy of the detector.

4.1.7 Conclusions

This chapter proposes a substantial improvement to the ISM-based detection system proposed by Leibe et al., e.g., in [68]. Unlike the original approach, our method is independent from the object scale in the train-

[1] visor_1196179837385_movie11_viper.mpg

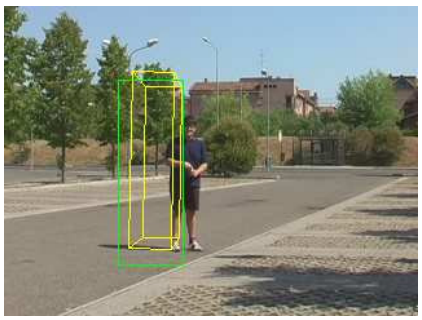


Figure 4.8: An example of the localization error for the VISOR dataset, turning into both a false negative and a false positive in the same image.

ing sets. This allows to work with much smaller training sets and to avoid to supply information about scale and size to the trainer. This has been done maintaining the robustness of the original approach. We also showed preliminary results on challenging datasets. We are currently working on creating a large statistic from the output of our system that will be used for comparison with other detection systems in a more structured way (ROCs, efficiency, etc.). The qualitative evaluation performed, in our opinion very satisfactory, showed that the percentage of correct detections, for the tested category of objects (pedestrians), is very high. False positives are quite rare. False negatives are primarily present in low contrast scenes, where interest region finders achieve worst performance. The size of the training sets (and, thus, of codebooks) are notably smaller than the ones required by the original approach.

4.2 Follow-up research work

4.2.1 Improvements for object detection - BISMD

Here two main improvements, with respect to the object detection approach presented so far, are presented, one for the training and one for the recognition phases.

Training

The codebook creation procedure described in Section 4.1.3 is divided into two steps, the second one consisting in revisiting all the codebook entries to mutually enrich similar ones. Experimental analysis pointed out that the similarity function used for the matching during the revis-

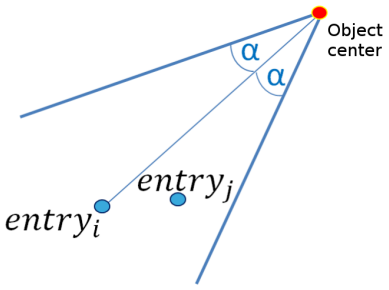


Figure 4.9: The spatial criterion for matching codebook entries during the enrichment process.

iting phase, as described there, is subject to two main week points that generate:

- Outliers due to histogram similarity of non-similar image patches. *E.g.* car wheels matched with car windshield in presence of reflections on this latter one.
- Outliers stemming from geometric symmetries of the object. *E.g.* front and rear car wheels, human limbs, etc.

These matching errors occur because the matching criterion is only based on the Euclidean distance of the descriptor's histograms. We propose to take into account also the spatial information of each image patch relative to the object's center. Thus, two codebook entries i and j will be regarded as similar according to

$$i \simeq j \Leftrightarrow (\|d_i - d_j\| < T_d) \wedge (|\text{SpatialError}| < T_s)$$

The first condition requires that the Euclidean distance between the descriptors d_i and d_j of the two codebook entries is below the threshold T_d, guaranteeing the appearance similarity (same as discussed in Section 4.1.3). The second condition can be explained referring to Figure 4.9: when matching two entries i and j, we attach to the entry i a spatial cone, which vertex is the object center, which axis connects the vertex and the spatial position of the entry and which angle is the threshold T_s. The entry j is said to be *spatially compatible* with the entry i, if its location falls within the spatial cone of i.

This expedient allows to prune out wrong matching during the revisiting phase of the codebook creation, thus preventing wrong or ambiguous votes during the recognition stage.

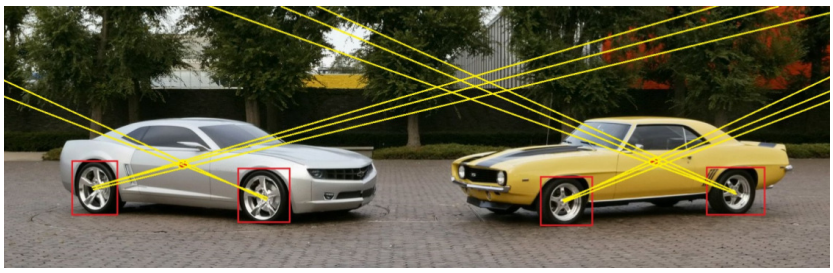


Figure 4.10: The main drawback of voting in terms of straight lines. Since lines propagate indefinitely, undesired interferences of voting lines from different objects in the image can generate strongly confident false positives.

Recognition

As discussed in Sections 4.1.4 and 4.1.5 both the original method presented in [68] and the method proposed in [124] rely on the Hough voting technique to convey the information from the single matched codebook entries into a common probabilistic framework. While in [68] each matched codebook entry casts votes for an object center in a 3D voting space (x, y, scale), in [124] votes are expressed in terms of straight lines starting from the matched image patch location and pointing towards the learned object center. In this latter case, the voting space reduces to a 2D one, since the scale information is embedded in the concept of "pointing towards", as opposed to the concept of "at this point". As discussed in Section 4.1.5, voting for a direction allows to naturally recognize objects at different scales, even those that were not present in the training set. However, also this approach has an important drawback, which is shown in Figure 4.10. It is clear from the drawing that voting in terms of straight lines inevitably leads to interferences between voting lines from different objects in the image, thus likely generating false local maxima in the voting space, i.e. false positive detections.

To overcome this important limitation of [124], here we propose to change the voting procedure so as to lessen the probabilistic strength (i.e. the intensity with which it is added to the voting space) of a voting line as it relinquishes from the image patch that generated it. Formally, if the line drawing procedure (Bresenham) is decomposed in its elementary steps, each pixel of the line will be drawn with probabilistic strength

$$p(x_k, y_k) = C_{Pr} \cdot p(x_{k-1}, y_{k-1})$$

where k denotes the drawing step and $C_{Pr} \in (0, 1]$ is the scaling

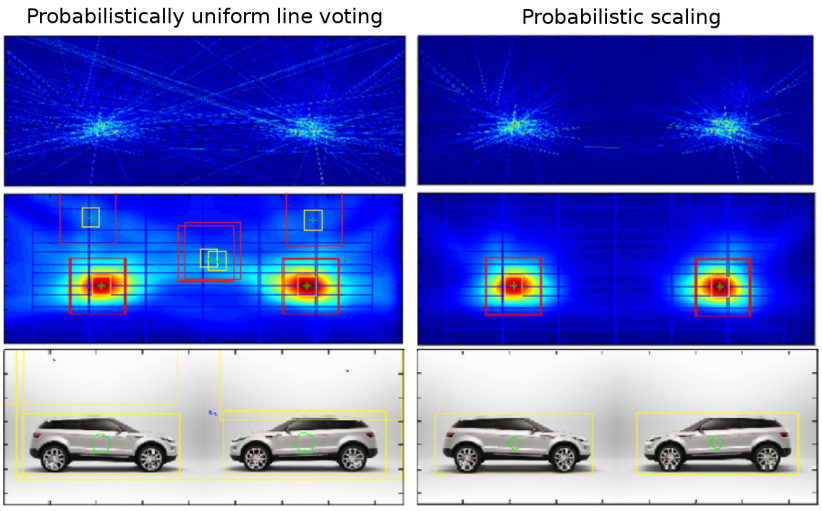


Figure 4.11: Overcoming the main drawback of voting in terms of straight lines. The probabilistic strength of a line decreases as it relinquishes the image patch that generated it. As a result, interferences between voting lines from different objects are most unlikely to occur.

factor that determines the decaying speed of the voting line probability. Figure 4.11 shows a comparison of the results when voting in terms of probabilistically uniform lines or by adopting the probability-scaled version proposed here. As it can be seen, the main drawback of [124] is overcome in the right column, resulting in a much cleaner voting space and in the absence of false positive detections.

Detection results

Preliminary results of extensive experimental activity are presented here. The training set adopted for the car detector is the one used in [67], shown in Figure 4.12. For a broader evaluation, several state-of-the-art datasets are considered:

- TuGraz [133]
- Kitti [134]
- Karlsruhe [135]
- UIUC Multiscale [136]
- VOC motorbikes test2 [137]

Figure 4.12: The training set used for training the car detector.

Figure 4.13: Some results achieved on the TuGraz datasets.

- Cow dataset [67]

Examples of recognition performances on these datasets are shown in Figures 4.13, 4.14, 4.15, 4.16, 4.17 and 4.18.

Also a video dataset (Vimodrone76) was acquired for tracking purposes (Section 4.2.1), framing lateral car views. Figures 4.19 and 4.20 show the qualitative results on this dataset, comparing the original method (ISM [67]) and the two proposed in this chapter (SIODISM and BISMD, respectively) in Figure 4.19 and, in Figure 4.20 SIODISM and BISMD performances with and without applying the MDL criterion [67]. As it can be seen from the charts of these preliminary results, the BISMD approach greatly improves over both ISM and SIODISM.

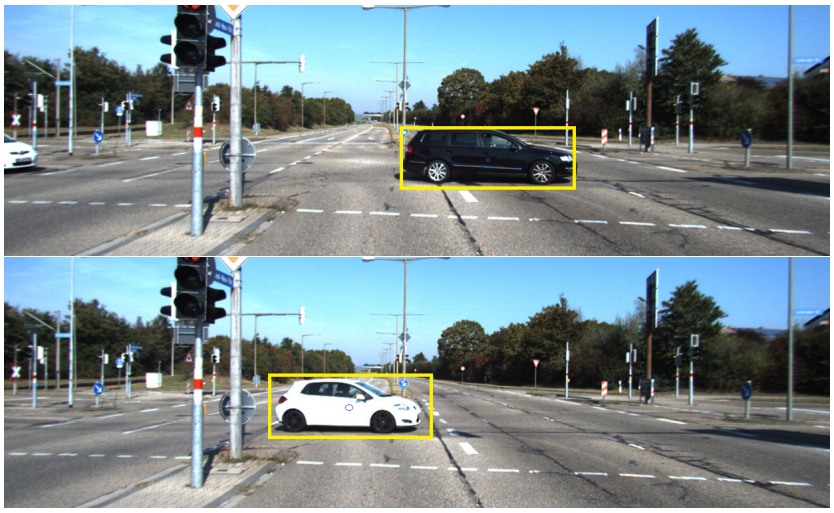


Figure 4.14: Some results achieved on the Kitti datasets.

Figure 4.15: Some results achieved on the Karlsruhe datasets.

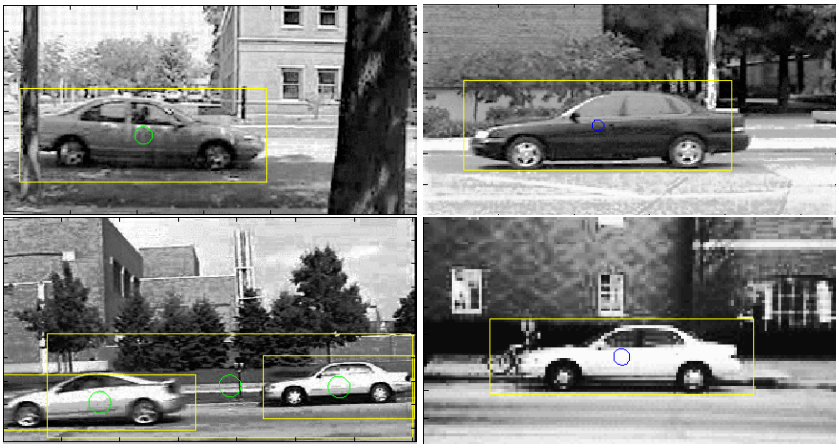


Figure 4.16: Some results achieved on the UIUC Multiscale datasets.

Figure 4.17: Some results achieved on bikes.

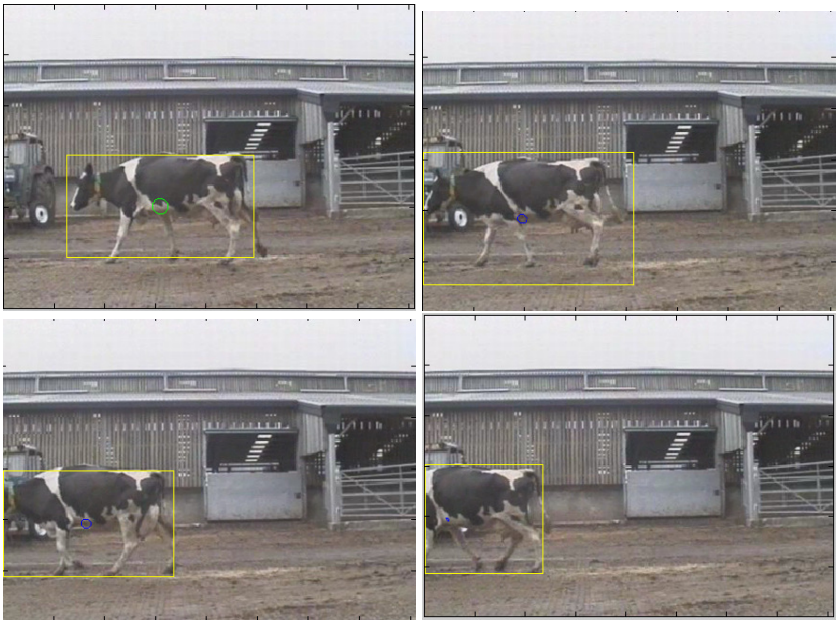


Figure 4.18: Some results achieved on cows.

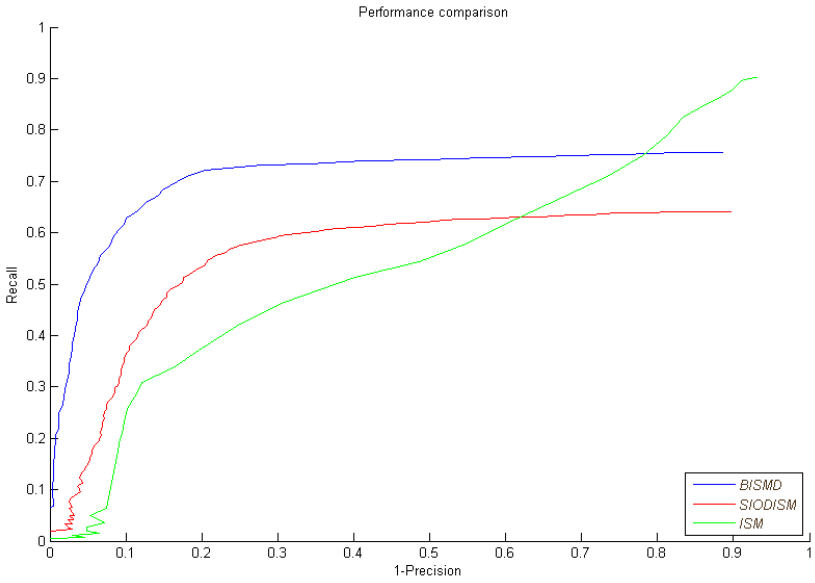


Figure 4.19: The RPC curve comparing ISM, SIODISM and BISMD on the Vimodrone76 dataset.

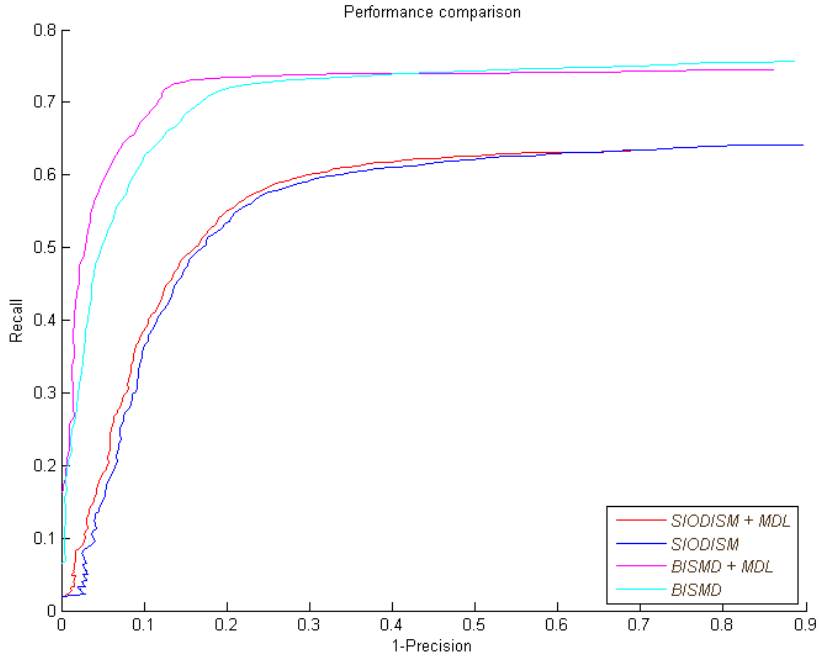


Figure 4.20: The RPC curve comparing SIODISM and BISMD on the Vimodrone76 dataset, with and without applying the MDL criterion [67].

Object tracking

In order to test and prove its effectiveness, the BISMD approach proposed in the previous section was used as detector for tracking objects in image streams. The implemented object tracker (BMTT) is a multi-target tracking algorithm inspired by [104], which instantiates a particle filter for each tracked object. Observations used by the filters are probabilistic combinations of BISMD object detections and adaptive model on-line boosting classifiers [101]. In this way, each adaptive model is trained on-line on a specific target and kept updated at each time frame, based on the new image evidence. Figure 4.21 shows the main structure of the BMTT multi-target tracker.

The results of the tracking on three datasets *Vimodrone73, Vimodrone76* and *Agrate Highway* are shown in Figures 4.22 through 4.28. Please note how the system is able to instantiate trackers also when objects appear in the middle of the scene, and that objects are successfully tracked even in the case of partial or total occlusion.

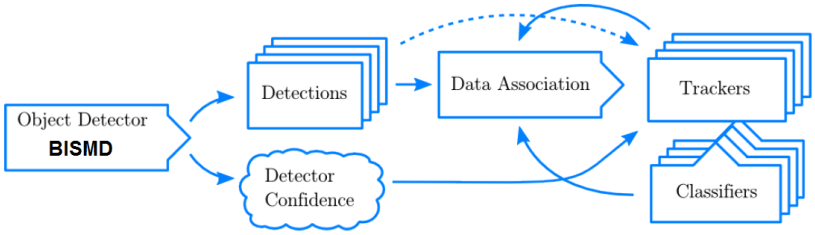


Figure 4.21: This Figure shows the interaction between the different modules of the BMTT multi-target tracker.

4.3 Acknowledgements

This work has been partially supported by the European Commission, Sixth Framework Programme, Information Society Technologies: Contract Number FP6-045144 (RAWSEEDS).

Special thanks to the Master Students L. Prisciantelli and G. Ventimiglia for their research work on this topic. Their work was funded by ST Microelectronics.

Figure 4.22: Tracking results on the dataset *Vimodrone76*.

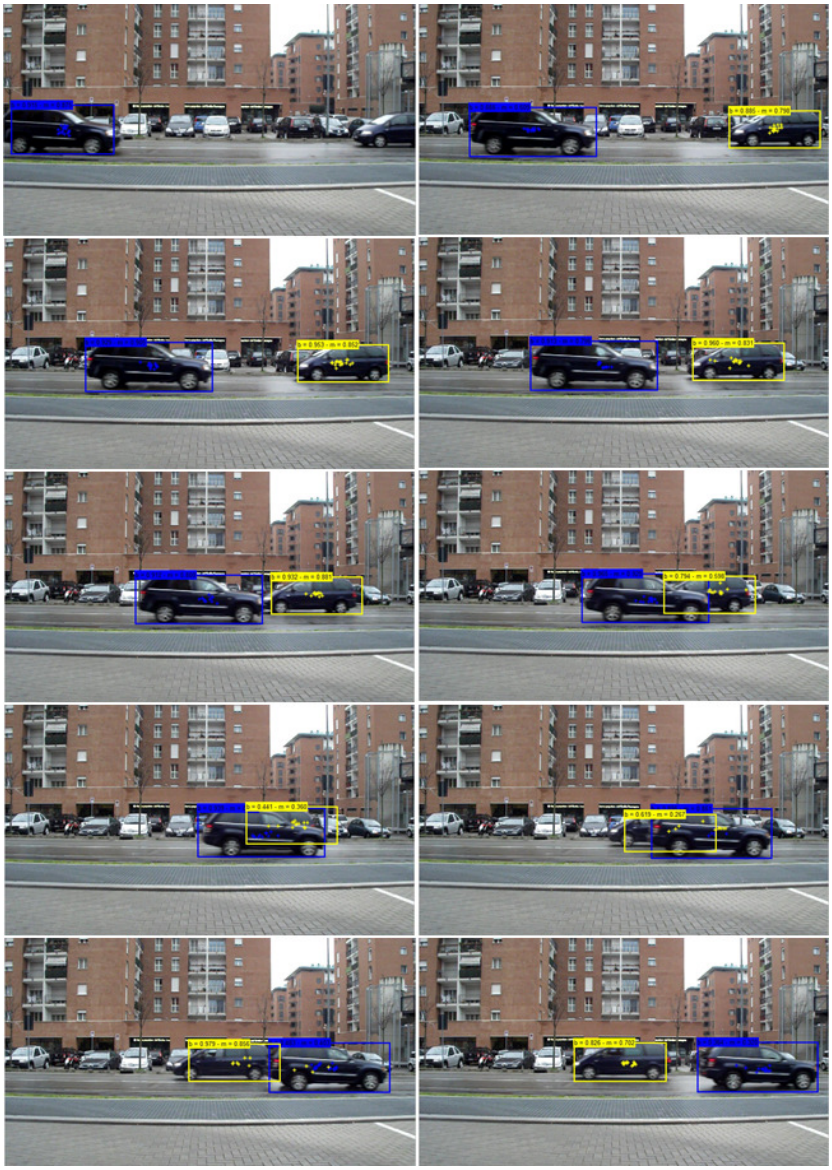


Figure 4.23: Tracking results on the dataset *Vimodrone76*.

