THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'Université Toulouse III - Paul Sabatier*
Discipline ou spécialité : *Systèmes embarquées*

Présentée et soutenue par *Cyrille Berger*
Le *15 Décembre 2009*

Titre : *Perception de la géométrie de l'environment pour la navigation autonome*

JURY
Walter Mayol, Rapporteur
Fawzi Nashashibi,Rapporteur
Jean-Denis Durou, Membre
Rachid Alami, Membre
Joël Morillon, invité
Simon Lacroix, invité

Ecole doctorale : *EDSYS*
Unité de recherche : *LAAS/CNRS*
Directeur(s) de Thèse : *Simon Lacroix*
Rapporteurs : *Walter Mayol et Fawzi Nashashibi*

Contents

Introduction

1 Context: field robotics

The work presented in this thesis is related to outdoor robots (figure 1), that have to autonomously achieve high level missions, such as *transport*, *surveillance*, *exploration*... The autonomous achievement of such missions is desirable for various reasons: because communications with distant operators are very constrained, to relieve the operator's work load, or even to reduce the numbers of involved operators in the mission.

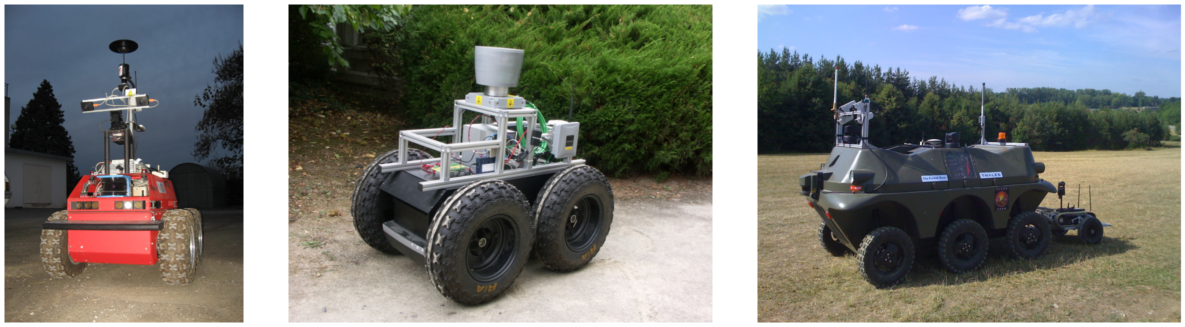


Figure 1: Three robots adapted to evolved in open and semi-structured terrains. From left to right: Dala and Mana from LAAS, respectively equipped with a stereovision bench and a panoramic Velodyne Lidar, and rTrooper from Thales TOSA, equipped with two stereovision benches.

On the importance of environment models

Executing autonomously such high level missions calls for the achievement of more elementary tasks, such as *motions*, *observations*, *communications*, *object grabbing*... These tasks are always defined with respect to the environment of the robot:

- *motions*: a model of the terrain is necessary to compute the easiest, safest and quickest path between two positions.
- *observations*: for instance, if the robot is tasked to take a picture of a given area, it needs a *visibility model* to know the positions from where the building can be observed.
- *communications*: similarly to the observations, a model of the environment is required to assess the places from where communications with other robots of the operator station are possible.

- *object grabbing* requires not only a 3D model of the object, but also a model of the surrounding environment to define the accessible positions from which the object can be grabbed.

This non exhaustive list of tasks shows that *several environment models* are required to plan and decide the robot actions: the planning and decisions processes call for the evaluation of the outcome of the possible actions, which is provided by confronting the environment models to the models of the robot actions (including perception actions). Each environment model is *dedicated* to the achievement of one particular task. For instance a traversability model can be exploited to plan motions, whereas a 3D volumetric model is required to plan observations. Nevertheless, there can be overlaps between the information represented in the various models – and the *environment geometry* is in particular an essential information, encoded in various forms in most of the necessary environment models.

On the importance of localisation

An estimate of the robot position is also required for various reasons:

- Missions are mostly expressed in localisation terms (*e.g.* `GoTo ‘‘position’’`, `Explore ‘‘area’’`...),
- The achievement of motions defined as geometric trajectories requires a precise knowledge of the robot position,
- To be spatially consistent, the environment models built on the basis of the data acquired by the robot require the knowledge of the robot position at the time of the data acquisitions.

Roughly, one can categorise the solutions to the localisation problem in three groups:

- Motion estimation: the current robot position is derived (or rather, “integrated”) from the composition of elementary motion estimates. Odometry and inertial navigation fall in this category.
- Absolute localisation with respect to known beacons: the determination of distances or angles of beacons whose position is known in a given reference frame provides a position in this reference frame. Beacons can be of various natures, *e.g.* lighthouses used by sailors, low frequency radio emitters for Loran, or satellites for GPS.
- Map-based localisation: a map of the environment can be seen as a generalisation of beacons, each element localised on the map (building, crossroads...) playing the role of a beacon.

Motion estimation has an important intrinsic drawback: the localisation information is indeed obtained by compounding elementary motions that are systematically measured with an error. As a consequence, however small this error can be, the localisation estimate eventually drifts over time or distance, this drift being not bounded.

Beacon-based localisation is very effective and provides a localisation estimate with a bounded error. But it requires the existence of beacons, which can not be ensured in every environment – and even though GPS satellites are present all around the earth, their signal can be outed by obstacles or jammers.

Similarly to beacon-based localisation, a map of the environment is required for map-based localisation solutions. However, it calls for the ability to *associate* map elements with perceived data (a problem that is straightforwardly solved by dedicated sensors in beacon-based localisation).

Simultaneous localisation and mapping

The links between environment models and localisation are pretty obvious: on the one hand environment models can be a solution to the localisation problem (map-based localisation), but on the other hand, the localisation problem must be solved in order to build spatially consistent environment models.

In robotics, in the absence of an *a priori* map of the environment, the robot is facing a kind of "chicken and egg problem": it makes observations on the environment that are corrupted by noise, from positions which estimates are also corrupted with noise. These errors in the robot's localisation have an influence on the estimate of the observed environment feature locations (these features being referred to as *landmarks*), and similarly, the use of the observations of previously perceived landmarks to locate the robot provide position estimates that inherits from both errors: the errors of the robot's pose and the map features estimates are correlated.

It has been understood early in the robotic community that the mapping and the localisation problems are intimately tied together [Chatila and Laumond, 1985, Smith et al., 1987], and that they must therefore be concurrently solved in a unified manner, turning the chicken and egg problem into a virtuous circle. The approaches that solve this now classic problem in robotics are commonly denoted as "Simultaneous Localisation and Mapping" (SLAM): roboticists have thoroughly studied this problem, and proposed numerous effective solutions for either indoor robots, field robots, aerial robots or submarine robots.

2 Thesis objective: towards rich geometric maps

We have mentioned that the environment geometry is a basic information that is required to build most of the environment models required for autonomy. The notion of *obstacle* or of *traversable area* is indeed essentially geometric, it is in particular closely related to the geometry of the considered robots. Geometry is naturally what defines the *visibility* models. Also, localisation (of the robot and of landmarks or other environment features) is by definition a position in a given Cartesian reference frame, and therefore a geometric concept.

Note that if geometry is an essential information to recover to build the environment models, it is not the *sole* necessary information: for instance traversability is also a function of the physical nature of the terrain, and the material of volume impacts the radio communications visibility model. Regarding localisation, some tasks can be achieved without the *explicit* knowledge of the robot position, such as servoing to a visually detected target. Similarly, some authors tackled the SLAM problem according to a non-metric (non-geometric) approach (*e.g.* [Cummins and Newman, 2008, Angeli et al., 2008]): the environment map is here a collection of signatures (or image indexes) and the localisation is only defined with respect to previously perceived data. In such approaches, not only the environment model does not exhibit any geometric structure, but also it is *sensor-centric*, as it is defined on the basis of the sensor signal.

However, geometry is an *intrinsic* characteristic of any environment. As opposed to visual appearance for instance, it is not dependant on illumination conditions or sensor photometric calibration. A purely geometric model based on geometric primitives (whatever they are, from the simplest 3D points to planar areas or higher order objects such as pieces of quadrics, or spline representations of surfaces) has therefore the following characteristics:

- *sensor independence*: some geometric primitives can be detected by a wide range of sensors: *e.g.* planar areas can be detected using either a camera, a 3D Lidar, a millimeter wave radar...
- *view-point independence*: a geometric primitive being *intrinsically* defined, its nature does not depend on the sensor viewpoints from which it is observed. The only issue is that some singular sensor/primitive configurations and occlusions may lead to partial observations of its parameters.
- *compatibility with existing models*: whatever the environment considered (even other planets of the solar system), there is now a huge amount of information that is available (*e.g.* through [Google, ,IGN,]). Most of these information are geometric (digital terrain models, 3D cities models, ...).

These properties of geometric representations make them a must in most of the mobile robotics systems. For instance, geometry is the only environment information that be exploited to define cooperation schemes between a micro-drone equipped with a camera and a ground robot equipped with a Lidar.

Approach

A very interesting property of landmark-based SLAM approaches is that they solve both the geometric localisation and mapping problems in a sound and consistent manner. The resulting environment model is therefore a geometric model, but remains however very restricted: it is a *collection of landmarks*, located in the environment with an estimated error, to which is sometimes attached a descriptor. In the early stages of SLAM, landmarks were 2D lines detected by a horizontal laser scanner, and nowadays most of the 3D SLAM approaches consider 3D points as landmarks (*e.g.* interest points when using vision). Such environment models are not very expressive: they cannot be used to assess the traversability or a given area or to compute visibilities for instance: landmark models are *dedicated* to the SLAM problem, the most important outcome of SLAM becoming the localisation of the robot.

Our approach to build a rich versatile geometric model consists in providing a SLAM algorithm with higher level geometric landmarks, namely planer facets and line segments. Since we rely on a SLAM approach, errors on the parameters of the modelled primitives are explicitly taken into account. The resulting landmarks model represents therefore more explicitly the geometric structure of the environment than point landmarks for instance: the most important outcome of our approach is a consistent rich geometric model of the environment.

Whatever the geometric primitives used to represent landmarks, the resulting model is a sparse description of the environment, a “cloud” of heterogeneous landmarks. We define a graph structure over the set of landmarks to allow its manipulation, and in particular to ease data association processes with a existing 3D model or between models built by different robots.

Contributions

In the direction towards endowing robots to build meaningful geometric environment models, the contributions of this thesis are the following:

- The definition of geometric landmarks to allow the application of a SLAM approach that leads to more structured landmark maps than most of the existing contributions. We introduce planar facets and line segments landmarks, and the associated vision algorithms to detect, track and match them in images.
- The exploitation of such landmarks in an EKF based SLAM approach. In particular, we have participated to the extension of the hierarchical SLAM approach introduced in [Estrada et al., 2005] to a multi-robot SLAM approach, and applied our algorithms to the building of an environment model using data gathered by a ground and an aerial robot.
- The definition of the graph structure on the basis of maps of heterogeneous landmarks. This structure is in particular dedicated to the definition of a map matching algorithm.

3 Manuscript outline

The manuscript is organized in three parts:

- The first part is an analysis of the environment modeling problem. Chapter 1 starts with a synthetic review of the state of the art in SLAM. It then presents a multi-robot multi-maps SLAM approach, and the overall organization of the use of heterogeneous landmarks geometric model in a SLAM approach. Chapters 2 and 3 then respectively present the essential techniques required to detect features in the perceived data and to associate them when perceived in different conditions.
- The second part gathers our contributions in the definition of landmark detection, tracking and matching algorithms:
  - Chapter 4 introduces an approach to match interest points, that exploits their repartition in the image.
  - Chapter 5 introduces a higher level of geometric landmarks: “facets” are oriented points that correspond to locally planar areas. The algorithms to detect them from a stereoscopic pair of images and to match them from various viewpoints are presented.
  - Chapter 6 is devoted to the detection and tracking of straight lines in images. A new approach to this classical vision problem is proposed, that yields more robust extractions than the existing approaches.

  Chapter 7 compares the various landmarks introduced in this part. It illustrates their use in a EKF based hierarchical SLAM approach, and presents some results of building an environment model from images acquired with a ground and an aerial robot.

- The third part of the manuscript is devoted to the definition of a generic geometric model composed of various kinds of geometric landmarks. A way to organize and structure the detected landmarks within a graph-based representation is presented in chapter 8, and an algorithm to exploit this representation for the data association problem is presented in chapter 9.

The manuscript ends with a discussion that draws some perspectives for further researches towards the achievement of a robust and versatile geometric environment modeling system.

Part I

The problems of environment modeling

Our approach to build a 3D geometric environment model stems on the use of SLAM, because this allows to explicitly deal with the various sensors noise

To be able to construct environment models, a robot must be endowed with the capacity to sense the environment, detect relevant features, track and match them, and estimate their positions. The feature detection is performed by a *signal processing* algorithm, and usually yield a feature description composed of two parts: metric information and a signal descriptor. The metric information is used by the *estimation* process to update the model. A critical part of that process is to match current observations with past observations.

In the context of an outdoor robot, possibly operating in cooperation with other robots, environment modeling raises two main challenges:

- *The robot travels long distances:* as a consequence the robot will collect a lot of information, that need to be managed in a way that the algorithms remains efficient.
- *The built model must explicit the environment geometry:* this is the sole intrinsic environment characteristics that can be fused among robots equipped with various kind of sensors, or with an existing environment map.

This part is an analysis of the various essential functions required to build a geometric environment model. Chapter 1 summarizes the SLAM problem and the main solutions of the literature, and introduces the multi-map approach that we are relying on. Chapter 2 recalls some basics of environment perception, and chapter 3 introduces the matching problem and its various instances.

Chapter 1

Simultaneous Localisation and Mapping

This chapter starts with a synthetic overview of the SLAM problem and the various solutions that have been proposed in the literature, insisting naturally on the nature of the resulting geometric environment models. It then briefly describes the development of a multi-robot multi-map SLAM approach to which we have contributed with a post-doctoral student from the lab, and that we have been exploiting to evaluate our landmarks detection, tracking and matching algorithms. The chapter ends with an overview of the organisation of our environment modeling approach within a multi-map SLAM context.

1.1 A short introduction to the SLAM problem

The SLAM problem has now been widely studied. A historical presentation of the main contributions can be found in [Dissanayake et al., 2001], and a recent synthesis of the state of the art is given in the two papers [Durrant-Whyte and Bailey, 2006] and [Bailey and Durrant-Whyte, 2006].

The principle of the incremental landmark-based solutions is presented figure 1.1. Such solutions encompasses the following four steps:

- *Landmark detection.* This consists in detecting in the perceived data elements of the environment that are salient, and whose position relatively to the current robot position can be estimated. This process naturally depends on the kind of the considered environment, on the sensor(s) used, and on the definition of the landmarks. It is a *perception process.*

- *Estimation of relative measures.* This consists in two distinct processes:

  - Estimation of the landmarks position with respect to the current robot pose: this is the *observation* process.
  - Estimation of the relative robot motions between two consecutive landmark observations. This estimation is the *prediction process.*

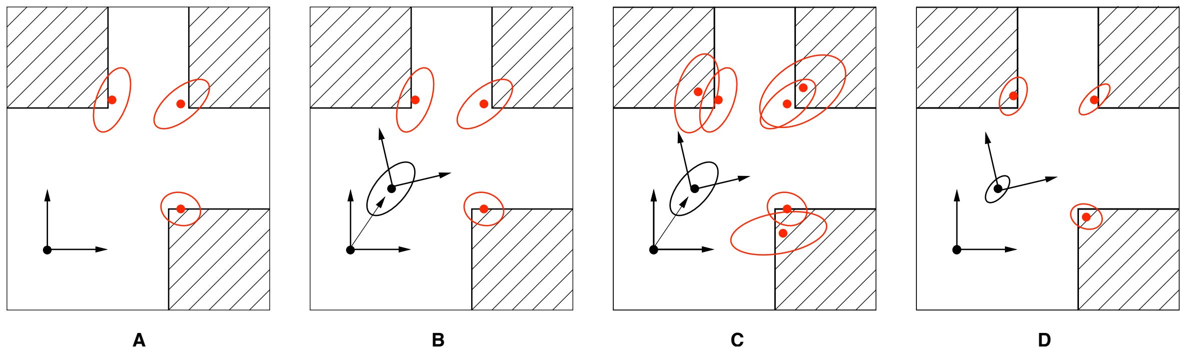


Figure 1.1: The four main steps of the SLAM process illustrated for a robot evolving in a plane: detection, prediction, matching, estimation. In (A), landmarks present in the environment are detected and located relatively to the robot position with given errors. In (B), the robot moves, and a measure of its displacement is provided by any other mean – also with a given error. Landmarks are perceived again and matched with the previously detected landmarks in (C), and finally (D) illustrates the outcome of the estimation process: both the robot and landmarks positions are known with a better precision in the global frame. This process is iteratively repeated as the robot moves and gathers information on the landmarks of the environment.

- *Data association.* Landmarks can help to refine the robot position estimates only if they can be observed from different positions. For that purpose, it is necessary to *associate* the landmarks perceived form different positions.

- **Estimation.** Estimation is the heart of any SLAM process: it consists in fusing the various relative measures to produce an estimate of the robot and landmarks positions in a global common frame.

  Stochastic approaches to this problem estimate an *a posteriori* distribution on the the landmarks and robot positions from all the acquired data. This distribution can be written as:

$$p(X_r(k), \{X_f(k)\}|\mathbf{z}, \mathbf{u}) \tag{1.1}$$

  where $X_r(k)$ is the current robot pose (at time k) and $\{X_f(k)\}$ is the set of landmarks poses, conditioned on all the relative landmark observations $\mathbf{z}$ and the relative motion estimates $\mathbf{u}$.

Besides these four processes, a SLAM solution must also deal with the following two issues:

- *Loop closing.* When the robot only discovers new landmarks (as when moving along a straight line for instance), the precision of the estimate of its position decreases. But when it navigates in a previously visited zone, the precision of its position estimate can be improved by re-perceiving previously mapped landmarks, which also improves the precision of all the mapped landmarks (figure 1.2). It is with such events, nick-named loop-closings, that makes SLAM approaches so relevant.

  Handling properly a loop-closure requires that the robot is able to associate the current landmark observations with previously mapped landmarks. This is a data association

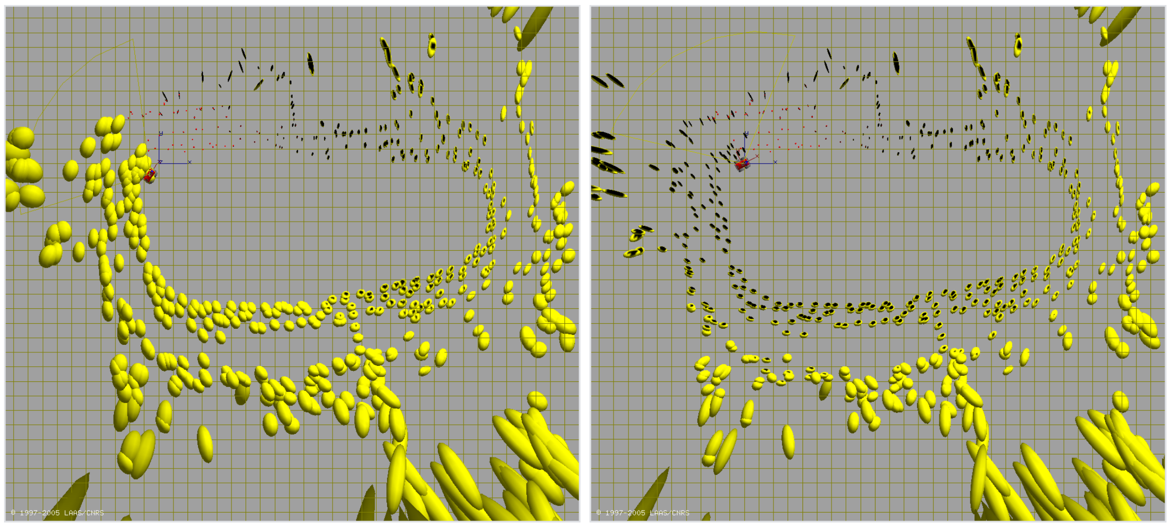


Figure 1.2: Errors on the landmark positions before a loop-closure (left) and after (right). Note the reduction of the ellipsoids whose volume represent the trace of the associated covariance matrices associated to the landmark positions – landmarks are here points detected and localized by stereovision (figure excerpt from [Lemaire et al., 2007]).

process, that can be very difficult if the precision on the current robot pose estimate is too low to allow the association on the sole basis of the landmarks pose estimates, or if the current viewpoint is very different from the one under which the landmarks have first been detected.

- *Map management.* Maintaining a single map of all the perceived landmarks yields an algorithmic complexity problem, and can hinder the overall consistency of the system – particularly for the EKF based SLAM approaches, whose consistency can not be guaranteed because of the various non-linearities present in the observation and prediction equations. Both problems can be resolved by a proper way of managing the map: various solutions have been proposed in the literature for this purpose.

So every SLAM process requires the implementation of the four steps mentioned above, plus a proper way to manage the landmark map and the loop closure events. The following sections summarizes the main contributions of the literature to achieve these steps.

1.1.1 Defining a landmark

The definition of the landmark to use is naturally essential. A minimum definition is that a landmark is defined by its geometric position, and must exhibit characteristics that allow its extraction from the perceived data. Data association can then be made on the sole basis of the estimation of its geometric position, which can only be achieved if the motion and landmark positions estimates are very precise. “2D segments” landmarks extracted from a horizontal laser scanner in the early ages of SLAM are of this kind [Wijk and Christensen, 2000, Thrun et al., 2000, Dissanayake et al., 2001].

Such a geometric description is required by the estimation processes, but can be enriched by information that ease the data association process. A classic example is the use of visual descriptors, such as the one associated to interest points (*i.e.* Harris points [Harris and Stephens, 1988a] or SIFT [Lowe, 2004]). Such descriptors allow to solve the data association problem without strongly relying on the current estimates of the geometric positions of the landmarks.

In 3D SLAM applications, most of the considered landmarks are points. A few recent contributions have focused on the use of higher level geometric primitives (segments detected in the images [Eade and Drummond, 2006, Smith et al., 2006, Lemaire and Lacroix, 2007a], or planar areas [Silveira et al., 2007]). We are convinced that this is the direction to take, because it can be the basis of the definition of a geometric environment model that exhibits more structure than point-based approaches.

1.1.2 Relative measures estimation

Relative estimation of the landmark positions. This measure is defined by the landmark detection process. Let's simply mention here that these observations can be complete (all the parameters that define the landmark position are observed), or partial, as in the case of bearing-only SLAM.

Relative estimation of the robot motions. Various means can be employed for this purpose, provided they are independent from the ones that provides the landmarks positions. One can use odometers for ground rovers, a dynamic model of the robot that predicts motions as a function of the control input, or simply an hypothesis on the robot behavior – such as a constant speed model for instance.

For both landmarks and robot motions measures, a key point is to have good error models: an optimistic error model rapidly leads to inconsistent estimates, while a pessimistic one yields estimates with a poor precision.

1.1.3 Data association

Data association consists in establishing correspondences between landmarks detected from two different positions. We distinguish two cases:

- Between two consecutive positions k and $k+1$, data association comes to a *tracking process.*
- Between arbitrarily distant viewpoints, data association comes to a *matching process.*

What actually distinguishes these two cases is the difference between the viewpoints:

- In the tracking case, data association is eased by the fact that the predicted position of the landmark in the data acquired at $k+1$ is precisely estimated: the robot motion being small, the associated error is also small. Furthermore, tracking a landmark does not necessarily requires a detection step at time $k+1$: landmarks detected at time k can be directly associated to the data of time $k+1$ (see *e.g.* the famous KLT tracker – [Shi and Tomasi, 1994a]).
- In the matching case, the error on the robot position estimate can be large, and the comparison between the predicted and observed positions of the landmarks can be of no help to associate them.

Generally speaking, whatever the landmark is, the tracking case is well solved, whereas the matching case raises difficult issues, as the difference of viewpoints can yield very different "appearances" of the landmarks. Even if some sound statistical approaches rely on the predicted and estimation positions to match landmarks [Neira and Tardos, 2001], we are convinced that approaches that match landmark *independently from their position estimates* are more robust. Interest points matching algorithms such as the one we present in chapter 4 fall in this category of solutions.

1.1.4 Estimation and stochastic maps management

Various estimation formalisms

Most of the contribution on SLAM deal with the estimation framework. In particular, stochastic approaches are well adapted [Thrun, 2002]. Among those, the use of Kalman filter is historically quite popular: it is proved that a Kalman filter is a solution to SLAM in the linear case [Dissanayake et al., 2001]. Besides non-linearities, one of the main drawback of this approach is that the algorithm complexity is quadratic with respect to the number of landmarks: various contributions deal with this issue – see *e.g.* [Guivant and Nebot, 2001, Knight et al., 2001].

The information filter, a dual version of the Kalman filter, enables the elimination of the less significant correlations, and hence lead to linear complexity approaches [Thrun et al., 2004]. The *FastSLAM* approach [Montemerlo et al., 2003] introduces the use of particle filters for SLAM. Other estimation approaches have been proposed, mainly to cope with the fact the the involved errors are not necessarily Gaussians (for instance, set-membership approaches [Kieffer et al., 2000]). Other authors tackle the problem as a global minimization problem – the first contribution is [Lu and Milios, 1997], and these approaches have recently gained a lot of interest: in particular, they can handle the non-linearities. GraphSLAM has been introduced in [Thrun and Montemerlo, 2005, Folkesson and Christensen, 2004], which relies on the use of sparse constraint graphs, where the nodes are landmarks and the different positions of the robot. Similarly, in [Olson et al., 2007], a method using incremental optimization of the graph has been proposed.

Map management

A proper management of the landmark map can limit the algorithmic complexity, but also inconsistencies (figure 1.3). In the standard approach, a single global reference frame is used: in this frame the robot pose and the landmark estimates can have arbitrary large errors, which are likely to produce large linearisation errors. The main idea of other map structures is to adopt a representation where these errors can be bounded.

At the opposite of the monolithic global map approach, the relative map representation has been proposed in [Newman, 1999]. Rather than estimating the global transformation of a landmark, relative transformations between neighbour landmarks are estimated. This raises the problems of choosing which relative transformations to put in the map, a problem of consistency for loops of landmarks, and also the problem of retrieving the global coordinates. These issues are addressed in [Newman, 1999] with the development of the Geometric Projection Filter (GPF).

In between the absolute map and the relative maps algorithms, there is the family of local maps, or sub-maps algorithms [Estrada et al., 2005]. The local maps are maintained using a

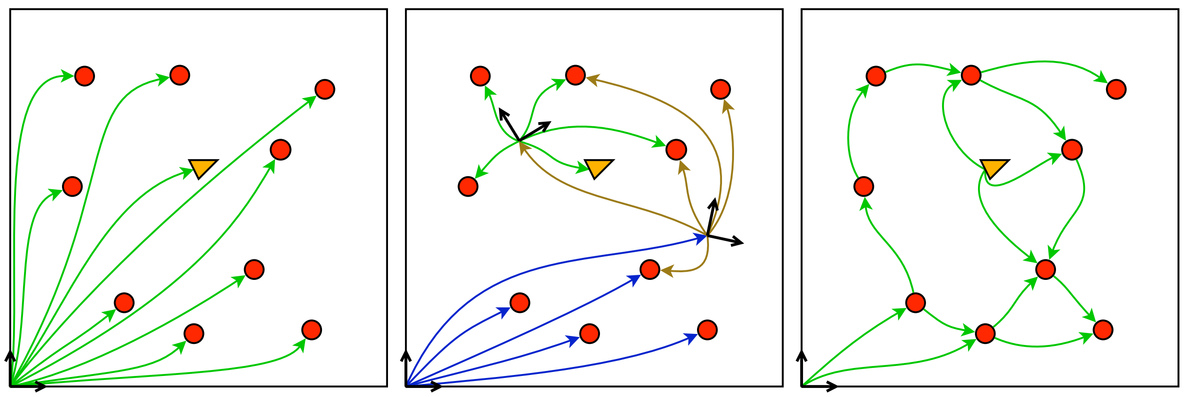


Figure 1.3: Various landmark maps structures (the arrows represent the position information that is memorized). From left to right: "classic" single global map approach, sub-0map approach, and relative maps approach.

simple estimation technique, for instance a Kalman filter. The local maps are controlled so as to maintain:

- a bounded map update processing time: when the number of landmarks is too high, a new sub-map is created,
- a bounded robot estimate error: when the uncertainty on the robot pose is too large, or when the robot has moved for a given amount of distance, a new sub-map is also created.

The case of partial observations

In the first classic SLAM approaches, the sensors used and landmark chosen allow a full observability of the landmark position parameters: each newly detected landmark is simply added to the landmark map.

But in the case of partial observability, a process of *landmark initialization* is required. A pioneer solution to this problem in the case of (monocular) vision-based SLAM has been proposed in [Davison, 2003], where the initially unobservable point landmark depth is represented by a uniform distribution in a given range by a set of particles. As the landmark is tracked in successive frames, this distribution is updated, and once it can be modelled by a Gaussian distribution, the landmark is added to the map. Various improvements to this approach have been proposed [Lemaire et al., 2007, Sola et al., 2005], until in 2006 where an undelayed initialisation process has been proposed in [Montiel et al., 2006], which is now the "gold standard" in EKF-based vision SLAM.

1.1.5 Loop closures

Even with matching algorithms that are independent of the landmarks pose estimates, detecting a loop closure remains challenging.

Efforts have been invested in the development of approaches that rapidly indicate if the current position corresponds to a previously visited area, mainly exploiting image indexing techniques [Cummins and Newman, 2008, Angeli et al., 2008] – these approaches being naturally well suited to panoramic images. In landmark based SLAM, by providing a qualitative estimate of the robot current position, these approaches can focus the search for matches with landmarks in the vicinity of this position, or can allow to retrieve the image stored in a database that is the closest to the current one, on which a landmark matching algorithm can be run [Lemaire and Lacroix, 2007b].

As we mentioned in the introduction of the manuscript, these approaches are sensor-centric, and can therefore not be used in a heterogeneous multi-robot context, nor be generalized to establish data associations with an a priori existing environment model.

1.2 Multi-maps and multi-robots

We mentioned that a limitation of the EKF-based SLAM approach is that it does not scale to large maps. Furthermore, while it is possible to have two robots in the same filter, this would mean that the mapping is centralized, which require that all robots are constantly communicating. For obvious reasons, this is not practical, the mapping need to be decentralized.

A solution to this problem is to use different maps for each robot, and to use small maps for each robot. We summarize in this section work made in collaboration with Teresa Vidal during her post-doctoral stay in the laboratory. The approach focuses on making possible for multiple robots to construct locally independent maps, and to connect them at a higher level, to create a global map and to improve the localization of the robots. But the actual communication problem remains to be solved, especially the aspect of synchronization of the global level, as well as ensuring the consistency and avoiding the duplication of information.

1.2.1 Multi-maps

In [Estrada et al., 2005], Estrada proposes a hierarchical approach, with two levels: at the low level the robot uses locally independent maps, that are connected together at the higher global level in a graph of maps.

In the graph of maps, each node corresponds to one map, each map is connected to other maps by an edge that contains a transformation between the two connected maps. This connection between maps is defined when a new map is started: the transformation between the two maps is equal to the position of the robot in the previous map. When there is a cycle in the graph of maps (a loop closure), the transformations between the maps of that cycle can be optimized.

Each of the local maps is constructed independently, when a map is started, the robot position in that map is set to 0, with a null covariance, and the features are reinitialized with new observations, then the robot moves and the map is updated, until a new map is started. See algorithm 1.

The multimap approach has been extended to multiple robots in [Vidal-Calleja et al., 2009]: the idea is that each robot constructs its own local map, and only the global level is shared, which allows to connect two maps coming from two different robots, improving the localization of both robots.

Algorithm 1 Multimap algorithm

```
1. Initialize a new map $\mathcal{M}_i$ , the robot is initialized in $(0, 0, 0)$ with a null covariance,
   initialize the features
2. The robot move and update the map
3. When the map is too large, or when the robot local uncertainty is too important, a
   new map $\mathcal{M}_{i+1}$ is started, and the graph is updated with a link between the previous
   map $\mathcal{M}_i$ and the new map $\mathcal{M}_{i+1}$, the transformation is the position of the robot in the
   previous map $\mathcal{M}_i$.
```

1.2.2 Multi-robots events

In the case of a single robot [Estrada et al., 2005], a loop can only be closed between two maps when the robot moves to a previously visited location. In a multiple robots system, various events generate loop closures [Vidal-Calleja et al., 2009]:

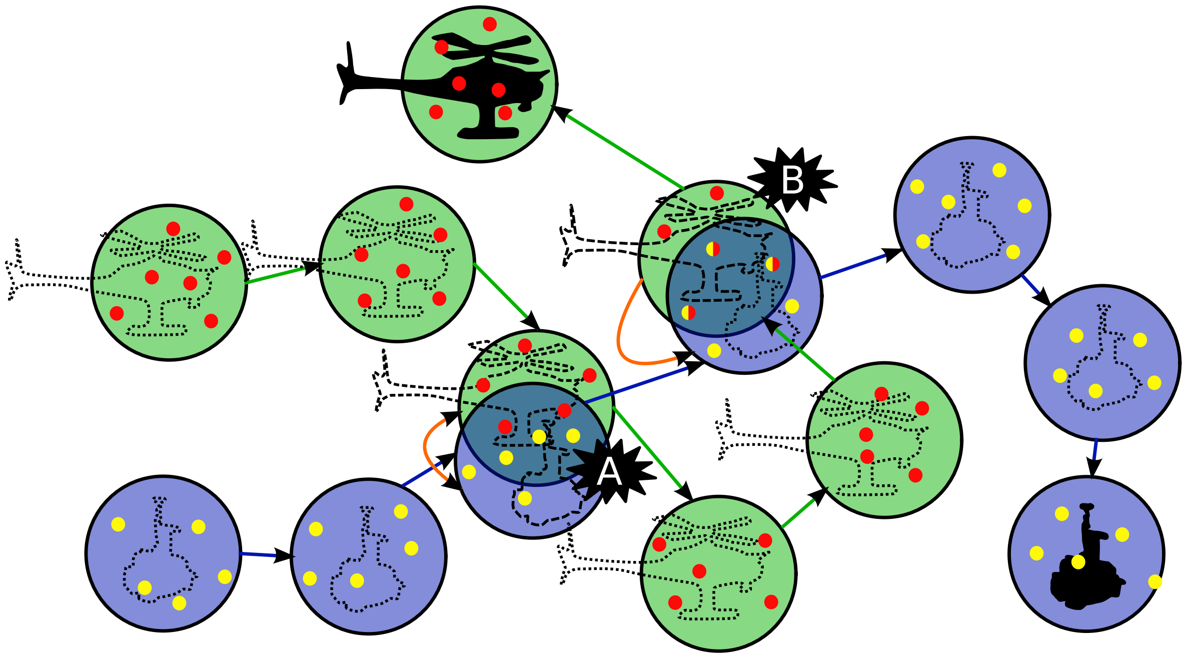


Figure 1.4: Multi-robot, multi-maps. The event A is triggered by a rendez-vous between the robots, while in event B, the helicopter detect an area mapped by the ground robot.