Figure 4.24: Tracking results on the dataset *Vimodrone73*.

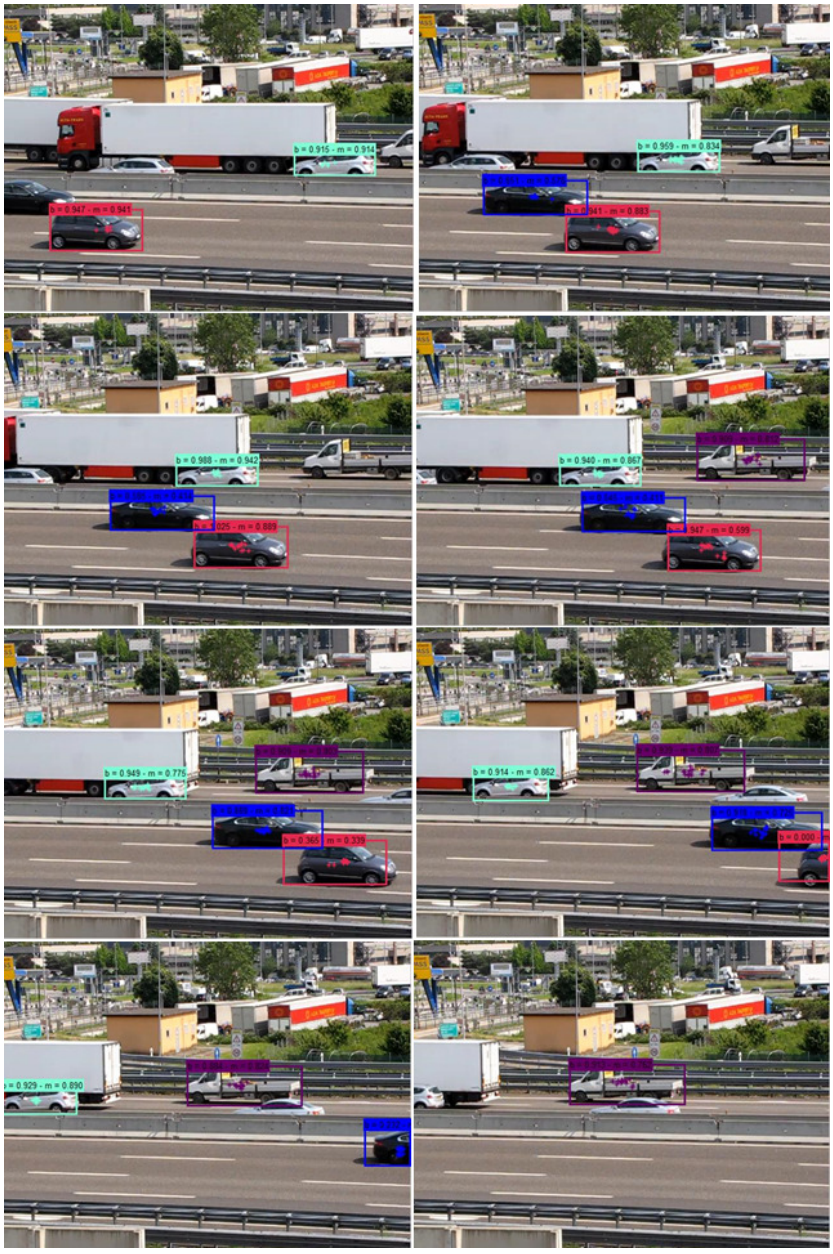


Figure 4.25: Tracking results on the dataset *Agrate Highway*.

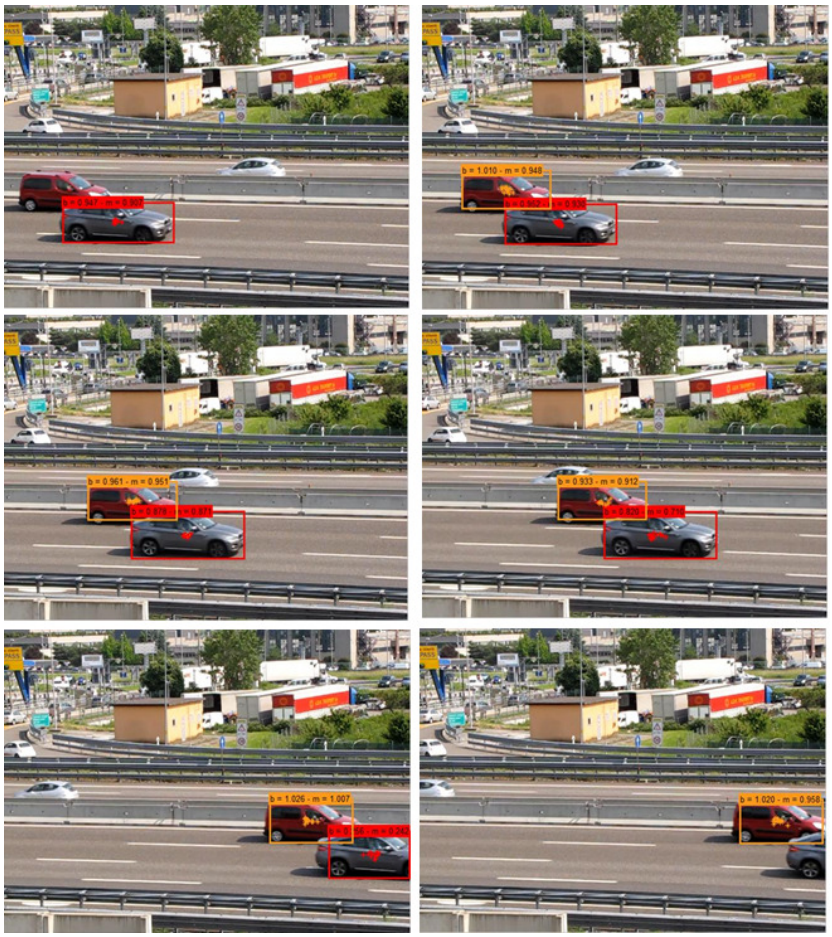


Figure 4.26: Tracking results on the dataset *Agrate Highway*.

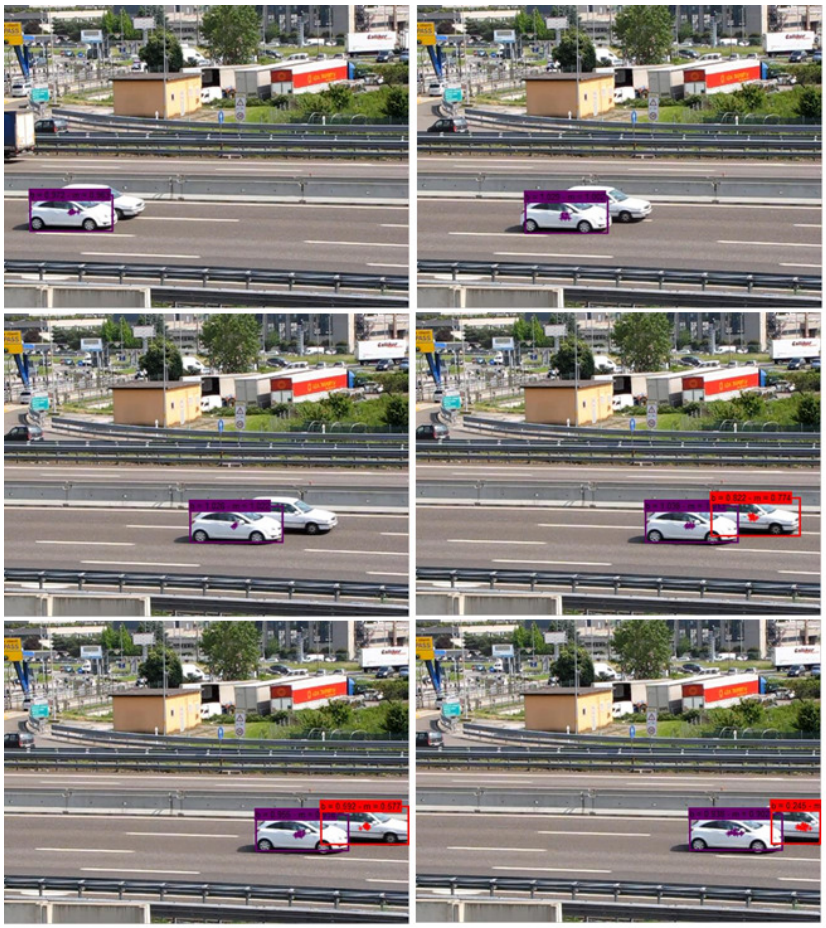


Figure 4.27: Tracking results on the dataset *Agrate Highway*.

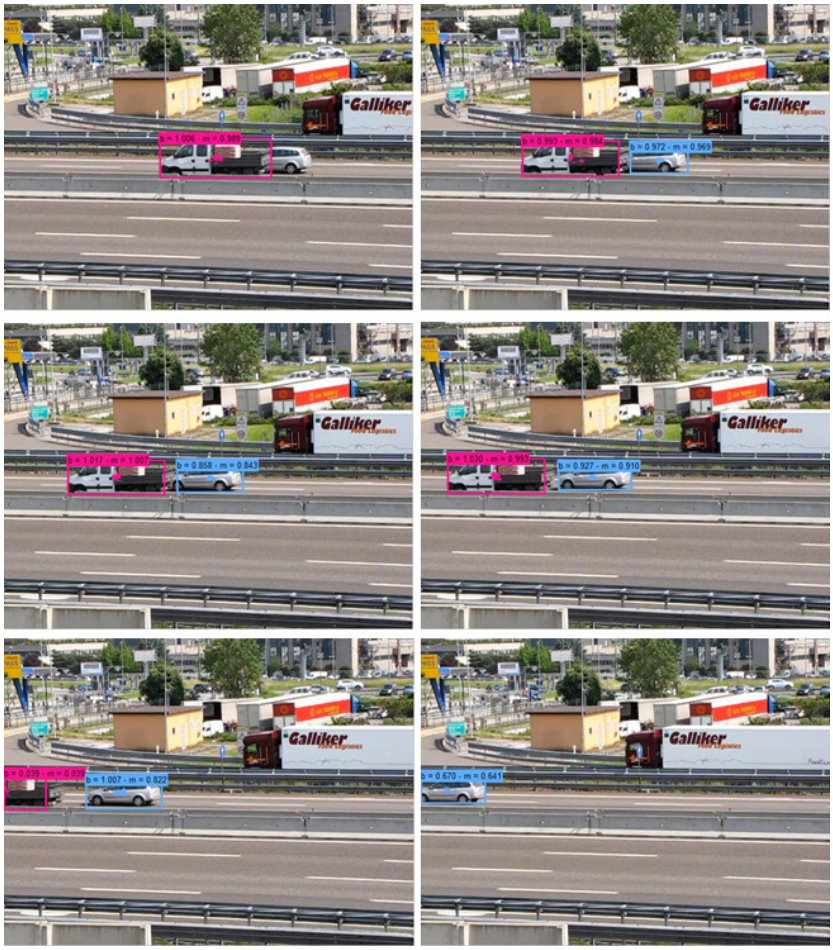


Figure 4.28: Tracking results on the dataset *Agrate Highway*.

Chapter 5

Indoor scene reconstruction

The main goal of the Scene Understanding problem is to reproduce the human ability of inferring the global structures and situations of an observed scene. Such ability could greatly improve performances of applications such as autonomous driving and human machine interaction. This chapter will focus on understanding indoor scenes from video sensors. Many works have been presented for indoor scene understanding, yet few of them combine structural reasoning with full motion estimation in a real-time oriented approach. In this work we address the problem of estimating the 3D structural layout of complex and cluttered indoor scenes from monocular video sequences, where the observer can freely move in the surrounding space. An effective probabilistic formulation is proposed, that allows to generate, evaluate and optimize layout hypotheses by integrating new image evidence as the observer moves. Compared to state-of-the-art work, this approach makes significantly less limiting hypotheses about the scene and the observer (e.g., Manhattan world assumption, known camera motion). Concomitantly, a new challenging dataset is introduced along with an extensive experimental evaluation, which demonstrates that the proposed formulation reaches near-real-time computation time and outperforms state-of-the-art methods while operating in significantly less constrained conditions.

5.1 Introduction

The indoor scene reconstruction problem has sparked lively interest in the research community in the past few years. The ability to under-

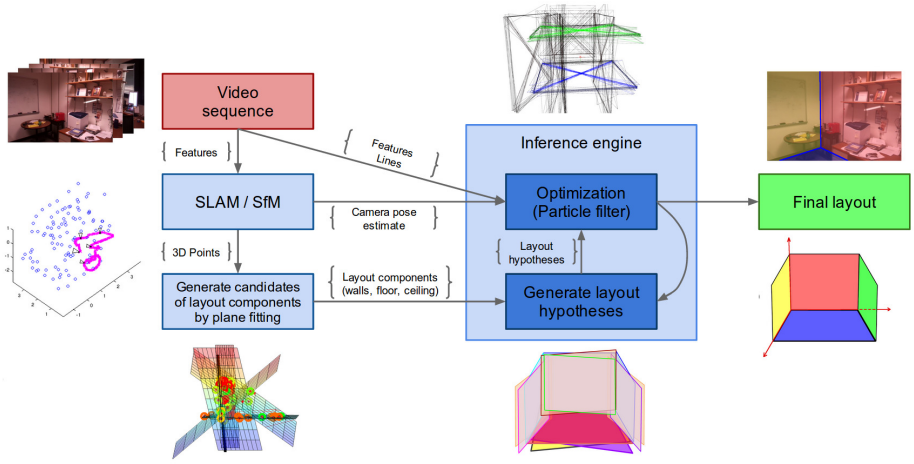


Figure 5.1: **3D scene layout estimation process.** The video sequence is first processed to obtain camera localization and sparse 3D point cloud reconstruction. Layout components (e.g. floor, ceiling, walls) are generated from the sparse 3D points and combined to generate layout hypotheses (see Section 5.2). Each layout hypothesis is evaluated and optimized by incorporating new image evidence. The final 3D scene layout is represented by the hypothesis that better describes the scene (see Section 5.3).

stand the structural geometry of living spaces allows, for example, autonomous robots to safely move in the surrounding environment, as well as the development of augmented reality applications. Many contributions have been proposed to either address the problem of recognizing semantically meaningful components, such as floor or walls in 2D images, or the problem of generating sparse 3D point-cloud maps of the observed scene while moving within it. By contrast, few works have addressed the problem of reconstructing semantically consistent indoor structures (also referred to as 3D scene layout, see Figure 5.1), and refining layout hypotheses by integrating new evidence acquired as the observer moves around. Among those that do, simplifications are made to solve the problem, such as the Manhattan world assumption, partially or fully known observer motion, and occasionally human intervention for a correct initialization.

In this work we pursue a semantically consistent reconstruction of the structural elements in indoor scenes, excluding any a priori knowledge of the observer's motion, human intervention, or hard Manhattan constraints. We start by observing in our experiments that state-of-the-art 3D reconstruction techniques, such as SLAM and SfM, are far from achieving reliable results that can be directly translated to a higher se-

mantic reconstruction. This phenomenon is particularly noticeable when the observed scenes are highly cluttered and/or mostly featureless. We therefore propose a probabilistic framework, to be fed with such noisy elaborations, that allows us to efficiently integrate new evidence to generate, evaluate and refine 3D layout hypotheses as the observer moves through the scene. Different types of information extracted from images (points, lines, regions) and, potentially, from other kind of sensors, can be easily merged into our probabilistic formulation, allowing us to cope with both clutter and featureless surfaces.

We present extensive experimental results on challenging sequences, demonstrating improvement over state-of-the-art approaches while operating in significantly less constraining conditions and in near-real-time (in the order of $\sim$20fps). We propose a new challenging dataset to prove the full capacities of the proposed method and which is available for future comparisons [138].

5.1.1 Contributions

In this work an efficient method for estimating 3D indoor layout from an arbitrary 6DoF moving monocular observer is proposed, whose motion is estimated using state-of-the-art techniques such as SLAM and key-framed SfM. With respect to previous work, the approach presented here introduces important improvements by:

- Eliminating the hard Manhattan world assumption.
- Requiring no *a priori* knowledge of the observer motion with respect to the scene.
- Operating at near-real-time speeds ($\sim$20fps).
- Introducing a new challenging dataset which features cluttered, non-Manhattan and non-box-shaped scenes.

The reminder of this chapter is organized as follows. Section 5.2 presents an overview of the proposed approach, the probabilistic framework is explained in Section 5.3, along with the details of the proposed layout parameterization, the generation and evaluation of the layout hypotheses. Section 5.4 presents the extended experimental activity and a thorough discussion of success and failure cases.

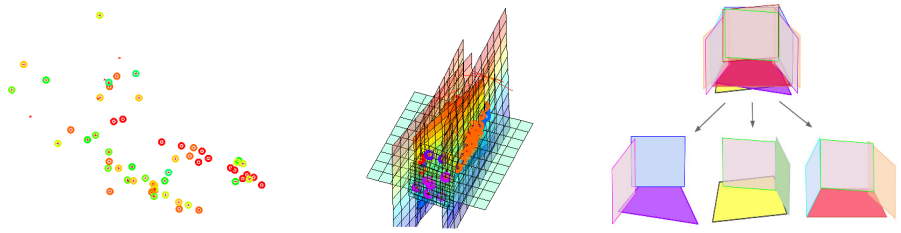
5.2 Method Overview

This section describes the framework within which layout hypotheses are generated, evaluated and updated or rejected. Figure 5.1 shows a pictorial representation and a schematic diagram of the whole process.

Sparse 3D reconstruction. As the observer moves in the surrounding environment and the image stream is acquired, we first pre-process sequences with a localization and sparse 3D reconstruction algorithm. Since the main contribution of this work is not related to the 3D reconstruction problem, the proposed framework is designed so as to be able to work with any algorithm that can provide a camera motion estimation as well as a cloud of 3D points. In the experimental activity two such approaches are compared: a real-time implementation of the Monocular V-SLAM approach proposed in [22] and the non-real-time VisualSfM [36]. Notice that these 3D reconstructions are in general noisy and sparse (Figure 5.2(a)).

Hypotheses initialization. The second step consists of generating a higher level representation of the 3D points estimated in the pre-processing phase. Several types of geometrical primitives are suitable for this purpose. In our case, we believe a piecewise planar representation is the most appropriate for indoor scene representation. Thus a large number of planes is fitted to the 3D points so as to generate a large number of (potentially inaccurate) candidates of layout components, i.e. walls, floor, ceiling (Figure 5.2(b)). For the experiments an iterative RanSaC plane fitting procedure was implemented, which is optimized for indoor scenes by allowing peripheral fitted points to be re-injected in the iteration process, since these points potentially lay on the intersection of two planes.

Layout estimation. In the last step, which constitutes the core of the proposed inference engine, layout hypotheses are generated as random combinations of candidate layout components (Figure 5.2(c)). Each layout hypothesis is evaluated at each time frame by measuring its compatibility with observations (*e.g.* image points and lines) and geometrical constraints across frames. During this process, each layout is "perturbed" (see Section 5.3.3) by locally adjusting, optimizing, merging or splitting layout components. There are different approaches to manage sets of



(a) (b) (c)

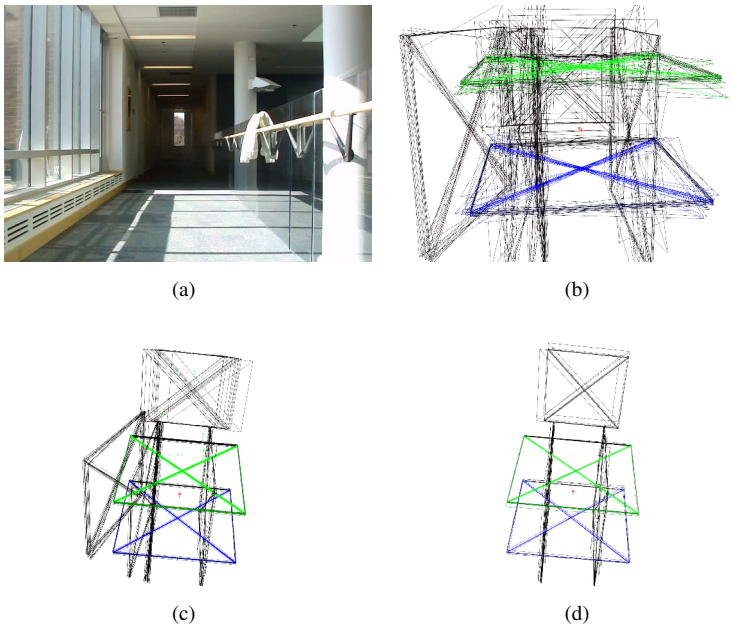
Figure 5.2: Left: a noisy sparse 3D reconstruction. Please note that when using Visual SLAM techniques for 3D reconstruction, the amount of estimated 3D points is very limited, typically not exceeding one or two hundreds of points. Center: a large number of candidate layout components obtained with plane fitting. It is clear from the image that fitting planes to 3D points inevitably produces a considerable amount of "wrong" planes. Right: The process of generating layout hypotheses as random combinations of candidate layout components (i.e planes).

hypotheses. In this work we choose to integrate the probabilistic framework within a particle filter structure. This choice allows to explicitly formulate the problem in a parallel-computing oriented fashion (particles are independent from each other), which can lead to high efficiency gains in computation time. The output of the optimization procedure is an estimation of the 3D scene layout, which is obtained by selecting the layout hypothesis with the best set of layout components (see Section 5.3).

Advantages. There are two important advantages stemming from the choices mentioned above:

i) The local optimization step applied to each layout component helps recovering from noisy initialization (see the transition from Figure 5.3(b) to Figure 5.3(c)) while keeping the computation amount affordable for real-time applications. This could not be achieved with standard "brute force hypothesize and test" approaches, *e.g.* [106, 110].

ii) The choice of embedding the proposed probabilistic formulation within a particle filter also allows to exploit some critical properties of particle filters, like multi-modal posterior representation, resampling, particle clustering and, most importantly, recover from substantially wrong initializations (see, *e.g.* transition from Figure 5.3(c) to Figure 5.3(d)).



(a) (b) (c) (d)

Figure 5.3: From top-left to bottom-right, an example of indoor scene and of the evolution in time of particles (i.e. layout hypotheses) throughout the optimization process. For clarity, only the 40 best hypotheses are drawn here. Note how the unlikely layout hypotheses disappear while the most plausible ones survive and get refined.

5.3 Probabilistic Layout Estimation

As stated above, the input to the proposed inference engine is an image sequence from which the camera motion and sparse 3D points are estimated. These 3D points are used to generate layout components (walls, floor, ceiling) by fitting a large number of planes through such points (see Section 5.2). The output of the inference engine is the layout hypothesis that better explains the scene in terms of compatibility with observations and geometrical constraints. At the heart of the proposed approach is a particle filter-based optimization [139] that is capable of processing candidate layout components so as to obtain plausible scene layouts, where the layout hypotheses represent the particles of the filter. As with any optimization strategy, this requires four key components: a principled choice of the layout parameterization (5.3.1); an initialization strategy (5.3.2) to generate the initial layout hypotheses; a method for exploring the state space (5.3.3); a score function (5.3.4) to evaluate the quality of layout hypotheses. In the following section, we will discuss each of these aspects in details.

5.3.1 Layout Parametrization

While much prior work has leveraged the Manhattan world assumption, we believe this is a limiting hypothesis. To overcome this limitation, in this work a representation similar to [119, 120] (sometimes referred to as Soft Manhattan) is adopted, which makes the following assumptions about the environment:

i) Ground plane and ceiling are parallel.

ii) Walls are only constrained to be orthogonal to the ground plane (and ceiling).

iii) There can be any number of walls and each wall can be displaced at any angle with respect to other walls.

Thus a room layout is fully parametrized by its gravity vector, the heights of its ground and ceiling, and a set of walls with only one degree of freedom for rotation and one for translation.

5.3.2 Initializing Layout Hypotheses

A layout hypothesis is generated as follows. First, the rough direction of gravity is determined as given by an IMU or, if none is present, by assuming that the camera optical axis is roughly horizontal when the sequence begins. The heights of ground and ceiling are then approximated by the lowest and highest features tracked by SLAM/SfM along this direction. If they are not observed at initialization time, their heights will be underestimated at first and adjusted as more parts of the scene are observed. Subsequently, sampling from the initial set of planes fitted to the 3D points obtained from the SLAM/SfM reconstruction (Section 5.2), each layout hypothesis is assigned a random number of candidate walls (Figure 5.2(c)). While being assigned to the layout hypothesis, each wall is transformed by finding the minimal transformation needed to make it orthogonal to the ground.

5.3.3 Exploring the State Space

The above step gives a very rough initial estimate of the scene layout. In order to refine this estimate, we propose to "perturb" the layout components assigned to each layout hypothesis. This can be done within the hypothesis itself by:

- Rotating wall i about the gravity vector by θ_w^i and optimizing its reprojection error
- Translating wall i by some distance d_w^i and optimizing its reprojection error

thus refining the hypothesis itself, or by generating a set of new hypotheses. Given a layout hypothesis at time $t-1$, a set of new hypotheses can be generated for the time t "perturbing" the existing one by:

- Rotating the ground plane about a random direction by some angle θ_g
- Translating the ground or ceiling by some distance, d_g or d_c
- Rotating wall i about the gravity vector by θ_w^i
- Translating wall i by some distance d_w^i
- Removing a wall which is currently hypothesized
- Adding a wall which had been removed
- Taking no action

where θ_g, d_g, d_c, θ_w^i, and d_w^i are normally distributed. For each layout hypothesis, only a single action may be performed per time step, as determined by a weighted coin flip. The choice of allowing only one action per hypothesis per time step is driven by:

i) Keeping the computation amount as low as possible to achieve real-time performances

ii) Processing image streams of up to 30fps (or more), the expected changes in the observed scenes are small from frame to frame. Thus, introducing too much perturbation at each time step could lead to a fast divergence of the estimation process.

5.3.4 Scoring Hypotheses

At each timestep t, we wish to evaluate the level of agreement of each hypothesis with the sensor data, i.e., to assign a probability of being a correct interpretation of the observed scene, to each hypothesis, taking into account new observations and geometrical constraints:

$$P_t = \prod_i P_f^i P_o^i(\theta_i) P_r^i(e_r^i) \prod_j P_m^{ij}(\phi_{ij}) P_s^{ij}(d_{ij}^{-1})^{p_{ij}} (P_w^{ij})^{a_{ij}} \tag{5.1}$$

where the exponents of the two rightmost terms, p_{ij} and a_{ij} are binary values that are initially set zero. p_{ij} is set to 1 if walls i and j are near-parallel by some angular threshold and a_{ij} is set to 1 if they are adjacent. This scoring function is designed to enforce a number of desirable properties.

Fitness: The initial pre-processing yields planes with varying goodness of fit. Thus, to each visible wall, we assign a corresponding fitness term P_f^i, represented by a zero-mean Gaussian over the residual least-square error after the plane fitting process.

Orthogonality to Ground: In Section 5.3.2 we described a method for generating orthogonal-to-ground wall candidates from non-orthogonal planes. The more these planes are altered, the less they are supported by data. This aspect is captured with a zero-mean Gaussian $P_o^i(\theta)$, where θ denotes the amount of rotation required to orthogonalize the wall to the ground.

Low Reprojection Error: Many feature points tend to fall on the ground, ceiling, or walls of a scene. To make use of these visual cues, we track features using the Kanade-Lucas-Tomasi method [140]. To be robust to outliers, those matches are discarded which i) could not have come from a plausible camera motion, or ii) don't share a homography with a minimum number of other points in the scene. At time t, we project the keypoints at $t-1$ onto all possible walls, and evaluate their hinge-loss 2D reprojection error. Each point is then assigned to the wall that minimizes its reprojection error, and the average e_r is computed for each wall. For each wall we then assign a probability $P_r^i(e_r)$, which is normally distributed about 0 pixels.

Manhattan Layout: While we do not leverage the Manhattan assumption to generate layout hypotheses, we recognize that angles are far more likely to fall in 90° or 45° increments than, e.g., 87°. To capture this fact, for each pair of visible walls we include a term $P_m^{ij}(\phi)$, where ϕ is their relative angle modulo 90° (or 45°) and P_m^{ij} is a zero-mean Gaussian.

Simplicity: Adding more walls will always improve reprojection error. Actual layouts, however, are fairly simple: they are far more likely to contain one large wall than many small ones. To enforce simpler interpretations, for two near-parallel walls we assign a probability $P_s^{ij}(d^{-1})$

which captures how redundant wall i is given the presence of wall j, d meters away. This is normally distributed about $\frac{1}{d} = 0$, or $d \to \infty$.

Wall-wall intersection: Small errors in the estimation of wall rotations can not be reliably captured by the reprojection error term. Yet, such small errors can lead to substantial errors in the displacement of the intersection between two walls. We exploit this intuition by assigning a probability P_w^{ij} that weights the image evidence supporting an intersection between walls i and j. To obtain this evidence, the 3D line segment resulting from the intersection is projected into the image and there compared against 2D line segments extracted with a Canny edge detector [141].

The final output of the optimization procedure is an estimation of the 3D scene layout, obtained by selecting the layout hypothesis that best describes the scene in terms of the score function in Eq. 5.1.

5.4 Results

This section presents the experimental results of the proposed method when tested on the state-of-the-art dataset [119], as well as on a new challenging dataset that is introduced in this work and that is available for future comparison [138]. To the best of our knowledge, the dataset [119] is the only state-of-the-art dataset that can be used for comparison for this type of problem. Since the proposed method requires video sequences as inputs, some datasets cannot be used for evaluation because they feature single images [107] or non-video (i.e. sparse) images [142, 143].

5.4.1 Experimental setup

First, two sparse 3D reconstruction techniques, RT-SLAM [22] and VisualSfM [36] are used to pre-process the image sequences, then the generated 3D point clouds and camera pose estimations are fed to our algorithm, which outputs the final 3D layout reconstruction. Final reconstruction results are compared to:

- **State-of-the-art approaches:** the video-based approach proposed in [119] and two well known single image methods, [144] and [107] (Section 2). For completeness, in Table 5.1 we report the results of [144] composed with a MRF over image frames. Please refer to [119] for a comprehensive description of this composition.

Method	Excluding ceiling	Including ceiling
[119]	**90.58**	82.17
[144]	82.62	83.30
[144] + MRF	81.44	82.13
[107]	84.70	84.33
Our + VSLAM	86.92	**87.01**

Table 5.1: Classification accuracy on the Michigan Indoor Corridor Video Dataset. Our results are compared to the results obtained with [119], [144] and [107].

Method	Classification accuracy	Average fps
Baseline	70.64	—
[144]	59.29	0.17
[107]	73.59	0.03
Our + VSLAM	**86.24**	**21.63**
Our + VSfM	75.94	16.90

Table 5.2: Classification accuracy on the proposed dataset. Our results (with SLAM and SfM) are compared to the results obtained with a naive baseline method, [144] and [107].

- **Baseline method:** in order to show the importance of the evaluation and optimization process (Section 5.3), we built a baseline method consisting in projecting all the possible combination of layout components (i.e. fitted planes, see Section 5.2) into the image and picking the combination that achieves the best classification accuracy.

In the experiments we evaluate the quality of the final reconstruction by means of the classification accuracy, which is a commonly adopted metric [107, 119, 145]. It is defined as the percentage of correctly labeled pixels when projecting the estimated 3D scene layout into the image. In order to evaluate if a pixel is correctly labeled, a groundtruth image is provided. Labels indicate if the pixel should belong to the ground floor, to the ceiling or to a wall numbered with an incremental counter. Please note that, for all the parameters described in Sections 5.3.3 and 5.3.4, the same configuration was used for all sequences.

5.4.2 Michigan Indoor Corridor Video Dataset

This dataset was proposed in [119] and consists of a set of image sequences collected in various indoor environments with a calibrated cam-

era mounted on a mobile robot. The camera is set up to present zero tilt (pitch) and roll angles and known fixed height with respect to the ground floor. The authors in [119] state that their approach strictly relies on these specific setup constraints and on the ground-walls' boundaries detected in the images. This implies that, if the observer does not move parallel to the ground with known height and if those boundary lines are not observed, the approach will not be able to generate initial layout hypotheses. On the other hand, our approach does not require any of these assumptions.

The quantitative results of the tests on this dataset are presented in Table 5.1, while Figure 5.4 shows a visual overview of our performance. There are a few sequences for which neither SLAM nor VisualSfM are able to produce any 3D reconstruction due to the very small amount of motion of the observer (insufficient parallax). These sequences were not taken into account for the evaluation. Please note that the method in [119] cannot recover the ceiling part of the scene layout, therefore the authors did not include these pixels in the evaluation of the performances. Since our approach as well as [144] and [107] are able to estimate the ceiling component of the scene layout, and in order to present a more complete comparison, we add in Table 5.1, beside the original values, the results where ceilings are included in the evaluation. Please note that, when excluding the ceiling, the proposed method is second only to [119] (which was designed to work specifically in such constrained scenarios), while, when taking into account the whole scene, including ceiling, the proposed method outperforms all other approaches, while operating in significantly less constraining conditions.

5.4.3 Proposed dataset

The sequences in the dataset [119] feature substantially simple environments, as can be seen in the 3D reconstructions in Figure 5.4. With this work we introduce a new dataset [138] to evaluate the full capabilities of the proposed approach. As opposed to the previous dataset, we let the observer freely move (6DoF) around to observe the scene. We collected 10 sequences in a variety of environments, spanning offices, corridors and large rooms. Most of the sequences frame ground-walls boundaries for short periods or do not frame them at all; some present scenes that cannot be represented by a simple box layout model or relying on the Manhattan world assumption. All the sequences were collected with

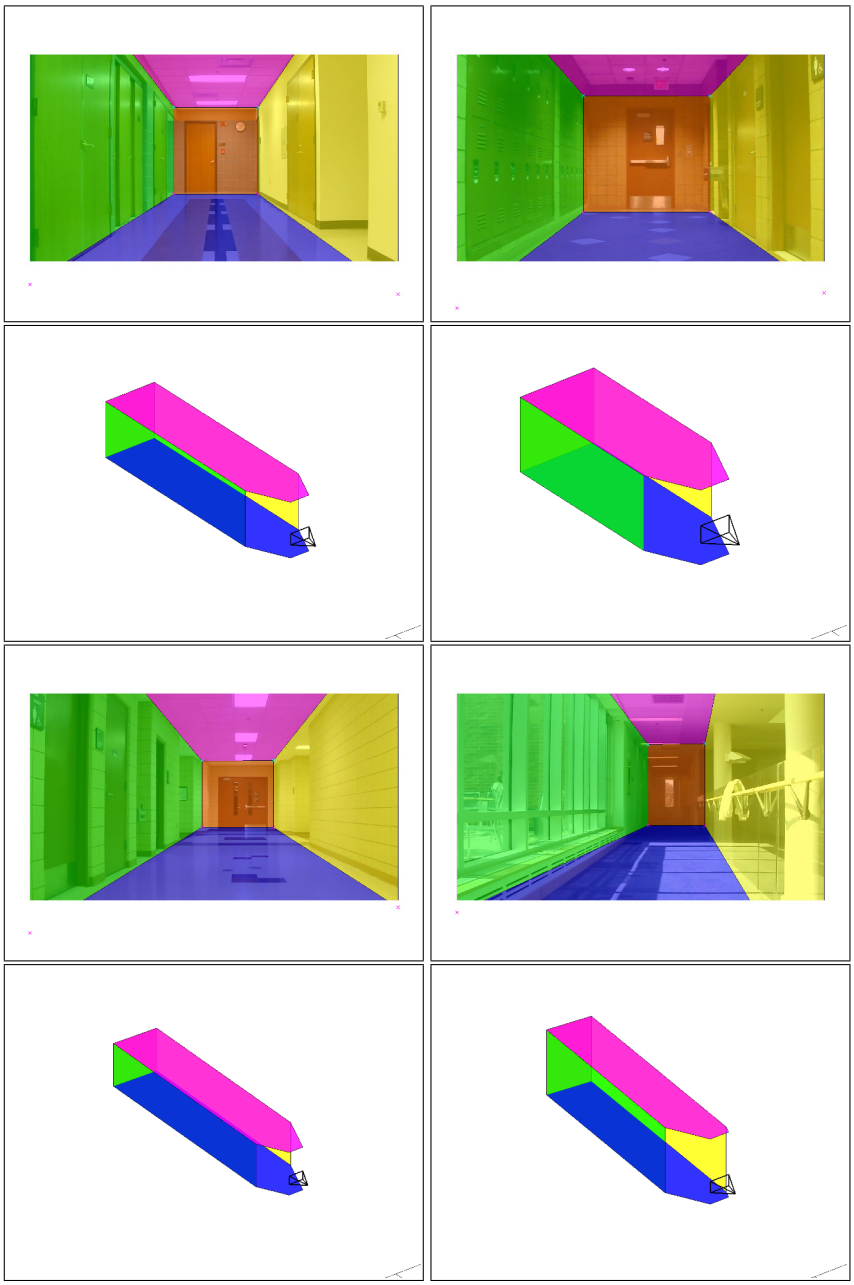


Figure 5.4: Some examples of our results on the dataset presented in [119]. The first and the third rows show the reprojection of the best layout hypothesis into the image, while the second and the fourth rows show the same layout hypothesis in the 3D space along with the camera pose.

common smartphones, in the attempt to test the proposed method in real-life scenarios with low-cost sensors.

We compare our proposed method, separately fed with the sparse 3D reconstructions by VSLAM [22] and VisualSfM [36], with the baseline method and with the single-image approaches proposed in [144] and [107]. The method proposed in [119] could not be used for comparison, since it requires the observer to move parallel to the ground floor and with known roto-translation between the ground plane and the camera.

The classification accuracy results and the mean execution speed (in fps) of the tests on this dataset are presented in Table 5.2, while Figure 5.5 shows a visual overview of the dataset and of our performance. In Table 5.2, please note that: i) the proposed method significantly outperforms state-of-the-art methods in both classification accuracy and execution time; ii) when feeding the proposed approach with the SfM reconstructions, in order to keep the execution time reasonable, both SfM and the optimization procedure were run on a small subset of frames which, despite the ability of SfM to produce denser reconstruction with respect to SLAM, led to worst reconstruction results.

Please refer to Section 5.4.4 for a discussion of failure and success cases, additional images and the complete table of the experimental results.

5.4.4 Failure and success cases analysis

Here we briefly discuss the failure and success cases on the proposed dataset. In Table 5.3 we present the classification accuracy for each sequence in the dataset achieved by the compared methods, while in Figures 5.6 through 5.10 we show the corresponding visual results. Figures are organized so that columns represents sequences (see the captions of the Figures), while rows are encoded with the following order:

- First frame of the sequence.
- Reconstruction result by [144].
- Reconstruction result by [107].
- Reconstruction result by the proposed method fed with VSLAM (projection).