1. *map-matching*: this event happen when a robot comes to a location that has already been mapped by any of the other robots

2. *rendez-vous*: this event happen when a robot detect another robot, the relative transformation between the two robots can be observed directly using a camera, for instance if each robot is tagged with a recognizable pattern, alternatively a match between what both robots are observing can be used

3. *GPS fix*: this event happen when a robot receives a GPS signal that locates it in the geo-referenced frame. Provided other GPS fixes occurred before, this event define a loop in the graph of maps.

All of those events trigger a link between maps at the global level.

1.2.3 Simulation results

We show here the effects of multiple loop closures between three robots. Two ground robots $\mathbf{r}_1$ and $\mathbf{r}_2$ move along circles in different locations on the environment. They never meet, and their maps never overlap. The third robot $\mathbf{r}_3$ is an aerial robot that moves in a circular trajectory which extends from the area covered by $\mathbf{r}_1$ to the one covered by $\mathbf{r}_2$.

The three robots start at a well known location, then at $240\,s$, $\mathbf{r}_1$ and $\mathbf{r}_3$ have a *rendez-vous*, later at $610\,s$ and $700\,s$ the robot $\mathbf{r}_3$ detects a loop closure with two maps of $\mathbf{r}_2$ (the first part of the video shows a run of this experiment). The uncertainties are expressed in the *wrf* so that the effects of the loop closure can be seen. The consistency plots for a single run are shown in Figure 1.5. The final map is shown in Figure 1.5(d).

1.3 Our approach to construct maps

In this thesis, high level geometric landmarks are mapped according to an EKF-based multimap approach, to be able to deal with large maps or multi-robot systems. The process of constructing the environment is divided in four functional blocks: localization (or motion estimation), mapping (SLAM), multimap management, map structuring.

Localization The localization process have access to the different incremental sensors: odometry, inertial, visual odometry [Lacroix et al., 1999]. These provide an estimation of the current movement of the robot, that is used for the prediction of the SLAM process (local level maps: the local position estimates are not integrated with the information coming from the multimap manager to get the global position of the robot).

Another input of the localization process is GPS data: such information is given to the *map manager* to optimise the position of the local maps.

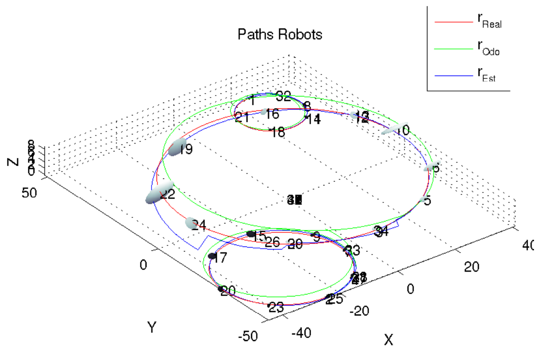
SLAM The SLAM process is used to construct the local map of the environment, and to refine the local position of the robot. It takes as inputs the motions estimates of the robot, and detects and track landmarks – as presented in part II. At the end of the estimation, the condition to terminate the current map and create a new map are checked.

Once a local map is terminated, it is transmitted to the map structuring block.

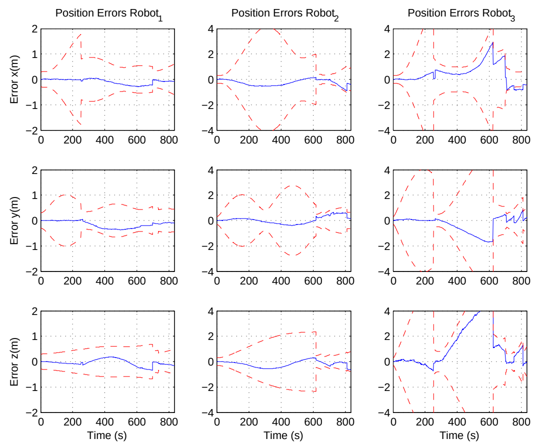
Map manager The role of the multimap manager is limited to keeping track of the position of previous map, to optimize the graph of maps when there is a loop-closure event (GPS or feature match) and to give absolute position of the robot and geometric objects.

Map structuring: "SPAF" This process creates and updates the geometric model, according to the framework detailed in part III. As an input, it gets a terminated local landmark map, and transforms it into a local graph of features[1]. This graph structure is exploited to

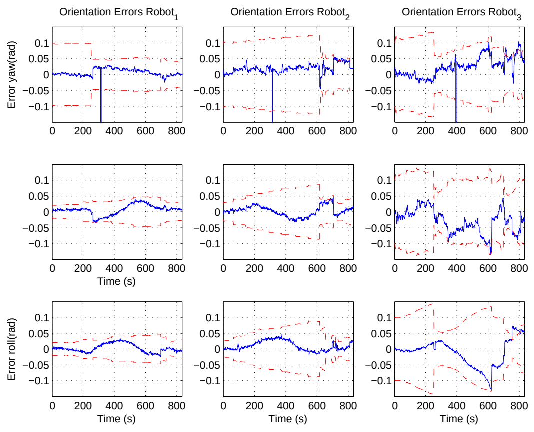
[1] "SPAF" stands for "spatial features"



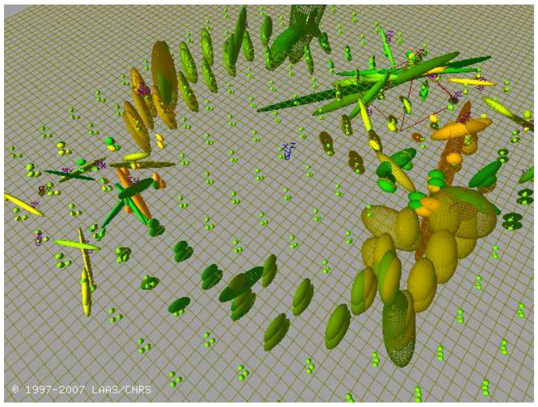
(a) Robot paths and Global level



(b) Position Errors $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$



(c) Orientation Errors $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$



(d) The resulting map

Figure 1.5: Simulation results for the 3 robots exploration; one aerial and two ground robots. In a) the odometry is shown in green, real and estimated trajectories are shown in red and blue respectively. 3σ ellipsoids are plotted on the basis of each *lrf*. b) shows the global position errors for each robot and their global 3σ uncertainty bounds. c) shows the global orientation errors for each robot and their global 3σ uncertainty bounds (the jump on the yaw curve around $t = 450$ is due to a switch from $-\pi$ to π).

match local maps to establish loop closures.

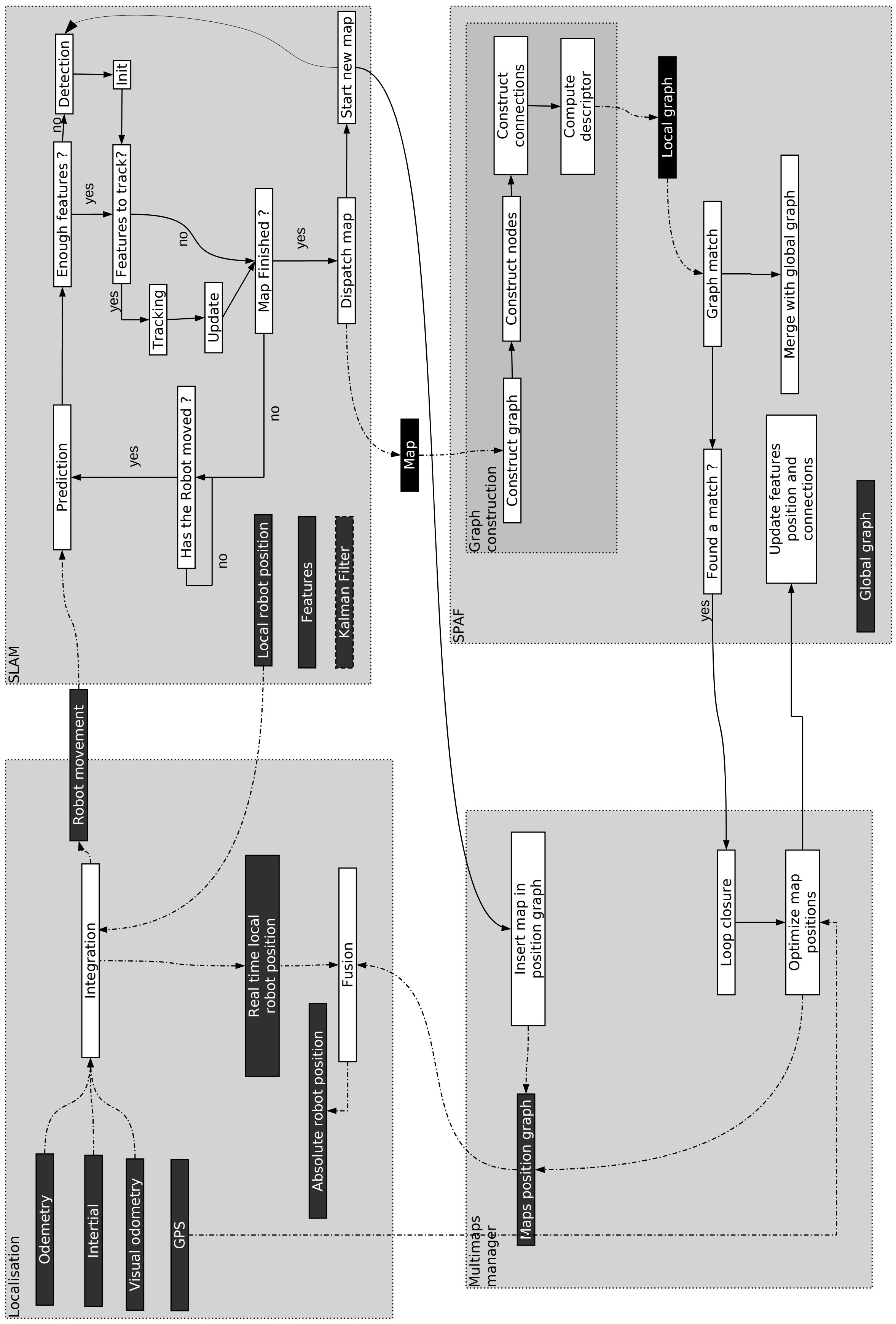


Figure 1.6: The four main processes involved in our approach to environment modeling

Chapter 2

Detection

When a robot needs to interact with its surrounding environment, before creating a model, it firsts need to be able to extract information from this environment, information that comes from exteroceptive sensors.

In this chapter we will show the sensors (section 2.1) that can be used to perceive the environment, what kind of information to extract (section 2.2), and a few basis of how to extract it (section 2.3 and section 2.4).

2.1 Sensors

A robot can use two types of sensors, *active sensor* (2.1.1) which detect the radiation emitted or reflected by an artificial source, while the *passive sensor* (2.1.2) are detecting radiation emitted by a natural source [Everett, 1995].

2.1.1 Active sensors

An active sensor works by emitting a radiation (light, electric wave...), the measurement of the delay between its emission and its detection gives the distance between the sensor and the object, this type of sensor is capable of measuring the distance between the robot and objects, they are often refered as range sensors.

Sonar A Sonar (SOund Navigation And Ranging, figure 2.1(a)) uses acoustic waves. It is mostly use underwater, but it can also be used in the air (bats use acoustic waves to sense their environment).

Radar A Radar (RAdio Detection And Ranging, figure 2.1(b)) emits electromagnetic waves.

The quality of the results is highly dependent on the reflection of materials, for instance, metallic surface have a good reflectance for a radar, while rock have very low reflectance making it more difficult to be detected. But this can be used to get information about the material, for instance, vegetation is transparent to radar waves, so a robot could have better detection of real obstacles.

Lidar A Lidar (LIght Detection And Ranging) is a sensor that measures the properties of the reflection of scattered light (usually a pulsating laser) on objects.

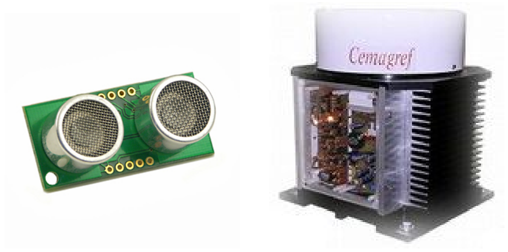


Figure 2.1: On the left, a cheap sonar, on the right the Cemagreph's K2Pi Radar

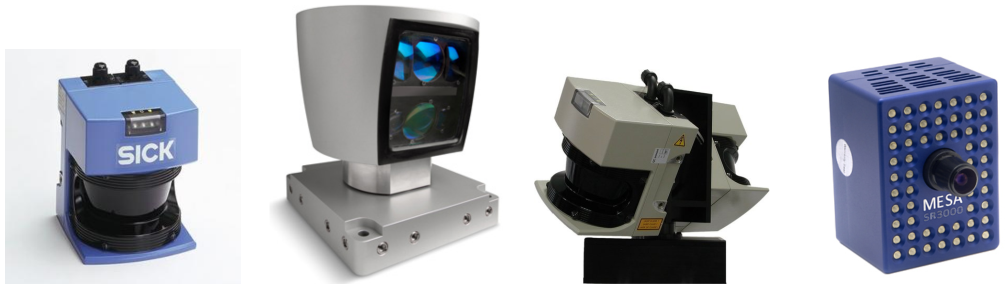

(a) Sick (2D) (b) Velodyne (3D) (c) 3DLS-K (d) Swissranger

Figure 2.2: Example of Lidar systems

On a 2D Lidar, there is a single laser rotating in a plane, and acquiring data at a high frequencies. Some 3D Lidar are based on a 2D Lidar, either by using two of them, in a 360° rotation (figure 2.2(c)), or by rotating a single one. An other type of 3D Lidar relies on a rotation of multiple lasers pointing in different direction (figure 2.2(b)). The last type of 3D Lidar is the Swiss ranger camera (figure 2.2(d), [AG,]), infrared light is pulsed on the scene, and an infrared camera is used to detect the return of the light, this sensor works mostly on the short range.

Lidar, based on a laser, usually have a better angular accuracy than Radar and are able to give more accurate data. But they also have a shorter range.

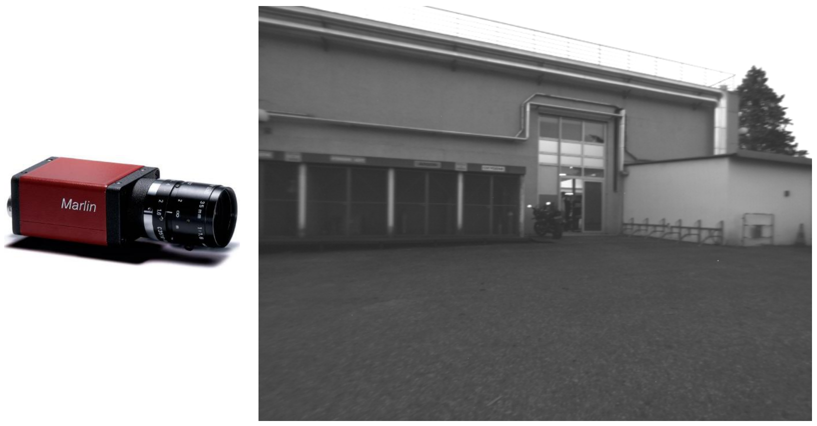
The main advantage of active sensor is that they are reasonably easy to use, since they give directly range information, they also have very good accuracy, while the price range from a few Euros (for a sonar) to several thousands Euro (for a 3D Lidar like the Velodyne).

Active sensors have a rather limited range, while biggest radar can have a range of a hundred kilometres, embedded radar (such as IMPALA, [Cemagref et al.,]) have a range of 150 meters, Lidars tend to have a range between 50 and 120 meters (for the Velodyne, figure 2.2(b), [Velodyne,]).

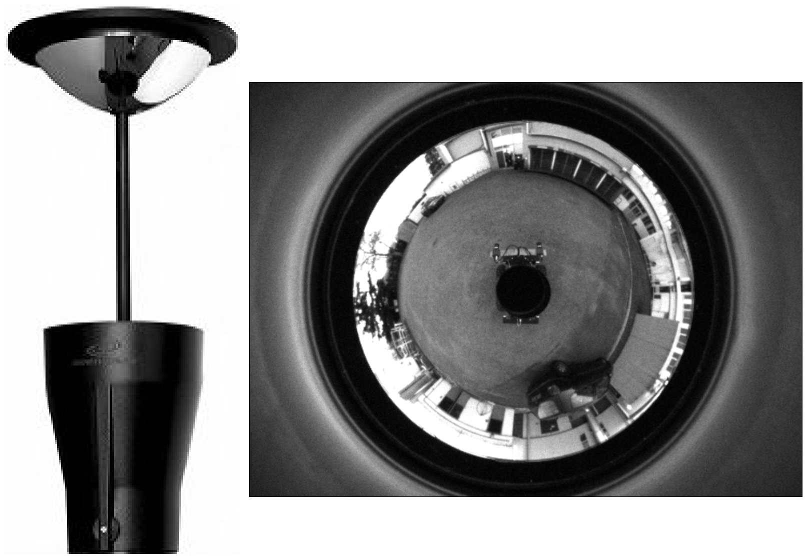
They are also subject to interferences, when multiple robots are located in the same area, or by a malicious agent that scrambles the signals. For military applications, another issue with active sensors is that the signals emitted by those sensors can be detected by the enemy, which would give away the position of the robot.

2.1.2 Camera

For environment modelling, the most useful passive sensor is the camera. There are different type of cameras, the two common models are perspective camera (figure 2.3(a)) and omnidirectional camera, which are either made by a few perspective cameras arranged on a circle, or by a single camera with a mirror in front of it (figure 2.3(b)).



(a) Perspective Camera

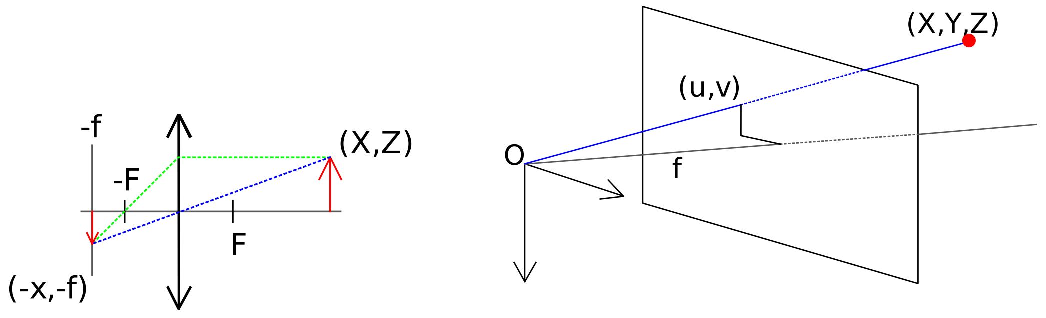


(b) Catadrioptic Camera

Figure 2.3: Different types of camera and what the image they acquire for the same scene.

This section only covers some basic element of camera models, the reader interested for more information would be advised to read [Hartley and Zisserman, 2000] for a more complete description of the perspective camera model as well as stereovision.

Perspective camera In a real system, camera are built using a multiple lenses system, which is modelled with a single *lens*, situated at the *optical center* (O) of the camera. The axis of the lens is called *optical axis*. The image is projected on a plane perpendicular to that axis and called *image plane* (at distance $-f$ from the O), F is called the *focal point* of the lens.



(a) With a focus lens (b) Projection

Figure 2.4: Model of a perspective camera

Perspective cameras are often modeled using *the pinhole model*, where the lens is replaced by a very small hole, and real world points are projected on the camera using lines (the blue line on figure 2.4) that pass through the optical center.

The following equation gives the projection from a 3D points (X, Y, Z) on the image plane:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.1}$$

$$u - u_0 = u_{res} \cdot x \tag{2.2}$$

$$v - v_0 = v_{res} \cdot y \tag{2.3}$$

(x, y) are the physical coordinate of the projection of the point on the sensor, (u_0, v_0) is the projection of the optical center on the resulting image, and (u, v) is the coordinate of the pixel in the image that correspond to the projection of the point(X, Y, Z). u_{res} and v_{res} are the horizontal and vertical resolution of the sensor. A calibration process [Bouguet,] allow to compute $\alpha_u = f \cdot u_{res}$ and $\alpha_v = f \cdot v_{res}$.

This model ignores the distortion that appears when using lenses. With $r = ||(x, y)||$, the following equations are used to model the distortion:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = (1 + \sum_{n>0} d_n \cdot r^n) \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.4}$$

Omnivision Omnidirectional camera offer the advantage of capturing information from all directions, as can be seen on figure 2.3(b). The model of omnidirectional camera can be found in [Geyer and Daniilidis, 1999] and [Barreto and Araujo., 2001].

Stereovision A monovision camera, with a single frame, does not allow to recover depth information. To recover depth information, it is necessary to have at least two shots from two different positions, either by taking new shots when the robots move, or by having two cameras on the robots, with a known transformation between them.

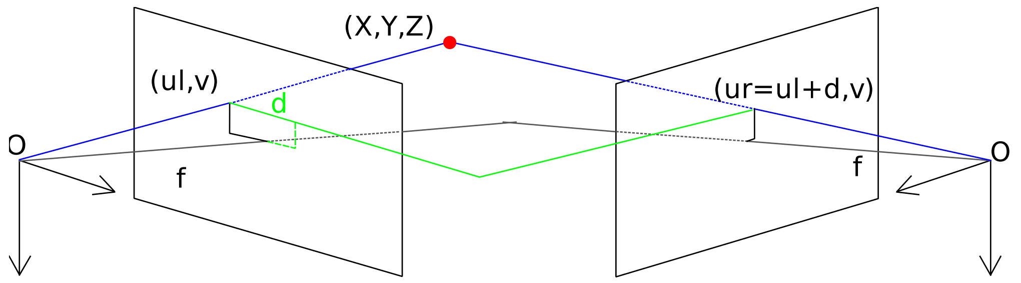


Figure 2.5: Epipolar geometry of a stereovision bench

Then a correlation algorithm [Dhond and Aggarwal, 1989] is used to match the pixels of the two images, and using the disparity d, the distance between the pixel in the two images. the geometry of the system and camera model (figure 2.5), it is possible to recover the depth, and the coordinates of the 3D point in the world [Hartley and Zisserman, 2000], using the following equations:

$$Z = \alpha / d \tag{2.5}$$

$$X = Z \cdot (u - u_0) / \alpha_u \tag{2.6}$$

$$Y = Z \cdot (v - v_0) / \alpha_v \tag{2.7}$$

Where α, u_0, v_0, $\alpha_u = f \cdot u_{res}$ and $\alpha_v = f \cdot v_{res}$ are estimated using a calibration process, for instance using the Calibration toolbox of mathlab [Bouguet,]. α is called the baseline and correspond to the distance between the two cameras, (u_0, v_0) are the coordinates of the optical center in the image and u_{res} and v_{res} are the resolutions in pixel per meter of the sensor.

2.2 Extracting information

Using the sensors presented in the previous section, the robot is capable of sensing the environment, this section focuses on what kind of information to extract from the data, and on giving basic rudiments of signal processing.

2.2.1 What information to extract ?

Since the different models of the environment store different information, they also need to extract different type of information.

Landmark Sea charts, used by sailors to localize their boat, highlight the most remarkable elements (lighthouse, mountain, rock...) of the environment, and then a sailor can use a compass to trace the direction between the landmark and the boat. The characteristic for a good landmark for a sea chart is to be seen from a long distance and to have a unique shape to be easily recognized and detected in the environment. A group of shapes (for instance, three rocks) is also a good landmark.

Good landmarks for localizing a robot would share similar characteristics, as they would need to be salient, easily detectable, measurable and matchable. For instance, a good landmark in a signal, can be associated to a local extrema of the signal.

A landmark contains two parts, geometric information and optionally appearance information. The minimum geometric information is the position in the world, but it can also contain information such as the size or the direction. The existence of appearance information is depending on the type of sensor, for a camera it can be the texture, for instance.

Occupancy grids As mentioned in section 2, another type of environment model relies on the use of occupancy grid, either for obstacle detection or for observability. Such grids are either in 2D or 3D, the main idea is to associate to each cell of the grid a probability of obstacle.

Several techniques exist to construct such a grid, but generally it is done by checking if there is a measurement of an object in a cell. For instance, the output of a laser is a set of distances between the sensor and an obstacle, by projecting each distance on the grid, it is possible to know the cell that contains the obstacle, and then to raise the probability of an obstacle in the cell.

A 2D grid [Moravec and Elfes, 1985] is made of a single horizontal layer of cells, which means that 2D grids suffer from the lack of vertical information, and therefore, they are more suitable for an indoor environment, where the ground is flat, and the robot will stay on the same plane. But they can be extended to outdoor if each cell of the grid also record the height of the ground in the cell, such grids are often referred as “Digital Terrain map” (DTM). They are very efficient in an outdoor environment to give a model of the ground, but they are not very efficient to when there is a vertical wall or when the robot goes under a bridge or indoor, in a multiple floor building, unless a multiple height grid is used.

A 3D occupancy grid [Moravec, 1996] is made of several layers, which allows to store volume information. But, the memory footprint of such a grid increases drastically compared to 2D grids, while most of the grid would be free.

The size of the cells give the precision of the grid. A too fined grain map will give accurate information, but will require more memory, more computational time for the update and for the use.

2.3 Rudiments of signal processing

In the previous section 2.2, we have shown what kind of information we need to extract from the data. This section will focus on the basic tools that are used to extract information from data to be able to use it in the environment model.

This section is a quick overview of signal processing, a reader interested in more details about signal processing would be advised to read [Moon and Stirling, 2000] for a deeper presentation of signal processing algorithms.

2.3.1 Signal

A signal is a time-varying or space-varying measurable quantity. In the real-world, that quantity is often a piecewise continuous function, we will note $f(t)$ a 1D continuous signal and $f(x, y)$ a 2D continuous signal. Discrete signals are noted $D(n)$ and $D(u, v)$.

Discretisation While a natural signal is usually a continuous function, sensors do measurement by sampling, either of time, or of space. The interval between two measurement t_1 and t_2 used for the sampling is called *sampling interval*, and noted $\epsilon = t_2 - t_1$.

We define the *sampling* function $s(t)$ as follow:

$$n \in \mathbb{Z}, \forall t \in [n \cdot \epsilon, (n+1) \cdot \epsilon, s_\epsilon(t) = n] \tag{2.8}$$

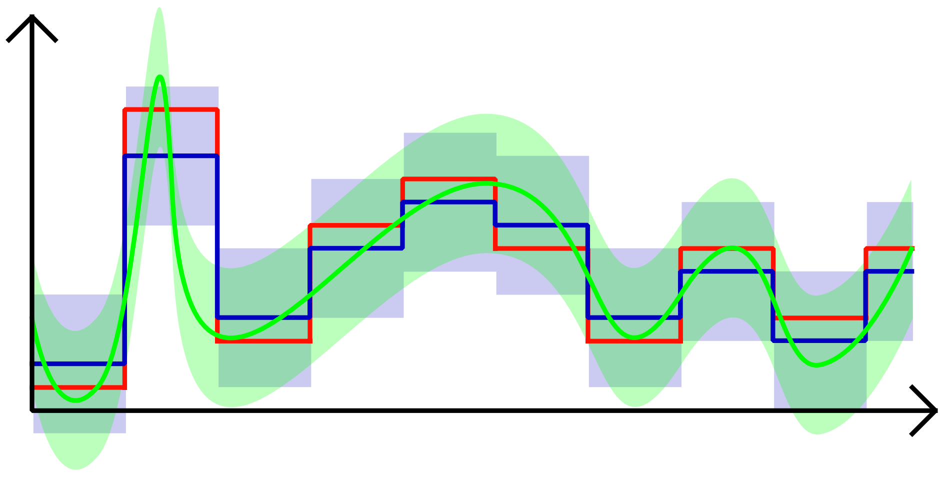


Figure 2.6: Discretisation of the signal, the green curve is the continuous signal $f(t)$ the green area show the standard deviation of the signal, the blue curve show the discretised signal, the blue area show the standard deviation of the error for the measurement, while the red one shows the observation of the signal with the measurement noise.

The discretised signal, and the continuous signal are linked by the following equations (Figure 2.6):

$$n \in \mathbb{Z}, \qquad D(n) = \int_{t=n}^{n+1} f(\tfrac{t}{\epsilon}) dt \tag{2.9}$$

$$(u, v) \in \mathbb{Z}^2, \quad D(u, v) = \int_{x=u}^{u+1} \int_{y=v}^{v+1} f(\tfrac{x}{\epsilon}, \tfrac{y}{\epsilon}) dy dx \tag{2.10}$$

Theorem 1 (Nyquist–Shannon) *If a function x(t) contains no frequencies higher than B hertz, it is completely determined by giving its ordinates at a series of points spaced 1/(2B) seconds apart.*

The theorem 1 of Nyquist-Shannon [Shannon, 1949] indicates the condition for a discetised signal to fully represent a continuous signal, provided that an infinite number of sampling are made and that signal is band-limited (all the coefficient of its Fourier decomposition are null above a finite frequency). While the theorem is expressed in 1D, it can easily be extended to multiple dimensions.

If those conditions are respected, it is possible to recover the original signal with the following formula:

$$f(t) = \sum_{n=-\infty}^{\infty} D(n) \cdot sinc(\frac{t - n \cdot \epsilon}{\epsilon}) \tag{2.11}$$

$$f(x,y) = \sum_{v=-\infty}^{\infty} \left(\sum_{u=-\infty}^{\infty} D(u,v) \cdot sinc(\frac{x - u \cdot \epsilon}{\epsilon}) \right) \cdot sinc(\frac{y - v \cdot \epsilon}{\epsilon}) \tag{2.12}$$

With $sinc(x) = \frac{sin(\pi \cdot x)}{\pi \cdot x}$.

Aliasing When the Nyquist-Shannon condition is not fulfilled, i.e. when the resolution of the sampling of the signal is too small compared to the frequency of the signal, the sampling of the signal creates unwanted artefacts, called aliasing.

Aliasing occurs when the sampling rate is too large compared to the frequency of the data, then the same measures can correspond to two different signals (see figure 2.7). A classical example of the aliasing problem occurred when displaying a line on a computer screen.

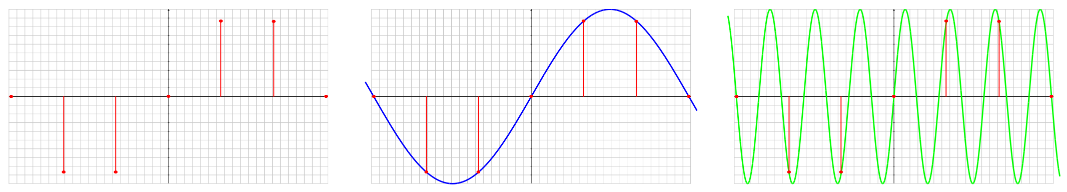


Figure 2.7: Same measures, different signals. The left graph show the measurement point, while the blue and green curve are two possible sinusoidal signals for that measurement.

In image processing, aliasing shows itself as a Moiré pattern, which is especially visible when downsampling a small repetitive pattern, such as a grid (Figure 2.8).

Aliasing raises problems in the processing of the signal: there is no good method to remove aliasing after the sampling and it is necessary to avoid aliasing before sampling the data. For instance, on a camera an optical anti-aliasing filter can be used in front of the sensor, this filter will work by reducing the frequency before the camera sensor.

Modeling of signal noise

$$f_o(t) = f(t) + w(t) \tag{2.13}$$

Where $w(t)$ is a white Gaussian noise, uncorrelated to data, of standard deviation σ.

Discretisation error Since the signal is detected by a sensor as a discretised signal, it means that the value measured correspond to multiple instants, or to an infinite number of

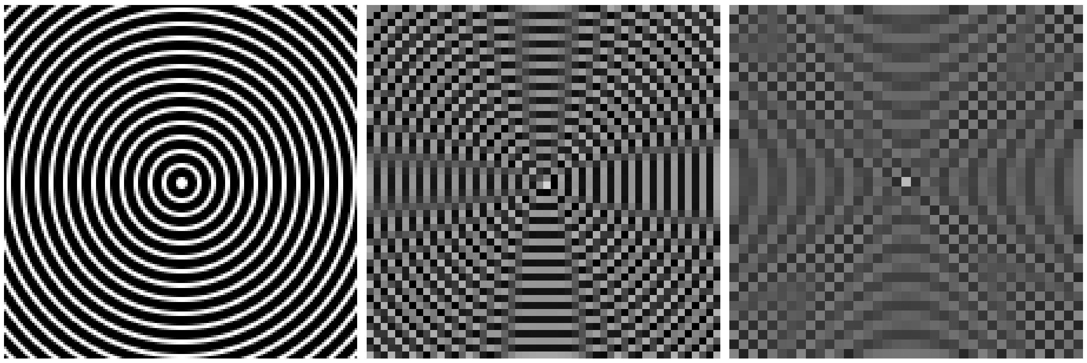


Figure 2.8: The Moiré effect, on left unscaled circles, on center a reduction by three and on right a reduction by four

real-world points for an image. This means that when an element of the dataset is selected as a landmark, there is an uncertainty on the point of the real world that really corresponds to that landmark.

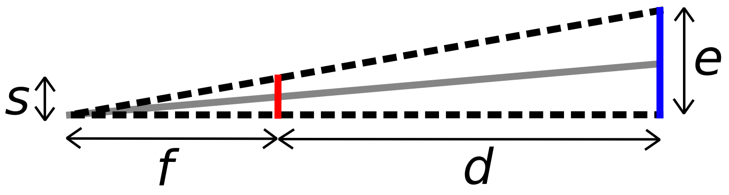


Figure 2.9: The red line shows the pixel of size s, while the blue line show the localisation error e of a point located at distance d and projected on the pixel with a camera of focal f.

For instance, in the case of stereovision, the Jacobian of the function that computes the coordinates of a 3D points (Equations (2.5) to (2.7)) from signal information is given by:

$$J = \begin{bmatrix} -\frac{\alpha}{d\cdot\alpha_u} & & \frac{\alpha\cdot(u-u_0)}{d^2\cdot\alpha_u} \\ & -\frac{\alpha}{d\cdot\alpha_v} & \frac{\alpha\cdot(v-v_0)}{d^2\cdot\alpha_v} \\ & & -\frac{\alpha}{d^2} \end{bmatrix} \tag{2.14}$$

Considering that the error around u and v is given by $\sigma_u = \sigma_v = 0.5px$, half the size of a pixel. Then, the error of the 3D point is given by:

$$E = J \cdot \begin{bmatrix} \sigma_u \\ \sigma_v \\ \sigma_d = 2 \cdot \sigma_u \end{bmatrix} \cdot J^t \tag{2.15}$$

2.3.2 Cross-correlation

The cross correlation is a measurement of the similarity of two signals, for two continuous signals, it is defined by:

$$(f \star g)(t) = \int_{-\infty}^{\infty} f^*(\tau) \cdot g(\tau + t) d\tau \tag{2.16}$$

Where f^* is the conjugate of the function f. For two discrete signals, this equation becomes:

$$(D_1 \star D_2)(t) = \sum_{\tau=-\infty}^{\infty} D_1^*(\tau) \cdot D_2(\tau + t) \tag{2.17}$$

But the absolute values of the signal can change when the condition of the data acquisition change, for instance, the values of the pixels of an image are very dependent to change of lightning and the time taken to acquire the data, hence the idea to subtract the mean and normalize by the standard deviation, giving the Zero mean Normalised Cross Correlation (ZNCC, also known, in statistics, as the sample Pearson correlation coefficient):

$$ZNCC = \frac{1}{n} \sum_{t=0}^{n} \frac{(D_1(t) - \overline{D_1}) \cdot (D_2(t) - \overline{D_2})}{\sigma_{D_1} \cdot \sigma_{D_2}} \tag{2.18}$$

Where $\overline{D_1}$ and $\overline{D_2}$ are the means of D_1 and D_2, and σ_{D_1} and σ_{D_2} are the standard deviations. The ZNCC score is equal to 1 when the two signals D_1 and D_2 are equal, it is equal to -1 when the $D_2 = -D_1$, a score of 0 means the two signals are uncorrelated.