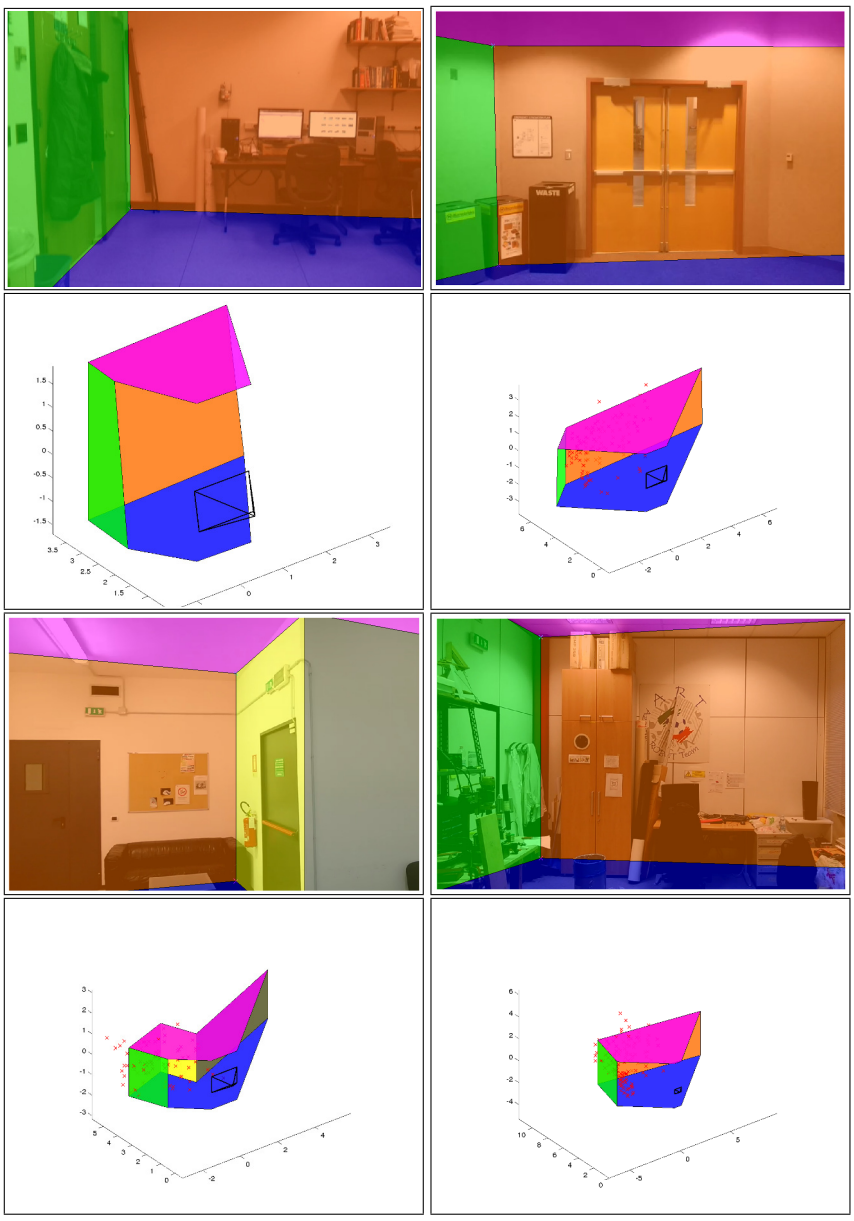


Figure 5.5: Some examples of our results on the proposed dataset. The first and the third rows show the reprojection of the best layout hypothesis into the image, while the second and the fourth rows show the same layout hypothesis in the 3D space along with the camera pose.

- Reconstruction result by the proposed method fed with VSLAM (3D model).
- Reconstruction result by the proposed method fed with VSfM (projection).
- Reconstruction result by the proposed method fed with VSfM (3D model).

There are two main types of scenarios where the proposed method achieves a poorer reconstruction result:

- Scenarios in which the observer do not explore enough the scene (*e.g.* Room 1, Room 5 and Lounge 1, where the right walls are framed for a very short period)
- Scenarios in which the ceiling is very dark and featureless. For example, in the sequence Room 1, the proposed method is perfectly able to recover the non Manhattan geometry of the left side of the room (unlike [144] and [107]), but fails in estimating the correct ceiling height, resulting in a poorer classification accuracy wrt to [107].

On the positive edge, there are also two main scenarios where the proposed method significantly outperforms state-of-the-art methods:

- Scenarios in which only a small portion of the scene falls within the field of view of the camera due to the small dimensions of the scene itself (the observer is pretty close to the walls) or to long focal length (pretty common on smartphones). A good example of this scenario is represented by the sequence Room 3, where [144] and [107] totally fail in reconstructing the correct scene layout, while the proposed method achieves very good performances.
- Scenes that cannot be represented with a box layout (*e.g.* Room 4) or that violate the Manhattan world assumption (*e.g.* Room 1).

Finally, a short note on the failure of the proposed method fed with the VSfM reconstruction in the sequence Lounge 2. Repetitive patterns like chessboards or some modern art drawings can lead the VSfM approach to generate a confuse cloud of 3D points around the patterns (see the 3D reconstruction in Figure 5.8 left column). In Lounge 2, a modern art drawing is present in the second half of the sequence and, in this

Sequence/Method	Baseline	[144]	[107]	Our + VSLAM	Our + VSfM
Corridor	66.92	61.17	73.69	**81.96**	68.68
Entrance 1	74.36	88.04	89.75	89.43	**93.72**
Entrance 2	69.15	67.55	64.95	**97.24**	94.06
Lounge 1	71.21	51.76	**86.28**	81.14	83.31
Lounge 2	57.56	25.11	68.74	**88.82**	31.28
Room 1	65.32	60.23	**76.17**	70.08	63.75
Room 2	83.27	74.54	82.69	**97.03**	91.88
Room 3	71.48	64.68	55.13	93.66	**95.52**
Room 4	67.26	33.20	54.98	**84.79**	59.34
Room 5	79.83	66.57	**83.52**	78.25	77.88
Average	70.64	59.29	73.59	**86.24**	75.94

Table 5.3: Classification accuracy on the proposed dataset. The scores of all the compared methods are shown for each sequence in the dataset.

case, led to the failure of the whole reconstruction. Please note that, on the same sequence, the proposed method fed with the VSLAM reconstruction achieved good results.

5.5 Conclusions

In this work we presented a real-time oriented approach for indoor scene understanding, addressing the problem of estimating the 3D structural layout of complex and cluttered indoor scenes from monocular video sequences, where the observer can freely move in the surrounding space. The proposed probabilistic framework allows us to generate, evaluate and optimize layout hypotheses by integrating new image evidence as the observer moves. The proposed effective inference engine allows us to make less limiting assumptions than other state-of-the-art methods (e.g., Manhattan world assumption, known and fixed camera height). In the extensive experimental evaluation we demonstrate that our formulation reaches near-real-time computation time and outperforms state-of-the-art methods in both classification accuracy and computation time, while operating in significantly less constraining conditions.

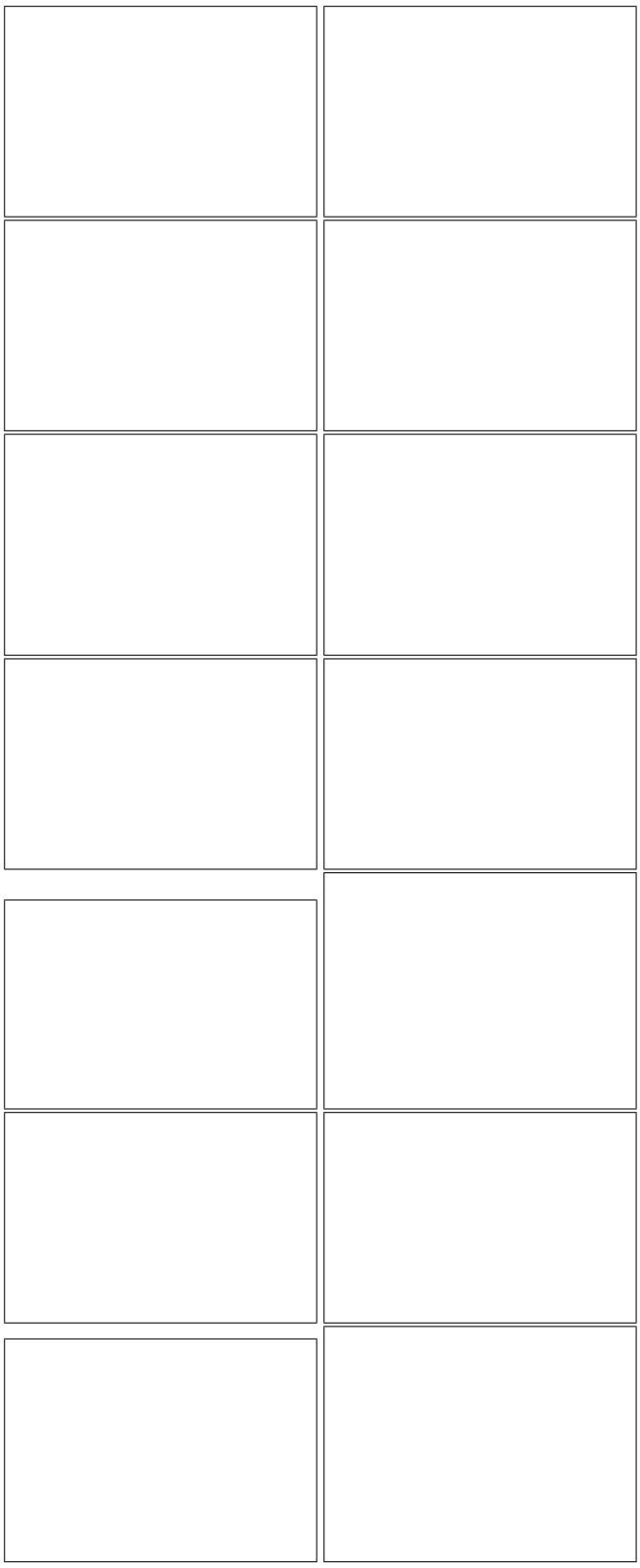


Figure 5.6: Sequence names: Corridor, Entrance 1.

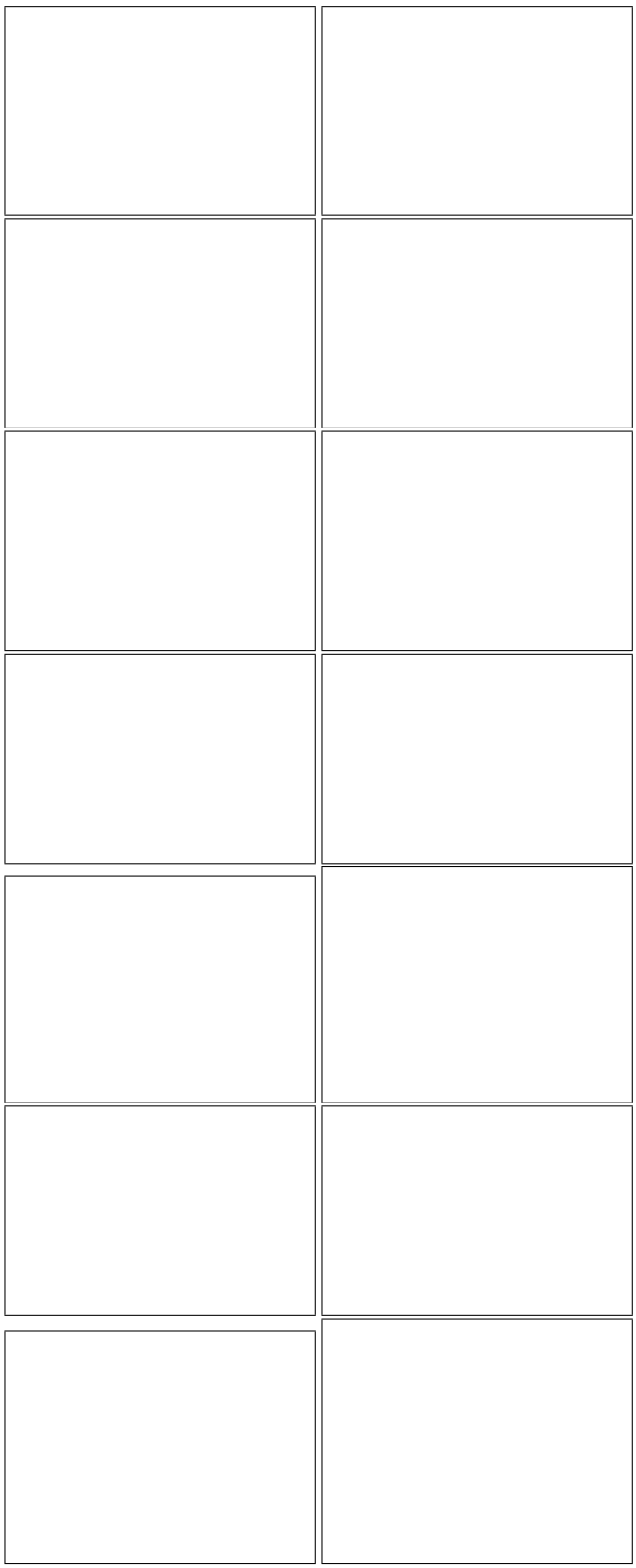


Figure 5.7: Sequence names: Entrance 2, Lounge 1.

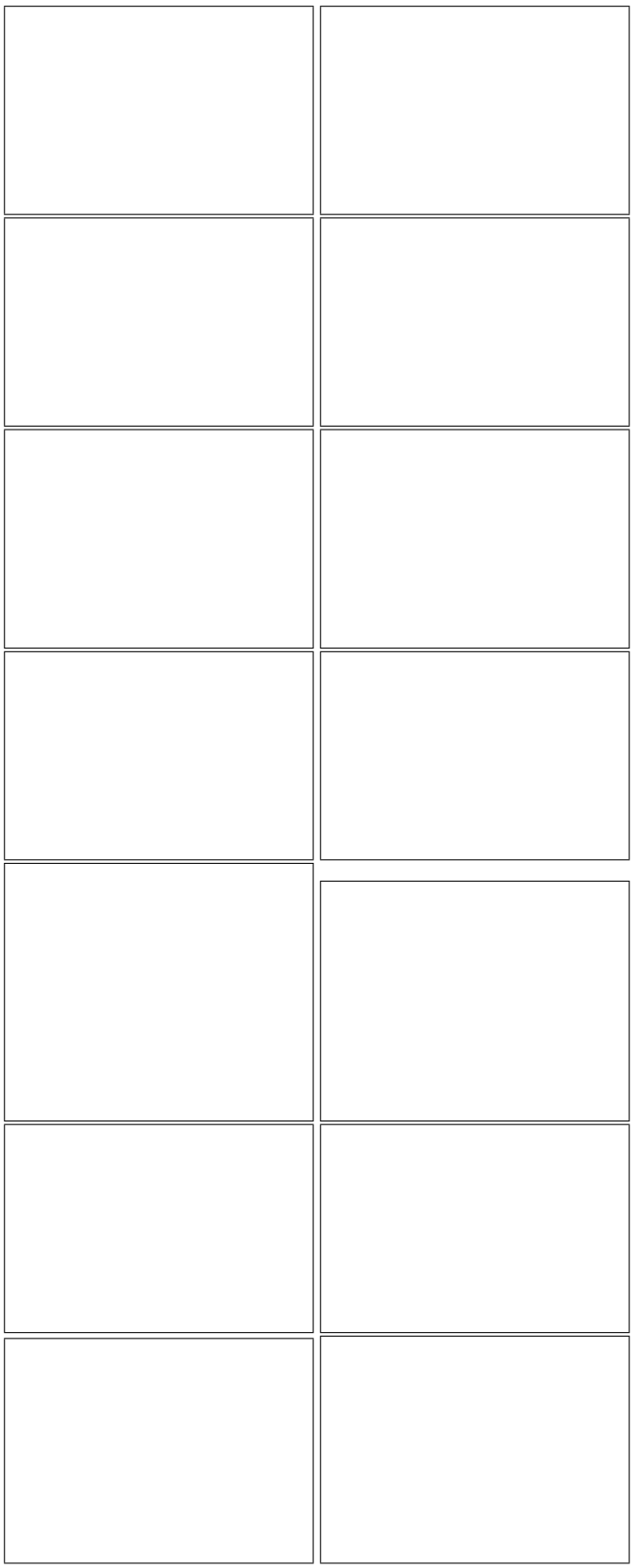


Figure 5.8: Sequence names: Lounge 2, Room 1.

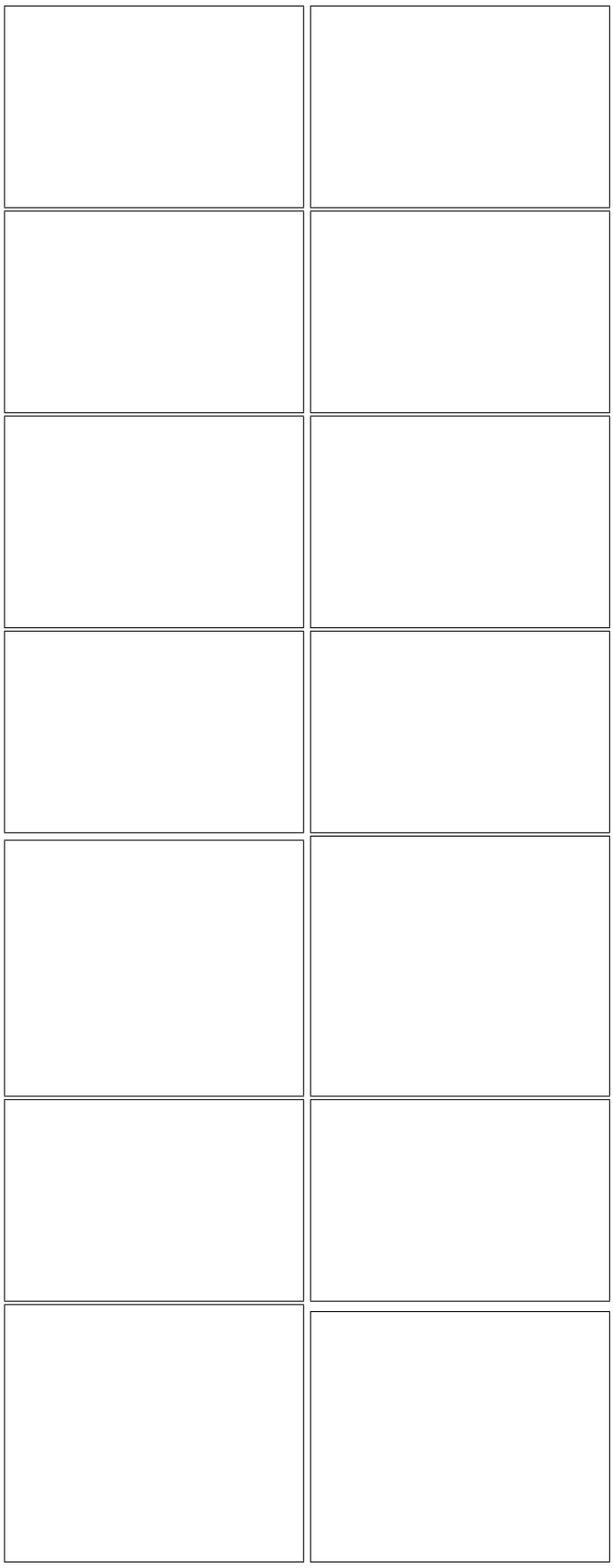


Figure 5.9: Sequence names: Room 2, Room 3.

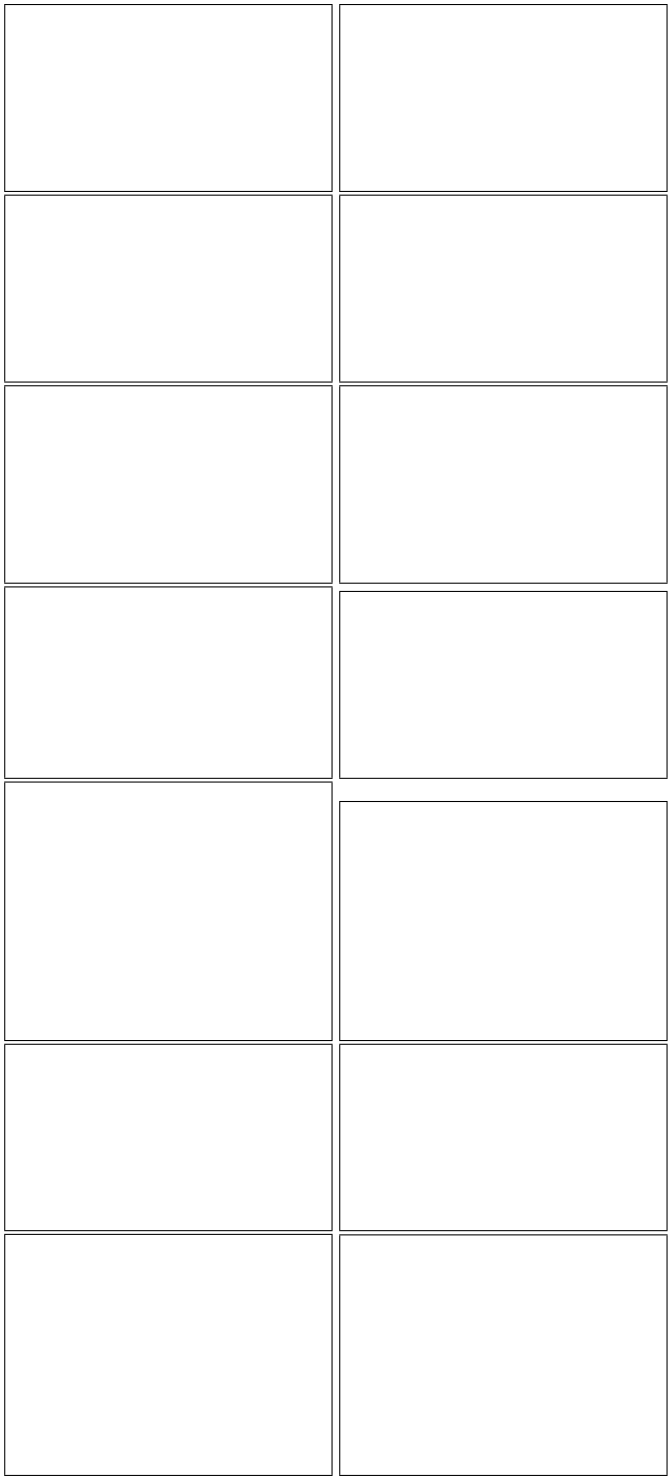


Figure 5.10: Sequence names: Room 4, Room 5.

5.6 Acknowledgments

We acknowledge the support of a NSF CAREER award N.1054127 and a gift award from HTC.
This work has been partially supported by the Italian Ministry of University and Research (MIUR) through the PRIN 2009 grant "ROAMFREE".
Stephen Miller is supported by the Hertz Foundation Google Fellowship and the Stanford Graduate Fellowship.

Chapter 6

Real world application - The USAD project

This projects aims at the development of vehicles capable of driving autonomously in a urban setting, to increase the offer of public transportation to citizens, even in very low-demand areas. These devices would allow reasonable costs for the service provider, and therefore for the customer, while providing a 24-hours on-demand service. We believe this is the only option to support a decrease in the number of private-owned cars. Another potential application is the handling and moving of goods in urban settings, so as to obtain a city-wide low-cost public logistic system.

6.1 Introduction

Research activities in the topics mentioned in Chapters 3 and 4 are conveyed in (or under integration with) this project for real-life application and testing. For the self-localization aspect, the biggest problem when navigating in outdoor, especially urban, environments, is the unreliability of the GPS localization information. GPS might be accurate enough in its differential RTK variant, but it does need a given number of satellites in the field of view of the vehicle antenna and, at the same time, the same set of satellites have to be in the filed of view of the antenna of a fixed base station; this configuration is hard to set up, not only because of the temporary lack of enough satellites in view, but also because dense clouds, vegetation, and buildings, contribute to reduce the visibility of

some satellites, therefore degrading the accuracy and/or fully disrupting the service. There are several possible research solutions to this problem. In the USAD project we are currently testing the two techniques presented in Chapter 3. For the first one, the Extended Monte-Carlo Localization, the laser-based approach requires to build very large maps of the city areas that will be traversed by the vehicle, a very computationally intensive activity that can be executed off-line. These maps will be used at runtime to localize the vehicle. The second solution, the Visual Odometry, has the potentiality of easily estimating the camera pose with respect to a visual 3D map. This can be performed with respect to a global map, acquired off-line or, as described in Chapter 3 by keeping a separate process that performs robust mapping within a temporal window, i.e. mapping only a small surrounding space by adding new map point and deleting old ones as the vehicle moves. Among the positive aspects of this latter approach, it is important to mention the much lower costs of camera devices with respect to the laser scanners, the dropping of the necessities of building a global 3D map of the city environment and of scanning the surrounding space by means of tens of thousands laser beams per second. On the negative side, illumination conditions and the indirectness of the 3D information could degrade the quality of the reconstruction and pose estimate.

Although obstacle detection for collision avoidance can be also performed by exploiting laser scanner information, the problem of detecting and tracking objects in the scene with cameras is tackled as described in Chapter 4. While the proof of concept and the experimental activity presented in Chapter 4 were performed with a Matlab implementation, a C++ real-time implementation is under development.

Finally, the work presented in Chapter 5 is suitable for indoor navigation and should be tested in such conditions, therefor it is not directly applicable to outdoor environments due to the different nature of the geometric structures, which cannot be limited to planar patches. Nevertheless, an accurate estimate of the surrounding space in outdoor navigation, in terms of higher level descriptions of scene components, is fundamental to overcome the localization problems mentioned above, as well as for effective navigation planning and security enforcement. We are currently working toward adapting this approach to urban outdoor settings. This implies recovering layouts that do not include just geometric structure, like planes of the road, buildings facades, etc., but also other, more complex structures like traffic lights, trees, parked cars, etc., which can be

Figure 6.1: The Alpaca golf cart.

either modeled with geometric primitives (cylinders, ellipsoids, boxes), or with 3D models of the proper object classes.
The vehicle we are currently using for this activity is an ordinary golf cart, which we transformed so as to be controlled directly by a computer (i.e., it has been robotized). The whole robotization process has been carried on by the IRALab research team, and in particular by the author of this thesis (Sections 6.3, 6.4 and 6.5).
The effectiveness of this real-life application has been proved with several live demos, as will be shown in Section 6.6.

6.2 Overview

The cart is an Alpaca golf cart by Ecology Runner (Figure 6.1) equipped with a Curtis 1244 MultiMode^{TM}engine control unit (ECU). Its 4KW, 48V direct current engine provides enough power to carry up to 4 people over leveled and light uphill grounds. The cart is not provided with any odometry sensor out-of-the-shelf and the ECU does not offer a digital interface for speed and brake control. Thus, the robotization process required a huge amount of work to generate odometry information, interface with the ECU simulating the analogical throttle and operate the steering wheel. On top of this, all the exteroceptive sensors (cameras, laser scanners), the computational units, the emergency system and the

Figure 6.2: Sensors on the golf cart.

power converters had to be designed and realized. In particular, for the sensor part, three single plane and one four planes laser scanner, two High Dynamic Range (HDR) and a normal black&white camera were installed. Figure 6.2 shows where the sensors are located on the golf cart.

6.3 The autonomous driving system architecture

As discussed above and in the previous chapters, the top level task of autonomous navigation is composed by a certain number of important subtasks. Figure 6.3 shows the global structure of the software system on the cart. Globally, the information flow can be summarized as follows: raw data are generated by sensors (lasers and cameras) and sent (green arrows) to the modules that implement robotic perception algorithms such as self-localization, obstacle detection etc. At the same time, map information is sent (blue arrows) to the planning modules, in particular the global planner, which is in charge of generating the sequence of actions the cart will have to perform in order to fulfill the assigned task. The processed information generated by the perception modules and the action plans are sent (orange arrows) to the local planner module, which is in charge of performing two main assignments:

- **Short term planning**. This is a very delicate task. Indeed, it tackles the problem of following the global path generated by the global

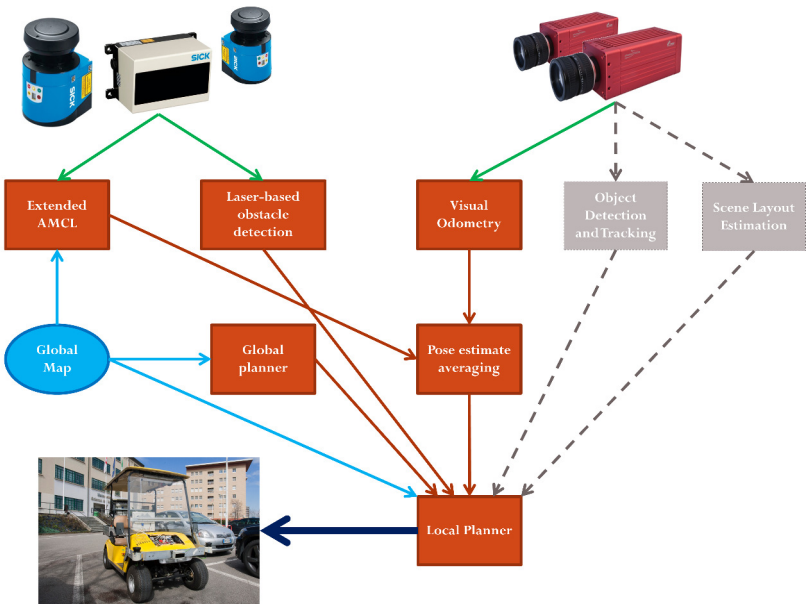


Figure 6.3: The global structure of the software system. Green arrows indicate the flow of raw data. Blue arrows indicate which modules use the global map information. Orange arrows show the flow of processed information. The thick dark blue arrow represents the velocity and steering angle set-points sent to the electronics for actuation.

planner, while being aware of the dynamic obstacles in the surrounding space, thus temporarily modifying the local trajectory to avoid them.

- **Generation of the vehicle desired dynamics.** The result of the short term planning must be converted in velocity and steering angle set-points, which must be then sent (thick dark blue arrow) to the electronic components for actuation.

6.3.1 Perception modules

Extended Monte-Carlo Localization. The work presented in Section 3.1 is here employed for global localization within a known 3D map. Earlier versions of this module implemented the standard 2D-3DoF localization approach AMCL, which proved to perform reasonably in many conditions, as far as the ground plane was sufficiently flat. As the live demo scenarios moved to less constrained outdoor environments, the need for a full 3D-6DoF motion model and localization system arose. In particular, hollows in the ground, ramps and uphill paths produce laser readings that cannot be matched with the 2D global map, thus misleading the localiza-

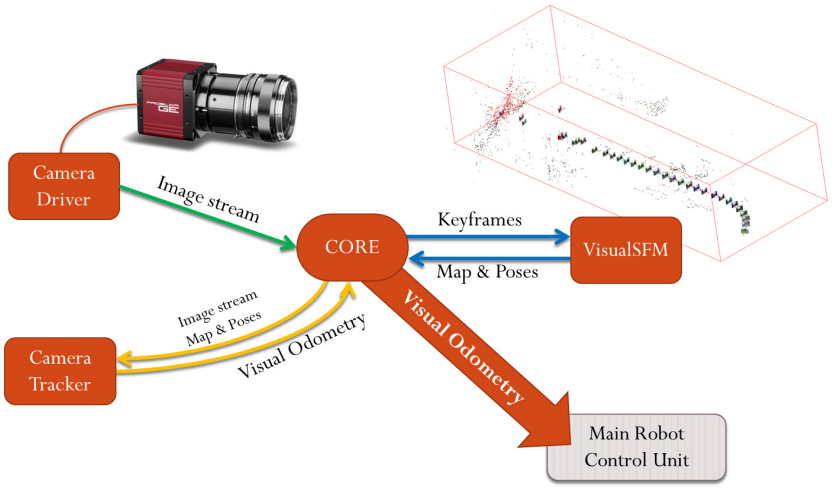


Figure 6.4: The visual odometry module. It exploits the state-of-the-art VisualSfM implementation in conjunction with the Lie algebra camera localization approach presented in Section 3.2.

tion process. Furthermore, these same readings produced a high number of false positive obstacle detections, leading the local planner to slow down or even stop the vehicle. The drawback of using a full 3D-6DoF global localization algorithm is the need of generating a 3D map of the environment; yet, with modern laser scanners and reconstruction techniques, such maps can be easily generated, at least for small to medium sized worlds.

Visual Odometry. The work presented in Section 3.2 is implemented here to compute an estimated relative camera motion, to be merged with the output of the global localization module and with the pure wheel odometry propagation. In presence of a global 3D visual map, this module is employed as is, while when such a map is not available, it is used in conjunction with the state-of-the-art VisualSfM implementation in [123]. Figure 6.4 shows how these two components interact to generate the final visual odometry camera motion estimate. The image stream is split up into sequence segments separated by key-frames ([30]). The key-frames from within a sliding temporal window are sent to VisualSfM for the off-line computation of a local map, which is then sent to the Lie camera tracker that will minimize the reprojection error in order to estimate the camera poses in the next sequence segment, until a new output of VisualSfM is available.

Object Detection and Tracking. This module is not yet available in a

real-time implementation, although the proof of concept of the work presented in Chapter 4 and the related experimental activity were performed with a Matlab implementation. This module should support the obstacle avoidance activity, improving the navigability skills of the autonomous vehicle.

6.4 Mechanics

The most important mechanical component of the cart robotization is the steering system. The Alpaca golf cart came with its normal steering wheel, to be operated by humans. The current requirements, as they have been interpreted in our research team, imply that, for safe autonomous driving, a human must be able, at any moment, to take over the complete control of the vehicle. Thus the steering robotized system was designed and developed so as to engage on the main steering column, preserving the steering wheel in its original position and function. The correct gear demultiplication factor, altogether with the steering engine power had to be chosen in order to allow the steering system to turn the wheels up to the extrema of the motion range, even if the cart is not moving. Figures 6.5 through 6.10 show the whole design and building process. The steering wheel will plug on top of the steering column. Please note that, while in emergency conditions humans can perfectly operate the steering system, the spring fastener allows to easily engage and disconnect the steering engine from the steering system, preventing a human driver to put additional force to win the engine inertia when operating the steering wheel.

6.5 Electronics

The process of equipping the cart with the electronic components needed for robotization required a huge amount of time. In fact, the electronic systems on-board control all the cart components, acting as an intermediary between the high level autonomous driving algorithms on the main computer on the one hand, while being in charge of springing all the safety mechanisms in emergency conditions on the other hand. The main electronic components on the cart include the steering, throttle and brake control, encoder reading and emergency disengaging of the autonomous control of the cart. Figure 6.11 shows a simplified schema of

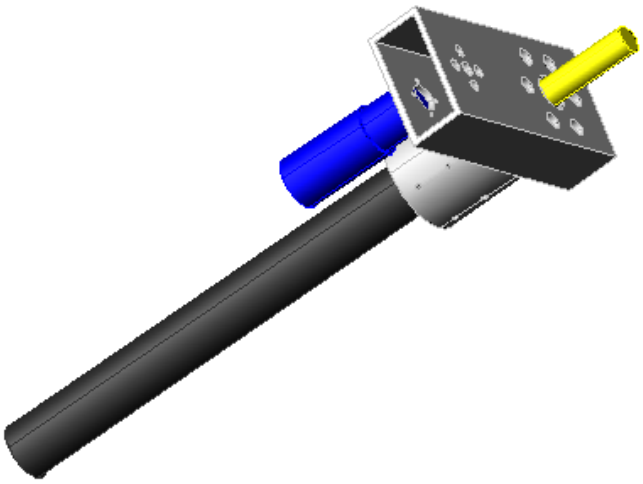


Figure 6.5: The 3D model of the steering system.

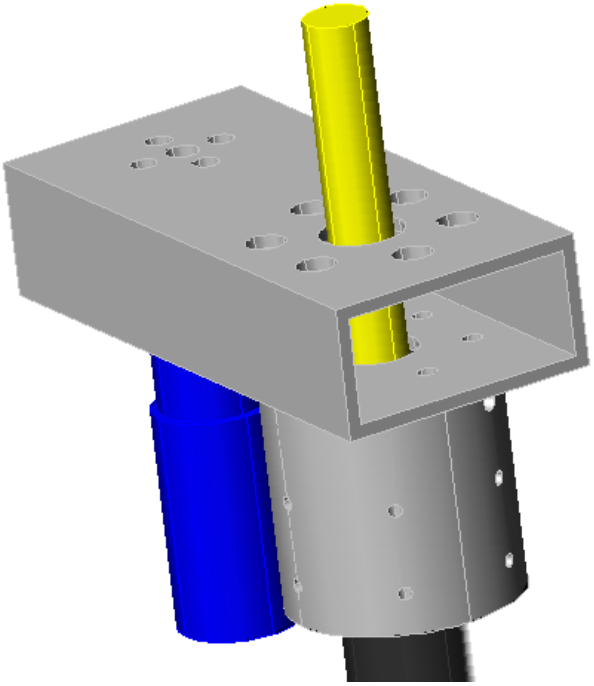


Figure 6.6: A detail of the 3D model of the steering system.

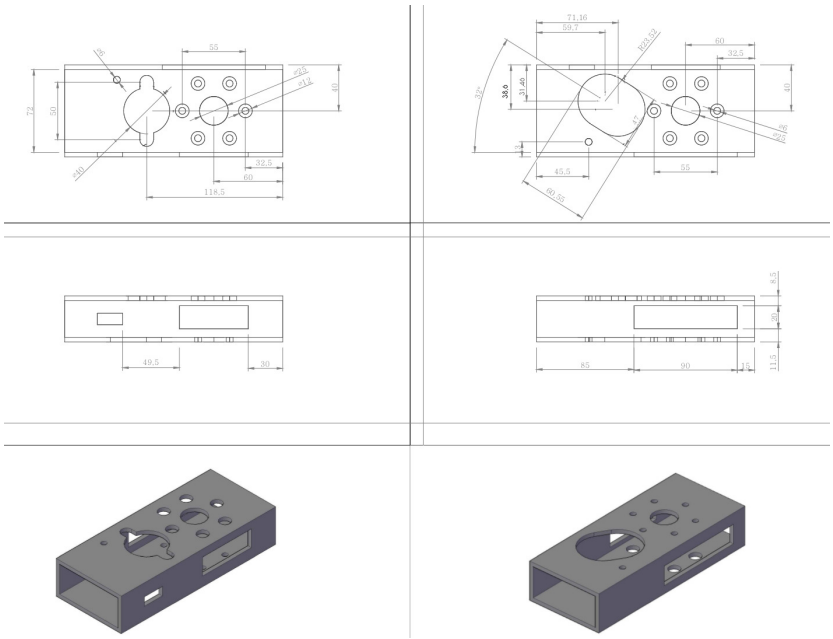


Figure 6.7: The design process of the steering system components.

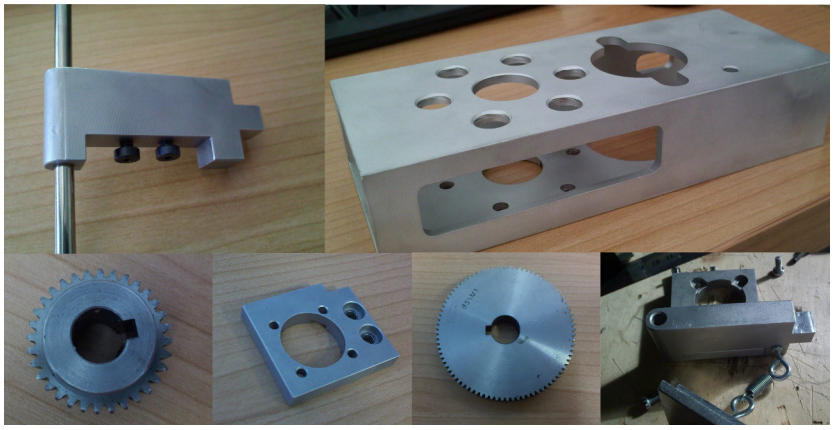


Figure 6.8: The assembling process.

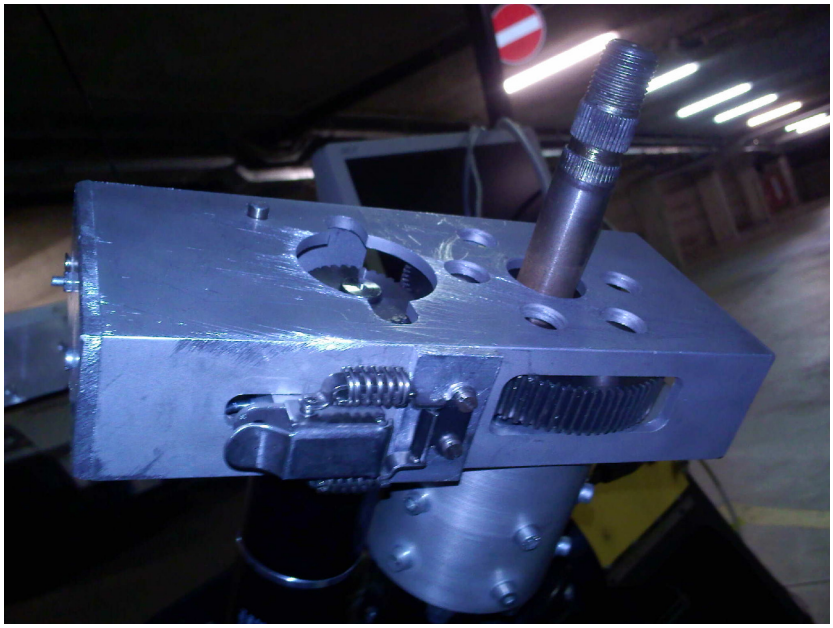


Figure 6.9: The final, fully operational robotized steering system. Steering wheel missing.

Figure 6.10: The final, fully operational robotized steering system. The steering wheel is plugged on top of the steering column, allowing humans to operate the steering system.