2.3.3 Derivation

In the section 2.2.1, we mention that a good landmark would be located on an extrema of the signal, which a good landmark is located when the derivative get null, for a 1D signal this gives:

$$\frac{\partial f}{\partial t}(t) = \lim_{h \to 0} \frac{f(t+h) - f(t-h)}{2 \cdot h} = 0 \tag{2.19}$$

In the strict mathematical sense, a derivative can not be defined for a discrete signal $D(n)$, but assuming it corresponds to a continuous signal, it is possible to have an approximation of the derivative of the signal, with the following formula:

$$D'(n) = \frac{D(n+1) - D(n-1)}{2} \simeq \frac{\partial f}{\partial t}(s_\epsilon(t)) \tag{2.20}$$

This is easily extended to 2D signal, a maximum of the signal is found when both derivatives on x and y are null:

$$\frac{\partial f}{\partial x}(x, y) = \frac{\partial f}{\partial y}(x, y) = 0 \tag{2.21}$$

$$D'(u, v) = \frac{D(u+1) - D(u-1)}{2} \simeq \frac{\partial f}{\partial x}(s_\epsilon(x), s_\epsilon(y)) \tag{2.22}$$

$$D'(u, v) = \frac{D(v+1) - D(v-1)}{2} \simeq \frac{\partial f}{\partial y}(s_\epsilon(x), s_\epsilon(y)) \tag{2.23}$$

But due to the discretisation, it is unlikely that the condition $D'(n) = 0$ is ever found, and it is much simpler to check $D(n-1) < D(n) < D(n+1)$. But computing derivatives allows to find a maximum on that derivative, which is very interesting, since a maximum of the derivative correspond to an edge in the signal.

2.3.4 Histograms

An histogram H is a piecewise constant function, which means it exists a sequence $a(v)$, such as:

$$\forall v \in \mathbb{N} \forall x \in [a(v); a(v+1)[, H(x) = H(a(v)) \tag{2.24}$$

For a robotic signal, the sequence $a(v)$ is usually finite, and $v \leqslant n$, and unless the information is limited (for instance the pixel value of an image), often $a(n+1) = \infty$.

A signal is an injective function, for each instant t there is a definite value of $f(t)$ or $D(t)$, but it is usually not bijective, each value of the signal can be reached at different instant t_1, t_2... An histogram of the signal is a count of how many times the signal have a given value. In case of a discrete signal, it is defined by:

$$H(a(v)) = \sum_t (vote_v(D(t))) \tag{2.25}$$

$vote_v$ is a voting function that defines how much the value $D(t)$ contribute to the bin v, the most basic voting function would simply count the number of time $D(t)$ is contains in the interval $[a(v), a(v+1)[$ of the bin v:

$$vote_v(x) = \left\{ \begin{array}{l} 1,\ x \in [a(v), a(v+1)[\\ 0,\ x \notin [a(v), a(v+1)[\end{array} \right. \tag{2.26}$$

But a more sophisticated procedure can be used to spread the votes in the neighbourhood of the value to take into account that the value can be noisy:

$$vote_v(x) = \left\{ \begin{array}{l} \frac{1}{2},\ x \in [a(v), a(v+1)[\\ \frac{1}{4},\ x \in [a(v-1), a(v)[\texttt{ or } x \in [a(v+1), a(v+2)[\\ 0,\ x \notin [v-1; v+1] \end{array} \right. \tag{2.27}$$

The voting function of an histogram can be a continuous, for instance, if the value is associated to a density of probability $p_x(t)$. In that case, the voting function is the integral:

$$vote_v(x) = \int_{a(v)}^{a(v+1)} p_x(t) dt \tag{2.28}$$

To make an histogram of a continuous signal, the first step is to discretised it. And an histogram can also be considered as a signal, so all that apply to signal (like correlation) also work for histograms.

2.3.5 Indexing of histograms

Histograms are widely used to index data, either indexing and retrieval of images [Stricker and Swain, 1994], or as a descriptor of interest points [Lowe, 1999]. This raises the need for fast and efficient methods to compare two histograms:

- euclidean distance:

$$\sqrt{\sum_v (H_1(v) - H_2(v))^2} \tag{2.29}$$

- histogram intersection [Swain and Ballard, 1991]:

$$\frac{\sum_v min(H_1(v), H_2(v))}{min(|H_1|, |H_2|} \tag{2.30}$$

Where $|H_1|$ and $|H_2|$ are the number of samples used to create H_1 and H_2.

But if the number of samples can be different, or the data is to noisy, the use of a cross-correlation gives better results for comparing histograms. For an histogram $H(v)$, to compute fast comparison, the zero mean normalized histogram can be used:

$$H'(v) = \frac{H(v) - \overline{H}}{\sigma_{D_1}} \tag{2.31}$$

Where $\overline{H}$ is the mean of the histogram and σ_{D_1} the standard deviation.

Then the comparison between two histograms H_1 and H_2 is equal to the sum of the product of values:

$$\sum_{v=0}^{n} H_1(v) \cdot H_2(v) \tag{2.32}$$

While computing this distance between two histograms is a very fast operation, the computational cost increase drastically when we want to retrieve an histogram in a database. To reduce the cost, [Beis and Lowe, 1997] proposes to use a kd-tree, a binary tree, where each node is a point of dimension k, then each subnode is a subdivision of the remaining points using the median of the *i th* coordinate. A search is performed in that tree by defining in which leaf the histogram belongs, then the histogram is compared to the other histogram in that branch.

2.4 Rudiments of image processing

This section introduces some of the basics tool that are specific to image processing, [Gonzalez and Woods, 2002] contains more specific details and algorithms on image processing. Most of those tools rely on the use of a convolution kernel.

2.4.1 Convolution kernel

A convolution is an integral that blends a function g towards an other function f, it is defined as:

$$(f \otimes g)(t) = \int_{-\infty}^{\infty} f(\tau) \cdot g(t - \tau) d\tau \tag{2.33}$$

For a 2D discrete signal this translates to:

$$(D_1 \otimes D_2)(u, v) = \sum_{\mu=-\infty}^{\infty} \sum_{\nu=-\infty}^{\infty} D_1(\mu, \nu) \cdot D_2(u - \mu, v - \nu) \tag{2.34}$$

Usually, for an image, the convolution is done between the image I and a kernel K of finite width w_K and finite height h_K, which leads to the following formula:

$$(K \otimes I)(u,v) = \sum_{\mu=-w_K/2}^{w_K/2} \sum_{\nu=-h_K/2}^{h_K/2} K(\mu,\nu) \cdot I(u-\mu, v-\nu) \tag{2.35}$$

An example of a convolution kernel, to blur an image (low pass filter):

$$K = \frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.36}$$

2.4.2 Gradient

As mentioned in section 2.3.3, a maximum of the derivative indicates an edge in the signal, for an image it is therefore interesting to compute the gradient, and then to look for maximum of gradient to find edges in an image.

Sobel The *Sobel* filter [Sobel and Feldman, 1968] is very useful in image processing to compute an approximate of the derivative for an image. It is defined by the two following convolution kernels:

$$K_u = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad K_v = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.37}$$

$$G_u = K_u \otimes I \qquad G_v = K_v \otimes I \tag{2.38}$$

G_u gives the derivative on the image on the horizontal direction, and G_v on the vertical direction. The magnitude of the gradient $G(I)$ and the angle $\Theta(I)$ are then computed by the following formula:

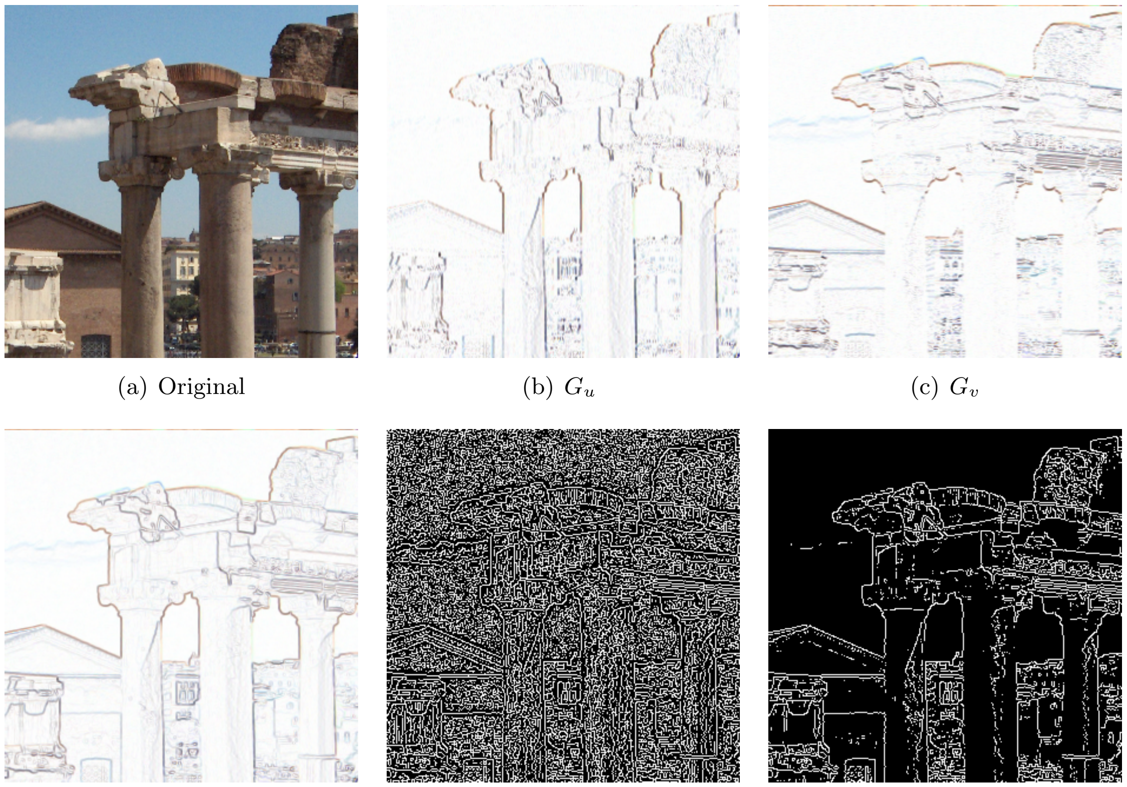
$$G(I) = \sqrt{(G_u^2 + G_v^2)} \quad \Theta(I) = arctan(\tfrac{G_v}{G_u}) \tag{2.39}$$

Canny In [Canny, 1986], Canny introduce an edge detector based on a maximum of gradient. Using $\Theta(I)$, the direction of the gradient is approximated, then the pixel is considered to be on an edge if it is a maximum in that direction and :

1. if $\Theta(I)(u,v) = 0°$, $G(I)(u,v) > G(I)(u,v+1)$ and $G(I)(u,v) > G(I)(u,v-1)$
2. if $\Theta(I)(u,v) = 45°$, $G(I)(u,v) > G(I)(u+1,v-1)$ and $G(I)(u,v) > G(I)(u-1,v+1)$
3. if $\Theta(I)(u,v) = 90°$, $G(I)(u,v) > G(I)(u+1,v)$ and $G(I)(u,v) > G(I)(u-1,v)$
4. if $\Theta(I)(u,v) = 135°$, $G(I)(u,v) > G(I)(u+1,v+1)$ and $G(I)(u,v) > G(I)(u-1,v-1)$

The resulting image of maximum of gradients is usually binarised with a threshold $\mathcal{T}$ to select only strong edges and to further reduce the noise.

Result of canny without threshold are shown on figure 2.10(e), and with a threshold set to $G(I)(u,v) > 100$ on figure 2.10(f).


(a) Original (b) G_u (c) G_v

(d) G (e) Canny with $\mathcal{T} = 0$ (f) Canny with $\mathcal{T} = 100$

Figure 2.10: Examples of the effect of a Sobel filter (sobel filters results are inverted for the purpose of printing) and Canny filter

2.4.3 Hough transform

The Hough method was introduced in [Hough, 1962], it allows to map an observation space, to a parameter space using a voting procedure, it was first applied to detecting segment in images in [Duda and Hart, 1972], while in [Ballard, 1981] the method was extended to the detection of any kind of shape in an image.

The main idea is to have the pixels of the image to vote for the parameter of the shape that we want to detect, for instance, the best parametrization for a line in 2D is $(\rho, \theta) \in \mathbb{R}^+ \times [0, 2\pi[$, where θ is the angle of the direction of the line, and ρ is the distance to the origin. The main advantage of this parametrization over $v = u \cdot a + b$ is that there is no singularity on vertical lines. With (ρ, θ), the line equation is given by:

$$\rho = u \cdot cos(\theta) + v \cdot sin(\theta) \tag{2.40}$$

Since that there are only two parameters, and an inifinite numbers of line passing through a pixel (u, v), each maximum of gradient in the image votes in the parametric space according to the following sinusoidal:

$$\rho_{u,v}(\theta) = u \cdot cos(\theta) + v \cdot sin(\theta) \tag{2.41}$$

When two sinusoidal $\rho_{u,v}$ and $\rho_{u',v'}$ intersects, that means that the pixel (u, v) and (u', v') are on the same segment.

2.4.4 Pyramid

A pyramid is a multi-scale representation of an image, it contains the same image at different scale. [Adelson et al., 1984] contains an overview of the different method to construct a pyramid: in the gaussian pyramid, the level G_0 contains the original image, and to constuct level G_{k+1} from level G_k a low-pass filter (for instance (2.36)) is applied on G_k and then the result is resized:

$$G_{k+1} = REDUCE(K \otimes G_k) \tag{2.42}$$

And the laplacian pyramid is constructed from computing a difference of the gaussian between two consecutive levels, the bottom level L_n is equal to the bottom level G_n of the gaussian pyramid, then the level L_k is equal to the difference between G_k and a scaled version of G_{k+1}:

$$L_k = G_k - EXPAND(G_{k+1}) \tag{2.43}$$

Chapter 3

Data association

Once objects are detected in the environment, it is necesserary to track them between frames, and to match what the robot is currently detecting with the models it has in memory.

When constructing a model of the environment, a false positive during the data association process can have devastating effects on the resulting model. Imagine a robot on a circular trajectory, and that two opposite points of that circle are matched and considered identical, instead of a circle the trajectory become two circles. But at the same time, if the matching algorithm is too selective, the number of good matches become too law, which hinders their exploitation.

In other words, the main challenge of data association is the balance between detecting the greatest number of correct matches while avoiding false positive.

The data association process can use signal information (like the pixels surrounding an interest point) and geometric information (position and size of objects), in the first section of this chapter, we list the tools and information available to compute a match (section 3.1), then we explain the problem of the loop closure (section 3.2) and we finish by giving some of the solution of a special case of data association: tracking (section 3.3).

3.1 Features matching

As mentioned in section 2.2.1, a landmark contains two parts, geometric information and signal information. Using the signal information and a correlation measure (section 2.3.2), it is possible to match the landmark with part of the signal that is currently observed. First, features are extracted from the current data set, and then each features is compared individually to the features in the database.

Since comparing all the descriptors with the content of the database can be a very slow process when the database grows, in the case of vision, when the signal descriptor used is an image patch (a small image around an interest point), it is possible to use a kd-tree as proposed in [Lepetit and Fua, 2006], to reduce the number of descriptor comparisons.

But individual matches isn't sufficient, since data is usually very repeatitive, especially in human made environment. There is a need to do a global match, and to define constraints on the relative position of the matches, the literature contains several method to solve that problem:

- The RANSAC [Fischler and Bolles, 1981] algorithm is a statistic method that uses a voting procedure to define a transformation between features in the observation set and the ones in the database. In a first step, the algorithms select a random number of features in the observation set, and using the individual matches, it computes a transformation, then the algorithm counts the number of matches that validate that transformation. This process is repeated multiple times, the final set of matches is determined by the transformation with the most votes.

- The model of the sensor can be used to set constraints on the position of the features in the signal, for instance, the epipolar geometry of a stereovision bench ([Hartley and Zisserman, 2000] and figure 2.5).
- Another possibility is to use the geometric constraint during the matching, in [Jung and Lacroix, 2001] Jung introduces the idea of a seed-and-growth algorithm, a graph of neighboring features is generated, then an initial hypothesis is found by comparing one observed feature with the features in the database, then the algorithm walks into the graph to find other matches, a more detailed version of that algorithm can be found in section 4.3 for matching points in the image plane, while a version extend to the 3D has been used for facets matching in section 5.2.1.

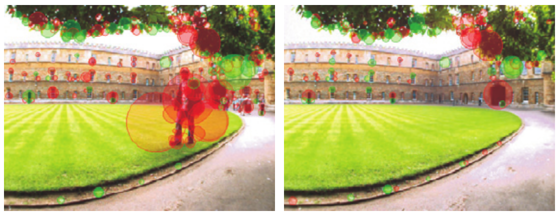
3.2 Loop closure

The loop closure is a specific problem of the SLAM context, it is the process of recognizing that the current location of the robot has already being mapped. In other words, it is an algorithm that detects that two parts of the map are the same in the real world. This is a complex problem, and different types of algorithms have been created to solve that problem, either by comparing two data sets, or projecting a model on a data set, or by comparing data to the content of the map and lastly by map-to-map matching.

3.2.1 Data-to-data

The first possibility to detect a loop closure is two compare the current output of the sensors, with the output at a previous location. For instance, the robots store the images of past locations, the images are indexed in a database, using interest points [Mikolajczyk and Schmid, 2001], a possible speed up can be done using image histograms like in [Gonzalez-Barbosa and Lacroix, 2002] and then to use an interest point matching algorithm such as presented in the chapter 4. This kind of loop closure detection is improved by the use of panoramic images, since the robot will be able to recognize a location independently of its orientation. The main issue with that solution is the need to store a large amount of data.

To reduce the amount of data used for the *image-to-image* matching, it is possible to use a dictionnary of features. In [Cummins and Newman, 2008], [Cummins and Newman, 2009], Cummins uses a fixed dictionnary of SIFT points [Lowe, 2004], for each image, a set of SIFT points is extracted in the image and compared to the dictionnary, this allows to associate each image with a probability that the image contains a given word. To reduce false positives, in [Cummins and Newman, 2008], statistics on whether words are often seen together in the image are computed, this is usefull in case of a repeating pattern, for instance on a brick wall, the same pattern is repeated multiple times, and looking at two pictures of different brick wall would lead to a false match. But in [Cummins and Newman, 2008], the dictionnary needs to be learned offline, and [Angeli et al., 2008] extended the idea with the use of a dictionnary learned inline, they also use epipolar geometry to discard false matches.


(a) Fabmap

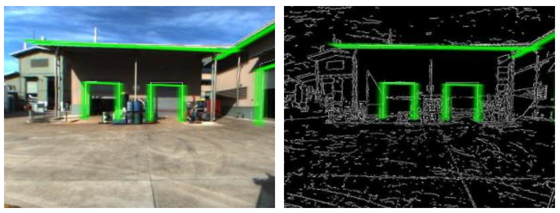

(b) Projection of a model

Figure 3.1: The two images of figure 3.1(a) are excerpts from [Cummins and Newman, 2008] and show a loop closure with scene changes, while figure 3.1(b) are excerpts from [Nuske et al., 2009] and show the projection of a model, green lines, on an image.

3.2.2 Model-to-data

It is also possible to match a model on the data, like in [Nuske et al., 2009], using approximate knowledge of the position of the robot, a wireframe model of the environment is projected on the image plane, then with a nearest-edge search [Deriche and Faugeras, 1990] the parameters of the segment are estimated, which gives a match between the wireframe model and the image.

3.2.3 Map-to-map

Almost all SLAM process will use gating to check that an observation is compatible with a model. Usually, the gating is done using the Mahalanobis distance, which allows to measure the similarity between two data sets, given the $x^o = (x^o_0, ...x^o_n)$ an observation of covariance $xCov^o$ and $x^f = (x^f_0, ...x^f_n)$ the feature of the model of covariance $xCov^f$, the Mahalanobis distance is:

$$D_M = \sqrt{(x^o - x^f)^T \cdot (xCov^f + xCov^o)^{-1} \cdot (x^o - x^f)} \tag{3.1}$$

While it is possible to use the gating of individual features to compute matches, using the *individual compatibility neirghest neighbor* algorithm, which simply match features that are the most compatible. But this doesn't guarantee that two matching are jointly compatible, this is the problem addressed in [Neira and Tardos, 2001] with *joint compatibility maps*, with the idea to make an initial individual match, then to complete that match with other matches that are also jointly compatible, if it has been established that $H = (E_i, F_j)...$ is a set of matches that are compatible among themselves then a new match (E_k, F_l) will be added to H if and only if it is compatible with all the other matches.

In [Bailey, 2002], Bailey proposes the use of a *combined constraint data association*, where all features of the map are connected in a graph, each edge contains the transformation information between features. To close the loop, the correspondance grah is constructed, the nodes of this graph contain all possible matches, and the edges are the compatible transformations created using the graph of features. The set of matches is determined by the maximum clique extracted from this graph.

3.2.4 Data-to-Map

Data-to-map is a loop closure method that uses both information coming from signal comparison, and from joint compatibility matching. In [Williams et al., 2008], during the map building a graph of which features has been observed at the same time is constructed and signal information (such as an image patch) is recorded. Then this graph is used to validate the output of a *map-to-map* algorithm, to make sure the geometric features do indeed match the signal data.

3.3 Tracking

When a good estimate of the position of the feature is available, a tracking method is an efficient way to compute data association, since it allows to avoid the detection step. One drawback of tracking is the danger of having the feature position drift in the data, but for some type of features (such as segments), tracking is the only way to collect data frame after frame.

The goal of the tracker is to find the displacement $\delta(u, t, \tau)$ between the previous position u_p of the the feature and the real position u of the feature in the signal (see figure 3.2), such as:

$$D(u, t + \tau) = D(u + \delta(u, t, \tau), t) \tag{3.2}$$

The offset $\delta(u, t, \tau)$ is computed using an optimisation process, initialised using the prediction of the position of the feature.

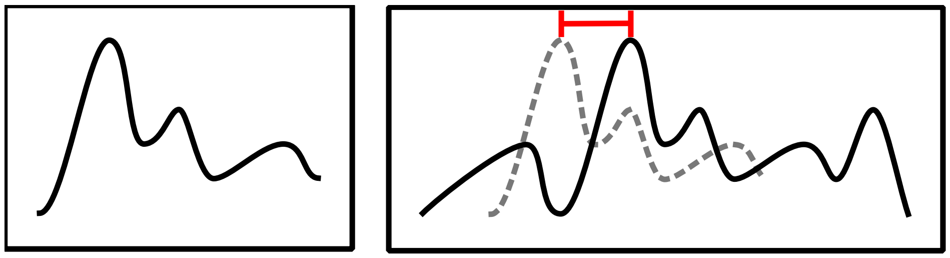


Figure 3.2: Tracking of signal, on the left the pattern representing the signal, on the right the signal at the current instant, the red offset will have to be corrected by the tracking process

Offset approximation In [Shi and Tomasi, 1994b], it is suggested that the displacement for an image can be approximated by an *affine motion field*, such as:

$$\delta(u, v) = D \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \mathbf{d} \tag{3.3}$$

$$\text{Where: } D = \begin{bmatrix} d_{uu} & d_{uv} \\ d_{vu} & d_{vv} \end{bmatrix} \text{ and } \mathbf{d} = \begin{bmatrix} d_u \\ d_v \end{bmatrix} \tag{3.4}$$

Where D is the deformation matrix, and $\mathbf{d}$ is equal to the translation of the feature window's center. $\mathbf{d}$ is also the value of interest in a feature tracking process. This is an approximation that works if there are very small changes between the current image and the original template, this is why the descriptor of the feature is updated at every frame, but this can cause a feature drift.

Another approximation of the offset using an homography H:

$$\delta(u,v) = \begin{bmatrix} \frac{\delta_u}{\delta_c} \\ \frac{\delta_v}{\delta_c} \end{bmatrix} \tag{3.5}$$

$$\text{Where: } \begin{bmatrix} \delta_u \\ \delta_v \\ \delta_c \end{bmatrix} = s \cdot H \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \text{ and } s \cdot H = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} \\ h_{2,1} & h_{2,2} & h_{2,3} \\ h_{3,1} & h_{3,2} & h_{3,3} \end{bmatrix} \tag{3.6}$$

The rank of homography matrix H is equal to 8, which means H is known up to a scale coefficient s (often defined such as $(s \cdot H)(3,3) = 1.0$). The interest of using an homography is that the window corresponding to a plan, can be transformed by an homography between two different view points.

Image alignement The goal of image alignment is to find the parameters of the function $\delta(\mathbf{x}, t, \tau)$, assuming the current image is I, and the template of the feature is T, the image alignment process will need to minimize the following function:

$$\sum_x \left[I(W(\mathbf{x})) - T(\mathbf{x})\right]^2 \tag{3.7}$$

$$\text{Where: } W(\mathbf{x}) = \mathbf{x} + \delta(\mathbf{x}, t, \tau) \tag{3.8}$$

The KLT tracker featured in [Shi and Tomasi, 1994b] presents a method in the case of the affine motion field (equation 3.3), [Baker and Matthews, 2004] contains an overview of all the methods to compute the minimization of equation 3.7, among them, the two most intersting algorithms in the case of the homography approximation are "Inverse Compositional Estimation" (ICE, [Baker and Matthews, 2001]) and "Efficient Second-order Minimisation" (ESM, [Malis, 2004]).

Part II

Landmarks models

Landmarks models play naturally a critical role in the definition of a SLAM solution: a good landmark must be salient in the data, it should be easily detected, tracked and matched from different points of view. And all these processes must allow the estimate of relative positions estimation, with an associated error model.

Any solution to the image feature matching problem calls for three steps [Brown, 1992]: definition of the feature space, definition of a similarity measure over the feature space, and match search strategy. The definition of the features to match is of course essential, as it conditions the whole process. Features can be directly the image signal, or edges, contours, lines, regions detected on the image, up to higher level semantic information. The literature contains many contributions on matching methods based on local gray values similarity scores [Shi and Tomasi, 1994b,Zabih and Woodfill, 1994,Martin and Crowley, 1995]. But in order to generate reliable matches, these approaches require to focus the match search (*e.g.* assuming the transformation between the two images is close to identity or known, so that the epipolar constraint can be applied). To establish matches when several unknown changes occur in the image, one must consider features that are as much invariant as possible with respect to any image transformation.

In this part, we present three different landmark models, and the associated algorithms to detect, track and match them in images. Chapter 4 deals with the detection and matching of points. In chapter 5, points are extended to small planar facets, and chapter 6 presents the extraction and tracking of segments.

Chapter 4

Interest points

Point features, often denoted as "interest points", are salient in images, have good invariant properties, and can be extracted with a small computational cost. A comparison of various interest points detectors is presented in [Schmid et al., 1998] and [Mikolajczyk and Schmid, 2004]: it introduces a modified version of the Harris detector [Harris and Stephens, 1988b] which uses Gaussian functions to compute the two-dimensional image derivatives, and that gives a better *repeatability* under rotation and scale changes (the repeatability being defined as the percentage of repeated interest points between two images).

4.1 Interest point detection

In this section we present some of the most classical algorithms used for point extraction, along with a point similarity measure as well as a descriptor ([Mikolajczyk and Schmid, 2005]). The point similarity is used to compute a quick comparison between points, while the descriptor is used in the confirmation step of the matching algorithm. If the comparison of descriptor is cheap to compute, it can be used as a point similarity measure.

4.1.1 Harris points

To locate points in the image where the signal changes bi-directionally, the Harris corner detector computes the local moment matrix M of two normal gradients of intensity for each pixel $\mathbf{x} = (u, v)$ in the image [Harris and Stephens, 1988b]:

$$M(\mathbf{x}, \tilde{\sigma}) = G(\mathbf{x}, \tilde{\sigma}) \otimes \begin{pmatrix} \mathcal{I}_u(\mathbf{x})^2 & \mathcal{I}_u(\mathbf{x})\mathcal{I}_v(\mathbf{x}) \\ \mathcal{I}_u(\mathbf{x})\mathcal{I}_v(\mathbf{x}) & \mathcal{I}_v(\mathbf{x})^2 \end{pmatrix} \tag{4.1}$$

where $G(., \tilde{\sigma})$ is the Gaussian kernel of standard deviation $\tilde{\sigma}$, and $\mathcal{I}_u(.)$ and $\mathcal{I}_v(.)$ are the first order derivatives of the intensity respectively in the u and v directions. The eigenvalues (λ_1, λ_2) of $M(\mathbf{x}, \tilde{\sigma})$ are the principal curvatures of the auto-correlation function: the pixels for which they are locally maximum are declared as interest points. It has been shown in [Schmid et al., 1998] that interest points are more stable when the derivatives are computed by convolving the image with Gaussian derivatives (see section 2.4.1):

$$\mathcal{I}_u(\mathbf{x}, \sigma) = G_u(\mathbf{x}, \sigma) \otimes \mathcal{I}(\mathbf{x})$$

$$\mathcal{I}_v(\mathbf{x}, \sigma) = G_v(\mathbf{x}, \sigma) \otimes \mathcal{I}(\mathbf{x})$$

where $G_u(.,\sigma), G_v(.,\sigma)$ are the first order derivatives of the Gaussian kernel of standard deviation σ along the u and v directions. The auto-correlation matrix is then denoted $M(\mathbf{x},\sigma,\tilde{\sigma})$. Note that to maintain the derivatives stable with respect to the image scale change s, the Gaussian functions can be normalised with respect to s – the auto-correlation matrix is then $M(\mathbf{x},\sigma,\tilde{\sigma},s)$ [Dufournaud et al., 2004].

Point similarity. If the geometric transformation $\mathcal{T}$ between two images $\mathcal{I}$ and $\mathcal{I}'$ is strictly equal to a scale change s and rotation change θ, the following equality is satisfied for two matching points $(\mathbf{x},\mathbf{x}')$ in the images:

$$\begin{pmatrix} \mathcal{I}_u(\mathbf{x},\sigma,\theta) \\ \mathcal{I}_v(\mathbf{x},\sigma,\theta) \end{pmatrix} = R(\theta) \begin{pmatrix} \mathcal{I}_u(\mathbf{x},\sigma) \\ \mathcal{I}_v(\mathbf{x},\sigma) \end{pmatrix} = \begin{pmatrix} \mathcal{I}'_{u'}(\mathbf{x}',s\sigma) \\ \mathcal{I}'_{v'}(\mathbf{x}',s\sigma) \end{pmatrix}$$

where $R(\theta)$ is the rotation and $\mathcal{I}_u(\mathbf{x},\sigma,\theta)$ and $\mathcal{I}_v(\mathbf{x},\sigma,\theta)$ are the steered Gaussian derivatives of the image in the direction θ [Freeman and Adelson, 1991]. As a consequence, we can write:

$$R(\theta)M(\mathbf{x},\sigma,\tilde{\sigma})R(\theta)^T = M(\mathbf{x}',\sigma,\tilde{\sigma},s)$$

Since

$$M(\mathbf{x},\sigma,\tilde{\sigma}) = U \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} U^T$$

and

$$M(\mathbf{x}',\sigma,\tilde{\sigma},s) = U' \begin{pmatrix} \lambda'_1 & 0 \\ 0 & \lambda'_2 \end{pmatrix} U'^T$$

where the columns of U and U' are the eigenvectors. The principal curvatures of the two matched points are therefore equal: $\lambda_1 = \lambda'_1$ and $\lambda_2 = \lambda'_2$.

For two matching points in two images of real 3D scenes, this equality is of course not strictly verified, because of signal noise, and especially because the true transformation of the image is seldom strictly equal to a rotation and scale change. We define the *point similarity* $\mathcal{S}_p$ between two interest points on the basis of their eigenvalues and their intensity:

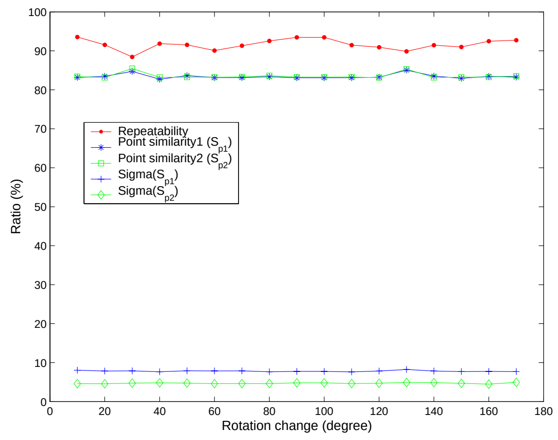
$$\mathcal{S}_P(\mathbf{x},\mathbf{x}') = \frac{\mathcal{S}_{p1}(\mathbf{x},\mathbf{x}') + \mathcal{S}_{p2}(\mathbf{x},\mathbf{x}') + \mathcal{S}_{pI}(\mathbf{x},\mathbf{x}')}{3}$$

where

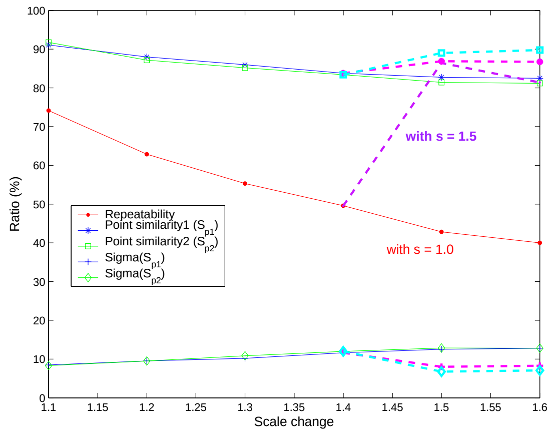
$$\mathcal{S}_{p1}(\mathbf{x},\mathbf{x}') = \frac{min(\lambda_1,\lambda'_1)}{max(\lambda_1,\lambda'_1)},\ \mathcal{S}_{p2}(\mathbf{x},\mathbf{x}') = \frac{min(\lambda_2,\lambda'_2)}{max(\lambda_2,\lambda'_2)},\ \text{and}\ \mathcal{S}_{pI}(\mathbf{x},\mathbf{x}') = \frac{min(\mathcal{I}(\mathbf{x}),\mathcal{I}'(\mathbf{x}'))}{max(\mathcal{I}(\mathbf{x}),\mathcal{I}'(\mathbf{x}'))}$$

The maximum similarity is 1.0. Statistics show that the evolution of $\mathcal{S}_{p1}$ and $\mathcal{S}_{p2}$ for matched points is almost not affected by rotation and scale changes, and is always larger than 0.8 (figure 4.1).

As shown on figure 4.1(b), the repeatability steeply decreases with significant scale changes: in such cases, a scale adaptive version of the Harris detector is required to allow point matching [Dufournaud et al., 2004]. When no information on scale change is available, matching features becomes quite time consuming, scale being an additional dimension to search through.