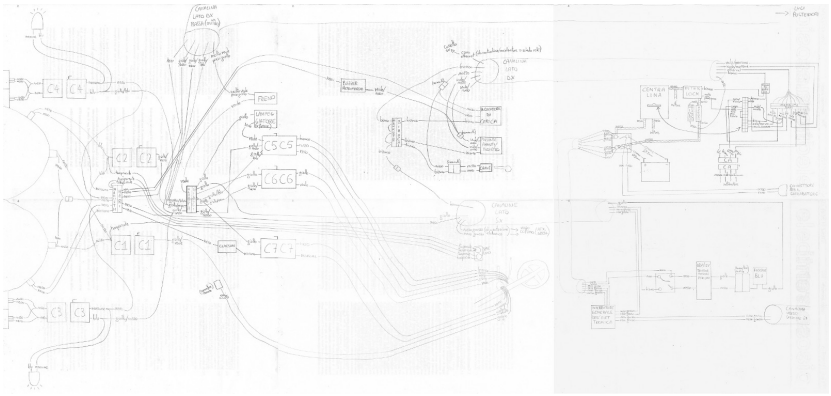


Figure 6.11: The global electrical schema of the cart's wirings.

the electronic components.

6.5.1 The main control board

This board (Figures 6.12 and 6.13) is in charge of actuating the steering and throttle commands from the main computer and to provide information about the steering position. In order to control the steering wheels, the steering commands received via USB are converted in angle setpoints and fed to a PID controller, together with the encoder readings from the steering engine, to close the control loop. The output of the PID is the duty-cycle of the pulse width modulated (PWM) signal that the control board sends to the power H-bridge (Figure 6.14). The H-bridge, which is opto-insulated from the rest of the circuit to avoid back-propagation of ripples, converts the PWM logical signal into a power output for the steering engine by means of a VNH3SP30 automotive fully integrated h-bridge motor driver, drawing the 24V power supply from the 48V-24V power converter.

Interfacing with the main cart engine control unit (ECU) required a slightly more complicate architecture, since the Curtis 1244 MultiMode^{TM} does not provide a digital interface to directly control the cart 4KW, 48V engine. The final solution was designed so as to emulate the throttle pedal, which presents a 4-wire interface, 2 wires for the security interlock switch and 2 for the analog value of a potentiometer whose end-of-range value is 5KOhm. This emulation is performed on a separate board (Figure 6.15) by means of a digital potentiometer MCP41010, which is controlled via I^2C from the main control board.

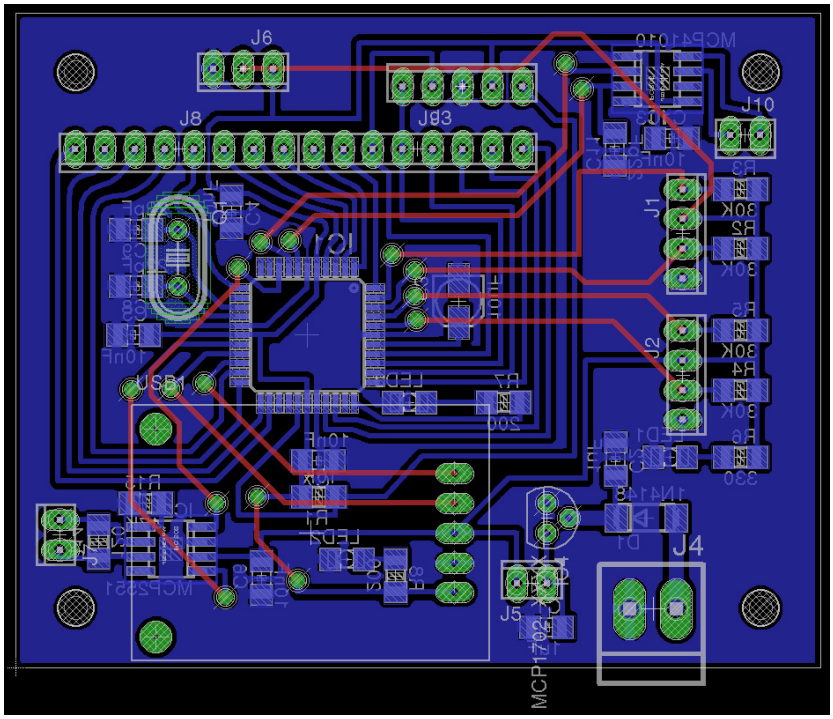


Figure 6.12: The main control board.

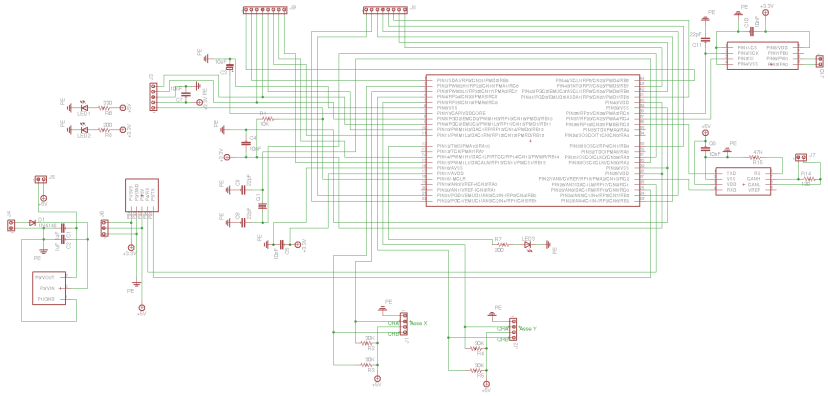


Figure 6.13: The main control board.

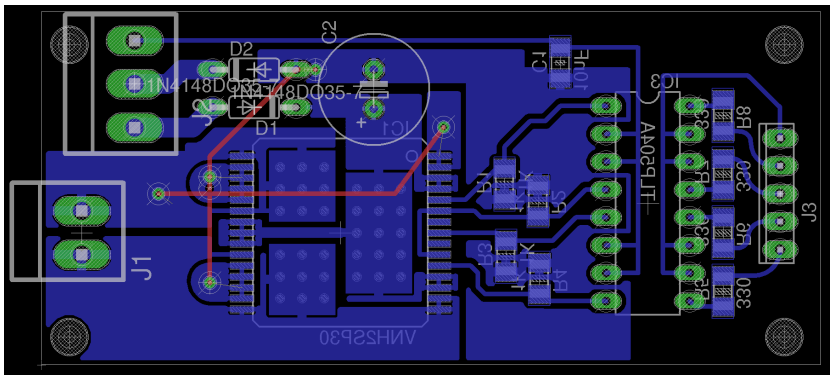


Figure 6.14: The steering H-bridge board. The logical PWM signal is converted by means of a VNH3SP30 automotive fully integrated h-bridge motor driver.

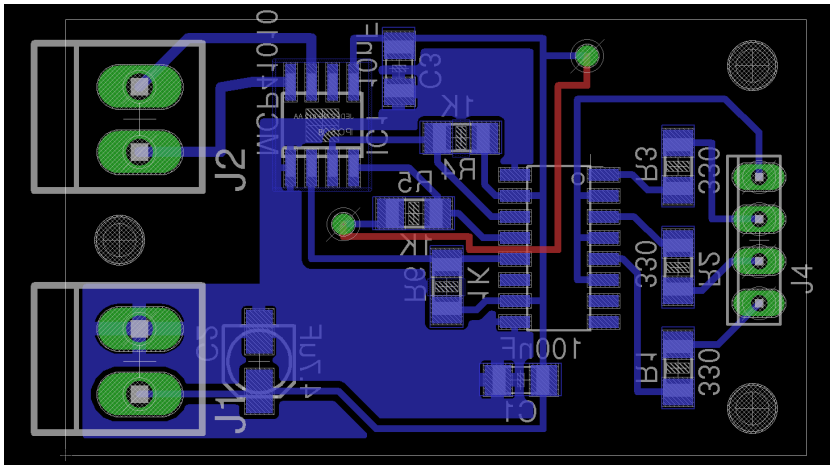


Figure 6.15: The potentiometer board.

6.5.2 Encoders and encoder board

The absence of an integrated odometry system in the original cart was the source of many problems. The developed solution is based on an electro-mechanical system, composed by mechanical parts engaged inside the drums of the rear wheels, just outside the brake shoes, and an electronic part made of two photo-diodes per wheel, plus a reference position strip of printed paper, also engaged (i.e., glued) into the drums. A mechanical part of the drum is substantially a large cylinder that is fixed to the wheel axis, and carries on its internal side a chess-printed paper strip. The photo-diodes are positioned on a metallic plate, suitably machined for fitting inside the drums so as to read the alternating of black and white on the chess strip, out of phase by $\frac{\pi}{2}$. The noisy signal from the photo-diodes is sent to a filter board that converts them into a logical square wave, which is, in turn, sent to the encoder board (Figure 6.16) that performs the quadrature and counts the ticks, thus allowing the measuring of the angular velocity of each wheel. The main problems that arose while developing the encoder odometry system were related to the issue of keeping the iron dust generated by the drum brakes outside the photo-diodes area and to the risk of misalignment of the photo-diodes due to the vibrations; both have been resolved, the first by means of a carefully designed toroidal machined part, on which the strip was glued, which had a specific protrusion aiming at keeping dust away from the strip and the sensors; the second by means of fixtures for the photo-diode sensors mounted on the sensor plates.

6.5.3 Emergency board

Last, but perhaps among the most important of all, the emergency board is in charge of disengaging the autonomous control of the vehicle and to deliver it to the human driver. It is a pretty sophisticated game of relays that allows to safely remove any automated control of the vehicle even in case of power loss and/or firmware/software failure. Figures 6.17 and 6.18 show the prototype design and its realization. Among the different features, this board is able to switch between automated and manual control of the steering system, the throttle and the forward/backward gear selection; the switch action can be triggered by means of both a non-critical operation button and a series of emergency stop buttons displaced in strategical positions around the cart.

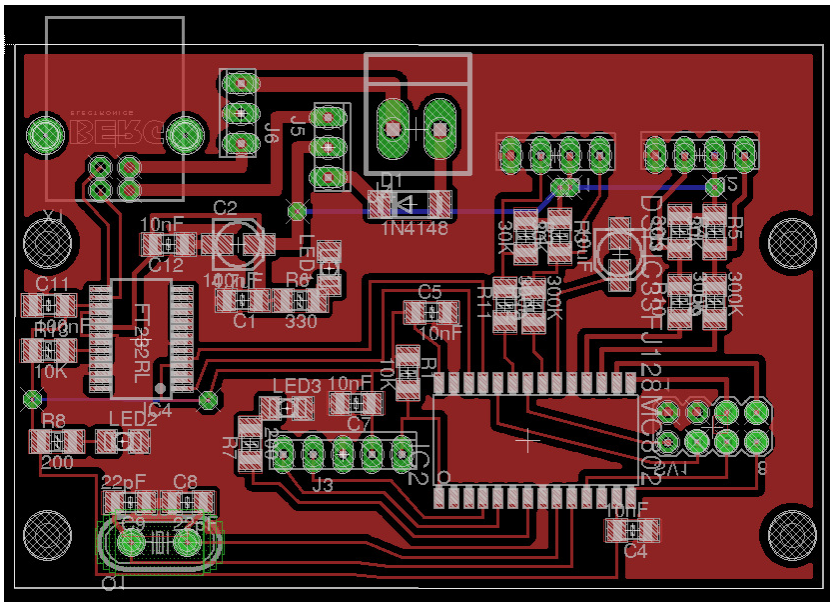


Figure 6.16: The encoder board.

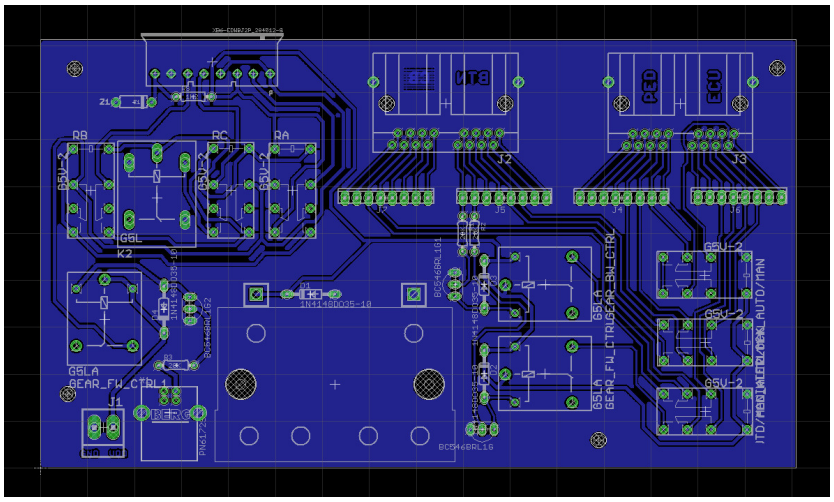


Figure 6.17: The relay board.

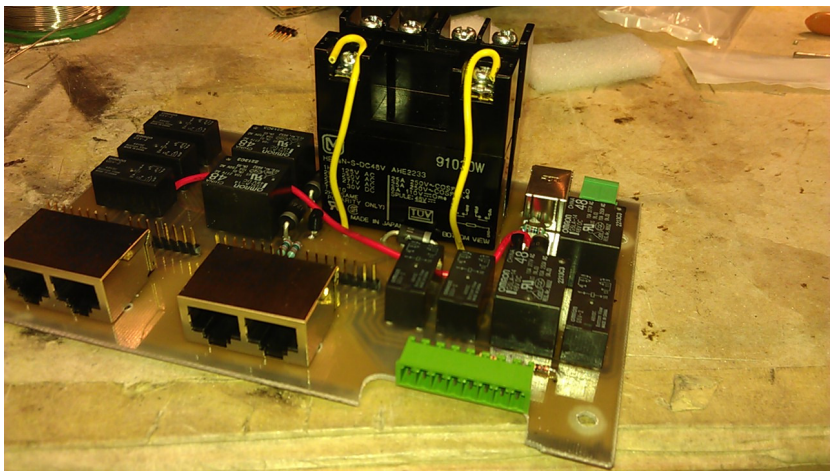


Figure 6.18: The relay board.

6.6 Live demos

- **November 2013 - Autonomous car driving live demo at the National Museum of Technology "Leonardo Da Vinci" in Milano.** Upcoming event. This demo is the first live public event in which the Visual Odometry system described in Section 3.2 will be employed. Our autonomous car will be driving people around the museum campus (outdoor and indoor) for 4 days in an heavily crowded environment.

- **May 2013 - Autonomous car driving live demo at Wired Next Fest.** For this demo we had our autonomous car driving people around for 3 days in an mild off-road (dirt/gravel road), heavily crowded environment at the city park of Porta Venezia in Milano. The major problems that needed to be dealt with were related to the self-localization due to the repetitiveness of the map patterns generated by the tree-lined paths within the park and the huge amount of people, kids and dogs present in the environment running here and there about the vehicle. Figures 6.19 through 6.21 show different moments during this demo.

- **March 2013 - Autonomous car driving demo, interview and photo session for "Quattro Ruote" (automobile magazine).** In this demo we showed the ability of our autonomous car to drive inside an indoor/outdoor parking lot. The main difficulties here were given by the restricted navigable space and by the risk of

Figure 6.19: The demo at the Porta Venezia city park in Milano.

Figure 6.20: The demo at the Porta Venezia city park in Milano.

Figure 6.21: The demo at the Porta Venezia city park in Milano.

sudden backward motion of the other parked cars. Figures 6.22 through 6.24 show different moments during this demo.

- **January 2011 - Outdoor autonomous car driving demo with RAI 3 (national television).** In this demo we recorded for RAI 3 some sequences of autonomous driving within the University of Milano - Bicocca campus. The main difficulties in this demo were related to the small amount of map elements available for self-localization, the restricted space available for maneuvering and the presence of diffused hollows in the terrain. Figures 6.25 through 6.27 show different moments during this demo.

- **November 2010 - Indoor autonomous car driving demo at the Electrical Intelligent Vehicles Fair 2010 (EIV2010).** In this demo we had our autonomous car driving people between stands of the fair for 4 days, moving through small spaces and in a highly crowded environment. Figures 6.28 through 6.30 show different moments during this demo.

6.7 Conclusions

This project represents the real life testing bed for the research topics presented in this thesis. The effectiveness of the software implementation of the presented algorithms, as well as the quality of the mechanical and electronic design and development were proven in several live

Figure 6.22: The demo for the automobile magazine “Quattro Ruote”.

Figure 6.23: The demo for the automobile magazine “Quattro Ruote”.

Figure 6.24: The demo for the automobile magazine “Quattro Ruote”.

Figure 6.25: The demo for RAI 3 (national television).

Figure 6.26: The demo for RAI 3 (national television).

Figure 6.27: The demo for RAI 3 (national television).

Figure 6.28: The demo at the EIV Fair 2010.

Figure 6.29: The demo at the EIV Fair 2010.

Figure 6.30: The demo at the EIV Fair 2010.

demos that stressed the cart resistance and the robustness of the autonomous navigation system. Among the difficulties encountered during the live demos, the most challenging were related to the crowded environments, hard localization conditions, tiny spaces for maneuvering and uneven terrains. All of these challenging conditions were met and all the demos were successfully performed.

6.8 Acknowledgments

Special thanks to Prof. Domenico Sorrenti, PhD Daniele Marzorati, Augusto L. Ballardini, Andrea Galbiati, Francesco Sacchi, Simone Fontana, Francesco Visin and all the students that, even if just for a brief time, participated in the research activity.

Figure 6.31: The IRA team.

Figure 6.32: The IRA team.

Appendix A

Lie algebra

The approach presented in Section 3.2 exploits the local properties of the SE(3) group and its associated $\mathfrak{se}(3)$ algebra. The purpose of this appendix is to give the reader an introduction to Lie groups and an explanation of the properties exploited in the approach mentioned above. This content is inspired by [146] and [147], as well as by several internal technical reports published by different research groups.

A.1 Groups

A group $(G, I, \oplus)$ is an *algebraic structure* represented by a set G, the neutral element I of G, and an associated operation $\oplus : G \times G \rightarrow G$, defined over the group elements, that combines two of them to generate a third one. A simple example of group is the set of integer numbers together with the sum operation. The operation $\oplus$ is called the *group law* and must satisfy, together with the underlying set G, the four group axioms:

- **Closure -** $\forall a, b \in G \quad a \oplus b \in G$
- **Associativity -** $\forall a, b, c \in G \quad a \oplus (b \oplus c) = (a \oplus b) \oplus c$
- **Neutral element -** $\forall a \in G \quad a \oplus I = I \oplus a = a$
- **Unique inverse element -** $\forall a \in G \; \exists b \in G \quad a \oplus b = b \oplus a = I$

Commutativity is not a required property for groups, thus, for two elements $a, b \in G$, the equation

$$a \oplus b = b \oplus a$$

is in general not true. Groups that are commutative are called *abelian groups*.
If a subset $S \subset G$ forms a group $(S, I, \oplus)$, then $(S, I, \oplus)$ is called a subgroup of $(G, I, \oplus)$.

A.1.1 Group homomorphisms

A homomorphism is a structure-preserving map between two algebraic structures. In particular, a *group homomorphism* $f : G \to H$ between two groups $(G, \oplus)$ and $(H, \otimes)$ is required to satisfy the following equation:

$$\forall a, b \in G \quad f(a \oplus b) = f(a) \otimes f(b)$$

An *isomorphism* is a bijective homomorphism. In our case, two groups G and H are called isomorphic if there exist two homomorphisms $e : G \to H$ and $f : H \to G$ such that

$$\forall h \in H \quad e(f(h)) = h$$

and

$$\forall g \in G \quad f(e(g)) = g$$

A.1.2 General Linear Group GL(n)

The General Linear group GL(n) is the group of all $n \times n$ non-singular, i.e. invertible, real-valued matrices, the group law being the matrix multiplication. The neutral element is the identity $n \times n$ matrix I_n and the inverse operation is the matrix inverse. All the groups we will encounter later (*e.g.* O(3),SO(3),SE(3), ecc.) are called *matrix groups* and are subgroups of the General Linear group GL(n).

A.2 Differentiable manifolds

A *manifold* is a topological space that can be locally approximated with an Euclidean space. More precisely, each point of an n-dimensional manifold has a neighborhood that is homeomorphic to the Euclidean space of dimension n. A manifold M is parametrized in terms of a set

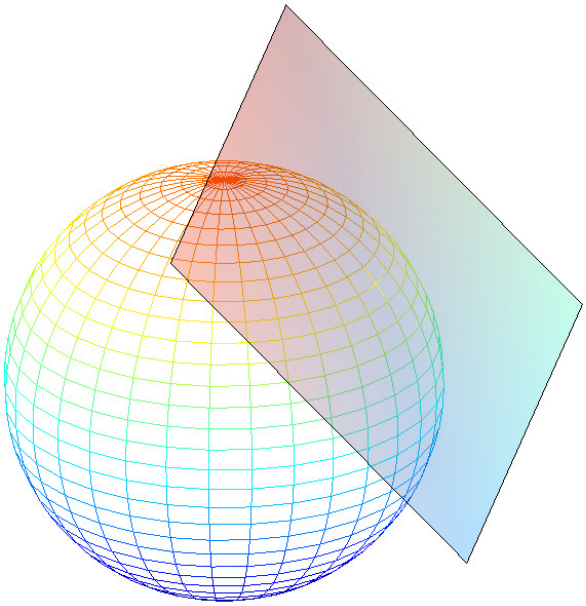


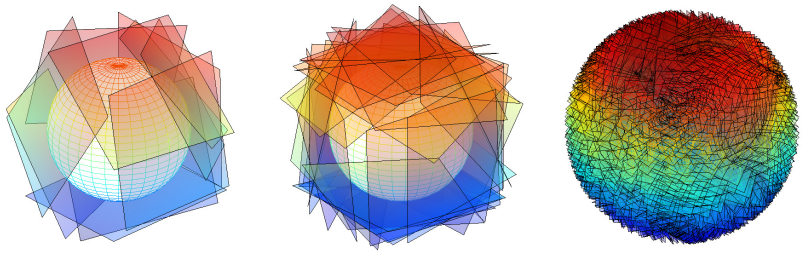
Figure A.1: Graphic representation of the tangent plane to a 2-sphere. It can be used for an intuitive explanation of the concept of tangent space.

(namely an *atlas*) of *local coordinate charts* of the form (ϕ, U), where $\phi : U \subset M \to \mathbb{R}^n$ maps points in the set U into $\mathbb{R}^n$ and is called a *smooth map*. An atlas in which, for any two local coordinate charts (ϕ, U) and (ψ, V), the function $f = \psi^{-1} \circ \phi$ is bijective and both f and its inverse f^{-1} are C^∞, is called a *smooth atlas*. A (smooth) differentiable manifold is a manifold that admit such a smooth atlas.

A.2.1 Tangent spaces

A *tangent space* of a smooth manifold M at a point p is the set of all derivations of the form $X_p : C^\infty(p) \to \mathbb{R}$, where $C^\infty(p)$ denotes the set of smooth, real-valued functions on M defined over some open neighborhood of p.

From a purely intuitive point of view, the concept of tangent space can be explained with the example of a sphere (our manifold M) and a tangent plane at a point p on the sphere. Figure A.1 gives a graphic representation of the sphere example. As shown in Figure A.2, tangent spaces can be merged together to form a two higher dimensional differentiable



(a) (b) (c)

Figure A.2: Graphic representation of the merging of tangent planes to a 2-sphere. The set of all the tangent spaces of a manifold M is called the *tangent bundle*.

manifold. The set of all the tangent spaces of a manifold M is called the *tangent bundle* of M.

The tangent space at p is made of tangent vectors, each representing a possible direction to pass through p. More formally, vectors of a tangent space can be thought of as partial (directional) derivatives of the smooth map $\phi : U \subset M \to \mathbb{R}^n$.

For example, the rotation matrix around the x axis

$$R_x(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(t) & -\sin(t) \\ 0 & \sin(t) & \cos(t) \end{bmatrix}$$

will be shown later to be a smooth map on the Lie group SO(3). Its tangent vector and, therefore, *a* tangent vector of the Lie group SO(3) is

$$\frac{\partial}{\partial t} R_x(t)|_{t=0} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\sin(0) & -\cos(0) \\ 0 & \cos(0) & -\sin(0) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

In general, the set of all tangent vectors at a point p on a manifold M forms a basis for the tangent space, allowing to write

$$X_p = X_1 \frac{\partial}{\partial x_1} + \cdots + X_n \frac{\partial}{\partial x_n}$$

with $(X_1, \ldots, X_n)$ being the local coordinate representation of X_p.

A.3 Lie groups

A Lie group is a group and at the same time a smooth manifold, for which the group operations of multiplication and inversion are smooth. In the theory of Lie groups, algebraic and calculus operations are not defined on the global object, the group, but on its local or linearized version, i.e. the associated Lie algebra.
Examples of Lie groups include:

- **The Euclidean space under addition.** It is, in particular, an *abelian* Lie group, since its elements commute under addition.

- **The general linear group** $GL(n, \mathbb{R})$**.** As mentioned above, it is the group of all the $n \times n$ nonsingular matrices ($\det \neq 0$) together with the matrix multiplication as the group law. It is a smooth manifold, since both matrix multiplication and matrix inversion are smooth in the matrix components.

- **The special orthogonal group** $SO(3)$**.** It is a subgroup of $GL(n)$ and its elements are all the $n \times n$ orthogonal matrices that represent a pure rotation. It is also a subgroup of the orthogonal group $O(3)$, which includes both pure rotation matrices ($\det = +1$) and matrices representing a rotation followed by a reflection ($\det = -1$).

- **The special euclidean group** $SE(3)$**.** It is the group of all rigid body transformations in $\mathbb{R}_3$ and is defined as the composition of a rotation R and a translation p. Formally, its elements are the mappings $g : \mathbb{R}^3 \to \mathbb{R}^3$ of the form $g(x) = Rx + p$ and can be written in matrix notation as

$$g = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$$

  where $R \in SO(3)$ and $p \in \mathbb{R}^3$. The SE(3) is a Lie group of dimension 6.

A.3.1 Lie algebra

A Lie algebra $\mathfrak{g}$ associated with a Lie group G, is an algebraic structure represented by a tangent (vector) space of G, together with the *Lie bracket*, a binary operation $[\cdot, \cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ subject to the axioms

- *Skew-symmetry*

  $\forall v, w \in \mathfrak{g} : [v, w] = -[w, v]$. This axiom implies *bilinearity* and *alternating* on $\mathfrak{g}$.

- *Jacoby identity*

  $\forall v, w, z \in \mathfrak{g} : [[v, w], z] + [[z, v], w] + [[w, z], v] = 0$

Please refer to Section 3.2.1 for examples of elements belonging to the $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$ algebras.

A.3.2 The exponential map

Now that we have defined the smooth manifolds and their associated algebras, we need a means of mapping elements between them. For matrix Lie groups this can be done using the *exponential map*, which is the generalization to all differentiable manifolds of the ordinary exponential function

$$e^x : \mathbb{R} \to \mathbb{R}^+, \quad e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

Let X be a $n \times n$ matrix, then the exponential map of X is defined as

$$\exp(X) : \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n}, \quad \exp(X) = \sum_{k=0}^{\infty} \frac{X^k}{k!}$$

where $X^0 = I$.
The exponential map for matrix Lie groups (the matrix exponential) can be proved to be invertible [147] (matrix logarithm), and it is easy to see that $\exp(0) = I$. These two properties are sufficient to state that the matrix exponential maps elements from the tangent space to its corresponding Lie group.

A.3.3 Exponential map for SO(3)

Despite the general definition of the matrix exponential is quite hard to implement, there is a closed form representation that helps in that sense. Let $\omega \in \mathbb{R}^3$, then $exp([\omega]_\times) \in SO(3)$ represents a rotation around ω by an angle $\|\omega\|$. The closed form solution is given by the *Rodriguez formula*

$$\exp([\omega]_\times) = I + \frac{[\omega]_\times}{\|\omega\|}\sin(\|\omega\|) + \frac{[\omega]_\times^2}{\|\omega\|^2}(1 - \cos(\|\omega\|)) \tag{A.1}$$

It is important to notice that when $\|\omega\| \to 0$, such formulation of $exp([\omega]_\times)$ would be indefinite. In this case we observe that

$$\lim_{\|\omega\|\to 0} \frac{\sin(\|\omega\|)}{\|\omega\|} = 1$$

and

$$\lim_{\|\omega\|\to 0} \frac{1 - \cos(\|\omega\|)}{\|\omega\|^2} = \frac{1}{2}$$

thus the Rodriguez formula simplifies to

$$\exp([\omega]_\times) = I + [\omega]_\times + \frac{1}{2}[\omega]_\times^2 = I$$

A.3.4 Exponential map for SE(3)

In a similar fashion as for $SO(3)$, there exists a closed form for the matrix exponential to map elements of $\mathfrak{se}(3)$ into roto-translation matrices of $SE(3)$. In particular, let $\mu = \binom{v}{\omega} \in \mathbb{R}^6$, the corresponding matrix form in $\mathfrak{se}(3)$ is

$$\begin{bmatrix} [\omega]_\times & v \\ 0_{1\times 3} & 0 \end{bmatrix}$$

and the closed form for the matrix exponential is

$$\exp(\mu) = \begin{bmatrix} exp([\omega]_\times) & Vv \\ 0_{1\times 3} & 0 \end{bmatrix} \tag{A.2}$$

where

$$V = I + \frac{[\omega]_\times}{\|\omega\|^2}(1 - \cos(\|\omega\|)) + \frac{[\omega]_\times^2}{\|\omega\|^3}(\|\omega\| - \sin(\|\omega\|)) \tag{A.3}$$

Again, when $\|\omega\| \to 0$, $exp(v, [\omega]_\times)$ is indefinite and, since

$$\lim_{\|\omega\|\to 0} \frac{\|\omega\| - \sin(\|\omega\|)}{\|\omega\|^3} = \frac{1}{6}$$

the Equation A.3 reduces to

$$\exp([\omega]_\times) = I + \frac{1}{2}[\omega]_\times + \frac{1}{6}[\omega]_\times^2 = I$$

A.4 Jacobians for camera pose estimation

As discussed in Section 3.2, in order to perform minimization of the reprojection error function $E(\mu)$ with respect to μ, we first need to compute its Jacobian $\frac{\partial E(\mu)}{\partial \mu}$. Let us then decompose the error function as in Section 3.2.2; in particular, we are interested in the roto-translation matrix M

$$\begin{pmatrix} u \\ v \end{pmatrix} = \mathbf{Pr}(MP^w) \tag{A.4}$$

where $Pr(x)$ is the pinhole-camera model projection. Let us start from the generic projection (not the pinhole-camera model one), i.e. the function

$$proj : \mathbb{R}^n \to \mathbb{R}^{n-1}, \quad proj(x) = \frac{1}{x_n} \begin{pmatrix} x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

Its Jacobian is given by

$$\frac{\partial proj(x)}{\partial x} = \frac{1}{x_n} \left[I_{n-1 \times n-1} \quad \begin{pmatrix} x_1 \\ \vdots \\ x_{n-1} \end{pmatrix} \right] \tag{A.5}$$

Now, the pinhole-camera model can be rewritten as

$$Pr(x) = f \cdot proj(x) + c$$

where f is the 2×2 diagonal matrix, whose diagonal (f_x, f_y) consists of the x and y components of the camera focal length, while c is the camera center. Its Jacobian is given by

$$\begin{aligned} \frac{\partial Pr(x)}{\partial x} &= \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \frac{\partial proj(x)}{\partial x} = \\ &= \frac{1}{x_3} \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \begin{bmatrix} 1 & 0 & -\frac{x_1}{x_3} \\ 0 & 1 & -\frac{x_2}{x_3} \end{bmatrix} \end{aligned} \tag{A.6}$$

Finally, expressing the roto-translation M by means of the exponential map of μ, leads us to [147, Appendix B]

$$\begin{aligned} \left.\frac{\partial proj(M\cdot x)}{\partial \mu}\right|_{\mu=0} &= \left.\frac{\partial proj(\exp(\mu)\cdot x)}{\partial \mu}\right|_{\mu=0} = \\ &= [I_{3\times 3} \ -[proj(M\cdot x)]_\times] \end{aligned} \tag{A.7}$$

where $proj(M\cdot x) = Rx + t$, R and t being, respectively, the rotational and translational components of M.
Putting together results obtained in Equations A.6 and A.7, the complete 2×6 Jacobian for the camera model, including the camera roto-translation with respect to the map points, is

$$\begin{aligned} \left.\frac{\partial E(\mu)}{\partial \mu}\right|_{\mu=0} &= \left.\frac{\partial (z_i - \hat{z}_j)}{\partial \mu}\right|_{\mu=0} = \left.\frac{\partial (z_i - Pr(proj(\exp(\mu)\cdot P^w)))}{\partial \mu}\right|_{\mu=0} = \\ &= -\frac{1}{x_3}\begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix}\begin{bmatrix} 1 & 0 & -\frac{x_1}{x_3} \\ 0 & 1 & -\frac{x_2}{x_3} \end{bmatrix}[I_{3\times 3} \ -[x]_\times] \end{aligned} \tag{A.8}$$

where $x = proj(\exp(\mu)\cdot P^w$, i.e. $x = RP^w + t$.

Appendix B

Pure reactive path follower

This appendix briefly describes the approach that was used, from 2010 to the first half of 2013, as the lateral control strategy in the real-life application presented in Chapter 6.

B.1 The lateral control problem

As discussed in the previous chapters, autonomous robots need a means of executing planned actions. Given a trajectory, the robot must be able to stick to it as close as possible, at the maximum possible speed. It is easy to see that an excessive oscillation around the trajectory will lead to uncomfortable motion and, most importantly, to the need of reducing the speed in order to keep the oscillation under control. This fact is not as impacting for indoor, slowly moving robots, but it becomes crucial when the mobile robot is an autonomous car, driving at considerable speed. There are few works on this topic, the most relevant being the one by Sotelo [148]. In this work, the author tackles the problem of lateral control strategy for vehicles implementing the Ackerman kinematics. After implementing this approach, it showed its sensibility to the calibration of the system parameters. Since finding the optimal set of parameters revealed to be a very hard task, a different approach was developed.

B.2 The proposed approach

Once the vehicle pose is expressed in terms of its translation and rotation error with respect to the desired trajectory, the expected output of a lateral control strategy is a set-point for the steering wheel angular position and a set-point for the vehicle tangential speed.
The proposed method consists in two pure reactive functions that map the translation-rotation error in the two desired set-points. It is designed to be easily tuned to accommodate for the vehicle physical capabilities and for the desired smoothness/abruptness of the resulting movement.

B.2.1 Computing the errors

Let assume we are given the odometry readings from the rear wheels of the vehicle, i.e. the amount of space traveled by each wheel in a time lapse Δt. Let us call them ΔL for the left wheel and ΔR for the right one. Let b be the baseline between the rear wheels. Then, the transformation that expresses the new vehicle pose $(X, Y, \theta)^{\texttt{New}}$ with respect to the old one $(X, Y, \theta)^{\texttt{Old}}$ is

$$\begin{pmatrix} \mathrm{Cos}\left[\frac{\Delta\mathrm{L}-\Delta\mathrm{R}}{b}\right] & \mathrm{Sin}\left[\frac{\Delta\mathrm{L}-\Delta\mathrm{R}}{b}\right] & \frac{b(\Delta\mathrm{L}+\Delta\mathrm{R})\mathrm{Sin}\left[\frac{\Delta\mathrm{L}-\Delta\mathrm{R}}{b}\right]}{2(\Delta\mathrm{L}-\Delta\mathrm{R})} \\ -\mathrm{Sin}\left[\frac{\Delta\mathrm{L}-\Delta\mathrm{R}}{b}\right] & \mathrm{Cos}\left[\frac{\Delta\mathrm{L}-\Delta\mathrm{R}}{b}\right] & -\frac{b(\Delta\mathrm{L}+\Delta\mathrm{R})\mathrm{Sin}\left[\frac{\Delta\mathrm{L}-\Delta\mathrm{R}}{2b}\right]^2}{\Delta\mathrm{L}-\Delta\mathrm{R}} \\ 0 & 0 & 1 \end{pmatrix} \tag{B.1}$$

and the transformation that expresses the new vehicle pose $(X, Y, \theta)^{\texttt{New}}$ with respect to the world is

$$\begin{pmatrix} \mathrm{Cos}(\gamma) & \mathrm{Sin}(\gamma) & \frac{2X^{\texttt{Old}}(\Delta\mathrm{L}-\Delta\mathrm{R})+b(\Delta\mathrm{L}+\Delta\mathrm{R})\mathrm{Sin}[\theta^{\texttt{Old}}]+b(\Delta\mathrm{L}+\Delta\mathrm{R})\mathrm{Sin}(\gamma)}{2(\Delta\mathrm{L}-\Delta\mathrm{R})} \\ -\mathrm{Sin}(\gamma) & \mathrm{Cos}(\gamma) & \frac{2Y^{\texttt{Old}}(\Delta\mathrm{L}-\Delta\mathrm{R})-b(\Delta\mathrm{L}+\Delta\mathrm{R})\mathrm{Cos}[\theta^{\texttt{Old}}]+b(\Delta\mathrm{L}+\Delta\mathrm{R})\mathrm{Cos}(\gamma)}{2(\Delta\mathrm{L}-\Delta\mathrm{R})} \\ 0 & 0 & 1 \end{pmatrix} \tag{B.2}$$

with γ being

$$\gamma = \left[\frac{\Delta\mathrm{L} - \Delta\mathrm{R} - b\theta^{\texttt{Old}}}{b}\right]$$

With the transformation in Equation B.2 it is possible to express the vehicle pose (X_c, Y_c, θ_c) with respect to the world, or, in the case a global map is not available, with respect to the same reference frame that is used to express the trajectory points. Let us now consider the point (X_p, Y_p, θ_p) on the trajectory closest the vehicle, which can be easily

computed in terms of the Euclidean distance. The transformation that expresses a world point with respect to the vehicle is

$$\begin{pmatrix} \text{Cos}\left[\theta_c\right] & \text{Sin}\left[\theta_c\right] & -\text{Cos}\left[\theta_c\right] X_c - \text{Sin}\left[\theta_c\right] Y_c \\ -\text{Sin}\left[\theta_c\right] & \text{Cos}\left[\theta_c\right] & \text{Sin}\left[\theta_c\right] X_c - \text{Cos}\left[\theta_c\right] Y_c \\ 0 & 0 & 1 \end{pmatrix} \tag{B.3}$$

and, accounting for the rotation error $\theta_e = \theta_c - \theta_p$

$$\begin{pmatrix} \text{Cos}\left[\theta_c + \theta_e\right] & \text{Sin}\left[\theta_c + \theta_e\right] & -\text{Cos}\left[\theta_c + \theta_e\right] X_c - \text{Sin}\left[\theta_c + \theta_e\right] Y_c \\ -\text{Sin}\left[\theta_c + \theta_e\right] & \text{Cos}\left[\theta_c + \theta_e\right] & \text{Sin}\left[\theta_c + \theta_e\right] X_c - \text{Cos}\left[\theta_c + \theta_e\right] Y_c \\ 0 & 0 & 1 \end{pmatrix} \tag{B.4}$$

Thus, the point (X_p, Y_p) expressed with respect to the vehicle reference frame is

$$\begin{pmatrix} -\text{Cos}\left[\theta_c + \theta_e\right] X_c + \text{Cos}\left[\theta_c + \theta_e\right] X_p + \text{Sin}\left[\theta_c + \theta_e\right] \left(-Y_c + Y_p\right) \\ \text{Sin}\left[\theta_c + \theta_e\right] X_c - \text{Sin}\left[\theta_c + \theta_e\right] X_p + \text{Cos}\left[\theta_c + \theta_e\right] \left(-Y_c + Y_p\right) \end{pmatrix} \tag{B.5}$$

And, in particular, the Y coordinate is the one that expresses the distance lateral error D_e of the vehicle with respect to the trajectory.

B.2.2 Generating the set-points

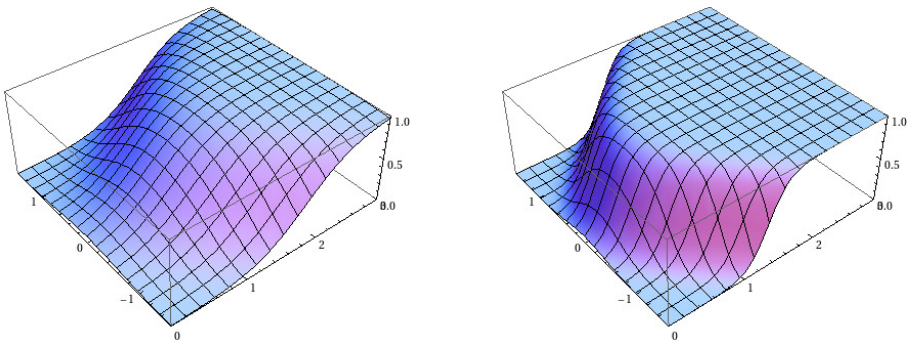
Using the errors D_e from Equation B.5 and θ_e, the bivariate function $f_{\text{Steering}} : \mathbb{R}^+ \times \mathbb{R} \to (0, 1)$

$$f_{\text{Steering}} = \frac{1}{2} + \frac{1}{2}\text{Tanh}\left[\left(-1 + \text{Cos}\left[\theta_e\right] + D_e\right) \alpha_{\text{Steering1}} - \alpha_{\text{Steering2}}\right] \tag{B.6}$$

maps the translation/rotation error into a $(0, 1)$ interval, representing the intensity of the steering action that needs to be applied to the steering wheel. Figure B.1(a) shows the plot of the steering function with parameters tuned for smooth driving actions, while in Figure B.1(b) the parameters are tuned for a more abrupt driving style.
Similarly, the bivariate function $f_{\text{Throttle}} : \mathbb{R}^+ \times \mathbb{R}^+ \to (0, 1]$

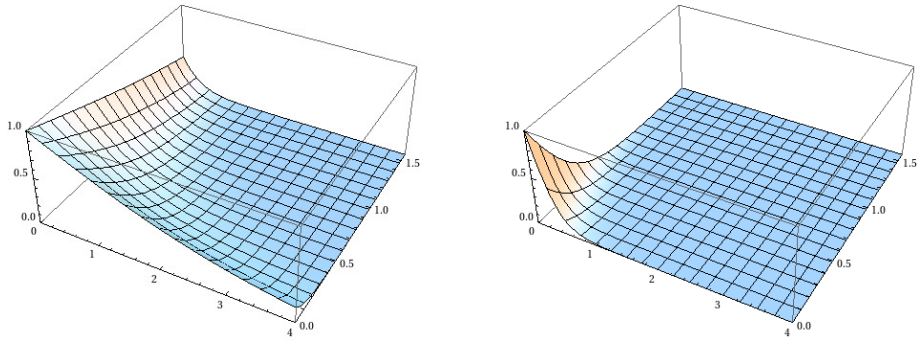
$$1 - \text{Tanh}\left[D_e \alpha_{\text{Throttle1}} + D_e |\theta_e| + \alpha_{\text{Throttle2}} |\theta_e|\right] \tag{B.7}$$

maps the translation/rotation error into a $(0, 1]$ interval, representing the intensity of the throttle that needs to be applied. Figure B.1(a) shows the plot of the throttle function with parameters tuned for smooth driving actions, while in Figure B.1(b) the parameters are tuned for a more abrupt driving style.