(a) Evolution of similarity with rotation change



(b) Evolution of similarity with scale change

Figure 4.1: Evolution of the mean and standard deviation of repeated points similarity with known rotation (4.1(a)) and scale changes (4.1(b)). On the right curve, the solid lines represent the evolution of the similarity when the scale of the detector is set to 1: the similarity decreases when the scale change between the images increases. The dashed lines show the values of the similarity when the scale of the detector is set to 1.5: the similarity is then closer to one when the actual scale change between the images is of the same order.

Descriptor The descriptor used for Harris point is a window of the image $\mathcal{I}$ centred around the interest point. To compare the descriptor of two interest point p_1 and p_2, a zero-mean normalised correlation score (ZNCC) is used (as shown in section 2.3.2).

$$ZNCC(p_1, p_2) = \frac{1}{card(W)} \sum_{u,v \in W} \frac{(\mathcal{I}_{p_1}(u,v) - \mu_{\mathcal{I}_{p_1}}) \cdot (\mathcal{I}_{p_2}(u,v) - \mu_{\mathcal{I}_{p_2}})}{\sigma_{\mathcal{I}_{p_1}} \cdot \sigma_{\mathcal{I}_{p_2}}} \tag{4.2}$$

Where $\mathcal{I}_{p_1}$ and $\mathcal{I}_{p_2}$ is the luminance around p_1 and p_2, $\mu_{\mathcal{I}_{p_1}}$ and $\mu_{\mathcal{I}_{p_1}}$ are the mean of the pixels of the image patch, and $\sigma_{\mathcal{I}_{p_1}}$ and $\sigma_{\mathcal{I}_{p_2}}$ are the standard deviation of the pixels, $card(W)$ is the number of pixels of the window W.

If $ZNCC(p_1, p_2) = 1.0$, then the two image patches are identical. The ZNCC score also has the interesting property of decreasing quickly when the data set are slightly different. It is also invariant to affine change in illumination.

4.1.2 Scale-invariant point

Scale-space theory A solution to the problem of the scale selection is to apply the scale-space theory ([Witkin, 1983] or [Lindeberg, 1991]), a framework that represents the different scales of a signal as a function of a parameter t, called *scale parameter*. An overview of the scale-space theory applied to computer vision can be found in [Lindeberg, 2009]: a pyramid is computed from an image, where each level of the pyramid corresponds to a different scale of the image (Section 2.4.4). But Lindeberg has shown in [Lindeberg, 1994] that the Gaussian kernel and its derivatives are the only possible smoothing kernels for a scale space representation.

In the case of interest point detection, the idea is to select all pixels that are a maximum or a minimum of a function F, across the pyramid, the scale that maximises the function is called *characteristic scale*. A pixel (x, y) at scale t is selected as an interest point if the pixel fulfils the following conditions:

$$\forall i, j \in -1, 0, 1^2, F(x+i, y+j, t-1) < F(x, y, t) \tag{4.3}$$

$$F(x+i, y+j, t+1) < F(x, y, t) \tag{4.4}$$

$$\forall i, j \in \{-1, 1\}^2, F(x+i, y+j, t) < F(x, y, t) \tag{4.5}$$

In [Lindeberg, 1998], Lindeberg has shown that a good function for interest point detection is the *scale normalised Laplacian of Gaussian*: $\sigma^2 \nabla^2 G$.

An extension of the Harris detector to the scale-space theory has been proposed in [Mikolajczyk and Schmid, 2002].

SIFT David Lowe introduces ([Lowe, 1999], [Lowe, 2004]) a scale-invariant detector, where the scale normalised Laplacian is approximated by the *difference of Gaussian*:

$$D(x, y, \sigma) = L(x, y, k\dot{\sigma}) - L(x, y, \sigma) \tag{4.6}$$

In [Lowe, 1999], a method to quickly compute the difference of Gaussian is explained. In [Lowe, 2004], Lowe mentioned that images usually contains a large number of maximum and minimum of the difference of Gaussian, but that a careful selection of the scale-space frequencies allows to select the most stable ones, then followed by a check on the eigenvalues of the Hessian matrix.

Lowe also proposes a descriptor, in a first step the orientation of the interest point is computed, then the window around the point is divided in four zones, in each pixel of each zone, the gradient is computed at the characteristic scale and then projected on eight orientations, this makes four histograms. This descriptor is invariant to illumination change, to scale change and to orientation change.

SURF In [Bay et al., 2008] Bay introduces another type of scale invariant interest point detector, it exploits an approximation of the Hessian matrix [Lindeberg, 1998], using integral images. An interest point is selected when it maximises the determinant of the Hessian matrix in the scale space:

$$H(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{4.7}$$

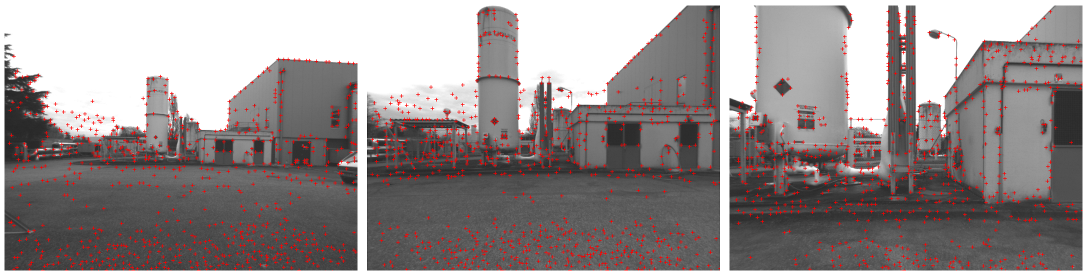
Like for the SIFT descriptor, the orientation of the interest point is computed, then the response to a Haar wavelet decomposition is computed.

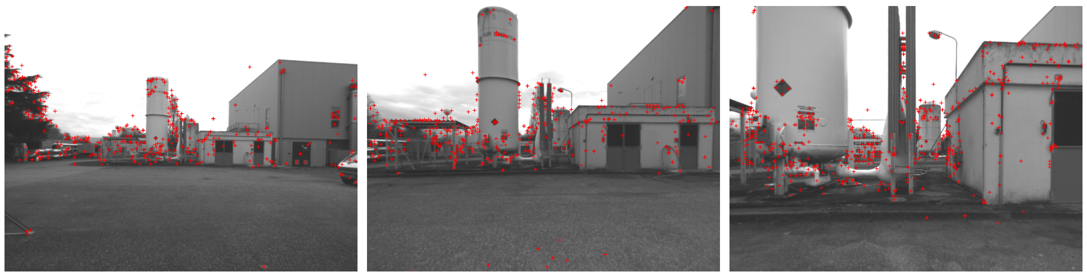
SIFT and SURF descriptors can be used as point similarity for the matching algorithm. Figure 4.2 show the result of extractions for Harris, Sift and Surf.

4.2 Tracking interest points

When a good estimation of the position of interest point between two images is known, for instance for two consecutive images in a video flow, it is possible to focus the data association problem as presented in section 3.3.

Between two images $\mathcal{I}_1$ and $\mathcal{I}_2$, assuming $\mathcal{T}_{1 \to 2}$ is the transformation between the two images, the most simple way to track interest point between $\mathcal{I}_1$ and $\mathcal{I}_2$ is to extract the


(a) Harris points


(b) Sift points

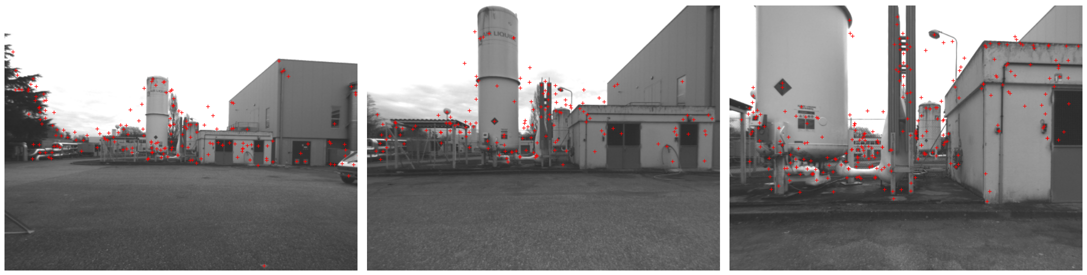

(c) Surf points

Figure 4.2: Extraction of interest point at different distance.

interest points $IP_1 = \{\mathbf{x}_1\}$ and $IP_2 = \{\mathbf{x}_2\}$ in each image, and then using the transformation $\mathcal{T}_{1\rightarrow 2}$, for each interest point of $\mathbf{x}_1 \in IP_1$, the list of track candidate in IP_2 can be computed, by selecting the points around $\mathcal{T}_{1\rightarrow 2}(\mathbf{x}_1)$. The descriptors can then be used to select the tracked point.

Other techniques do not require to compute an interest point detection of $\mathcal{I}_2$, and instead compute a deformation of the local image between the interest point detected in $\mathcal{I}_1$ and their prediction in $\mathcal{I}_2$ using $\mathcal{T}_{1\rightarrow 2}$, either using an affine deformation model [Shi and Tomasi, 1994b] or an homographic deformation model [Malis, 2004] and [Baker and Matthews, 2001].

4.3 Matching interest points

To match interest points, a cross-correlation measure of the signal can be used [Lhuillier and Quan, 2003], but this requires a precise knowledge of the search area. To cope with this, local grey value invariants can be used, as in [Schmid and Mohr, 1997]. The approach we propose here imposes a combination of geometric and signal similarity constraints, thus being more robust than approaches solely based on point signal characteristics (a simpler version of this algorithm has been presented in [Jung and Lacroix, 2001]). It relies on *interest point group* matching: an interest point group is a small set of neighbouring interest points, that represents a small region of the image. With groups composed of a small number of interest points, the corresponding region is small enough to ensure that a simple rotation θ approximates fairly well the actual region transformation between the images – the translation being ignored here. The estimate of this rotation is essential in the algorithm, as a group match hypothesis (*i.e.* a small set of point matches) is assessed on both the signal similarity between interest points and the point matches compliance with the rotation. The matching procedure is a seed-and-grow algorithm initiated by a reliable group match (see Algorithm 2).

Algorithm 2 Overview of the interest point matching algorithm

```
Given two images $\mathcal{I}$ and $\mathcal{I}'$:

 1. Extract the interest points $\{\mathbf{x}\}$ and $\{\mathbf{x}'\}$ in both images

 2. In both images, establish the groups of extracted points. This defines two sets of groups
    $\mathbf{G} = \{\mathcal{G}\}$ and $\mathbf{G}' = \{\mathcal{G}'\}$, and the neighbourhood relations establish a graph between
    the detected points – this procedure is depicted in section 4.3.1.

 3. **While** $\mathbf{G} \neq \emptyset$:

    (a) Establish an initial group match $\mathcal{M}(\mathcal{G}_i, \mathcal{G}'_j)$, which defines a rotation $\theta_{i,j}$, and
        remove $\mathcal{G}_i$ from $\mathbf{G}$ – this procedure is depicted in section 4.3.2.
    (b) Recursive propagation: starting from the neighbors of $\mathcal{G}_i$, explore the neighboring
        points to find group matches compliant with $\theta_{i,j}$. Remove the new matched groups
        from $\mathbf{G}$, and iterate until no matches compliant with $\theta_{i,j}$ can be found – this
        procedure is depicted in section 4.3.3.

 4. Check the validity of the propagated group matches
```

4.3.1 Grouping process

The sets of interest points $\{\mathbf{x}\}, \{\mathbf{x}'\}$ detected respectively in the two images $\mathcal{I}, \mathcal{I}'$ are structured in *local groups*, formed by a pivot point $\mathbf{g}_0$ and its n closest neighbors $\{\mathbf{g}_1, \ldots, \mathbf{g}_n\}$ (figure 4.3). To ensure that the image region covered by the points of a group is small enough, n is rather small (*e.g.* we use $n = 5$). The groups are generated by studying the neighborhood of each point following a spiral pattern: the grouping process is stopped if the spiral meets the image border before n neighbors are found. Also, a maximum threshold on the distance between the neighbor points and the pivot is applied, to avoid the formation of groups that cover a too large image region (in low textured areas for instance, where there are scarce interest points). This implies that a few points do not belong to any group: their matching is processed individually (see section 4.3.3).

After the grouping process, we end up with two group sets $\mathbf{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_N\}$ and $\mathbf{G}' = \{\mathcal{G}'_1, \ldots, \mathcal{G}'_M\}$, $\mathcal{G}_i$ denoting the local group generated with the point $\mathbf{x}_i$ as a pivot:

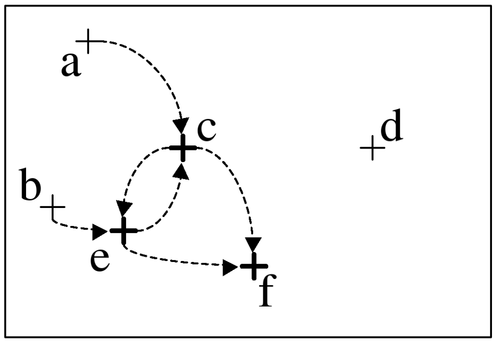


Figure 4.3: Illustration of the point grouping procedure, with $n = 2$ for readability purposes. Groups have not been generated around points **a** and **b** as they are too close to the image border, and neither around **d** as no neighbor is close enough. Three groups have been generated, with points **c**, **e** and **f** as a pivot. $\mathbf{b} \rightarrow \mathbf{e}$ means "**b** is a neighbor of **e**", which defines a graph relation between the points.

$$\mathcal{G}_i = \{\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_n\}, \ \mathbf{g}_0 = \mathbf{x}_i$$

The neighbors $\{\mathbf{g}_1, \ldots, \mathbf{g}_n\}$ are ordered by their distance to the pivot:

$$\| \mathbf{v}_1 \| < \cdots < \| \mathbf{v}_n \|$$

where the vectors $\mathbf{v}_i$ are defined as $\mathbf{v}_i = \mathbf{g}_i - \mathbf{g}_0$ and $\| \, . \, \|$ is the norm operator. For each neighbor of the group, we also compute its angle, defined as:

$$\theta_{\mathbf{g}_p} = tan^{-1}\left(\frac{\mathbf{v}_p \cdot \mathbf{v}}{\mathbf{v}_p \cdot \mathbf{u}}\right)$$

where $\mathbf{u}$ and $\mathbf{v}$ are the image axes.

4.3.2 Group matching

The procedure to establish a group match is essential in our approach: in particular, a wrong initial group match would cause the algorithm to fail. The procedure consists in three steps depicted in the next paragraphs:

1. Given a group $\mathcal{G}_i$ in $\mathcal{I}$ with $\mathbf{g}_0$ as a pivot, all the groups $\mathcal{G}'_j$ in $\mathcal{I}'$ whose pivot $\mathbf{g}'_0$ is *similar* to $\mathbf{g}_0$ are candidate group matches.

2. For each candidate group match $\mathcal{G}'_j$, determine all the *group match hypotheses* $H(\mathcal{G}_i, \mathcal{G}'_j)$ on the basis of the possible individual neighbor matches, and select the best one $H^*(\mathcal{G}_i, \mathcal{G}'_j)$.

3. Select the best group match among all the $H^*(\mathcal{G}_i, \mathcal{G}'_j)$, and apply a validation criteria.

Point similarity. Two points $\mathbf{x}, \mathbf{x}'$ are defined as *similar* if their similarity measure is above a threshold $T_{\mathcal{S}_p}$:

$$\mathcal{S}_P(\mathbf{x}, \mathbf{x}') > T_{\mathcal{S}_P}$$

This test is used to asses the similarity of points in steps 1 and 2 of the group matching procedure.

Building group match hypotheses. Given two groups $(\mathcal{G}_i, \mathcal{G}'_j)$ whose pivots have passed the point similarity test, one must evaluate all the possible associated group match hypotheses, *i.e.* the various combinations of matches between the neighbor points of the groups. A group match hypothesis $H(\mathcal{G}_i, \mathcal{G}'_j)$ is defined as:

- a rotation θ
- a set $M(\mathcal{G}_i, \mathcal{G}'_j)$ of interest point matches which respect the rotation θ and whose similarity score is above the threshold $T_{\mathcal{S}_P}$:

$$M(\mathcal{G}_i, \mathcal{G}'_j) = \{(\mathbf{g}_p, \mathbf{g}'_q) \in \mathcal{G}_i \times \mathcal{G}'_j \mid \mathcal{S}_P(\mathbf{g}_p, \mathbf{g}'_q) > T_{\mathcal{S}_P} \text{ and } |\theta_{\mathbf{g}_p} - \theta_{\mathbf{g}'_q}| < T_\theta\} \cup \{(\mathbf{g}_0, \mathbf{g}'_0)\}$$

- a group hypothesis similarity score $\mathcal{S}_G$, defined as the sum of the similarity of the corresponding matched interest points:

$$\mathcal{S}_G(H(\mathcal{G}_i, \mathcal{G}'_j)) = \sum_{(\mathbf{g}_p, \mathbf{g}'_q) \in M(\mathcal{G}_i, \mathcal{G}'_j)} \mathcal{S}_P(\mathbf{g}_p, \mathbf{g}'_q)$$

The best group match hypothesis among all the ones that can be defined on the basis of two candidate groups $(\mathcal{G}_i, \mathcal{G}'_j)$ is determined according to Algorithm 3: this provides the best group match hypothesis H^*, if it exists, between $\mathcal{G}_i$ and $\mathcal{G}'_j$. Note in this procedure the introduction of a threshold $\Delta_{\mathcal{S}_G}$ in the comparison of hypotheses, to ensure that the best hypotheses has a much better score than the second best: this is useful to avoid wrong group matches for images with repetitive patterns, in which many points are similar.

Selection of the best group match hypothesis. Now that we have determined the best group match hypothesis H^* for each candidate group match $\mathcal{G}'_j$ for the group $\mathcal{G}_i$, one must determine the one that actually corresponds to a true group match. This is simply done by comparing their similarity $\mathcal{S}_G$, applying the same threshold $\Delta_{\mathcal{S}_G}$ as above to make sure the best match is not ambiguous.

Finally, the validity of the found group match is confirmed by comparing the descriptors of the pivots of $(\mathbf{g}_0, \mathbf{g}'_0)$, when the orientation of the point isn't associated to the descriptor (like the image patch of 4.1.1), then the knowledge of the rotation θ defined by the group match hypothesis can be used to compute a more accurate score. For instance, for the descriptor used for Harris points, θ is used to compute a rotation of the image patch.

Algorithm 3 Determination of the best match hypothesis for two groups

```
• Init: $\mathcal{S}_G^* = 0$
• For $p = 1$ to $|\mathcal{G}_i|$, For $q = 1$ to $|\mathcal{G}'_j|$:
• if $\mathcal{S}_P(\mathbf{g}_p, \mathbf{g}'_q) > T_{\mathcal{S}_P}$ then create and evaluate a group match hypothesis $H_{p,q}(\mathcal{G}_i, \mathcal{G}'_j)$:
    – Set $M(\mathcal{G}_i, \mathcal{G}'_j) = \{(\mathbf{g}_0, \mathbf{g}'_0), (\mathbf{g}_p, \mathbf{g}'_q)\}$. This defines the rotation $\theta$ for this hypothesis: $\theta = \theta_{\mathbf{g}_p} - \theta_{\mathbf{g}'_q}$
    – complete $M(\mathcal{G}_i, \mathcal{G}'_j)$ with the other points in $\mathcal{G}_i$ that are similar to points in $\mathcal{G}'_j$, such that:
          $\forall s > p \text{ and } t > q,\ \mathcal{S}_P(\mathbf{g}_s, \mathbf{g}'_t) < T_{\mathcal{S}_P} \text{ and } |\theta - (\theta_{\mathbf{g}_s} - \theta_{\mathbf{g}'_t})| < T_\theta$
      Note here that the fact that the neighbours are ordered by their distance to the pivot reduces the search for additional point matches – see figure 4.4.
    – Evaluate the hypothesis $H_{p,q}(\mathcal{G}_i, \mathcal{G}'_j)$:
      if $\mathcal{S}_G(H_{p,q}(\mathcal{G}_i, \mathcal{G}'_j)) > \mathcal{S}_G^* + \Delta_{\mathcal{S}_G}$, then $H^* = H_{p,q}(\mathcal{G}_i, \mathcal{G}'_j)$ and $\mathcal{S}_G^* = \mathcal{S}_G(H^*)$.
```

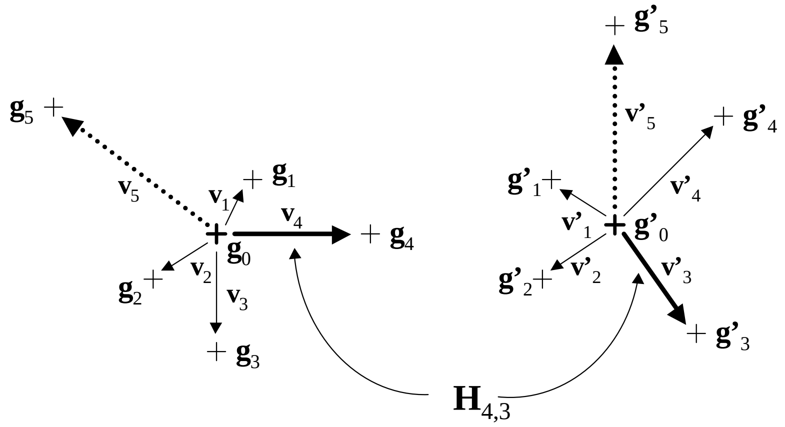


Figure 4.4: Completion of a group match hypothesis. Given the hypothesis $H_{4,3}$ defined by the point matches $(\mathbf{g}_0, \mathbf{g}'_0)$ and $(\mathbf{g}_4, \mathbf{g}'_3)$, the best potential match for $\mathbf{g}_5$ is determined by evaluating geometric and point similarity constraints. The indexes of the neighbors being ordered according to their distance to the pivot, only the matches $(\mathbf{g}_5, \mathbf{g}'_4)$, and $(\mathbf{g}_5, \mathbf{g}'_5)$ are evaluated – on this example, $(\mathbf{g}_5, \mathbf{g}'_5)$ is the sole valid match.

4.3.3 Finding further matches

Propagation process. Once a reliable group match hypothesis is established, a propagation process searches for new matches. The principle of the propagation is to exploit the graph defined by the grouping process and the estimated rotation associated to the current hypothesis: additional point matches consistent with the current rotation estimate are searched in the neighbourhood of the current group match. This process is depicted in Algorithm 4 .

Algorithm 4 Propagation process

```
Given a group match $(\mathcal{G}_i, \mathcal{G}'_j)$ and the associated rotation $\theta$ :
  • Init: set $M^{propage} = M(\mathcal{G}_i, \mathcal{G}'_j) \setminus \{(\mathbf{g}_0, \mathbf{g}'_0)\}$.
  • While $M^{propage} \neq \emptyset$:
      – Select a point match $(\mathbf{g}_p, \mathbf{g}'_q) \in M^{propage}$. $\mathbf{g}_p$ and $\mathbf{g}'_q$ are respectively the pivots of the groups $\mathcal{G}_p$ and $\mathcal{G}'_q$.
      – For $s = 1$ to $|\mathcal{G}_p|$, For $t = 1$ to $|\mathcal{G}'_q|$:
        if $\mathcal{S}_P(\mathbf{g}_s, \mathbf{g}'_t) < T_{\mathcal{S}_P}$ and $|\theta - (\theta_{\mathbf{g}_s} - \theta_{\mathbf{g}'_t})| < T_\theta$, add $(\mathbf{g}_s, \mathbf{g}'_t)$ to $M^{propage}$
      – Remove $(\mathbf{g}_p, \mathbf{g}'_q)$ from $M^{propage}$
```

During the propagation, the translation between matched points is computed: when the propagation ends, this allow to focus the search for new matches, as illustrated in figure 4.5.

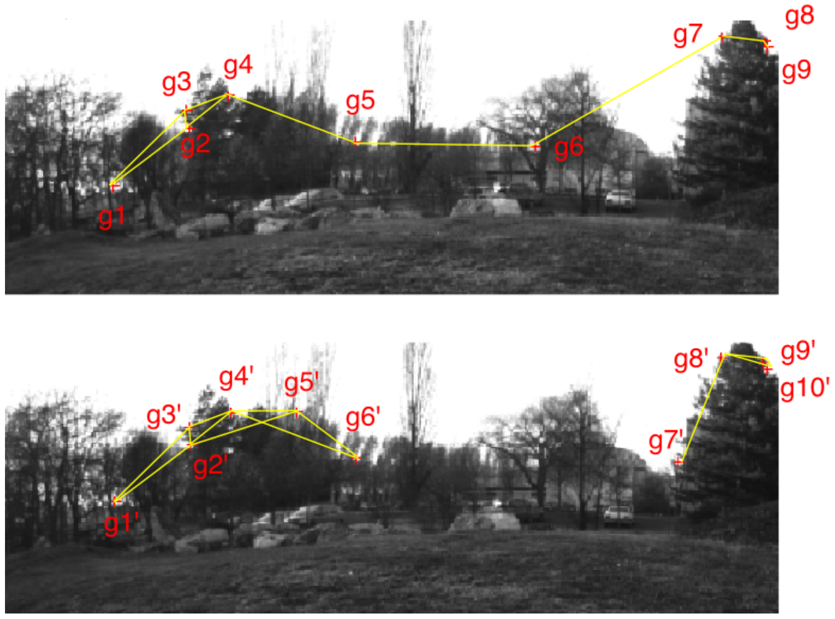


Figure 4.5: Illustration of the propagation process. Red crosses are interest points, yellow lines indicate neighbourhood relations defined by the grouping process. Here, $\mathbf{g}_2$ and $\mathbf{g}'_2$ are the pivots of the initial group hypothesis $H(\mathcal{G}_2, \mathcal{G}'_2)$, and the corresponding list of individual points matches is $M(\mathcal{G}_2, \mathcal{G}'_2) = \{(\mathbf{g}_2, \mathbf{g}'_2), (\mathbf{g}_1, \mathbf{g}'_1), (\mathbf{g}_3, \mathbf{g}'_3), (\mathbf{g}_4, \mathbf{g}'_4)\}$. During the propagation, matches for points neighbouring the ones of $M(\mathcal{G}_2, \mathcal{G}'_2)$ are evaluated – here the match $(\mathbf{g}_5, \mathbf{g}'_6)$ is added and the propagation stops. Thanks to the estimate of the translation between the points matched so far, the group match hypothesis $H(\mathcal{G}_7, \mathcal{G}'_8)$ can be evaluated, and new matches are added for a little computational cost.

Propagation monitoring. Repetitive patterns with a size similar to the group size can lead to false matches, although the initial group match has passed the tests described in section 4.3.2. The occurrence of such cases can be detected by checking whether the propagation process succeeds or not around the first group match: if it fails, it is very likely that the initial group match hypothesis is a wrong one, and it is then discarded (figure 4.6). Note that this test also eliminates group matches if a group is isolated or if the overlap between the two images $\mathcal{I}$ and $\mathcal{I}'$ is restricted to the size of the group: these are degenerated cases in which the algorithm does not match the groups.

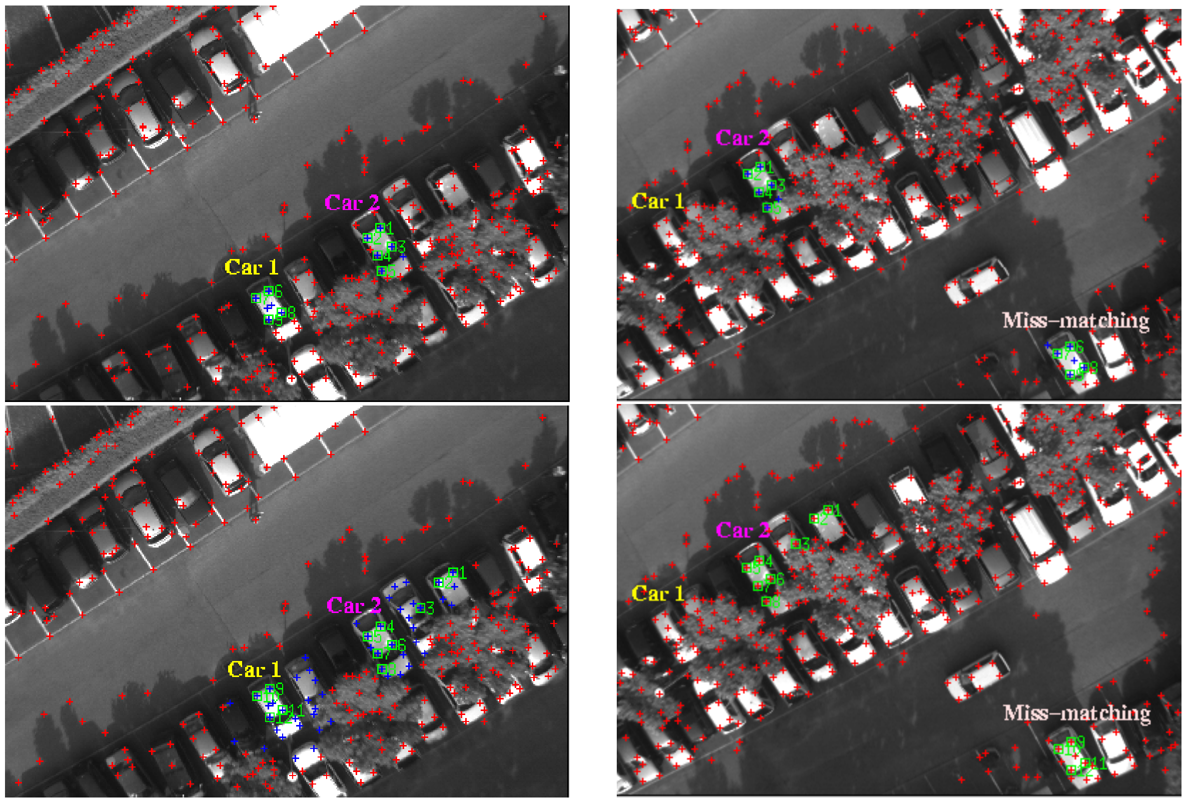


Figure 4.6: Illustration of the propagation monitoring. The top images show two group matches independently established according to the process of section 4.3.2: the “Car 2” group is properly matched, whereas “Car 1” has been incorrectly matched. The bottom images show the additional matches established by the propagation process: no additional matches have been determined around the “Car 1” group, whereas other matches around the “car 2” have been determined: the “Car 1” group match is a false one.

Non-grouped points matching. As mentioned in section 4.3.1, some points are not associated to groups after the grouping procedure, mainly near the image borders. Once the propagation procedure is achieved, for each non grouped point $\mathbf{x}_b$ of $\mathcal{I}$, matches are searched among the set of points X_b in the image $\mathcal{I}'$:

$$\mathbf{X}_c = \{\mathbf{x}|\mathbf{x} \in W(\hat{\mathbf{x}}'_b)\}$$

where $\hat{\mathbf{x}}'_b$ is the estimated position of $\mathbf{x}_b$ in $\mathcal{I}'$ provided by the application of the transformation defined by the mean of the rotations and translations estimated so far, and $W(\hat{\mathbf{x}}'_b)$ is a search window centred on $\hat{\mathbf{x}}'_b$. The points comprised in $W(\hat{\mathbf{x}}'_b)$ are evaluated according to the hypothesis pruning process presented in section 4.3.2: test on the point similarity measure $\mathcal{S}_P$ and verification with the comparison of descriptors.

4.4 Results

The algorithm provides numerous good matches while keeping the number of outliers very small, in different kinds of scenes and in a wide variety of conditions, tolerating noticeable scene modifications and viewpoint changes. Figure 4.7 presents matches obtained in various conditions, with the computational time required for the detection and matching – the processed image size is 512×384, and time measures have been obtained on $3.2GHz$ Pentium IV). The algorithm does not explore various scale changes: when a scale change greater than half a unit occurs, it must be provided as a parameter to the interest point detection routine. This is a limitation as compared to scale invariant point features, but a coarse knowledge of the scale change is sufficient: in a SLAM context, such an estimate is readily obtained.

Maximum group size	5
Minimum group size	2
Size of correlation window for ZNCC	9×9
Threshold $T_{\mathcal{S}_P}$ on point similarity $\mathcal{S}_P$	0.7
Threshold on ZNCC T_{ZNCC}	0.6
Threshold $\Delta_{\mathcal{S}_G}$ to discriminate group hypotheses	0.1
Threshold on rotation change, T_θ	$0.2rad$

Table 4.1: Thresholds and parameters of the matching algorithm.

Table 4.1 lists the parameters required by the matching algorithm and their values. These values are used for all the results presented throughout the thesis, and during our everyday use of the algorithm: no parameter tuning is required.

The first three parameters are used during the grouping process presented section 4.3.1. The *Minimum group size* is naturally set to 2, the minimum size that allows to run the group matches determination presented section 4.3.2. The *Maximum group size* is a compromise: on the one hand, the more members in a group, the more reliable are the group match hypotheses. On the other hand, a big number of points in a group tends to violate the hypothesis that a simple rotation approximates its transformation between the two images: empirical tests show that a value of 5 offers a good balance. Finally, the *Maximum distance between pivot and group member* threshold is set to $3\sqrt{\mathcal{D}}$, where $\mathcal{D}$ is the density of interest points in the image.

The threshold $T_{\mathcal{S}_P}$ on the similarity measure is used to evaluate if two points match: its value is set to 0.7, according to the variations of the point similarities presented in section 4.1.1. The threshold T_{ZNCC} on the correlation score to confirm a point match is set to 0.6, a value smaller than usually used for this score (*e.g.* in dense stereovision): this is due to the fact that a rotation is imposed to one of the correlation window before computing the score, which smooths the signal in the window, and also to the fact that we aim at matching points seen from different viewpoints. Finally, the threshold on rotation change T_θ is set to $0.2rad$, a

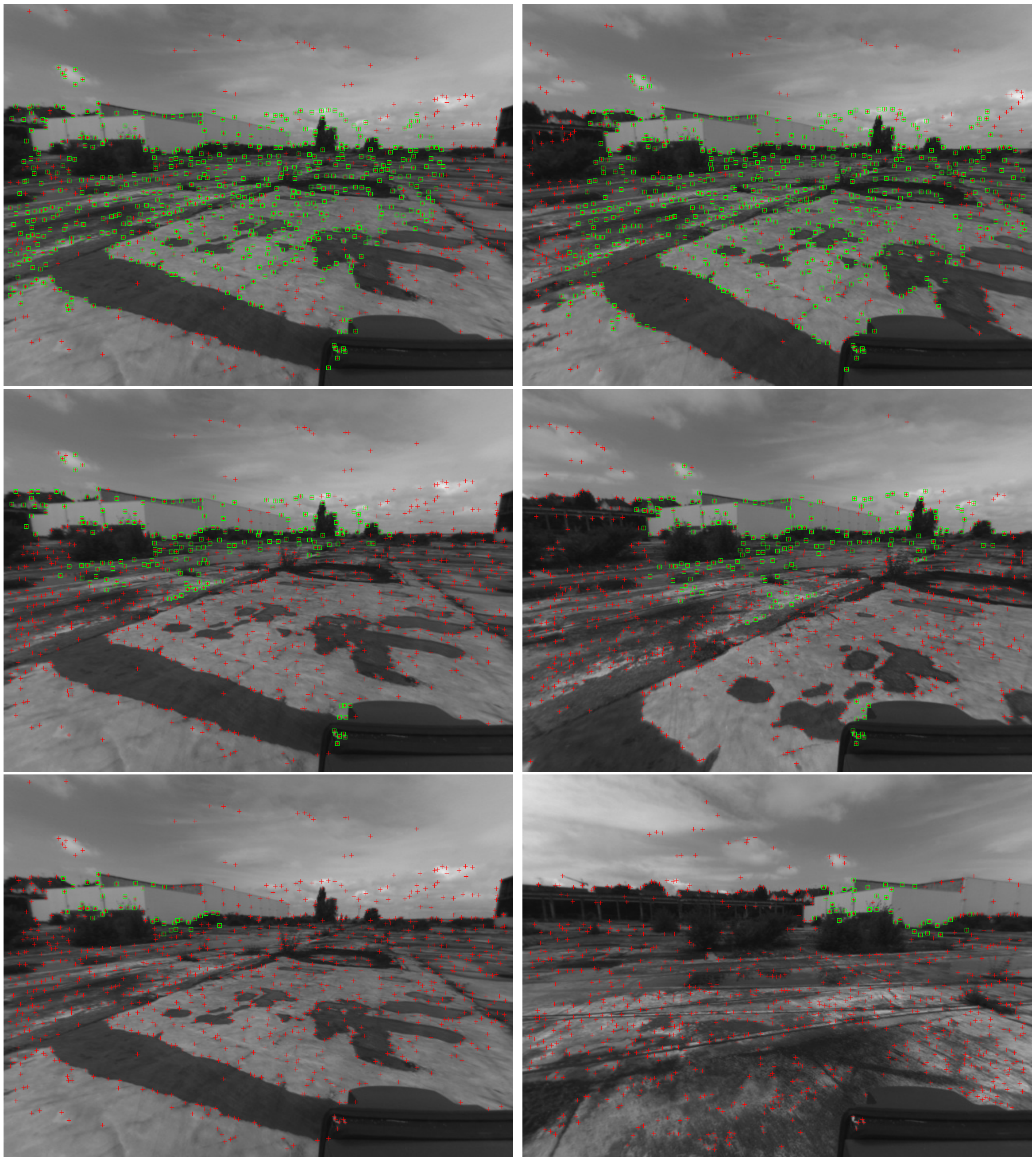


Figure 4.7: Points matched with an increasing change in viewpoint, red crosses show the interest points, and green squares indicates successful matches. There are 619 matches between the first image, 299 and then 43.

quite large value that is necessary to cope with the errors on the interest points detection, that can reach at most 1.5 pixel [Schmid et al., 1998].

Chapter 5

Facets

In this chapter, we propose a landmark model based on planar patches, called "facets" detected using stereovision ([Berger and Lacroix, 2008]). Relying on interest points, this model contains six geometric parameters and image patch information: this description gives a better observability of the robot position by the perception of a small number of landmarks, as opposed to [Molton et al., 2004], in which facets are only used to ease the matching process, and makes the matching process easier when perceiving a landmark from different view points. Section 5.1 presents this landmark model and the corresponding detection process in a pair of stereoscopic images. Section 5.2 describes tracking and matching algorithms.

Related work There are various contributions that represent the environment with small planar patches. For instance, [Murray and Little, 2005a] presents a method to extract patchlets from stereo images in order to model the perceived surfaces, but do not register multiple views. The main approaches that consider planar surfaces in a SLAM context are the following:

- In [Gee et al., 2007], the authors present a monocular SLAM approach in which 3D points are augmented with normal information. When points are found to be on the same plane, their state vector in the EKF filter is "collapsed", so as to reduce the computational cost.
- In [Chekhlov et al., 2007], the authors use SLAM with point landmarks, and find the planes among the point cloud using a RANSAC process, thus allowing to derive a map with higher level structural information.
- [Castle et al., 2007] presents an approach that recognizes known planar objects that have been previously modelled and stored in a data base.
- In [Silveira et al., 2007], the authors present a method to detect and track larger planar patches in the environment using a monocular camera.

It is worth to notice that besides [Silveira et al., 2007], these contributions deal with the problem in small confined environments.

5.1 Planar facets detection

Facets correspond to planar areas detected around interest points, by checking whether an homography between their two stereoscopic views can be fitted or not.

5.1.1 Facet model

A facet is a set of geometric properties that represent its position and orientation, completed by signal information. Figure 5.1 shows an example of facets extracted from a pair of stereoscopic images.

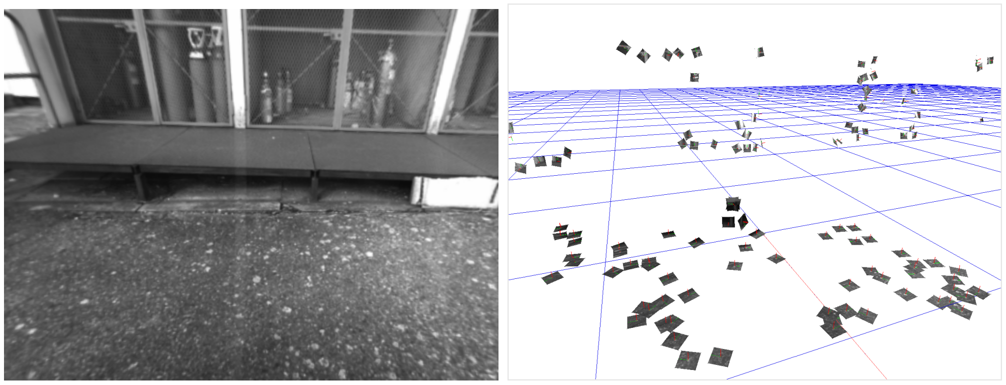


Figure 5.1: Left image of a stereoscopic image pair, and extracted facets.

Two equivalent geometric models are defined:

- A matrix representation of the position and orientation of the facet (12 parameters: the facet centre, plus the 3 vectors of the associated frame)
- A minimal representation (six Euler parameters)

The matrix representation is used to compute comparisons and transformations during detection and matching, whereas the Euler angles are used for the SLAM estimation.

To simplify the matching process, facets correspond to a constant size of planar patches in the environment (we typically use a size of 10×10 centimetres), and the associated image patch is stored in a fixed size image (25×25 pixels in our implementation).

5.1.2 Facets extraction

Interest point detection A facet can be associated to a Harris point or to scale invariants points – the later offer a better repeatability, at the expense of a much higher computation time.

Homography estimation Dense pixel stereovision could be used to estimate the normal vector of the surface corresponding to an interest point, with a least square plane fitting algorithm applied to the neighbouring 3D points. But fast stereovision algorithms yields noisy coordinates of the 3D points, which make the estimation of the normal very unstable.

An approach based on the homography estimation is more robust and reliable. The two image projections I_1 and I_2 of a plane P corresponding to different viewpoints are linked by a homography $s \cdot H$, where H is a 3x3 matrix, and s is an unknown scale factor (often defined such as $(s \cdot H)(3,3) = 1.0$). So two image patches I_1^p and I_2^p extracted from I_1 and I_2 correspond to a planar patch in the environment if there is a matrix H that satisfies:

$$\mathcal{T}(H, I_2^p) = I_1^p \tag{5.1}$$

Where $\mathcal{T}(H, I)$ is the image resulting from the transformation applied to the image I using the homography H.

Alignment algorithms which compute the value of H are optimisation procedures whose goal is to minimise:

$$E = \mathcal{T}(H, I_2^p) - I_1^p - (\mu(\mathcal{T}(H, I_2^p)) - \mu(I_1^p)) \tag{5.2}$$

Where $\mu(\mathcal{T}(H, I_2^p))$ and $\mu(I_1^p)$ are the mean of the pixels of $\mathcal{T}(H, I_2^p)$ and I_1^p, which reduce the influence of lightning change between two images.

[Baker and Matthews, 2001] provides an analysis of various alignment algorithms, and also introduce the "Inverse Compositional Estimation" (ICE) method for homography estimation. [Malis, 2004] introduce the "Efficient Second-order Minimisation" (ESM) used for tracking planar areas using an homography estimation.

For small image areas, both methods are able to estimate an homography which either precisely corresponds to the associated plane or is totally erroneous. Experimental trials show that when an erroneous homography is estimated, the resulting normal is completely unpredictable and not reproductible: those cases can therefore be identified by analysing successive observations (see 5.2.3). Figure 5.2 shows some evaluations of the two algorithms on synthetic images. It appears that ICE gives more facets but with a bigger error, while ESM tracks less facets, but is more accurate.

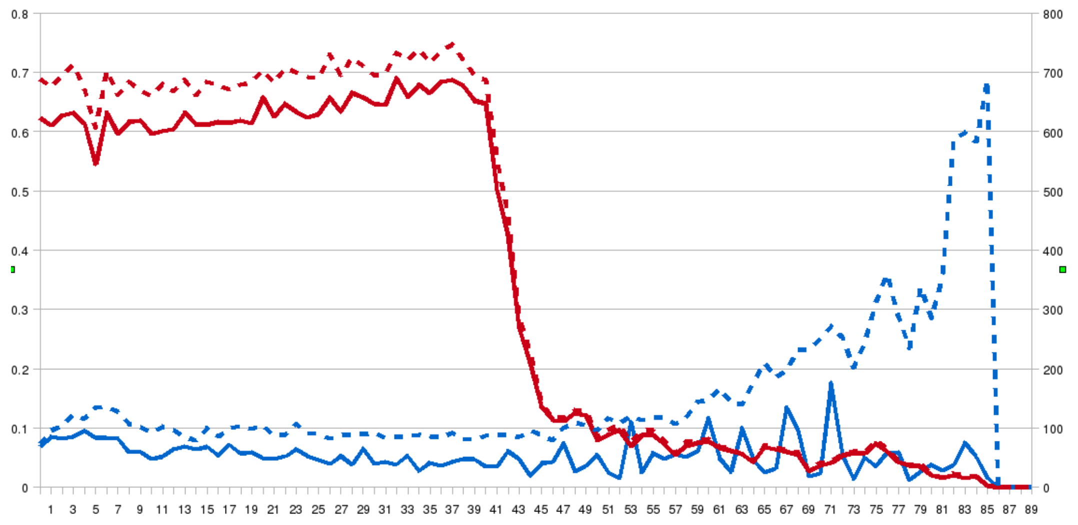


Figure 5.2: Comparison of the ICE (dash lines) and ESM (solid lines) algorithms. A plane textured with a real image is rotated in front of the camera: the plot shows the estimated normal error (left y-axis, blue lines) and the number of detected facets (right y-axis, red line), as a function of the plane orientation with respect to the camera. The collapse of the facet numbers around 40° is due to the fact that the interest point matching algorithm can hardly match points under large viewpoint changes.

Normal estimation Once the homography is computed, the normal of the facet is computed using the geometric parameters of the stereovision bench – *e.g.* by computing the coordinates of three points of the plane using the homography.

Completing the facet orientation information The facet orientation is defined by three vectors: it is only necessary to compute two of them, the third one being the result of their cross product. The first vector is the normal vector, and the second vector is computed using the image patch of the facet, so as to represent its orientation: the gradient is computed on each pixel P of a square window W around the interest point IP, using Sobel masks. The facet orientation is then defined as the following weighted sum:

$$Orientation = \frac{\sum_{P \in W} w(d(P, IP) \cdot atan2(Gy(P), Gx(P))}{\sum_{P \in W} w(d(P, IP))} \tag{5.3}$$

Where $d(P, IP)$ is the distance between the pixel P and the interest point IP and $w(x)$ is a Gaussian weighting function.

Unfortunately, despite the decrease of sensitivity to noise and to viewpoint changes brought by the weighted sum, the orientation is either very stable (in most cases) or very unpredictable and not reproductible. As for the computation of homography, facets whose orientation is not stable can be eliminated by analysing successive observations (see 5.2.3). In our convention, this orientation is the third Euler angle of the facet ("roll", denoted w).

The orientation is a reliable information for the basis, since the physical size (in real world coordinates) of the image patch is constant and the perspective of the image patch of window W is corrected using the normal.

5.1.3 Image patch

The image patch of a facet F is interpolated from the image of the camera, using the geometric properties of the facet. Each point p_t of the image patch correspond to a 3D point $P \in F$, this point P is then projected on a pixel p_c of the camera.

Let $\mathcal{P}_{Camera}$ the projection matrix of a point in the environment on the focal plane of the camera, OF the vector from the origin of the world to the centre of the facet F, and v and w, the orientation vectors parallel to the facet plane. Assuming the image patch pixels are indexed from the facet centre by i and j, and given r the resolution of the image patch, the following equation gives the value for each pixel of image patch as shown figure 5.3 :

$$p_t(i, j) = p_c(\mathcal{P}_{Camera}(OF + i \cdot v \cdot r + j \cdot w \cdot r)) \tag{5.4}$$

By applying this interpolation to memorise the facet image patch, it is represented the way it would have been perceived with the camera "aligned" to the facet, *i.e.* with the optical axis parallel to the facet normal, and the horizontal axis aligned to the facet orientation w. Thanks to this representation, during matching, a pixel by pixel comparison of the image patch allows to get a similarity score between the observed image patch and the memorised one. Note that to avoid situations of undersampling, facets which are too far from the robots are not used so that a 10x10cm patch correspond to at least 25x25 pixels.

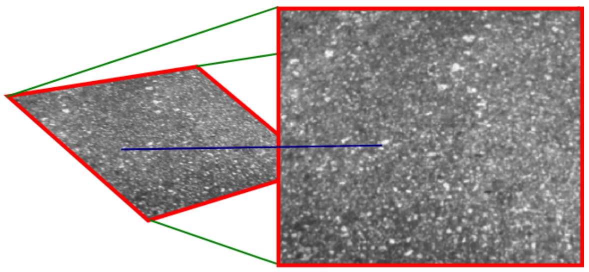


Figure 5.3: Interpolation of the image patch of a facet. The blue line shows an association of an image pixel to a pixel of the memorized image patch.

5.1.4 Error model

The error model for the minimal geometric representation of a facet is composed of covariances of its centre coordinates and of its Euler angles. The centre coordinates and the orientation angles being computed by independent processes, the centre/orientation covariances are equal to 0. Similarly, the facet normal estimate is provided by the homography estimate, and its orientation by an analysis of the image patch: these parameters variances are therefore independent. This yield a covariance matrix with the following form:

$$\begin{bmatrix} M^{stereo}_{[3\times3]} & & 0_{[3\times3]} & \\ & \sigma_u^2 & \sigma_{u/v}^2 & 0 \\ 0^2_{[3\times3]} & \sigma_{v/u}^2 & \sigma_v^2 & 0 \\ & 0 & 0 & \sigma_w^2 \end{bmatrix} \tag{5.5}$$

Where $M^{stereo}_{[3\times3]}$ is the stereovision usual error model [Xiong and Matthies, 1997]. The variance and covariance values for the angles are empirically set as follows: $\sigma_u = \sigma_v = \sigma_w = 0.01rad$ and $\sigma_{u/v} = 0.1rad$.

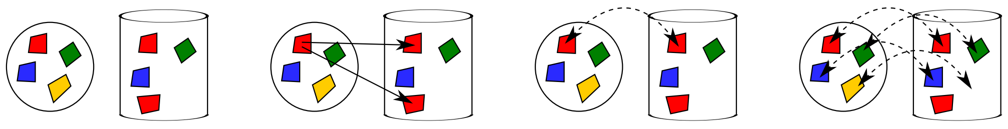
5.2 Facets matching

5.2.1 General Algorithm

The method used for facets matching is an extension to the third dimension of an interest point matching algorithm described in [Jung and Lacroix, 2001]: the idea is to mix signal information with geometric relations between neighbouring facets to assess robust matches.

Let $\mathcal{F}_1$ and $\mathcal{F}_2$ two sets of facets within which we are looking for matches. The algorithm is a hypothesise-and-test procedure: it starts by establishing a first match between a facet from $\mathcal{F}_1$ and one from $\mathcal{F}_2$ using only signal information (figure 5.4(b)). This first match hypothesis gives a geometric transformation $\mathcal{T}_{1\to2}(f)$, which is used to focus the search of additional matches, the establishment of additional matches reinforcing the initial hypothesis.

1. Given $f_1 \in \mathcal{F}_1$, let $f_2 \in \mathcal{F}_2$ the facet whose image patch is the closest to the one of f_1 – in other words, the facet $f \in \mathcal{F}_2$ which maximises $CompareImagePatch(f_1, f)$ where $CompareImagePatch$ is an image patch comparison function (for instance the ZNCC score)



(a) Facets to match (b) Initial match hypothesis (c) Transformation (d) Propagation and confirmation

Figure 5.4: This figure illustrates the three steps of the matching algorithm. The facets in the circle, on the left, is the list of facets detected at given time, while the facets on the right are the facets stored in memory.

2. This first match allows to compute the geometric transformation (figure 5.4(c)) $\mathcal{T}_{1\rightarrow 2}(f)$ such that:
$$\mathcal{T}_{1\rightarrow 2}(f_1) = f_2 \tag{5.6}$$

3. In the last step, to confirm the match we attempt to propagate the match in the neighbourghood of the facets using the geometric transformation (figure 5.4(d)): $\forall f_1' \in \mathcal{F}_1$, if there is $f_2' \in \mathcal{F}_2$ which satisfies the following two conditions:
$$\mathcal{T}_{1\rightarrow 2}(f_1') \approx f_2' \tag{5.7}$$
$$CompareImagePatch(f_1', f_2') > \mathcal{T}_{imagepatch} \tag{5.8}$$
Then the couple (f_1', f_2') is a match.

Figure 5.5 shows two examples of facet matching results.

5.2.2 Facets tracking

One of the advantages of using planar facets is the possibility to re-project them and to predict how a camera will observe them from a different viewpoint. Especially, if the transformation is precisely known, it is very easy to compare the observation with the image patch in memory. This is of a limited interest for SLAM when the change of view point is not very well known – typically when closing a loop. But between t and $t+1$, the estimation of the viewpoint change $T_{t\rightarrow(t+1)}$ provided by the prediction step is precise enough to predict the position and orientation of the facets observed at time t to *track* them.

Let $\mathcal{I}p(I^l_{t+1})$ and $\mathcal{I}p(I^r_{t+1})$ the list of interest points detected at time $t+1$ in the left and right images I^l_{t+1} and I^r_{t+1}, and $\mathcal{F}(t)$ the set of facets detected at time t.

1. $\forall f \in \mathcal{F}(t)$, the projection P^l_f of f on the image I^l_{t+1} is computed

2. Let $\mathcal{C}$ the list of interest points located close to the predicted position of the facet on the image:
$$\mathcal{C} = Ip^l \in \mathcal{I}p(I^l_2), |Ip - P_F| < \epsilon \tag{5.9}$$
Using the motion estimate $T_{t\rightarrow(t+1)}(base)$, it is possible to predict the facet parameters, and especially to use its predicted normal to compute the image patch for each point of

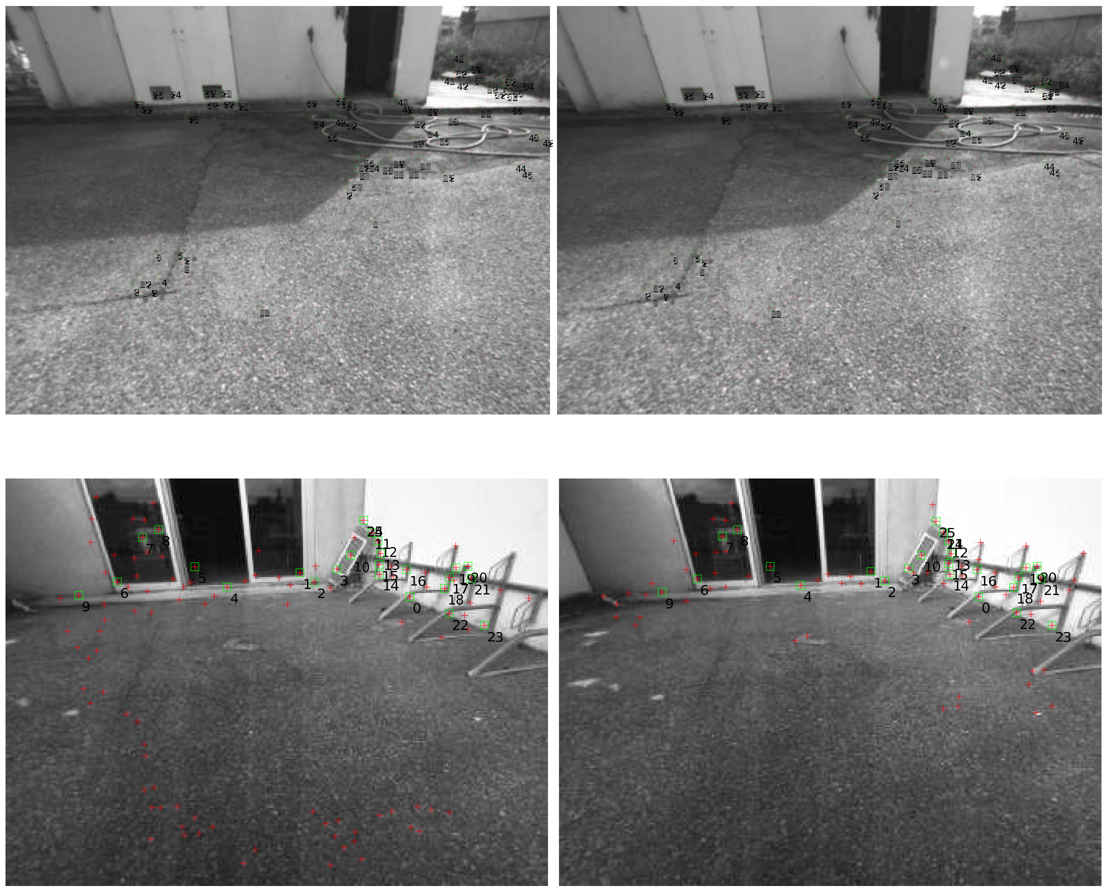


Figure 5.5: Two results of facets matching. Red "+" denote the detected facets, and green numbered squares show the ones that have been matched.

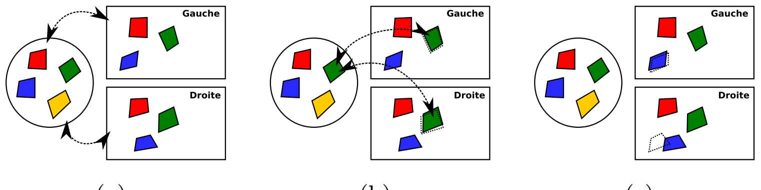


Figure 5.6: This figure illustrates the tracking process, the set of facets detected at $t-1$ (in the circle) is projected on the left and right images.

$\mathcal{C}$ as in section 5.1.3. Let $I_p^l(F) \in \mathcal{C}$ the interest point whose image patch is the closest to the one of the facet.

3. The same method is used to find $Ip^r(F)$ in the right image, with the added constraint that the two interest points must satisfy the epipolar constraint

4. Using the couple (Ip^l, Ip^r), the parameters of the facet f_{track} are computed as in sec-

tion 5.1, this allow to check that $f_{track} = T_{t \rightarrow (t+1)}(f)$

With respect to other tracking methods (such as [Shi and Tomasi, 1994a] or [Malis, 2004]), this approach offers the interest to get a direct control on the facets parameters, the possibility to update their models and to filter out the ones for which an erroneous homography has been estimated, as shown in the following sections. For 200 facets, a tracking step takes 300 ms (including all processing: image rectification, interest point detection and facets tracking), whereas an initial facet detection requires 500ms, and the matching without any prior motion estimate requires a second [1].

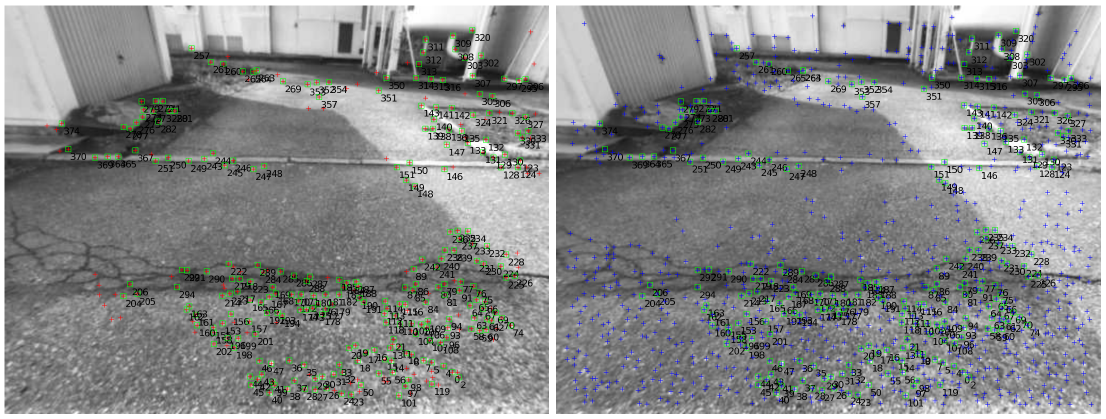


Figure 5.7: Tracked facets in two consecutive images. The red "+" denote detected facets, the blue points are Harris points, and green squares shows tracked facets.

5.2.3 Unreliable facets elimination

After the application of the matching or tracking algorithms, some facets remain unmatched, or their observation is not consistent with the matched facets observation. Such facets correspond either to an interest point with a too small repeatability, or to an erroneous normal or rotation estimate (see section 5.1.2). This can be due to various causes: for instance, if the neighbourhood of an interest point has a weak image patch, this can lead to a wrong homography (a black point on a white wall is a strong interest point, but the resulting homography is very likely to be erroneous).

Unmatchable, untrackable and inconsistent facets are considered to be weak facets, and are simply discarded as illustrated in Figure 5.6(c).

[1]Time measured on a Intel core Duo @ 2GHz using only one thread, on 512×392 images.

Chapter 6

Line Segments

Image line segments carry a lot of information on the perceived scene structure, and are basic primitives on which several vision applications are built on (*e.g.* structured object recognition [David and DeMenthon, 2005], or structured scenes 3D reconstruction in geomatics or robotics [Smith et al., 2006,Lemaire and Lacroix, 2007a]). Figure 6.1 shows, on a same image, how extracting segments give more information about the structure of the environment than points. Line segments have indeed numerous advantageous properties: they of course perfectly represent the straight lines of a scene, they can be used to estimate various geometric transformations, and they are invariant to large viewpoint and scale changes. But this latter property actually only holds if the line segment extraction process is itself robust and stable: classic segment extraction processes remain fragile with respect to image noise and viewpoint changes.

Over the last decade, achievements in the extraction and identification of stable feature points (Harris, SIFT features...) have diminished the interest in line segments. Nevertheless, however stable and robust are they, point primitives lack the structure expressivity of line segments. A stable, robust and fast segment detection and tracking process is therefore desirable, and is still the object of recent contributions [Chen and al, 2007,R. Grompone von Gioi and Randall, 2008].

6.1 Line Segment Detection

6.1.1 Related work

There are two classical approaches to the line segment detection problem, that both rely on the Canny filter [Canny, 1986]. The first approach consists in applying a threshold on the gradient, grouping neighboring local maxima of the gradient into contours, and applying a split and merge line fitting process to define the line segments [Etemadi, 1992]. The second approach originally proposed in [Ballard, 1987] exploits a generalized Hough transform computed on the gradient, and an edge splitting step.

Some other approaches ([Burns et al., 1986], [Steger, 1998], [Desolneux et al., 2000], [R. Grompone von Gioi and Randall, 2008]) relies on detecting a region, and then line segment parameters are fit on the region.

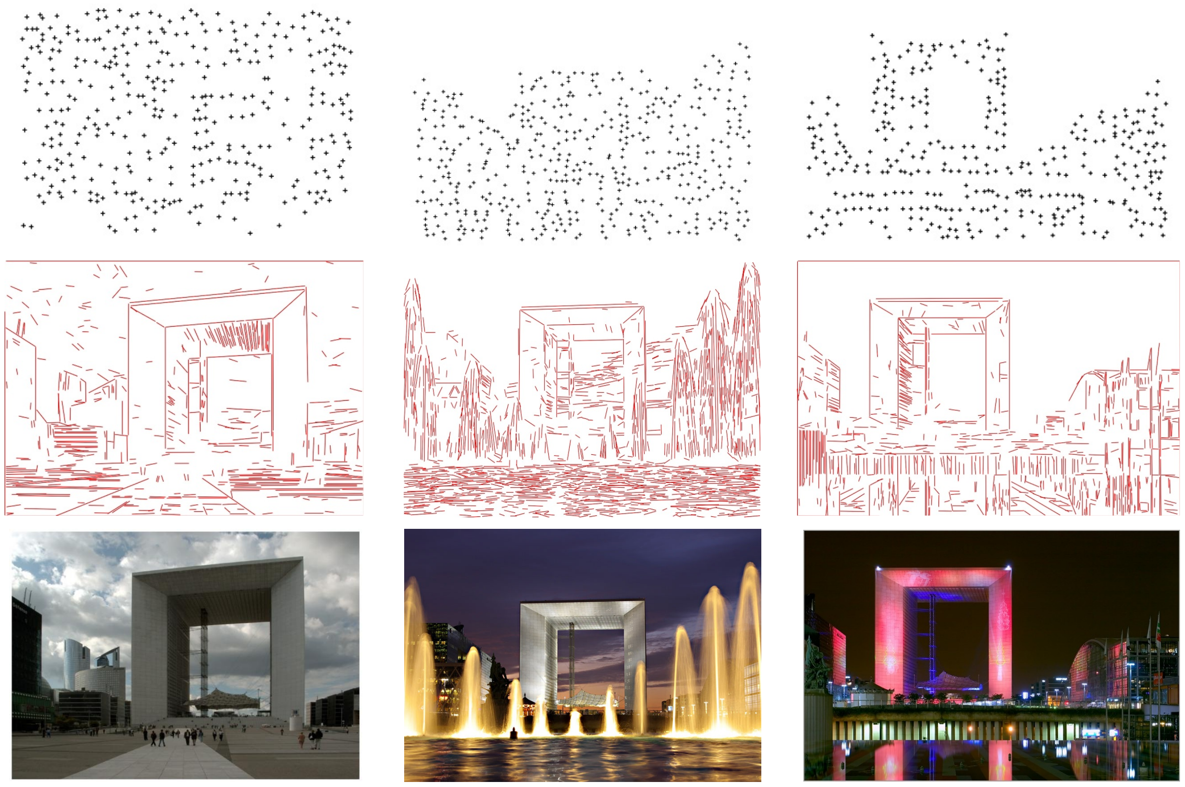


Figure 6.1: Segments and points

Canny/Linking First the Canny filter (see section 2.4.2) is applied on the image, this gives the position of the maxima in the direction of gradient in the image. Then the maxima are linked to their neighbourgh, then the chains are splited [Etemadi, 1992] to create line segments.

Hough The Hough method was introduced in [Hough, 1962], it allows to map an observation space (in our case an image), to a parameter space (segments) using a voting procedure (Section 2.4.3). The Canny filter [Canny, 1986] is used to find the maximums of gradient, then each maximum of gradient vote for a set of segments parameters. Since this method only allows to find the infinite line support, it is needed to apply a spliting algorithm to find line segments in the image. One of the main advantage of the Hough transformation is that it allows to detected any kind of shape in the image.

Many improvements to the method have been introduced over the years. For instance, in [Ballard, 1987] the voting procedure was extend to also use the direction information. While in [Kiryati et al., 1991], random samples of the observations are used to speed up the process, in [Guo et al., 2008], the votes are weigthed to give more importance to pixel with a clear boundary. In [Xu and Velastin, 1994] a Kalman filter is used to improve voting procedure of the Hough transformation, and in [Yacoub and Jolion, 1995] a hierarchical approach has been proposed.

Region based In [Burns et al., 1986], pixels with a similar gradient orientation are grouped, and then line parameters are estimated from the rectangle. In [Desolneux et al., 2000], region hypotheses are generated, then for each region the number of pixels with aligned gradients are counted, if there is a sufficient number the region is considered to be a segment.

Recently, Grompone in [R. Grompone von Gioi and Randall, 2008], proposed a Line Segment Detector (LSD) built upon the work of Burns [Burns et al., 1986] and Desolneux [Desolneux et al., 2000], mixing both approaches, and offering improvements:

1. Like in [Burns et al., 1986], the image is segmented in regions according to a gradient direction criteria. The segmentation is done by a region growing algorithm, a first pixel is selected (based on the gradient strength), the angle of the gradient around that pixel defines the angle θ for the region, then the angle of the gradient in the neighbourgh pixels are compared to that angle θ, and if they are close enough, the pixels are added to the region, after each step, the angle θ is updated to be the mean of all the angles in the region.

2. Regions are then approximated as a rectangle, using the center of mass of the region as the center of the rectangle ([Kahn et al., 1990]), and the first inertia axis is used to select the orientation. The norm of the gradient defines the mass of each pixel.

3. The last step applies the idea of [Desolneux et al., 2000] to the rectangle generated in the previous steps, which adjust the parameters of the rectangle until a linear segment is found.

One of the main improvements of this method over the previous ones is that it does not require the tuning of parameters, it also gives more accurate segments.

Other region based detectors are extracting lines with their width, for instance, in [Steger, 1998], Steger propose a method to extract curvilinear structures, but this does not give directly geometric parameters.

Model-driven All the previous methods are data-driven, as they consist in grouping pixels using data characteristics in a first step, a line segment model being introduced afterwards. In [Mansouri et al., 1987,Chen and al, 2007], a model-driven approach has been proposed, that directly fits a segment on the data, using measures on the gradient to adjust the parameters.

6.1.2 Direct line segment detection (DSeg)

After introducing the parametric model chosen to represent the line segments, this section depicts the detection process, that relies on a seed-and-grow scheme: null length line segments are initialized on the basis of gradient and phase information, and further extended and updated thanks to the Kalman filter. The section ends with an analysis of the influence of the algorithm parameters.

Line model A line is represented in the image frame with the following parametric linear model:

$$x(t) = a \cdot t + x_0 \tag{6.1}$$
$$y(t) = b \cdot t + y_0 \tag{6.2}$$

Where (a, b) is the direction vector of the line and (x_0, y_0) its origin. Although this model is over-parametrized, it brings forth two essential advantages: it can represent any 2D line (no singularity), and the single parameter t allows to specify any point on the line. The fact that this model is not minimal is not an issue for the estimation process, as no constraints link its parameters.

Detection process initialization The main idea of the approach is to fit the line segment model on the image. It would be naturally very time consuming to search for segments around every pixel: the process is therefore only initiated for *seeds*, *i.e.* pixels that correspond to a local gradient maximum.

Let $G_{i,j}$ and $\phi_{i,j}$ the norm and the phase of the gradient on pixel (i,j) computed by a Canny filter. A pixel (i,j) is considered as a seed if it satisfies the following conditions[1]:

$$G_{i,j} - G_{i+cos(\phi_{i,j}),j+sin(\phi_{i,j})} > \tau_{gradmax} \tag{6.3}$$
$$G_{i,j} - G_{i-cos(\phi_{i,j}),j-sin(\phi_{i,j})} > \tau_{gradmax} \tag{6.4}$$
$$G_{i,j} > G_{i-sin(\phi_{i,j}),j+cos(\phi_{i,j})} \tag{6.5}$$
$$G_{i,j} > G_{i+sin(\phi_{i,j}),j-cos(\phi_{i,j})} \tag{6.6}$$

The first two conditions ensure that the considered pixel is likely to belong to a line segment, while the two others state that the considered pixel gradient is a local maximum along the hypothetical line segment, and are mostly needed to give more robust seeds.