(a) (b)

Figure B.1: The steering function. The symmetric axis ranging from $\frac{-\pi}{2}$ to $\frac{\pi}{2}$ represents the θ_e while the axis ranging from 0 to 3 represents the D_e. Parameters are tuned for smooth (a) and abrupt (b) driving styles.



(a) (b)

Figure B.2: The throttle function. The axis ranging from 0 to π represents the θ_e while the axis ranging from 0 to 3 represents the D_e. Parameters are tuned for smooth (a) and abrupt (b) driving styles.

Bibliography

[1] S. O. Madgwick, A.J. Harrison, and A. Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *ICORR*, 2011.

[2] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000.

[3] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT Press, 2005.

[4] C.F. Olson. Probabilistic self-localization for mobile robots. *Robotics and Automation, IEEE Trans. on*, Feb 2000.

[5] Rainer Kümmerle, Rudolph Triebel, Patrick Pfaff, and Wolfram Burgard. Monte carlo localization in outdoor terrains using multilevel surface maps. In *In Proc. of the Int. Conf. on Field and Service Robotics*, 2007.

[6] T. Suzuki, M. Kitamura, Y. Amano, and T. Hashizume. 6-dof localization for a mobile robot using outdoor 3d voxel maps. In *IROS*, 2010.

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

[8] Cody C. T. Kwok, Dieter Fox, and Marina Meila. Adaptive real-time particle filters for robot localization. In *ICRA*.

[9] Kai Lingemann, Hartmut Surmann, Andreas Nüchter, and Joachim Hertzberg. Indoor and outdoor localization for fast mobile robots. In *IROS*, 2004.

[10] Ian Baldwin and Paul Newman. Road vehicle localization with 2d push-broom lidar and 3d priors. In *ICRA*, Minnesota, USA, May 2012.

[11] Simon Lacroix, Anthony Mallet, David Bonnafous, Gerard Bauzil, Sara Fleury, Matthieu Herrb, and Raja Chatila. Autonomous rover navigation on unknown terrains: Functions and integration. *Int. Jour. Robotics Research*, 21(10-11), 2002.

[12] Austin I. Eliazar and Ronald Parr. Learning probabilistic motion models for mobile robots. In *21st Int. Conf. Machine Learning*, 2004.

[13] N. Roy and S. Thrun. Online self-calibration for mobile robots. In *ICRA*, 1999.

[14] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2002.

[15] J. Montiel, J. Civera, and A. Davison. Unified inverse depth parametrization for monocular slam. In *Proceedings of Robotics: Science and Systems*, 2006.

[16] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29:2007, 2007.

[17] Ethan Eade and Tom Drummond. Monocular slam as a graph of coalesced observations. In *ICCV*, 2007.

[18] Daniele Marzorati, Matteo Matteucci, Davide Migliore, and Domenico G. Sorrenti. On the use of inverse scaling in monocular slam. In *ICRA*, 2009.

[19] Javier Civera, Andrew J. Davison, Oscar G. Grasa, and J. M. M. Montiel. 1-point ransac for ekf filtering. application to real-time structure from motion and visual. *JFR*, 27(5), 2010.

[20] J. Solá. Consistency of the monocular ekf-slam algorithm for three different landmark parametrizations. In *ICRA*, 2010.

[21] Hauke Strasdat, Andrew J. Davison, J. M. M. Montiel, and Kurt Konolige. Double window optimisation for constant time visual slam. In *ICCV*, 2011.

[22] Cyril Roussillon, Aurélien Gonzalez, Joan Solà, Jean-Marie Codol, Nicolas Man-sard, Simon Lacroix, and Michel Devy. Rt-slam: A generic and real-time visual slam implementation. *CoRR*, 2012.

[23] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. Real-time monocular slam: Why filter? In *ICRA*.

[24] Pedro Piniés and J. D. Tardós. Scalable slam building conditionally independent local maps. In *IROS*, 2007.

[25] A. Ranganathan, M. Kaess, and F. Dellaert. Loopy SAM. In *IJCAI*, 2007.

[26] Pedro Piniés and J. D. Tardós. Large scale slam building conditionally independent local maps: Application to monocular vision. *TRO*, 24(5), 2008.

[27] J. D. Tardós Lina María Paz, Pedro Piniés and J. Neira. Large scale 6dof slam with stereo-in-hand. *TRO*, 24(5), 2008.

[28] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *TRO*, 24(6), 2008.

[29] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Fast incremental smoothing and mapping with efficient data association. In *ICRA*, 2007.

[30] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *ISMAR'07*, 2007.

[31] R. O. Castle, D. J. Gawley, G. Klein, and D. W. Murray. Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In *ICRA*, 2007.

[32] Georg Klein and David Murray. Improving the agility of keyframe-based slam. In *ECCV*, 2008.

[33] E. Mouragnon, M. Lhuillier, Dho M., F. Dekeyser, and P. Sayd. Generic and real-time structure from motion using local bundle adjustment. *IVC*, 27, 2009.

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

[35] Bill Triggs, Philip Mclauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment - a modern synthesis. In *Vision Algorithms: Theory and Practice, LNCS*. Springer Verlag, 2000.

[36] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M. Seitz. Multicore bundle adjustment. In *CVPR*, 2011.

[37] Nico Cornelis, Bastian Leibe, Kurt Cornelis, and Luc Van Gool. 3d urban scene modeling integrating recognition and reconstruction. *IJCV*, 78, 2008.

[38] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3d reconstruction from video. *IJCV*, 78(2-3), 2008.

[39] Keith N. Snavely. *Scene reconstruction and visualization from internet photo collections*. PhD thesis, Seattle, WA, USA, 2009.

[40] Changchang Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). `http://cs.unc.edu/~ccwu/siftgpu`, 2007.

[41] Hans Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, 1977.

[42] Hans Moravec. Visual mapping by a robot rover. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, 1979.

[43] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, 1988.

[44] Jianbo Shi and Carlo Tomasi. Good features to track. In *CVPR*, 1994.

[45] Stephen M. Smith and J. Michael Brady. SUSAN - A new approach to low level image processing. *IJCV*, 1997.

[46] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *ICCV*, 2005.

[47] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *ECCV*, 2006.

[48] Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *ICCV*, 2001.

[49] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[50] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *PAMI*, 2005.

[51] David G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[52] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *CVIU*, 2008.

[53] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*, 2002.

[54] Yan Ke and Rahul Sukthankar. PCA-SIFT: a more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE computer society conference on Computer vision and pattern recognition*, 2004.

[55] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2002.

[56] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, 2004.

[57] Gabriela Csurka, Christopher R. Dance, Florent Perronnin, and Jutta Willamowski. Generic visual categorization using weak geometry. In *Lecture Notes in Computer Science: Toward Category-Level Object Recognition*, 2006.

[58] Jianguo Zhang, Marcin Marszałek, Svetlana Lazebnik, and Cordelia Schmid. Local features and kernels for classification of texture and object categories: a comprehensive study. *IJCV*, 2007.

[59] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Simultaneous object recognition and segmentation from single or multiple model views. *IJCV*, 2006.

[60] Juho Kannala, Esa Rahtu, Sami S. Brandt, and Janne Heikkilä. Object recognition and segmentation by non-rigid quasi-dense matching. In *CVPR*, 2008.

[61] Greg Mori, Xiaofeng Ren, Alexei A. Efros, and Jitendra Malik. Recovering human body configurations: combining segmentation and recognition. In *CVPR*, 2004.

[62] Greg Mori. Guiding model search using segmentation. In *ICCV*, 2005.

[63] Xuming He, Richard S. Zemel, and Debajyoti Ray. Learning and incorporating top-down cues in image segmentation. In *ECCV*, 2006.

[64] Chunhui Gu, Joseph J. Lim, Pablo Arbelaez, and Jitendra Malik. Recognition using regions. In *CVPR*, 2009.

[65] C. Schmid. Constructing models for content-based image retrieval, 2001.

[66] Bastian Leibe and Bernt Schiele. Interleaved object categorization and segmentation. In *In BMVC*, pages 759–768, 2003.

[67] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *In ECCV workshop on statistical learning in computer vision*, pages 17–32, 2004.

[68] B. Leibe, K. Mikolajczyk, and B. Schiele. Segmentation based multi-cue integration for object detection. In *British Machine Vision Conference (BMVC'06)*, September 2006.

[69] Sanjiv Kumar and Martial Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *ICCV*, 2003.

[70] Xuming He Richard, Richard S. Zemel, and Miguel Á. Carreira-perpiñán. Multiscale conditional random fields for image labeling. In *CVPR*, 2004.

[71] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 2007.

[72] Ramesh Jain and H.-H. Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. *TPAMI*, 1979.

[73] Chris Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *CVPR*, 1999.

[74] P. Kaewtrakulpong and R. Bowden. An improved adaptive background mixture model for realtime tracking with shadow detection. In *EWAVBSS*, 2001.

[75] Marko Heikkilä, Matti Pietikäinen, and Senior Member. A texture-based method for modeling the background and detecting moving objects. *TPAMI*, 2006.

[76] Koichi Sato and J. K. Aggarwal. Temporal spatio-velocity transform and its application to tracking and interaction. *Comput. Vision Image Understand*, 2004.

[77] Andreas Griesser, Stefaan De Roeck, Alexander Neubeck, and Luc Van Gool. GPU-Based Foreground-Background Segmentation using an Extended Colinearity Criterion. In *Proceedings of Vision, Modeling, and Visualization (VMV) 2005*, 2005.

[78] Michal Irani, Benny Rousso, and Shmuel Peleg. Computing occluding and transparent motions. *International Journal of Computer Vision*, 1994.

[79] Anurag Mittal and Daniel P. Huttenlocher. Scene modeling for wide area surveillance and image synthesis. In *CVPR*, 2000.

[80] Robert Kaucic, A. G. Amitha Perera, Glen Brooksby, John P. Kaufhold, and Anthony Hoogs. A unified framework for tracking through occlusions and across sensor gaps. In *CVPR*, 2005.

[81] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Pose tracking from natural features on mobile phones. In *ISMAR*, 2008.

[82] Mustafa Ozuysal, Vincent Lepetit, François Fleuret, and Pascal Fua. Feature harvesting for tracking-by-detection. In *ECCV*, 2006.

[83] Gregory D. Hager and Peter N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *TPAMI*, 1998.

[84] Bruce D. Lucas and Takeo Kanade. An itimage registration technique with an application to stereo vision. In *IJCAI*, 1981.

[85] Michael J. Black and Allan D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *IJCV*, 1998.

[86] Shai vidan. Support vector tracking. *TPAMI*, 2004.

[87] D. Koller, K. Danilidis, and H.-H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *IJCV*, 1993.

[88] Ragini Choudhury, Cordelia Schmid, and Krystian Mikolajczyk. Face detection and tracking in a video by propagating detection probabilities. *TPAMI*, 2003.

[89] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc J. Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *ICCV*, 2009.

[90] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *IJCV*, 2004.

[91] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *ARTIFICAL INTELLIGENCE*, 1981.

[92] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *TPAMI*, 2011.

[93] G.R. Bradski. Real time face and object tracking as a component of a perceptualuser interface. *Applications of Computer Vision, 1998. WACV '98. Proceedings., Fourth IEEE Workshop on*, 1998.

[94] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, 2000.

[95] Amit Adam, Ehud Rivlin, and Ilan Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, 2006.

[96] Suha Kwak, Woonhyun Nam, Bohyung Han, and Joon Hee Han. Learning occlusion with likelihoods for visual tracking. In *ICCV*, 2011.

[97] Junseok Kwon and Kyoung Mu Lee. Tracking by sampling trackers. In *ICCV*, 2011.

[98] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *TPAMI*, 2003.

[99] Robert Collins, Yanxi Liu, and Marius Leordeanu. On-line selection of discriminative tracking features. *TPAMI*, 2005.

[100] Shai Avidan. Ensemble tracking. *TPAMI*, 2007.

[101] Helmut Grabner and Horst Bischof. On-line boosting and vision. 2006.

[102] Kenji Okuma, Ali Taleghani, Nando De Freitas, O De Freitas, James J. Little, and David G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *ECCV*, 2004.

[103] Xiaofeng Ren. Finding people in archive films through tracking. In *CVPR*, 2008.

[104] Michael D. Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, and Luc Van Gool. Online multiperson tracking-by-detection from a single, uncalibrated camera. *TPAMI*, 2011.

[105] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31:824–840, 2009.

[106] David C. Lee, Martial Hebert, and Takeo Kanade. Geometric reasoning for single image structure recovery. In *CVPR*, 2009.

[107] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009.

[108] Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. In *Proceeding of International Conference on Computer Vision (ICCV)*, 2009.

[109] Huayan Wang, Stephen Gould, and Daphne Koller. Discriminative learning with latent variables for cluttered indoor scene understanding. In *Proceedings of the 11th European conference on Computer vision: Part IV*, pages 497–510, 2010.

[110] David Changsoo Lee, Abhinav Gupta, Martial Hebert, and Takeo Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. *NIPS*, 2010.

[111] Varsha Hedau, Derek Hoiem, and David A. Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. In *ECCV*, 2010.

[112] Abhinav Gupta, Alexei A. Efros, and Martial Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *European Conference on Computer Vision(ECCV)*, 2010.

[113] Abhinav Gupta, Scott Satkin, Alexei A. Efros, and Martial Hebert. From 3d scene geometry to human workspace. In *Computer Vision and Pattern Recognition(CVPR)*, 2011.

[114] A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Efficient structured prediction for 3d indoor scene understanding. In *CVPR*, 2012.

[115] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. Multi-view stereo for community photo collections. In *Proceedings of the 11th International Conference on Computer Vision (ICCV 2007)*, pages 265–270, Rio de Janeiro, Brazil, 2007. IEEE.

[116] Sudipta N. Sinha, Drew Steedly, and Richard Szeliski. Piecewise planar stereo for image-based rendering. In *ICCV*, 2009.

[117] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Brian Curless, Steven M. Seitz, and Richard Szeliski. Reconstructing rome. *IEEE Computer*, 43, 2010.

[118] David Crandall, Andrew Owens, Noah Snavely, and Daniel P. Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2011.

[119] Grace Tsai, Changhai Xu, Jingen Liu, and Benjamin Kuipers. Real-time indoor scene understanding using bayesian filtering with motion cues. In *ICCV*, 2011.

[120] Grace Tsai and Benjamin Kuipers. Dynamic visual understanding of the local environment for an indoor navigating robot. In *IROS*, 2012.

[121] Brian Gerkey. Amcl.

[122] Augusto L. Ballardini, Axel Furlan, Andrea Galbiati, Matteo Matteucci, Francesco Sacchi, and Domenico G. Sorrenti. An effective 6dof motion model for 3d-6dof monte carlo localization. In *In Proceedings of 4th Workshop on Planning, Perception and Navigation for Intelligent Vehicles (IEEE/RJS IROS 2012)*, 2012.

[123] Changchang Wu. VisualSFM: A visual structure from motion system. `http://homes.cs.washington.edu/~ccwu/vsfm/`, 2011.

[124] A Furlan, D Marzorati, and DG Sorrenti. Scale-independent object detection with an implicit shape model. In *Crime Detection and Prevention (ICDP 2009), 3rd International Conference on*.

[125] Lucia Prisciantelli. *BISMD: sistema robusto per l'identicazione ed il riconoscimento di categorie multiple di oggetti basato sull'apprendimento del modello implicito di forma*. Master's thesis, 2013.

[126] Giuseppe Ventimiglia. *Sistema flessibile per il multi-tracking di oggetti in un framework probabilistico basato sul detector BISMD*. Master's thesis, 2013.

[127] Bastian Leibe, Nico Cornelis, Kurt Cornelis, and Luc Van Gool. Dynamic 3d scene analysis from a moving vehicle. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

[128] Bastian Leibe, Konrad Schindler, Nico Cornelis, and Luc Van Gool. Coupled object detection and tracking from static cameras and moving vehicles. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(10):1683–1698, 2008.

[129] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1):259–289, May 2008.

[130] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, October 2005.

[131] Website of the rawseeds project, checked aug. 2009. `http://www.rawseeds.org`.

[132] Website of the visor repository, checked aug. 2009. `http://www.openvisor.org`.

[133] Graz University of Technology (TUGraz). Tugraz-02 dataset, 2007.

[134] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[135] Karlsruhe Institute of Tecnology. Karlsruhe dataset, 2007.

[136] University of Illinois at Urbana-Champaign. Uiuc image database for car detection.

[137] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 2010.

[138] `http://vision.stanford.edu/3Dlayout/` `http://www.ira.disco.unimib.it/free_your_camera`.

[139] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[140] C. Tomasi and T. Kanade. Detection and tracking of point features. *IJCV*, 1991.

[141] J. Canny. A computational approach to edge detection. *PAMI*, 8(6), 1986.

[142] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Manhattan-world stereo. In *CVPR*, 2009.

[143] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Reconstructing building interiors from images. In *ICCV*, 2009.

[144] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Recovering surface layout from an image. *IJCV*, 75(1), 2007.

[145] Alexander G. Schwing and Raquel Urtasun. Efficient exact inference for 3d indoor scene understanding. In *ECCV*, 2012.

[146] Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1994.

[147] Hauke Strasdat. *Local Accuracy and Global Consistency for Ecient Visual SLAM*. PhD thesis, 2012.

[148] Miguel Angel Sotelo. Lateral control strategy for autonomous steering of ackerman-like vehicles. *Robotics and Autonomous Systems*, 2003.

Acknowledgments - Ringraziamenti

Un sentito grazie ai miei genitori, per il supporto morale e materiale; e per tutto il bene che ci siamo voluti e che ci vogliamo.

Special thanks to Domenico Sorrenti, for all the great opportunities, for believing in me, supporting me in the most difficult moments and for the unique humanity shown in these years.
Special thanks to Silvio Savarese, for the great opportunity of working in his research group, for encouraging me in the tough moments and for all the great time at UoM.
Special thanks to Fei-Fei Li, for welcoming me in her great research group and for the energy put in our research work.
Special thanks to all the lab-mates of the IRALab in Milan, of the Vision Lab in Michigan and of the Stanford Vision Lab in Stanford, for all the help and the amazing moments spent together, within and outside the lab.
Grazie agli zii e cugini del Monferrato, Beppe Lida Stefano Andrea ed Erica, per il supporto, l'affetto, le camminate.. insomma, semplicemente per tutto (non basterebbe una tesi per fare l'elenco).
Un grazie particolare ad Augusto ed Andrea per l'affetto e la forza con cui, insieme, abbiamo sempre superato i problemi di lavoro e non.
Un agraïment especial a la Laura per totes les coses que m'ha ensenyat, tant en els moments bons com en els difícils.
Un grazie speciale a Roby e Renata per l'affetto, le cene, le risate e, soprattutto, per avermi preparato, chi in un modo chi nell'altro, ad affrontare e superare i problemi complessi ed a spiegare quelli complicati.

Grazie ad Andrea il 'pellegrino' ed alla famiglia Martelli tutta per l'affetto, le deliziose, pantagrueliche mangiate e le lunghe chiacchierate.
Grazie a Fulvio ed a Daniela per tutti gli sprazzi di divertimento nelle lunghe ore di lavoro!
Grazie, inoltre, a tutti coloro cui, anche se per un breve periodo, ho avuto l'onore di camminare a fianco in questi anni.