Given a seed, the parameters of the line model that initializes the state of the Kalman filter are:

$$x_0 = i, y_0 = j \tag{6.7}$$
$$a = -sin(\phi_{i,j}), b = cos(\phi_{i,j}) \tag{6.8}$$

Line extension process Once an initial point has been found, an iterative extension process is applied: it searches additional support points along the current estimated direction of the line model. This is made according to the following procedure (figure 6.2):

1. $t = 1$
2. $t \leftarrow t \pm \delta_t$, where δ_t is the distance along the line at which new support points are searched.
3. Prediction: using the current estimated parameters of the line and their associated variances, an estimation of the coordinates (x_t, y_t) of the next support point is computed, and scalar error e that represents the error across the line direction is computed.
4. Observation: a set of measures are made along the normal of the current line that intersects (x_t, y_t), and a selection process defines the one that will be incorporated as a support point to update the line model.

The observation selection process is as follows. Let $a' = a/\sqrt{a^2+b^2}$, $b' = b/\sqrt{a^2+b^2}$. $\mathbf{s} = (-b' \cdot e/n_o, a' \cdot e/n_o)^T$ is the direction vector of the search[2], with a norm that depends on

[1]In all the algorithm steps, subpixel values are obtained by bi-cubic interpolation.
[2]Bold notation denotes vectors

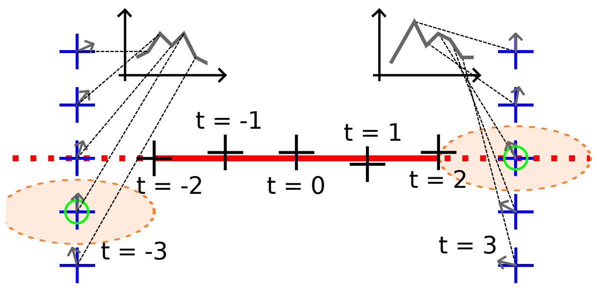


Figure 6.2: Segment extension process. The red line represents the current segment, the black crosses are the current support points. Additional support points are searched at a distance δ_t (here equal to 1), along a direction normal to the current line, within a range cross-distance to the line that depends on the variance on the current segment estimation. The blue crosses are the $2n_o + 1$ measure points that are checked ($n_o = 2$ here): the ones that correspond to a local gradient maxima are considered as *observations* of the current line segments. When there is more than one gradient maxima among the measure points, the closest to the current line is selected as an observation (at $t = 3$ in the picture), and when two local maxima are at an equal distance to the line (at $t = -3$ in the picture), the one with the closest phase angle to the current line orientation is selected. The orange ellipses show the position uncertainty of the two selected observations (see section 6.1.2).

the error e and on n_o, a constant that defines the number of considered measures. The set of measure points is:

$$M = \{(x_t, y_t) + i \cdot \mathbf{s}\}, i \in [-n_o, n_o] \tag{6.9}$$

and the measure $m_i \in M$ that is selected as an observation of the current line is the one that satisfies the following conditions:

- Local gradient maximum:

$$G_{(x_t,y_t)+i\cdot\mathbf{s}} > G_{(x_t,y_t)+(i+1)\cdot\mathbf{s}} \tag{6.10}$$
$$G_{(x_t,y_t)+i\cdot\mathbf{s}} > G_{(x_t,y_t)+(i-1)\cdot\mathbf{s}} \tag{6.11}$$

- Direction gradient compatible with the current line:

$$cos(\phi_{(x_t,y_t)+i\cdot\mathbf{s}} - atan2(a, -b)) > \tau_{angle} \tag{6.12}$$

If two measures m_{i1} and m_{i2} pass these tests, their distance to the line is first checked, and in case of equal distances, the one which the most compatible phase angle with the current line is selected. *e.g.* m_{i1} is selected if $\mid i_1 \mid < \mid i_2 \mid$, and if $\mid i_1 \mid = \mid i_2 \mid$, m_{i1} is selected if:

$$cos(\phi_{(x_t,y_t)+i_1\cdot\mathbf{s}} - atan2(a, -b)) < cos(\phi_{(x_t,y_t)+i_2\cdot\mathbf{s}} - atan2(a, -b)) \tag{6.13}$$

When no further support points are found, the search is extended a step further ahead to avoid spurious segment splits due to image noise: $t \pm \delta_t$, and if no additional observations are then found, the extension process ends.

Line Parameters Estimation

System model The line parameters are estimated by a Kalman filter: they are initialized as in section 6.1.2, and updateded using the observations produced by the line extension process.

The state in the Kalman filter is the set of line parameters: $\mathbf{x_k} = (a, x_0, b, y_0)^T$. Since there is no process noise and the model is stationary, the state equation is:

$$\mathbf{x_k} = \mathbf{x_{k-1}} \tag{6.14}$$

The observation equation is:

$$\mathbf{z_k} = H_k \cdot \mathbf{x_k} + \mathbf{v_k} \tag{6.15}$$

It is a linear observation model, where v_k is the observation noise with covariance P, and H_k the observation matrix:

$$H_k = \begin{bmatrix} t & 1 & 0 & 0 \\ 0 & 0 & t & 1 \end{bmatrix} \tag{6.16}$$

The initial values for the state vector are computed as in section 6.1.2, and the associated covariances are initialized to:

$$P_{0|0} = \begin{bmatrix} \sigma_a^2 & 0 & 0 & 0 \\ 0 & \sigma_{x0}^2 & 0 & 0 \\ 0 & 0 & \sigma_a^2 & 0 \\ 0 & 0 & 0 & \sigma_{y0}^2 \end{bmatrix} \tag{6.17}$$

Observation error. The observation error is defined by two uncorrelated variances in the frame associated to the current line estimate, that are set by the discretisation applied for the line extension process (orange ellipses figure 6.2). The along-track variance is δ_t^2, the cross-track variance is 0.5^2, error of the image discretisation – note that these are conservative values. In the frame associated to the line, the observation error matrix is:

$$\begin{bmatrix} \delta_t^2 & 0 \\ 0 & 0.5^2 \end{bmatrix} \tag{6.18}$$

A frame transformation is applied to obtain the observation error matrix $S_k$in the image reference frame:

$$S_k = R \times \begin{bmatrix} \delta_t^2 & 0 \\ 0 & 0.5^2 \end{bmatrix} \times R^T \tag{6.19}$$

where $R = \begin{bmatrix} cos(\alpha) & -sin(\alpha) \\ sin(\alpha) & cos(\alpha) \end{bmatrix}$ is the rotation matrix between the line frame and the image reference frame, α being the current estimated angle of the line in the image reference frame.

We now have all the parameters necessary to the application of the linear Kalman filter after each new observation (support point) provided by the line extension process.

Line merging After the detection process, it can occur that the newly detected line segment S_n overlap with a previously detected segment S_p (figure 6.3). In such cases, a merging process is applied: a chi-square test is applied to check whether the extremities of S_p can be considered as observations (support points) compatible with S_n. If yes, S_n and S_p are merged, and the parameters of S_n are updated with the new observations by iterating the Kalman filter.

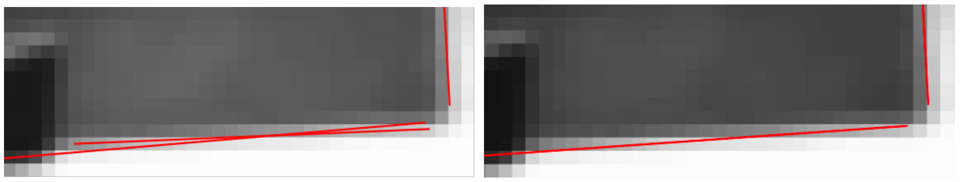


Figure 6.3: Line merging process. Left, two overlapping segments. Right, the line resulting from the merging process.

Parameters The detection process involves the definition of some parameters and thresholds. Large conservative values for these parameters are easy to specify, as they do not impact the detection results. Table 6.1 lists the parameters and their values – used in all our trials and for the results shown in the next section.

$\sigma_a = \sigma_b$	$\sigma_{x0} = \sigma_{y0}$	δ_t	τ_{angle}	$\tau_{gradmax}$	n_o
0.05	1.0	1px	1.0 - σ_a	10	2

Table 6.1: Parameters required and associated values

The two parameters that are defined empirically are $\sigma_a = \sigma_b$ and $\tau_{gradmax}$, since $\sigma_{x0} = \sigma_{y0}$ indicates how well the position of the local maximum of a gradient is known, and the resolution of the image is known up to one pixel. Figure 6.4 show the lack of influence of $\tau_{gradmax}$ and τ_{angle} on the number of detected segments. $\tau_{gradmax}$ determines whether low contrast segments will be extracted or not. For τ_{angle}, we choose a quite large value (0.95, $acos(0.95) = 31°$), so that the process is not disturbed by noise and still ignore drastic change in orientation.

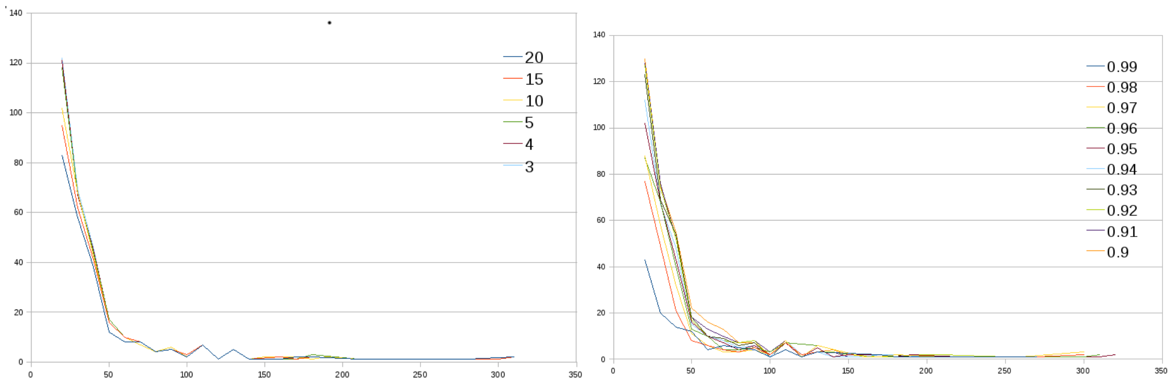
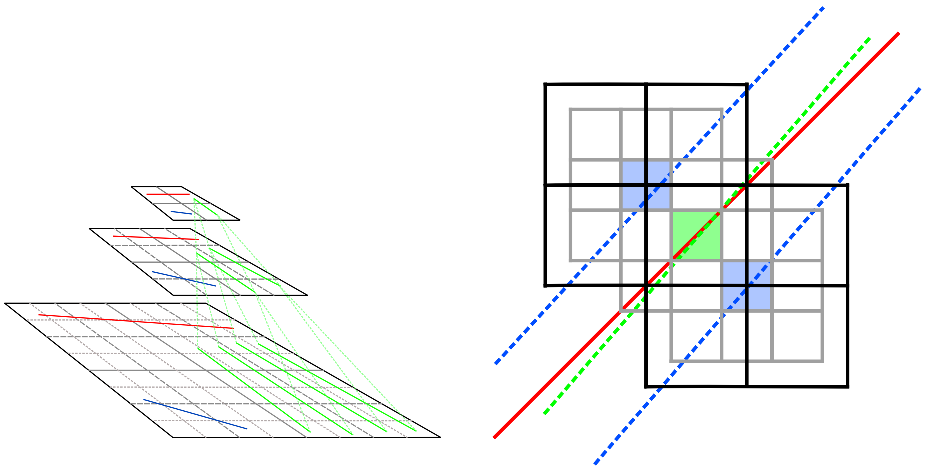


Figure 6.4: Distribution of the detected segment lengths for various values of $\tau_{gradmax}$ (left) and τ_{angle} (right).

Results Finally figure 6.7 present some results obtained with *DSeg*.

6.1.3 Hierarchical direct segment detection

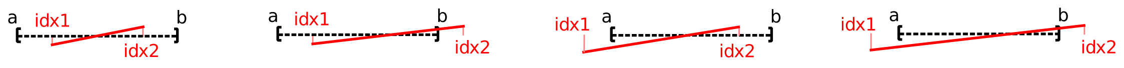

(a) Detection on multiple levels.

(b) The red segment is detected at the coarser level, at the next level, it can correspond to either the green segment, or to one of the two blue segments, or both of them.

Figure 6.5: Hierarchical direct segment detection

In this section we present an extension of the previous algorithm, where segments are first detected on a scaled down image, and then their position is tracked on the different levels of the pyramid (Figure 6.5). The main interest is that scaling down the image reduces the noise level, but it also gives less precise segments, therefore, with a hierachical approach, segments are initialized using less noisy data, while retaining the precision given by using the full size image.

Because scaling down is emulating a distance change, another interest is to only detect segments that are likely to be visible on a broader range, when the robots move backwards, the segments are more likely to be stable with respect to scale change.



(a) $[a, idx1]$, $[idx1, b]$ (b) $[a, idx1]$ (c) $[idx2, b]$ (d) $\emptyset$

Figure 6.6: These figures show how intervals are splitted after a detection of a segment. $idx1$ and $idx2$ are respectively the distance of the first and second extremity of the newly detected segment to the origin (which is defined as one of the extremity of the segment detected at a upper level). One situation is not shown here, when $idx2 < a$ or $b < idx1$, in which case the interval $[a, b]$ is not changed.

Detection process From the image, a n levels pyramid is computed (see Section 2.4.4). The level $level_{i=0}$ correspond to the bottom of the pyramid with the full size image, while $level_{i=n-1}$ is the level with the smallest image;

1. On the top $level_{i=n-1}$, segments are detected as in DSeg (6.1.2)
2. Given a segment seg_j^i detected at $level_i$, its parameters at $level_{i-1}$ are obtained by multiplying by two the parameters at $level_i$: $seg_j^{i-1} = 2 \cdot seg_j^i$. Given $O_j^i(x_{i-1}, y_{i-1})$ the coordinates of one of the extremities, $\mathbf{u}$ the vector director of the segment, $\mathbf{v}$ is the vector orthogonal to $\mathbf{u}$, and l the length of the segment, we initialize a list of intervals $L = \{[0, l]\}$.
   (a) Given an interval $I = [a, b]$ from L, look for a maximum of gradient in the direction of v, around the point:
   $$c = \frac{a+b}{2} \tag{6.20}$$
   $$(x_I, y_I) = (x_{i-1}, y_{i-1}) + c \cdot u \tag{6.21}$$
   (b) If there is a maximum of gradient in (x_I, y_I), $(x_I, y_I) + v$ or $(x_I, y_I) - v$, we initiate a segment extension as described in section 6.1.2.
   (c) For each segment detected at $level_{i-1}$, compute the projection $P(ext1)$ and $P(ext2)$ of its extremeties $ext1$ and $ext2$ on $2 \cdot seg_j^i$. Then, using $idx1 =< O_j^i P(ext1), \mathbf{u} >$ and $idx2 =< O_j^i P(ext2), \mathbf{u} >$, we can remove from the intervals of L, the area where a segment has already being detected, as explained in figure 6.6.
   (d) If no segment is found, the interval $I = [a, b]$ is splited in $[a, c]$, $[c, b]$, which are inserted in L.
   (e) Return to step **(a)** until L is empty.

   An interval is inserted in L only if its length is superior to the minimum segment length.

Parameters "Hierarchical DSeg" has only two more parameters than "DSeg": the number n_p of images in the pyramid, and the scale s_p between two images. The number of detected segments is too small if the top of the pyramid uses too small images, which sets a limit on n_p.

The scale parameter s_p should be inferior to 2 ($s_p < 2$), as for a bigger value, it will be needed at step 2(b) to check for a maximum of a gradient around more points. It is better interesting to have more images in the pyramid than to use a bigger scaling coefficient.

We usually use $s_p = 2$ and $n_p = 3$, which means the top image in the pyramid is divided by four.

Results Figure 6.7 shows the result of a detection on different images.

6.1.4 Comparison

We present here a comparison of our approach (*DSeg*, *Hierarchical DSeg*), with the *Probabilistic Hough* transformation approach ([Guo et al., 2008] *Hough*, OpenCV implementation),

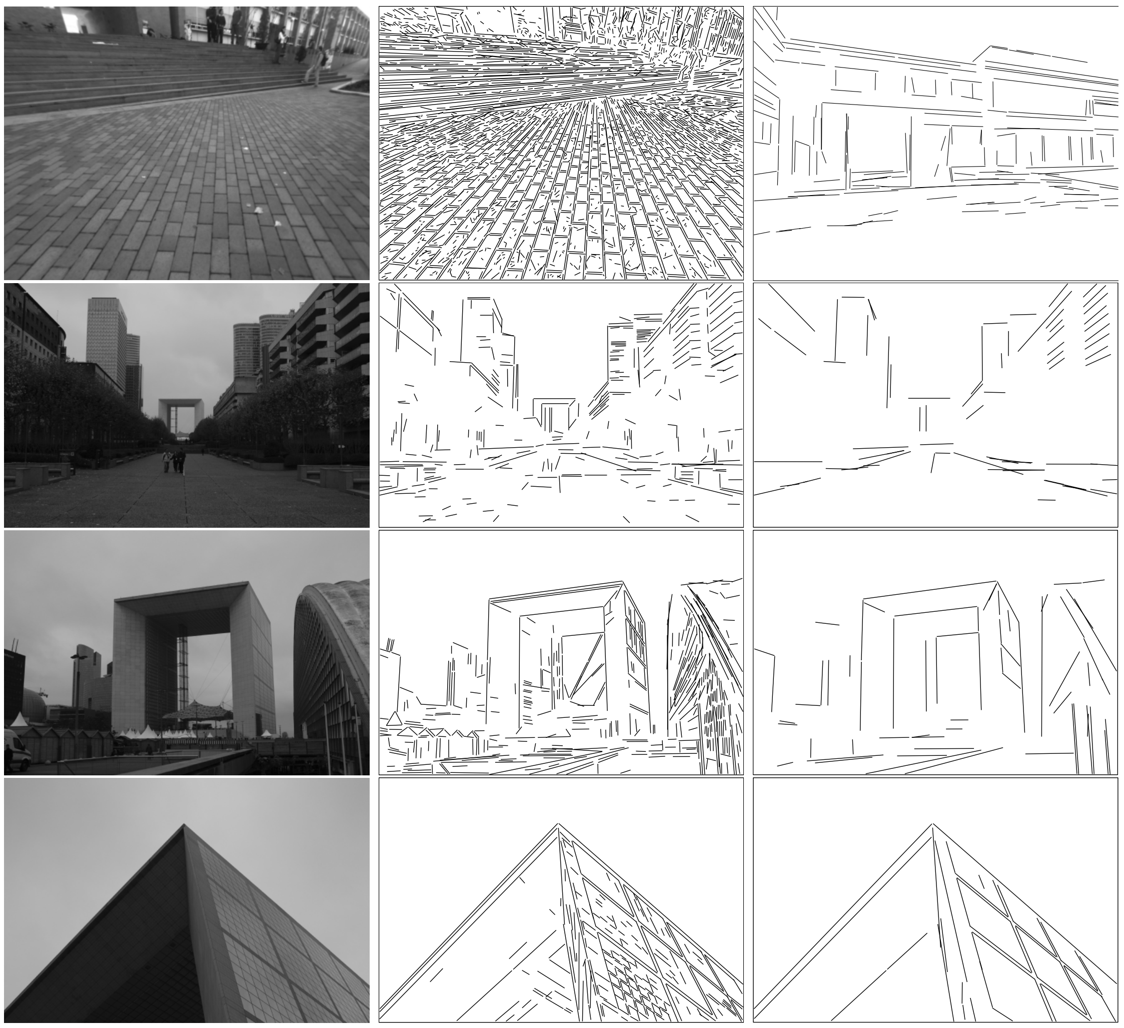


Figure 6.7: Some results of *DSeg* and *Hierarchical DSeg*. Note the stability of long segments extracted on the square arch building with respect to large scale changes. We can also see that we get less segments than with the hierarchical approach than with full image approach.

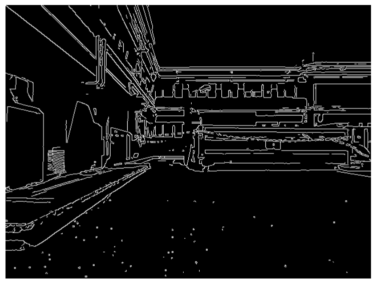
the segments chaining approach (*Split and merge*, LAAS implementation), and the approach presented in [R. Grompone von Gioi and Randall, 2008] (*LSD*, using the authors implementation).

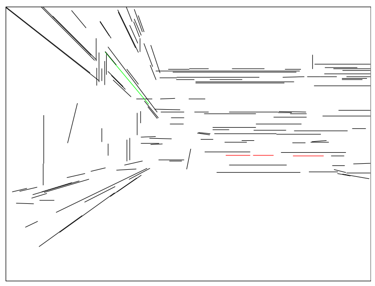
Note that we had to adjust parameters for each image with the Hough and Chaining approaches to obtain good results – especially the two threshold on the results of the Canny filter, which needs to be tuned depending on the contrast and noise level of the image. Whereas for the LSD approach and ours, it has not been necessary to change any parameter.

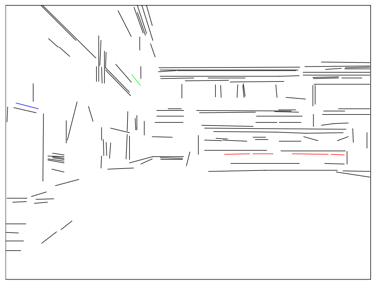
Figure 6.9 shows the line segments detected by the four approaches, with the associated computed times[3] – only segments longer than 20 pixels are shown. *DSeg* extracts more segments than the three other approaches, and especially more long segments – as highlighted by red and blue segments in the figures. Note that on more contrasted images (Figure 6.8), *DSeg* and *LSD* behave quite similarly, but still better than the Hough and Chaining approaches.

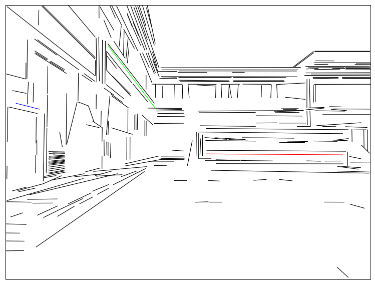
[3]Assessed on an 2.2 GHz Intel processor

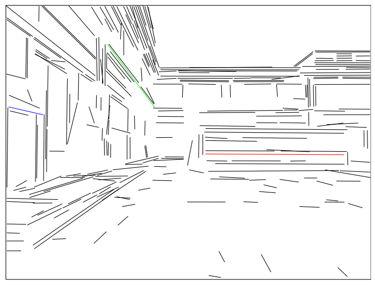
(a) Image 640x480

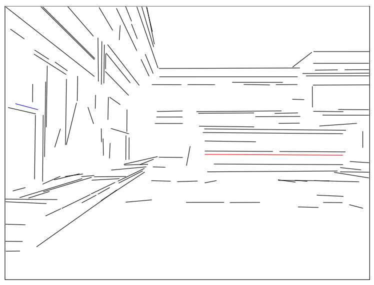

(b) Canny

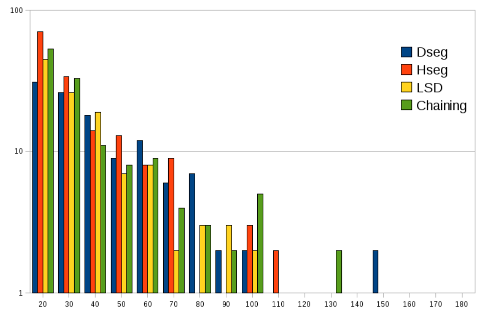

(c) Probabilistic Hough (82ms)


(d) Chaining (60ms)


(e) LSD (86ms)


(f) DSeg (168ms)


(g) Hierarchical DSeg (181ms)



(h) Segment length

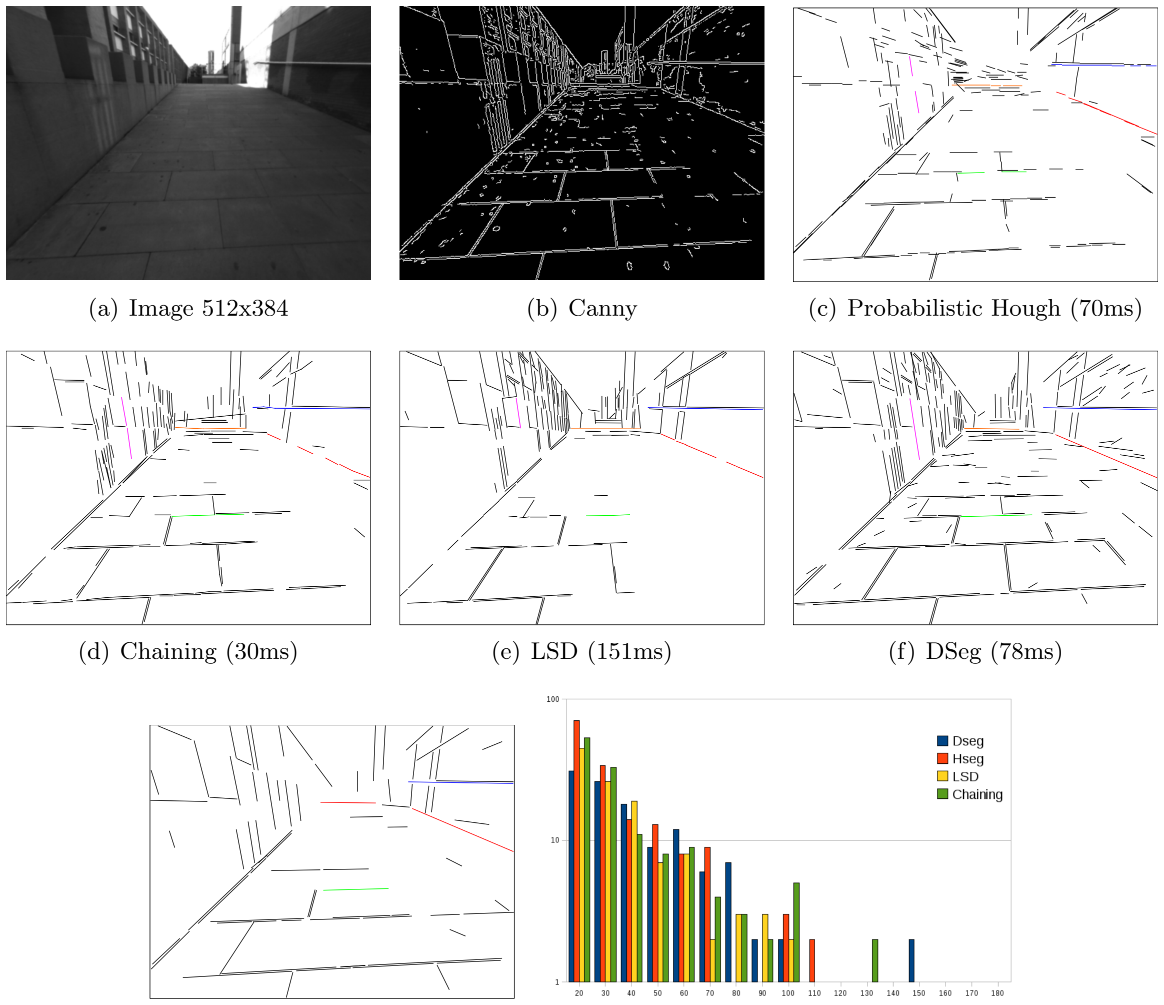
Figure 6.8: Line segment extracted on a well contrasted image.

All approaches take a rather similar amount of computational time, with an advantage for Hough and Canny. One might note that *Hierarchical DSeg* does not provide a performance enhancement compared to *DSeg*, the reason is that the most costly part of both algorithms is the segment growing process, and *Hierarchical DSeg* uses that process several times for each segments at every level.

More quantitative results on the number and length of the detected segments are provided by the histogram of figures 6.9(h) and 6.8(h) : *DSeg* finds the longest segments, and in total a larger number of segments. Naturally, *Hierarchical DSeg* finds less segments.

6.2 Sensitivity analysis

To assess the robustness of the algorithm with respect to image noise and to illumination changes, we analyse the *repeatability* of the detected segments by adding noise to an image, and on a sequence of images taken during 24 hours with a still camera.



(a) Image 512x384 (b) Canny (c) Probabilistic Hough (70ms)

(d) Chaining (30ms) (e) LSD (151ms) (f) DSeg (78ms)

(g) Hierarchical DSeg (83ms) (h) Segment length

Figure 6.9: Line segment extracted on a low contrast image.

6.2.1 Segment repeatability

To be able to automatically compute the repetability of the detection of segments with a fixed camera filming a static environment, it is necesserary to be able to compute which segment of the reference frame corresponds to the segment detected at a given time. To do this, we define a similarity measure, that uses the area contained between the two segments, ponderated by the length, angle and overlap (Figure 6.10):

$$sim(AB, CD) = \frac{\mathcal{A}(AB, P(AB, CD))}{(|P(AB, CD)| \cdot \mathcal{R}(P(AB, CD), CD) \cdot |cos(angle(AB, CD))|)} \quad (6.22)$$

$$distance(AB, CD) = sim(AB, CD) + sim(CD, AB) \quad (6.23)$$

Where $P(AB, CD)$ is the projection of the segment AB on CD, $\mathcal{A}(AB, P(AB, CD))$ is the area between the segment AB and its projection on the segment CD, $|P(AB, CD)|$ is the length of the projected segment, $\mathcal{R}(P(AB), CD)$ measures the overlap between the projection and the segment CD, and $angle(AB, CD)$ is the angle between the two segments.

The similitude defined in equation (6.22) is used to define the distance (6.23), which allows to find the segment of the reference image which is the closest to the segment in the current image. The distance (6.23) is not a distance in the mathematical sense, since $distance(AB, CD) = 0$ means that $\mathcal{A}(AB, P(AB, CD)) = 0$, which can happen when A, B, C and D are aligned, or AB and CD are on orthogonal lines, but in that case, $cos(angle(AB, CD)) = 0$. But it respects $distance(AB, CD) = distance(CD, AB)$, and if $distance(AB, CD) < distance(AB, EF)$, then CD appears to be closer of AB than EF.

Given two sets of segments Σ_{ref} and Σ_t, we can compute the repeatability as the number of segment $S^i_{ref} \in \Sigma_{ref}$ that satisfy the following conditions:

$$\exists S^j_t \in \Sigma_t / \forall k \neq j, distance(S^i_{ref}, S^j_t) < distance(S^i_{ref}, S^k_t) \tag{6.24}$$

$$\forall l \neq j, distance(S^i_{ref}, S^j_t) < distance(S^l_{ref}, S^j_t) \tag{6.25}$$

$$distance(S^i_{ref}, S^j_t) < \tau_{dist} \tag{6.26}$$

Since it's always possible to find a closest segment, a threshold τ_{dist} is used to eliminate segments that are not close to the current segment.

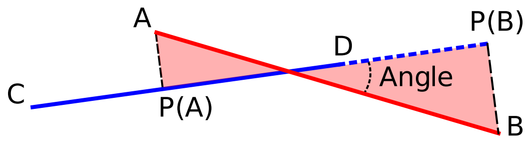


Figure 6.10: Distance between two lines segment. Here the overlap is $\mathcal{R}(P(AB), CD) = \frac{|P(A)D|}{|P(B)C|}$.

6.2.2 Sensitivity to noise

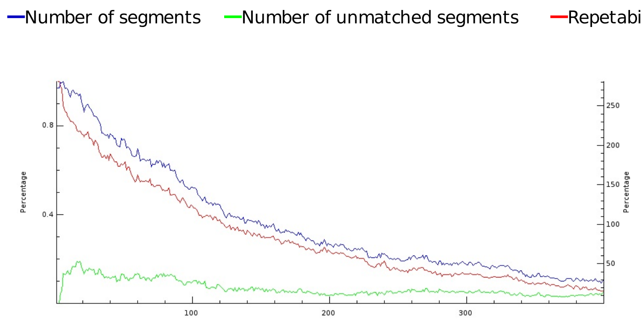
To test the sensitivity of the algorithms to the noise, additive and multiplicative noise is added to the luminance value of a reference image $\mathcal{I}_{ref}$:

$$\mathcal{I}(i) = R(\sigma^i_m) \cdot \mathcal{I}_{ref} + R(\sigma^i_a) \tag{6.27}$$

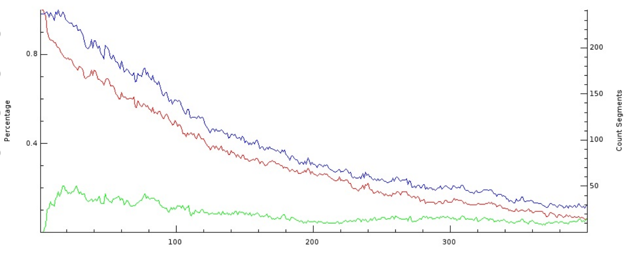
where $R(\sigma)$ is a random number generator with a Gaussian distribution of standard deviation σ and a null mean.

Direct segment detection Figure 6.11 shows the number of segments detected for different values of $\tau_{gradmax}$, as a function of the image noise. $\tau_{gradmax}$ has no influence on the repeatability of segment extraction, and that the algorithm is resistant to a significant level of noise.

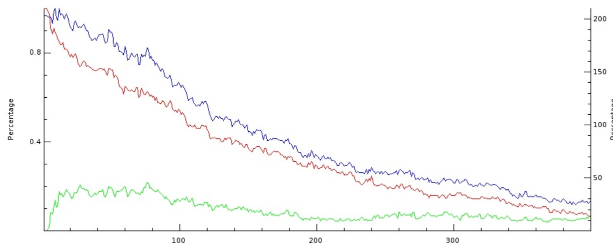
Comparison Figure 6.12 shows the comparison of the sensitivy to the noise for the different algorithms, the number of segments detected by *Probabilistic Hough* increases drastically with the noise, which triggers a lot of unmatched segments as well as splited segments, this is due



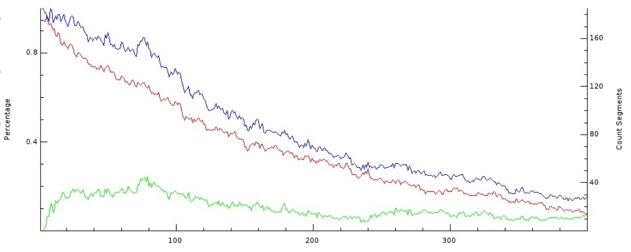
(a) $\tau_{gradmax} = 0$



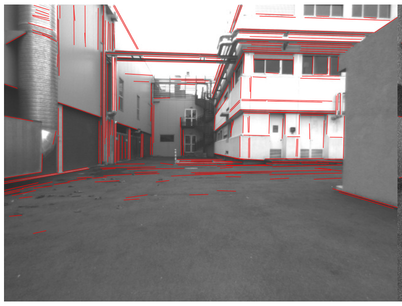
(b) $\tau_{gradmax} = 10$



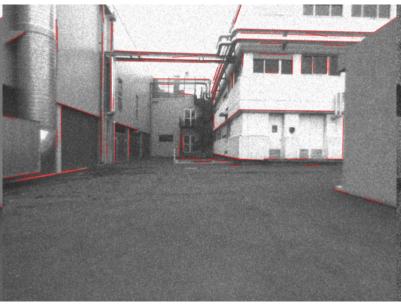
(c) $\tau_{gradmax} = 20$



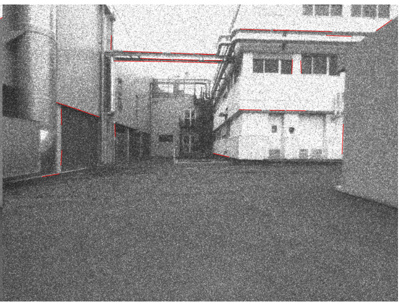
(d) $\tau_{gradmax} = 30$



(e) Reference image



(f) Image half noise



(g) Image full noise

Figure 6.11: Sensitivity to noise for DSeg.

to the nature of the algorithm, since the increase of noise increases the number of segment directions represented in the image, but it could be limited by tweaking the parameters. The *LSD* algorithm is very conservative to noise, which can be seen by the rapid drop in number of detected segments, and therefor in repeatability, but it has slightly less unmatched segments compared to *DSeg*. Unsurprisingly, the two algorithms with the strongest noise filter (through image scaling) *DSeg* on image divided by four, and *Hierarchical DSeg* have the highest repetability. But *DSeg* on image divided by four have a higher number of unmatched segments, which might be explained by the decrease on precision.

6.2.3 Sensitivity to day light changes

Here, we took images with a still camera every 5 minutes during 24 hours.

Direct segment detection Figure 6.13 and 6.14 show the performance of the "DSeg" algorithm on the 24 hours images, on figure 6.14 the drop visible at the end of the day is caused by the fall of the night, but since the lights were on, some segments (mostly inside the room) were still visible. The figures show that the repeatability remains around 0.9 during daylight, and that some segments are still correctly extracted during the night.

Comparison Figure 6.15 shows how the various algorithm performs when the light of the day changes, we can see that *Hierarchical DSeg* detects as much segments as *DSeg* on an image divided by four, that *Probabilistic Hough* has the highest number of detected segments but to the price of a lower repeatability and a higher number of splited segments. During the day *Hierarchical DSeg*, *DSeg* and *LSD* have a similar repetability, which is higher than *Split and merge* and *Hough*, while during the night, *DSeg* on image divided by four gets the highest repetability, followed by *Hierarchical DSeg*.

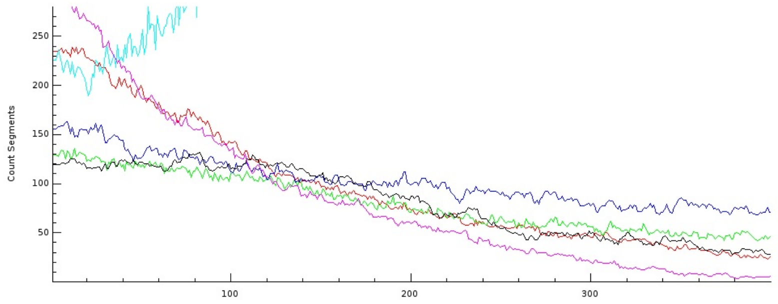
6.3 Line Segment Tracking

We present in this section the approach to track segments from one image to another. The process relies on a Kalman filter, as in [Deriche and Faugeras, 1990], that fuses a prediction provided by a motion model with observations that are made in a way similar to which the segments are extended during the detection phase – making our tracking scheme akin to the approach to fit rigid object models in an image introduced in [Harris, 1992].

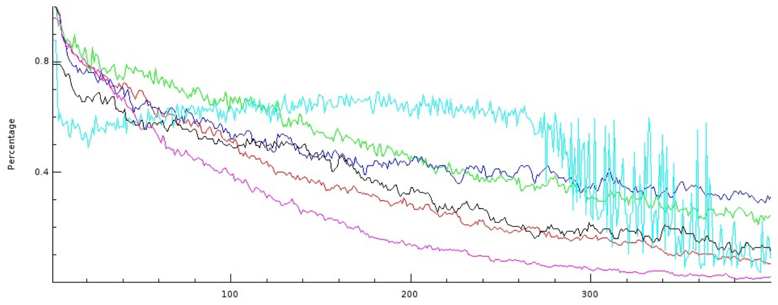
When tracking segments, the main difficulties are caused by the presence of close parallel segments. Indeed, even a precise motion model may yield predictions that are off up to a few pixels, which eventually leads to wrong associations between close parallel segments (for instance, in figure 6.16, segment 2 would jump to the position of segment 3). We focus on this issue here: a selection process discards the segments that may lead to false associations, and information related to the gradient along the segment are exploited to enforce the association provided by the tracking process.

6.3.1 Segments selection

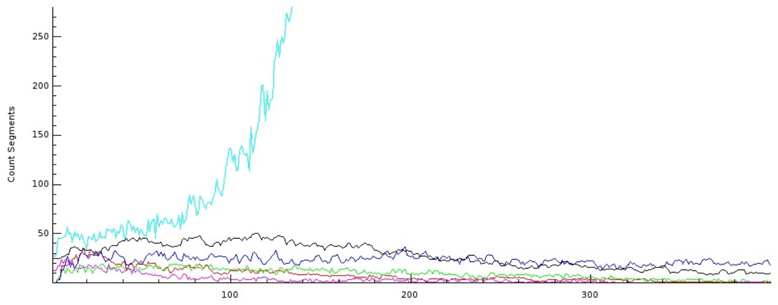
The *direction* of a line segment is defined on the basis of the corresponding gradient phase angles, as shown figure 6.16. Two segments (S_1 and S_2) are considered to be closely parallel,



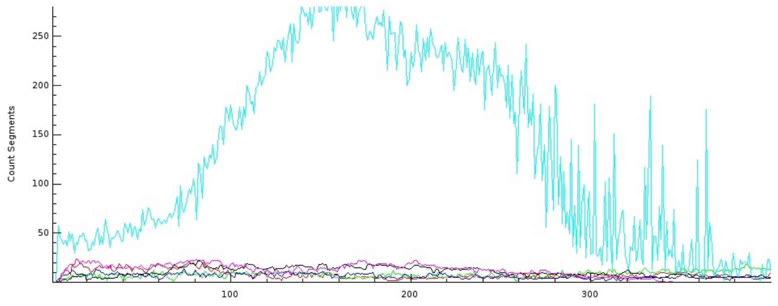
(a) Number of detected segments



(b) Repetabilty



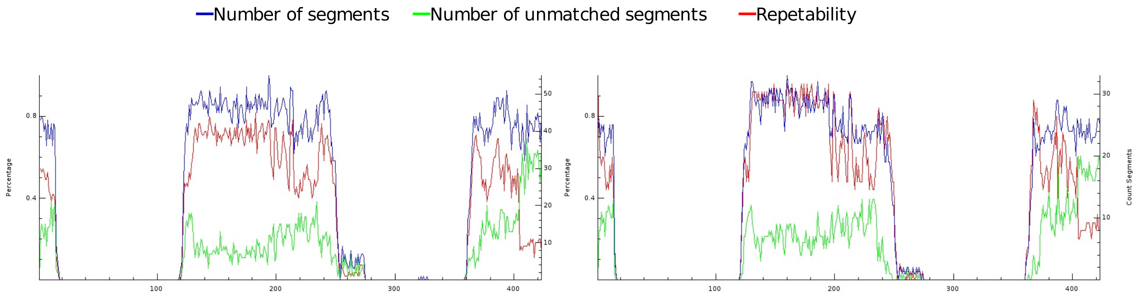
(c) Number of unmatched segments



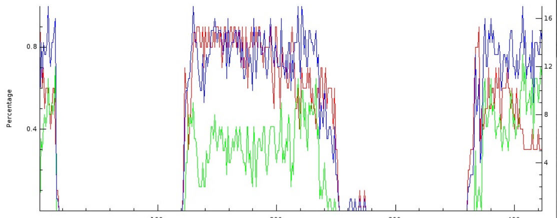
(d) Number of splited segments

— DSeg (scale 0.25) — Hierarchical DSeg — DSeg
— Probabilistic Hough — Split & Merge — LSD

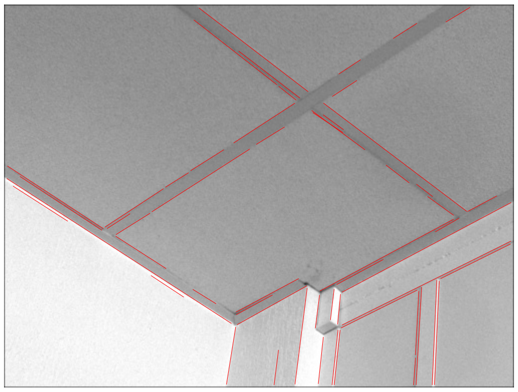
Figure 6.12: Comparison of the sensitivity to noise.



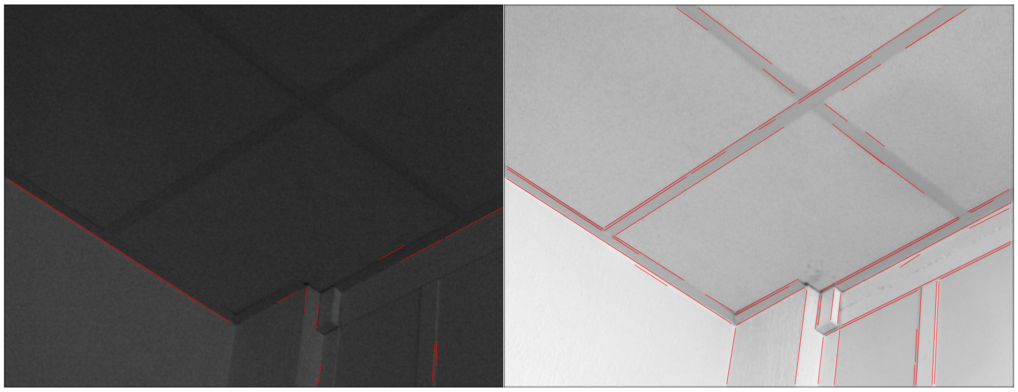
(a) *length* = 20 (b) *length* = 50



(c) *length* = 100

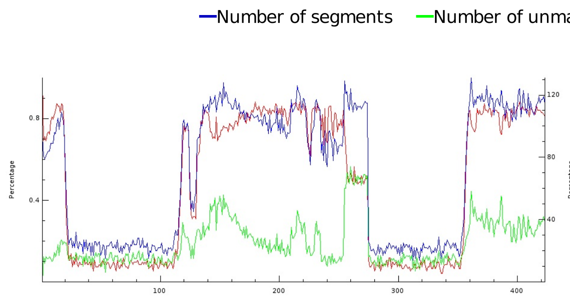


(d) Image 0

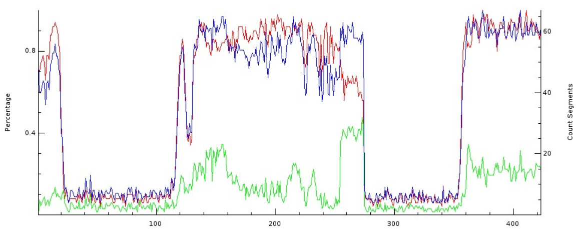


(e) Image 14 (f) Image 200

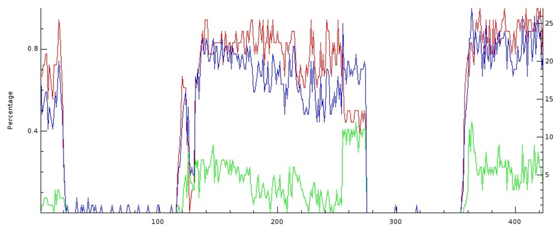
Figure 6.13: Sensitivity to change of illumination for "DSeg", when changing the minimal *length* of segments with *DSeg*.



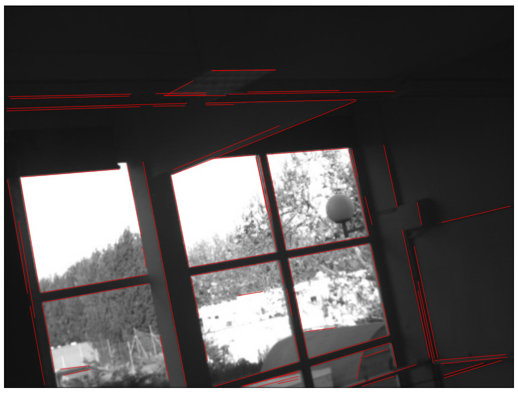
(a) *length* = 20



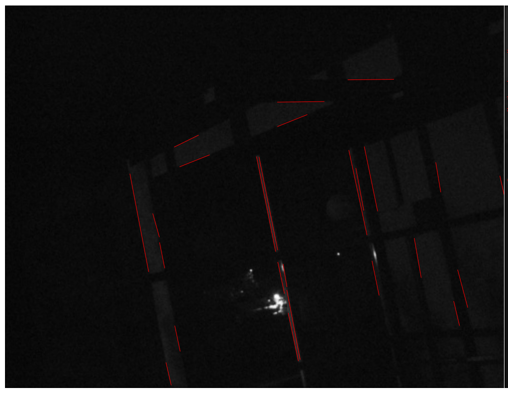
(b) *length* = 50



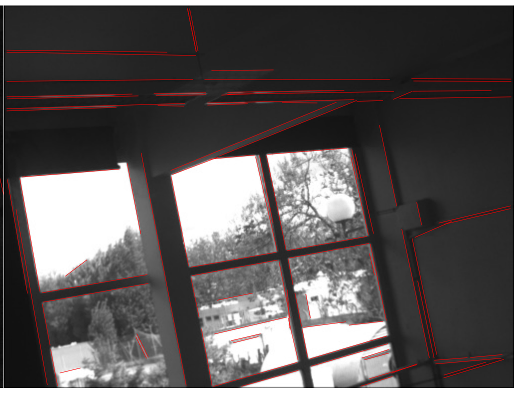
(c) *length* = 100



(d) Image 0

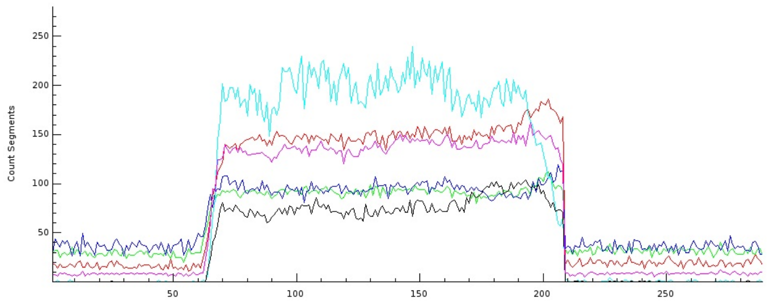


(e) Image 100

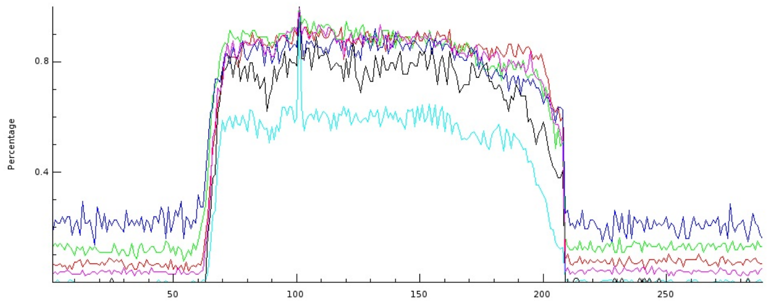


(f) Image 200

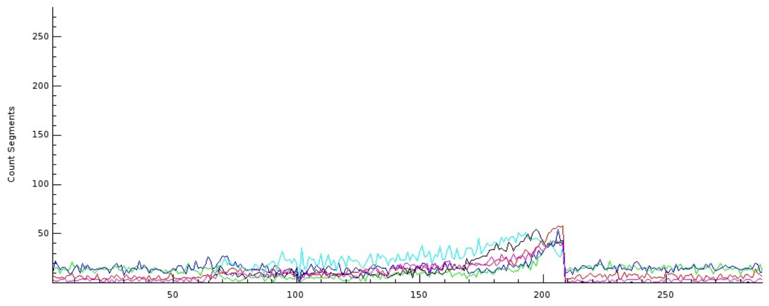
Figure 6.14: Robustness to the change illumination when changing the minimal *length* of segments with *DSeg*.



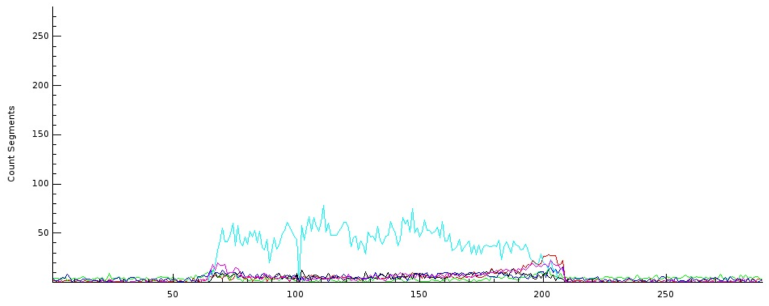
(a) Number of detected segments



(b) Repetabilty



(c) Number of unmatched segments



(d) Number of splited segments

🟦 DSeg (scale 0.25) 🟩 Hierarchical DSeg 🟥 DSeg
🟦 Probabilistic Hough ⬛ Split & Merge 🟪 LSD

Figure 6.15: Comparison of the robustness to change of light.

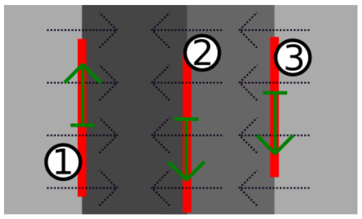


Figure 6.16: Line segment direction: the green vectors indicates the line direction, derived from the gradient phase angles (black arrows).

and in the same direction, if the following conditions are met:

$$cos(dir_{S1} - dir_{S2}) > \tau_{angle} \tag{6.28}$$

$$d(ext^1_{S1}, segment_2) < \tau_d \tag{6.29}$$

$$d(ext^1_{S2}, segment_2) < \tau_d \tag{6.30}$$

Where ext^1_{S1} and ext^1_{S2} are the two extremities of the segment S_1, and $d(point, segment)$ is the distance in pixels between a point and a segment.

If two closely parallel segments are of similar lengths, they are both discarded from the tracking process, and when small segments are closely parallel to a longer one, the small segments are discarded.

6.3.2 Association hypothesis

The tracking of a segment is initialized by the selection of an hypothesis. Then, an algorithm similar to the line segment extension process described in section 6.1.2 is used to refine the position of the segment.

Two steps are involved to find an association hypothesis: a prediction step, and a confirmation step, that evaluates a small set of observations to assess the likelihood of the hypothesis.

Prediction. Several prediction models can be considered. The most simple is a static model, *i.e.* in which the same segment parameters of the source image are considered. A constant velocity model is often used (*e.g.* in [Neubert et al., 2008]): in such case, one needs to use four Kalman filters for the tracking process, one for each parameter, which are independent from each others. An optimal solution would be to have the speed parameters in the Kalman filter used for detection but that would increase the computational cost of updating the filter while observing the segment. Nevertheless, such a model does not work well in case of irregular camera motions. Finally, if the actual camera motion can be measured, as when mounted on a robot for instance, it can be used to estimate a motion model for the segments – this model being even better if the 3D parameters of the line segments are estimated, as in a simultaneous localization and mapping process for instance [Lemaire and Lacroix, 2007a].

Each prediction model gives an estimated position of the origin (x_0, y_0) of the segment, and of its orientation ϕ_{pred}.

Initial guess from the signal. Given the predicted state of the line segment, n_{kp} key points are selected along it, and the tracking algorithm searches for a local gradient maximum along the normal of the key points. For each key point, we select the local maximum (i, j) that maximizes:

$$cos(\phi_{i,j} - (\phi_{pred} + \pi/2)) \cdot ratio(G_{i,j}, G_{mean}) \tag{6.31}$$

Where G_{mean} is the mean of the gradient on the segment computed at the previous tracking (or detection) step, and $ratio(x, y)$ is the function that returns x/y if $x < y$ or y/x otherwise.

The first part of this equation allows to check that the direction of the gradient is compatible with the direction of the predicted segment. This is what ensures that the direction of the segment is maintained, which is important for the case of close segments with similar directions. The second part ensures that the selected local maximum has a value close to the gradient in the previous frame.

Once a local maximum has been found for the n_{kp} key points, the median of the phase angles ϕ_{median} is computed. Key points for which $|cos(\phi_{i,j} - \phi_{median})| > 1.0 - \tau_{angle}$ are discarded. With the remaining maximums, we compute the median of the translation, which is used to discard some outliers. The mean of the translation and of the angles are then computed, and used to initialize the Kalman filter for the extension procedure of section 6.1.2.

6.3.3 Parameters

The parameters τ_d is depending on the displacement between images, the bigger the displacement the bigger the value of τ_d is. It also depends on the quality of the prediction, it is necessary to select $\tau_d > max(\sigma_t)/2$ (where σ_t is the error on the translation).

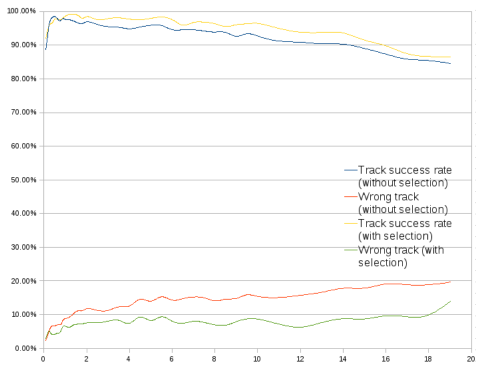
For the sequence used in our experiments, the pixel displacement between two images was of around 2 pixels. To be safe, we used $\tau_t = 5.0$. We used $n_{kp} = 20$ key points (a higher number of key points ensures a better estimation, at the cost of additional computational time).

6.3.4 Results

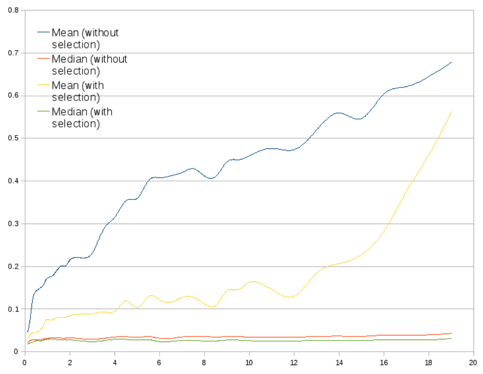
Static evaluation. A first evaluation of the tracker is done using twice the same image, but applying a translation $\mathbf{t}$ on the predicted segments. The results of this experiment are shown on figure 6.17. The false positives in 6.17(a) are counted using the Mahalanobis distance between the parameters of the segment in the reference extraction and after the tracking procedure. For this experiment, the error in position for the segment is equal to the norm of the translation vector $\sigma = |\mathbf{t}|$, which means, since the search area for key points is equal to $3.0 \cdot \sigma$, for a translation of 10 pixels, the key points are selected among 60 pixels (30 pixels in each direction): this clearly indicates the need for an accurate prediction.

The two graphs also show the benefit of the selection process, since the segments tracked after selection are also tracked without the selection, it is logical that the number of false positive is inferior with the selection. We observed that the wrong matches are mostly caused by segments which are located on curved features and that do not correspond to a real line segment. The graph 6.17(b) shows that errors remains small for a translation smaller than 4 pixels, and remain acceptable up to 10 pixels, whereas without the selection the number of mismatched segments increases considerably.

Tracking along a sequence. In figure 6.18 we show the result of tracking segment over a set of images using only the segments detected in the first frame. The time for tracking 35 segments at each step is around 60ms. For this experiment, we used a static prediction



(a)



(b)

Figure 6.17: (a): tracking success rate as a function of the translation difference between the real segment pose and their predicted position; (b) mean error between real segments and prediction. Without the segment selection, 109 segments were used, while after selection, only 35 segments are used.

model, increasing the error in translation for the Kalman filter by sigma $\sigma_t = 2$ and on the angle $\sigma_{angle} = 0.31 rad$.

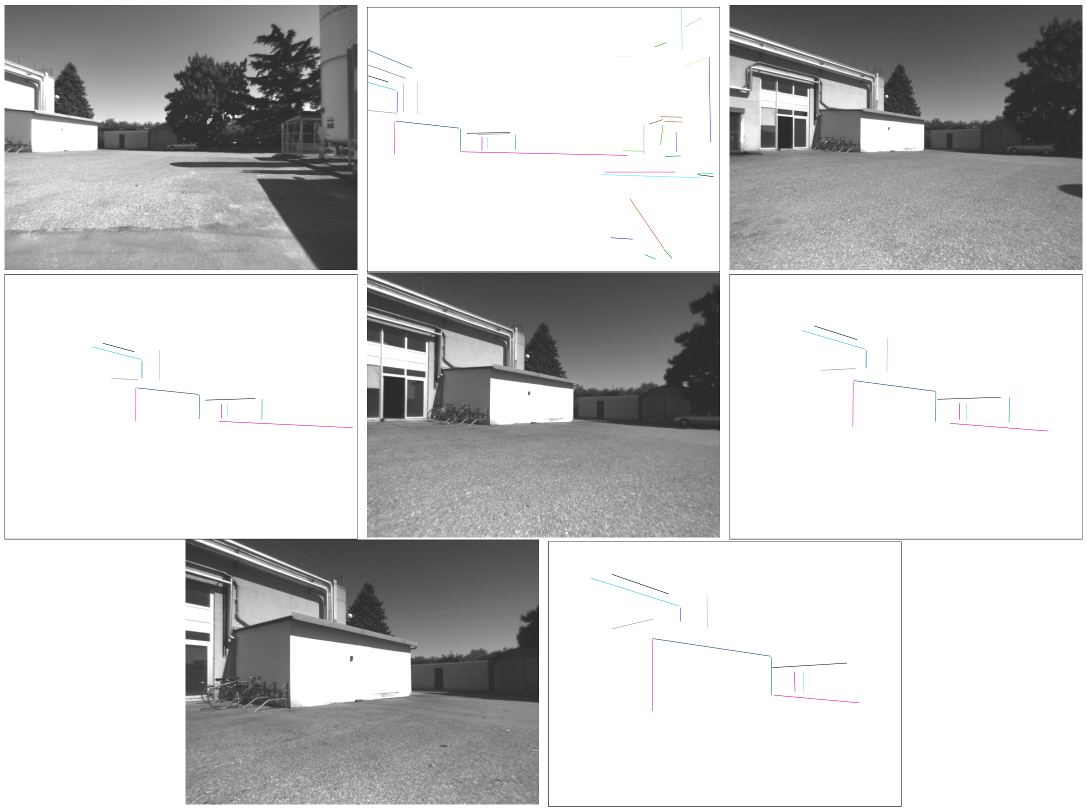


Figure 6.18: Tracking results.

Chapter 7

Discussion

We briefly compare in this chapter the three considered landmark types, and present some results of their use in a SLAM context.

7.1 Comparison of landmarks types

Interest points. The main advantage of using interest points is that they are easily detected, and the literature presents various mature algorithms for this purpose. Similarly, point-based SLAM processes are well established. But points yield to a sparse low-density model of the environment, that does no not exhibit its geometric structure. The extraction of points in images is also not robust with respect to large viewpoint changes.

Facets. We have proposed an algorithm that extends interest points to local planar facets in chapter 5. Facets carry more information on the environment geometric structure than interest points, and yield improvements on the localisation of the robot in a SLAM approach: this is due to the fact that each facet contains enough geometric information to fully observe the robot position – while a minimum of three points is needed for this purpose. The drawbacks of facets is the computational time required to compute the homography and their normals. Also, since they are centered around interest points, they suffer from the same problem of detection robustness with respect to large viewpoint changes.

Line Segments. We have proposed two new algorithms to extract and track segments in images in chapter 6. The choice between *Hierarchical DSeg* and *DSeg* depends on whether the application requires more segments from the image, or robust segments. Our experiments show that segment detection is as fast as interest point. Most importantly, segments exhibit more structure of the environment than points or facets. However, despite some attempts (*e.g.* [Schmid and Zisserman, 2000]), there is no good matching process of segments in the image space.

Segment parametrisation is an important issue in a SLAM context. In [Gee and Mayol, 2006] and [Smith et al., 2006] lines are parametrised in the SLAM filter using two inverse depth points, while we used the Plücker representation described in [Solà et al., 2009].

Planes. Planes would exhibit more spatial information than facets and segments. While we have not contributed to the definition of algorithms to extract planes from data, we mention here the main possiblities to achieve this.

- *Plane extraction in depth images.* On the basis of depth images provided either by dense pixel stereo of a Lidar, several techniques are possible to extract planes, *e.g.* the Hough transform [Vosselman and Dijkman, 2001] or planar regions segmentation [Baumstarck et al., 2006]. In [Viejo and Cazorla, 2006], the normal of each points is computed, and then points with similar normal are grouped together in a plane. This can also be done by first creating a set of facets [Murray and Little, 2005b, Weingarten and Siegwart, 2006], and then grouping the facets that have a similar normal into one plane.

  However, these techniques remain fragile and are very sensitive to the noise level of the data.

- *Plane extraction in vision.* In [Silveira et al., 2007], Silveira shows how to use the ESM homography estimator [Malis, 2004] to track planes in a sequence of monocular images, from frame to frame in a monovision process, and how to estimate the plan parameters. We believe that these approaches are very promising, especially when the *detection* of candidate planar areas is focused by the detection of line segments.

- *Fusing laser and vision information.* The main problem when using images for extracting planes is that vision algorithm fails in textureless environments. Exploiting both Lidar range imagery and vision provides more reliable plane detection [Wolf et al., 2005]. But 3D lasers are either very expensive, or slow to acquire data (the 3D laser used in [Baumstarck et al., 2006] takes one minute to scan the environment). In an indoor environment, one possible solution is to fuse information coming from a 2D laser with camera data [Baltzakis et al., 2003].

7.2 SLAM Results

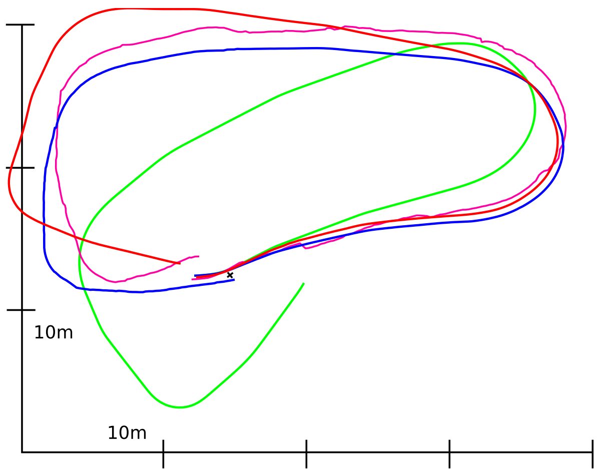


Figure 7.1: The different SLAM trajectories obtained with points (in red), facets (in blue) and line segments (purple). The green trajectory is the one estimated by odometry. The black dot indicates the final position of the robot.

The same data set has been used to compare an implementation of SLAM using either interest points, planer facets and line segments. Figure 7.1 compares the resulting robot trajectory estimation, without enforcing any loop closure. Using a visual motion estimation like in [Lacroix et al., 1999], it is possible to compute the distance between the last frame and the first frame, and to determine the final error of localisation for each type of feature. It shows an error of $3m$ for points, $0.78m$ for facets and $2.5m$ for segments. As expected, facets give a more accurate results: this comes from the fact that a single observation gives the position of the robot, while segments give similar results to points.

More interestingly, figure 7.2 show the resulting landmark maps.

Multi-maps, multi-robots, multi-landmarks In this experiment, we use two robots, a ground robot and an helicopter, that are mapping the same area, extracting interest points and segments. Figure 7.4 show the resulting map of extracting images from a house, by the ground robot and by the helicopter.

Figure 7.3 show the result of a loop closure event, that was triggered by manually computing a transformation between the ground and aerial robot. In the future, such an event should be computed automatically. For this purpose, we believe structuring the geometry of the landmark maps can yield to robust and efficient map matching algorithms: the next part of the manuscript is devoted to this problem.

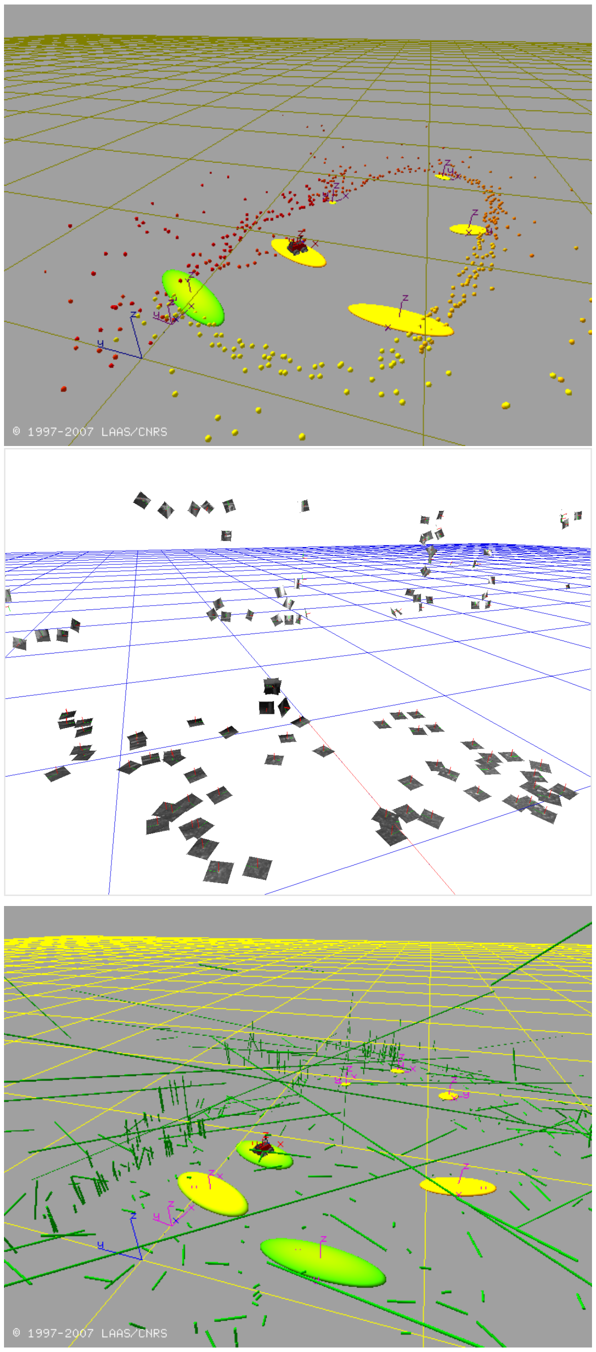


Figure 7.2: From top to bottom: landmark models with points, facets and segments, built with the same set of images.

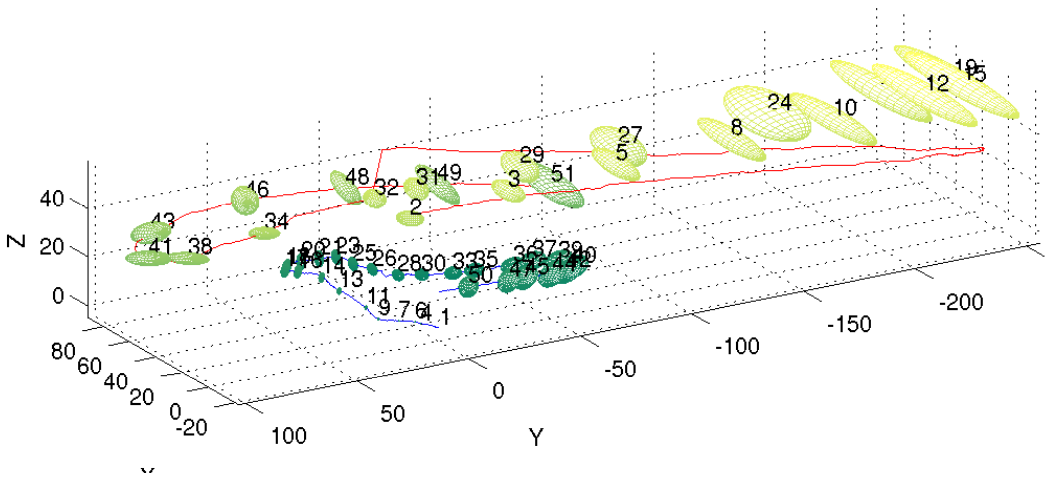
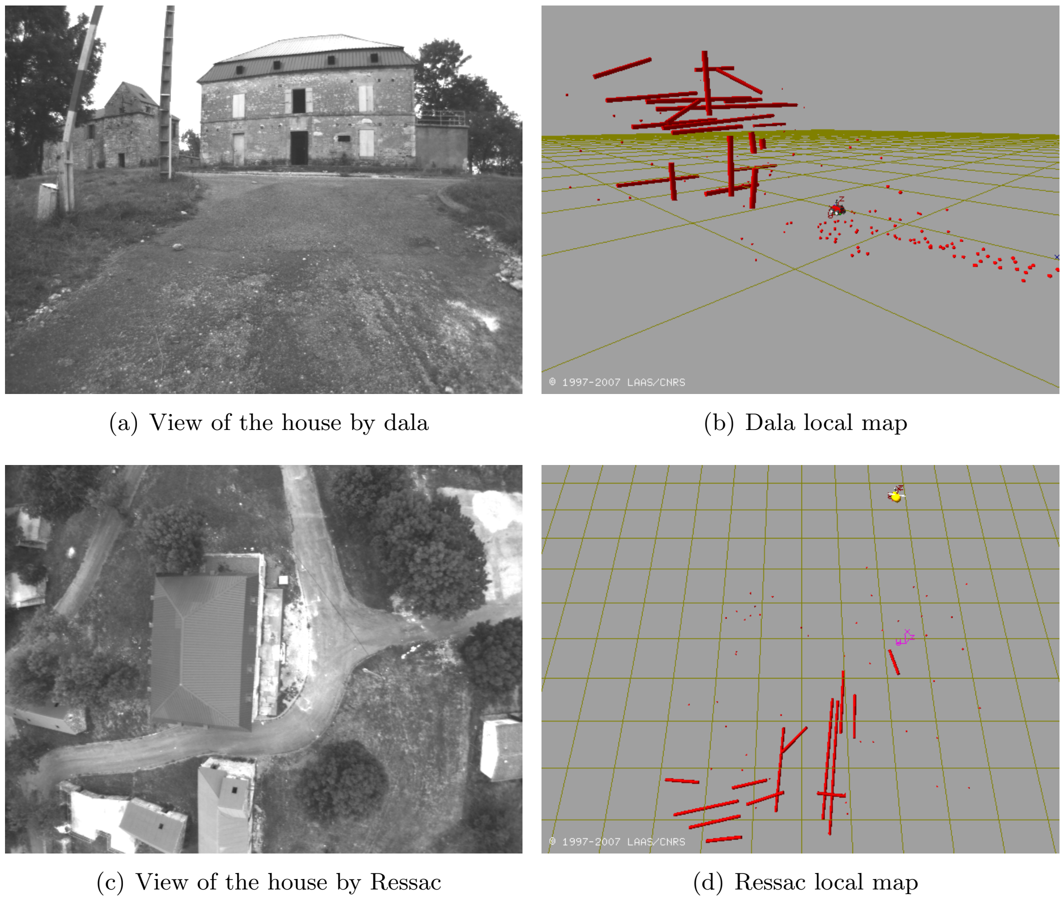


Figure 7.3: Final global graph (the global level) with Dala's trajectory in blue and Ressac's trajectory in red in the *wrf*.


(a) View of the house by dala

(b) Dala local map

(c) View of the house by Ressac

(d) Ressac local map

Figure 7.4: Plot-view of 3D local maps. Robot pose, points and line segments are displayed in *local reference frame*.

Part III

Geometric model

One of the limitations of the multi-robots multi-maps experiment of the chapter 7 is that there is no automatic loop closure detection between both kinds of robots. The signal information (such as image patches) is indeed too different from a robot to another – not to mention that it can change depending on the scene illumination.

Since the environment geometry is invariant, one must therefore rely geometric landmarks: but since individual features are very similar from one another, the only way to have accurate matching at the geometric level is to use the neighbourhood of a feature, and to compute multiple matches at the same time. In this part we introduce a structuring of the geometry of the environment into a graph (chapter 8). This graph structure is then used in chapter 9 to solve the data association problem at a geometric level.

Chapter 8

Graph based structuration of a cloud of geometric landmarks

In section 8.1, we define the geometric obects and their parametric representations. Section 8.2 presents the different relations that exist between geometric objects for modeling the environment, then section 8.3 presents how to compute the partial transformation vector, used to define the geometric environment around a feature.

The last two sections cover the structure of the graph (section 8.4) and its construction (section 8.5).

8.1 Geometric objects

The geometric objects used in this chapter are part of the Euclidean geometry system, and their parameters are expressed in the Cartesian space.

A representation of an object is "multiform", the representation used in the estimation process does not need to be the same as the one used in the matching process, or to express relations among objects. For instance, a point in monocular SLAM is represented by six parameters [Davison, 2003], but to express relations between objects the minimal three parameters representation is enough.

This section first gives the notation and definitions of geometric objects and operators in section 8.1.1, then later in section 8.1.2 we detail the possible parametrisation of atomic objects.

8.1.1 Notations and definitions

Geometric objects There are two types of geometric objects, objects that can be self-defined, (for instance, using a set of equations), referred to as *atomic objects*, and geometric objects that are defined as a set of other objects, referred to as *composite objects.*

Definition 1 *An* ***atomic object*** *is a geometric object that can be used as a basic building block. It is usually self contained, can be detected in the data, and can not be expressed as a finite set of other atomic objects.*

Definition 2 *A* ***composite object*** *is a geometric object which can be constructed using,* *directly or indirectly, a finite set of atomic objects. A **composite object** is indirectly constructed from atomic objects, when it is defined by a set of other **composite objects**.*

A *geometric object* (atomic or composite) is noted o, while an *atomic object* is noted $\mathring{a}$. The list of *atomic objects* we consider is:

- a point (P) is a sizeless 0D object
- a line (L) is an infinite 1D object
- a line segment (S) is a line bounded by two points
- a circle (Ci) is a set of points at equal distance to the center
- a plane (Π) is an infinite 2D object

Some examples of *composite objects*:

- a polygon (Po) is a set of line segments connected to each other
- a facet (F) is a plane bounded by a polygon
- a cylinder (Cy) is a volume defined by two circles
- a box (Bo) is a 3D volume made of any set of planes

Operators Since we are interested in how geometric objects relate to each other, we need to define the list of operators that can be used to define these relations:

- $d(o_1, o_2)$ is the distance between two objects o_1 and o_2
- $\theta(o_1, o_2)$ is the angle between two objects (if it can be defined)
- $l(S)$ is the length of a line segment. Given P_1 ans P_2 the extremities of the segment S:

$$l(S) = d(P_1, P_2) \tag{8.1}$$

- $p(Po)$ is the perimeter of a polygon:

$$p(Po) = \sum_{S \in Po} l(S) \tag{8.2}$$

- $s(F)$ is the surface of a facet.
- $v(\mathring{a})$ is the volume of an object.
- $\mu(\mathbf{x_1}, \mathbf{x_2}, M_1 + M_2)$ is the Mahalanobis distance between a vector $\mathbf{x_1}$ (of covariance matrix M_1) and $\mathbf{x_2}$ (of covariance matrix M_2).
- $proj(P, L)$ and $proj(P, \Pi)$ are the respective projections of the point P on a line L and on a plane Π

8.1.2 Atomic objects

There are four common atomic objects, each corresponding to an increase in the number of dimensions: point (0D), line (1D), plane (2D) and basis (3D).

Point A point is defined by three coordinates (x, y, z).

Line There are multiple representations of a line, the minimum number of parameters is four:

- the coordinates of **two points** (P_1, P_2), six parameters: $(x_1, y_1, z_1, x_2, y_2, z_2)$
- the coordinate of **one point and one direction vector** $(O_L, \overrightarrow{u})$, six parameters: (x, y, z, u, v, w)
- **two equations**, the general form would be the interesection of two planes:

$$xa + yb + zc = d \tag{8.3}$$
$$xe + yf + zg = h \tag{8.4}$$

  But for all lines which are not contains in a plane $z = z_P$, it can be reduced to:

$$x = a' + b'z \tag{8.5}$$
$$y = c' + d'z \tag{8.6}$$

  For lines contains in a plane $z = z_P$, the following equations can be used:

$$z = z_P \tag{8.7}$$
$$y = a' + b'x \text{ or } x = c' + d'y \tag{8.8}$$

- the **direction vector and the distance** to the origin, four parameters: (u, v, w, d)
- the **Plücker** representation uses six parameters, and is defined by the direction vector $\overrightarrow{u}$, and the moment vector $\overrightarrow{m}$, with the following constraint $< \overrightarrow{u}, \overrightarrow{m} >= 0$.

  For two homogeneous points on the line (x_1, x_2, x_3, x_0) and $y = (y_1, y_2, y_3, y_0)$, the algebraic definition of the Plücker coordinates are $(p_{0,1}, p_{0,2}, p_{0,3}, p_{2,3}, p_{3,1}, p_{1,2})$, with $p_{i,j} = x_i y_j - x_j y_i$.

  The geometric definition is given by $(\overrightarrow{d} = \overrightarrow{y} - \overrightarrow{x}, \overrightarrow{m} = \overrightarrow{x} \times \overrightarrow{y})$ (where $\times$ is the cross product between vectors).

The main interest in the Plücker representation is that it makes possible to quickly check some properties of line, for instance two lines $(\overrightarrow{d_1}, \overrightarrow{m_1})$ and $(\overrightarrow{d_2}, \overrightarrow{m_2})$ are coplanar, if and only if:

$$< \overrightarrow{d_1}, \overrightarrow{m_2} > + < \overrightarrow{d_2}, \overrightarrow{m_1} >= 0 \tag{8.9}$$

It is possible to easily switch from any representation to an other. For instance, to change from a two points (P_1, P_2) representation to a one point and one direction vector $(O_L, \overrightarrow{u})$:

$$O_L = P_1 \text{ and } \overrightarrow{u} = \frac{\overrightarrow{P_1P_2}}{\|\overrightarrow{P_1P_2}\|} \tag{8.10}$$

Plane Plane is a two dimensions object, the minimal representation is four parameters, but there are many different representations:

- The coordinates of **three points**, nine parameters
- An equation, four parameters:

$$P = (x, y, z) \in \Pi \iff ax + by + cz + d = 0 \tag{8.11}$$

- the **normal vector and the distance** to the origin, four parameters: (u, v, w, d)
- A line (d, m) and a point $y = (y_1, y_2, y_3, y_0)$ defines the following plane equation:

$$x = (x_1, x_2, x_3, x_0) \in \Pi \iff 0 =< y, \overrightarrow{m} > x_0 + < (y \times \overrightarrow{d} - y_0\overrightarrow{m}), x > \tag{8.12}$$

Segment Lines and planes are infinite objects. In the context of environment modelling this can be a problem, since the physical appearance of a line is limited in space. But they provide support for segments: line segments or facets or a partition of space.

There is no good parametrisation of a facet, other than to define the plane and the boundary as a polygon. While a line segment can be described by the parameters of a line and information on the position of the extremities a minimum of six parameters:

- the line parametrisation with the coordinates of **two points** (P_1, P_2) gives six parameters: $(x_1, y_1, z_1, x_2, y_2, z_2)$ and is a minimal representation for a segment.
- the **one point and one direction vector** $S = (P, \overrightarrow{u})$ parametrisation can be considered as a minimum representation of a segment, if $|\overrightarrow{u}| = d$, and the point $P_2 = P + \overrightarrow{u}$ is the second extremity.
- the equation parametrisation can be extended to segment by adding a constraint on the parameter, using one of the following equations:

$$a < x < b \;\; or \;\; a < y < b \;\; or \;\; a < z < b \tag{8.13}$$

- the **direction vector and the distance** to the origin can be augmented with two parameters s_1 and s_2 which represent the distance between the projection of the origin on the line and the two extremities.

Basis It is not a geometric object, it allows to define the coordinates of other object, the minimal representation for a basis is six parameters:

- Euler angles, and translation, six parameters
- The origin, and three vectors, 12 parameters

8.2 Relations between geometric objects

In this section we describe the relations that can be defined between two geometric objects, whether topological (i.e. a point as the intersection of two lines), constructive (i.e. a cube is made of six facets) or scale (the detail of the scene changes as a function of the distance).

8.2.1 Definitions

Definition 3 *A **topological relation** between a set of geometric objects is a relation that decreases the degree of freedom (the minimal number of parameters of the object) of at least one of the object.*

Definition 4 *There is a **constructive relation** between one object o and a set of objects $\{o_i\}$ when all parameters of o are defined by the parameters of the objects $\{o_i\}$.*

Straightforwardly from the definition, the *constructive relation* is a *topological relation* which reduces the degree of freedom of one object to zero:

Lemme 1 *A **topological relation** between an object o and the set of objects $\{o_i\}$ decreases the degree of freedom of o to zero, if and only if, there is a **constructive relation** between o and $\{o_i\}$.*

Definition 5 *There is a **scale relation** between two sets of objects $\{o_i\}$ and $\{o_k\}$ when the objects of both sets correspond to the same object at two different scales.*

8.2.2 Topological relations

A general formalism for topological relations is described in [Egenhofer, 1991], but for atomic geometric features, we prefer the less general definition 3.

A topological relation is defined for atoms with numerical parameters, points, lines, planes and segments. While in definition 3 of *topological relations* it is mentioned that the relation decreases the degree of freedoms of one object, in reality, it is the degree of freedom of the group of objects that is decreased. For instance, if two lines are parallel, it is possible to move the plane that contains the two lines as well as the distance between them.

Point Table 8.1 list topological relations for a point.

Notation	Description	Degree of freedom of the first object	Degree of freedom of the system
P		3	3
$P \in L$	A point on a line	1	5
$P \in S$	A point on a segment	1 (bounded)	7
$P \in \Pi$	A point on a plane	2	6
$P = L_1 \bigcap L_2$ ($L_1 \neq L_2$ and $L_1 \nparallel L_2$)	The intersection of two lines	0	7
$P = \Pi \bigcap L$ ($L \notin \Pi$ and $L \nparallel \Pi$)	The intersection of a line with a plane	0	7

Table 8.1: Topological relations for a point with associated degree of freedom.

$P = L_1 \bigcap L_2$ and $P = \Pi \bigcap L$ are also *constructive relations*, since the point P is totally defined by the relation.

In case of $P = L_1 \bigcap L_2$ and $P = \Pi \bigcap L$ the degree of freedom of the system is reduced by 1 to enforce an intersection constraint between L_1 and L_2, or between P and L.

Line Table 8.2 lists of topological relations for a line.

Notation	Description	Degree of freedom of the first object	Degree of freedom of the system
L		4	4
$L \in \Pi$	A line in a plane	2	6
$L_1 \parallel L_2$	Two parallels line	2	6
$L_1 \perp L_2$	Two perpendicular line	2	6
$L \parallel \Pi$	A line parallel to a plane	2	6
$L = \Pi_1 \bigcap \Pi_2$ $(Pi_1 \neq Pi_2)$	The intersection of two planes	0	8

Table 8.2: Topological relations for a line with associated degree of freedom.

In table 8.2, the degree of freedom for $L_1 \perp L_2$ are expressed for the case where there is an intersection between the two lines, but this relation can be extended to the case where there is no intersection, which adds an extra degree of freedom: the distance between the lines.

Plane Table 8.3 lists of topological relations for a plane.

Notation	Description	Degree of freedom of the first object	Degree of freedom of the system
Π		3	3
$\Pi_1 \parallel \Pi_2$	Two parallels planes	1	5
$\Pi_1 \perp \Pi_2$	Two perpendicular planes	2	6

Table 8.3: Topological relations for a point with associated degree of freedom.

Line segment they have the same topological relations as lines (Table 8.2). One can also define a relation for sharing one extremity, which has a degree of freedom for one segment of 5 and for the system of 9. Segments can also be connected to two points: $S = [P_1, P_2]$.

8.2.3 Constructive relations and composite objects

Composite objects are defined as a set of geometric objects (see definition 2). There is a *constructive relation* between a *composite object* o and each of the object o_i that define this object.

Besides being directly defined by atomic objects, it is also possible to define a composite object with other composite objects, for instance a cube is made of eight square facets, each of which are made of a plane and a polygon made of lines segments. In this case, the composite object has a *constructive relation* with other composite objects.

Table 8.4 gives some examples of composite objects.

In an environment, it will not always be possible to detect all parts of an object, it is then possible to define virtual geometric objects, that complete an object. For instance, the bottom face of a box that lies on the ground cannot be observed.

Name	Description
Polygon	A polygon is a list of segment S_i, where $\forall i$, S_i and S_{i+1} share a common extremity
Facet	A facet F is made of a plane P which is the support for a polygon Po_F, such as $\forall S \in Po \Rightarrow S \in P$
Box	A box Bo is a volume of space, closed by a set of facets: $\forall F_i \in Bo \forall S \in Po_{F_i}, \exists ! F_j \in Bo / S \in Po_{F_j}$
Parallelepiped	A parallelepiped is a special case of facets, containing only six perpendicular facets: $\forall F_i \in Pa, \forall F_j \in Pa, Po_{F_i} \bigcap Po_{F_j} \neq \emptyset \Rightarrow F_i \perp F_j$
Cube	A cube Cu is a special case of parallelepiped, where all facets have the same size: $\exists s / \forall F \in Cu, \forall S \in F, l(S) = s$

Table 8.4: Most common composite objects.

Among the atomic objects, there is one that does not have a physical appearance, and cannot be detected by usual mean: the basis. It is still possible to have virtual basis, like the origin of the world, or the origin of a local map. Otherwise it is possible to define a basis using other elements. The following set of atomic objects define a basis:

- three points
- a point and a segment (or line)
- two segments
- two non parallel lines
- a plane and a point
- a plane and two points (included in the plane)
- a plane and a segment
- a plane and a line (non parallel to the plane)
- a facet (as defined in chapter 5)

8.2.4 Numeric relations

Since objects can either be represented as parameters (section 8.1.2) or as a set of other parametrisable atoms (section 8.2.3), it is possible to compute the following measures between two atoms: *angle*, *distance*, *coplanarity* and a *partial transformation vector* (a vector that links two atoms together, see section 8.3.

angle To be able to compute an angle between two objects o_1 and o_2, each object needs to be defined by a vector. Points does not include a vector and can not be used to compute an angle, for a line it is the vector director, and for a plane the normal.

Assuming that an object o_1 is defined by the vector $\overrightarrow{v_1}$ of covariance matrix M_1 and o_2 by $\overrightarrow{v_2}$ of covariance matrix M_2, then:

$$\theta(o_1, o_2) = acos\,(ip) \tag{8.14}$$

$$ip = \frac{< \overrightarrow{v_1}, \overrightarrow{v_2} >}{\| \overrightarrow{v_1} \| \| \overrightarrow{v_2} \|} \tag{8.15}$$

$$cov_{\theta(o_1,o_2)} = J_1 \cdot M_1 \cdot J_1^t + J_2 \cdot M_2 \cdot J_2^t \tag{8.16}$$

$$J_i = \frac{\overrightarrow{v_j}}{\sqrt{1 - ip^2}} \text{ where } (i = 1, j = 2) \text{ or } (i = 2, j = 1) \tag{8.17}$$

distance The distance between two points P_1 of covariance M_1 and P_2 of covariance M_2 is given by:

$$d(P_1, P_2) = \| P_1 - P_2 \| \tag{8.18}$$

$$cov_{d(P_1,P_2)} = J \cdot (M_1 + M_2) \cdot J^t \tag{8.19}$$

$$J = \frac{P_1 - P_2}{d(P_1, P_2)} \tag{8.20}$$

A general definition of the distance between two objects o_1 and o_2 is the minimal distance between a point of o_1 and a point of O_2:

$$\forall (P_1, P_2) \in o_1 \times o_2,\ d(o_1, o_2) \leq d(P_1, P_2) \tag{8.21}$$

$$\exists (P_1, P_2) \in o_1 \times o_2,\ d(o_1, o_2) = d(P_1, P_2) \tag{8.22}$$

The distance between a point and a line, or between a point and a plane is given by computing the projection of the point on the line or plane (see appendix A), then equation (8.20) is used to compute the distance between the point and its projection.

Appendice A contains more on the equations on how to compute the distance between two geometric objects.

Coplanarity Between two lines, using equation (8.9) it is possible to define a coefficient of coplanarity.

Composite objects Since composite objects are made of a set of atomic objects, it means that several values for the *angle*, *distance* and *partial transformation vector* could be computed, but for most of the composite objects it is possible to define a basis that can then be used to compute the measures.

Expressing topological relations from numerical relations Many of the topological relations defined in section 8.2.2 can be expressed with a numerical relation, as shown in table 8.5.

Topological relation	Numerical relation
$\mathring{a}_1 \in \mathring{a}_2$	$d(\mathring{a}_1, \mathring{a}_2) = 0$
$\mathring{a}_1 = \mathring{a}_2 \bigcap \mathring{a}_3$	$d(\mathring{a}_1, \mathring{a}_2) = 0$ and $d(\mathring{a}_1, \mathring{a}_3) = 0$
$\mathring{a}_1 \parallel \mathring{a}_2$	$angle(\mathring{a}_1, \mathring{a}_2) = 0$
$\mathring{a}_1 \perp \mathring{a}_2$	$angle(\mathring{a}_1, \mathring{a}_2) = 90°$

Table 8.5: Link between topological relations and numerical relations

8.2.5 Link between numeric relations and topological relations

Topological relations are a special case of a numeric relations, the equivalence is given by the following list:

- $P \in o \Leftrightarrow d(P, o) = 0$
- $P = o_1 \bigcap o_2 \Leftrightarrow d(P, o_1) = d(P, o_2) = 0$
- $L \in \Pi \Leftrightarrow d(L, \Pi) = 0$ and $\theta(L, \Pi) = \frac{\pi}{2}$
- $o_1 \parallel o_2 \Leftrightarrow \theta(o_1, o_2) = 0$
- $o_1 \perp o_2 \Leftrightarrow \theta(o_1, o_2) = \frac{\pi}{2}$
- $L \parallel \Pi \Leftrightarrow \theta(L, \Pi) = \frac{\pi}{2}$
- $L = \Pi_1 \bigcap \Pi_2 \Leftrightarrow d(L, \Pi_1) = d(L, \Pi_2) = 0$ and $\theta(L, \Pi_1) = \theta(L, \Pi_2) = \frac{\pi}{2}$

8.2.6 Scale relations

An object has a different appearance when it is seen from a close distance, it has more details, the *scale relation* connects geometric objects that belong to the same real-world object but seen from a different distance.

8.3 Partial transformation vector

Angles and distances allow to give a rough estimate of the locus of the location of a feature compared to one other. For instance, let us take two points P_1 and P_2, if the coordinates of P_1 and the distance $d = |\overrightarrow{P_1P_2}|$ are known, then we know that the point P_2 is included on the sphere centred on P_1 and of radius d, but the information on how to retrieve the location of the second point is lost.

Let us consider four points, P_1, P_2, P_3 and P_4, with P_2, P_3 and P_4 at the same distance $d = |\overrightarrow{P_1P_2}| = |\overrightarrow{P_1P_3}| = |\overrightarrow{P_1P_4}|$ from P_1. If we also now the distances between the other points, we can reconstruct the geometry (up to a given rotation). But let us assume that we do not know the distance between all those points, for instance, because we are only able to make measurements between P_1 and each of the other points (this would be the case in environment modelling if during the observation phases the other points are hidden from one other). Let us assume that later we can make an observation with all four points, and we want to be able to match each points. With only the knowledge of d, it is not possible to make a distinction

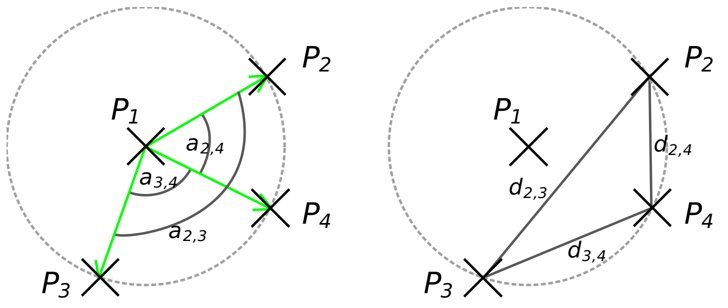


Figure 8.1: Angles between *partial transformation vector*

between P_2, P_3 and P_4, but if we have been also able to compute the angles $\widehat{\overrightarrow{P_1P_2}, \overrightarrow{P_1P_3}}$, $\widehat{\overrightarrow{P_1P_2}, \overrightarrow{P_1P_4}}$ and $\widehat{\overrightarrow{P_1P_3}, \overrightarrow{P_1P_4}}$ (which is possible since we can do a measurement of the vector $\overrightarrow{P_1P_2}$, $\overrightarrow{P_1P_3}$ and $\overrightarrow{P_1P_4}$), then it is possible to get the structure of the objects (see Figure 8.1).

We call that vector the *partial transformation vector*. Section 9.2.3 shows why such a vector is useful, and demonstrates the interest of having the information centred on the feature rather than relying only on distributed information, such as distance. The remaining of this section details how the *partial transformation vector* is computed for each type of feature.

Definition 6 *A* ***partial transformation vector*** *is a unique vector $\overrightarrow{ptv}(o_1, o_2)$, between two geometric objects o_1 and o_2, which is anchored on points of the two objects: $\exists(P_1, P_2) \in (o_1, o_2)$ such as $\overrightarrow{ptv}(o_1, o_2) = P_2 - P_1$.*

The anchors of the vector are stable by a transformation: given a transformation $\mathcal{T}$ (rotation and/or translation), if P_1 and P_2 are two anchors of $\overrightarrow{ptv}(o_1, o_2)$, then $\mathcal{T}(P_1)$ and $\mathcal{T}(P_2)$ are two anchors for $\overrightarrow{ptv}(\mathcal{T}(o_1), \mathcal{T}(o_2))$:

$$\overrightarrow{ptv}(o_1, o_2) = P_2 - P_1 \Leftrightarrow \overrightarrow{ptv}(\mathcal{T}(o_1), \mathcal{T}(o_2)) = \mathcal{T}(P_2) - \mathcal{T}(P_1) \tag{8.23}$$

The actual values of a *partial transformation vector* are dependent of the orientation of each feature, but the angle and anchor points between two of those vectors is not. From the definition, it comes that the *partial transformation vector* is antisymetric:

$$\overrightarrow{ptv}(o_1, o_2) = -\overrightarrow{ptv}(o_2, o_1) \tag{8.24}$$

Table 8.6 shows the possible *partial transformation vectors*, it is worth to note that it is not possible to define a $\overrightarrow{ptv}$ between a line and a plane, and between two planes.

8.3.1 Points

The *partial transformation vector* between two points P_1 and P_2 is simply defined by:

$$\overrightarrow{ptv}(P_1, P_2) = \overrightarrow{P_1P_2} \tag{8.25}$$

This definition fulfils equation (8.24), since $\overrightarrow{ptv}(P_1, P_2) = \overrightarrow{P_1P_2} = -\overrightarrow{P_2P_1} = -\overrightarrow{ptv}(P_2, P_1)$, and it is a unique vector.

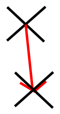
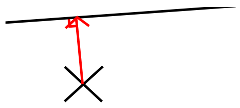
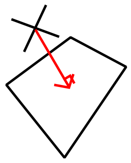
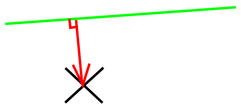
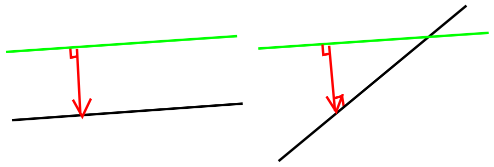
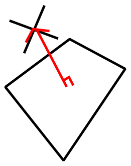
	points	lines	planes
points			
lines			
planes			

Table 8.6: *Partial transformation vector* for points, lines and planes.

The *partial transformation vector* between a point P and a line L is defined by:

$$\overrightarrow{ptv}(P, L) = \overrightarrow{Pproj(P, L)} \tag{8.26}$$

This definition fulfils equation (8.24), since $\overrightarrow{ptv}(P, L) = \overrightarrow{Pproj(P, L)} = -\overrightarrow{proj(P, L)P} = -\overrightarrow{ptv}(L, P)$, and it is an unique vector.

The *partial transformation vector* between a point and a facet is defined in the same way as for lines.

8.3.2 Lines

The *partial transformation vector* is undefined for two lines that intersect.

For two parallels line L_1 and L_2, given a point $P_1 \in L_1$, its projection on L_2 is $P_2 = proj(P_1, L_2)$, the *partial transformation vector* is then defined by:

$$\overrightarrow{ptv}(L_1, L_2) = \overrightarrow{P_1P_2} \tag{8.27}$$

Given that $proj(P_2, L_1) = P_1$, this definition fulfils equation (8.24). The choice of P_1 does not matter: $\forall P \in L_1$, $\overrightarrow{P_1P_2} = \overrightarrow{Pproj(P, L_2)}$, which means the uniqueness is guaranteed.

8.4 Geometric graph structure

8.4.1 Rudiments of graph theory

Definition 7 *A* ***graph*** $G = (V, E)$ *is a representation of a set of objects, called* ***vertices*** $V = \{v_i\}$*, that are linked together by* ***edges*** $E = \{e_{i,j} = (v_i, v_j)\}$*.*

The edges and vertices can contain information, used to discriminate edges or vertices one from the other. If the information is a number or a name, it is called a **label**, otherwise the information is called **attribute**. Edges can be oriented or not, if an edge is oriented then $e_{i,j} \neq e_{j,i}$.

Definition 8 *A **subgraph** $G^p = (V^p, E^p)$ of the graph $G = (V, E)$, is a graph such that $V^p \subset V$ and $E^p \subset E$.*

Definition 9 *A graph $G = (V, E)$ is called a **complete graph** if all vertices of V are connected by an edge of E to all other vertices:*

$$\forall v_i \in V,\ \forall v_j \in V,\ v_i \neq v_j \Rightarrow \exists e_{i,j} = (v_i, v_j) \in E \tag{8.28}$$

Definition 10 *A **clique** is a complete subgraph $G^p = (V^p, E^p)$ of the graph $G = (V, E)$. A **maximal clique** G^p is a clique for which it is not possible to add another vertices of the graph G that is connected to all other vertices of G^p:*

$$G^p \text{ is a } \textbf{maximal clique} \Leftrightarrow \forall v_i \in V \setminus V^p,\ \exists v_j \in V^p \text{ such as } e_{i,j} = (v_i, v_j) \notin E \tag{8.29}$$

Definition 11 *The **complement** graph $\overline{G} = (V, \overline{E}$ of the graph $G = (V, E)$ is the graph such as there is an edge $e_{i,j} = (v_i, v_j)$ in the graph $\overline{G}$ if and only if there is no edge $e_{i,j}$ in the graph G.*

8.4.2 Geometric vertices

A vertex of the graph represents a single geometric object o. If the object is an atom, then the vertex is labelled with the type and its attributes are the geometric parameters of the object o (later we will add to the attributes a geometric descriptor, in section 9.2). When the object is an output of a vision process, it is also possible to add a vision-based descriptor to the label.

While the vertices and their associated objects are two different concepts, the first one is a mathematical abstraction in a graph of the second is the geometric appearance of a landmark. Since there is a one to one mapping between vertices and geometric objects, it is convenient to refer directly to the object as being in the graph, and vice versa, to use the vertex to refer to the object in a geometric context.

8.4.3 Geometric edges

Edges in the graph correspond to relations between two objects represented by two vertices. For each type of relations defined in section 8.2, a type of edge can be defined.

Numeric edge The backbone of the graph is made by the *numeric edges* $e^n = (v_1, v_2)$, defined by the *numeric relation*, that connect the features together.

For each edge $e^n = (v_1, v_2)$, two *characteristic vectors* $\overrightarrow{c_1}$ and $\overrightarrow{c_2}$ are associated, computed with respect to each vertex v_1 and v_2. Given o_1 the object associated to v_1, and o_2 the object associated to v_2, if either o_1 or o_2 is a point, then the partial transformation vector is used and $\overrightarrow{c_1} = \overrightarrow{ptv}(o_1, o_2)$ and $\overrightarrow{c_2} = \overrightarrow{ptv}(o_2, o_1)$. If o_2 is a line, then $\overrightarrow{c_1}$ is equal to the vector director of the line o_2, and if o_2 is a plane, then $\overrightarrow{c_1}$ is equal to the normal of the plane of o_2.

The attributes of the edge $e^n = (v_1, v_2)$ are the distance between two features, and the cosinus of the angle. But it also contains the two characteristic vector $\overrightarrow{c_1}$ and $\overrightarrow{c_2}$.

Despite that the attributes of a *numeric edge* contains an information is dependent of the extremities of the edge, the *numeric edges* are not oriented in the graph and $e^n = (o_1, o_2) = (o_2, o_1)$, as the information carried by this edge is bidirectional.

Composite edge A *composite relation* defines an oriented *composite edge*. For a composite object o, made with the set of object o_i, there is an edge $e^c = (o, o_i)$ between o and all object o_i, o is the starting point of the edge, while o_i is the end.

Topological edge A *Topological relation* between two objects define a *topological edge* e^t, it is labelled with the type of topological information.

The intersection relation $o_1 = o_2 \bigcap o_3$ is equivalent to $o_1 \in o_2$ and $o_1 \in o_3$, therefore the intersection relation is represented in the graph using two belonging relations.

There are two types of *topological edges*, oriented edges and unoriented edges. Edges labelled with a belonging relation are oriented, while edges labelled with parallel or perpendicular relation are unoriented.

Scale edge The *scale relation* allows to define a *scale edge* which is also oriented from the coarser level to the more detailed level.

8.4.4 Geometric graph

A graph is made of the whole set of features of the environment, connected by the different edges of section 8.4.3. Figure 8.2 shows a graph with different type of features, and the label associated to edges and nodes. The graph $G = (V, E)$ of the environment can be split in four different subgraphs, one for each type of edge, the *numeric graph* $G^n = (V, E^n = e^n)$, the *topological graph* $G^t = (V, E^t = e^t)$, the *composite graph* $G^c = (V, E^c = e^c)$ and the *scale graph* $G^s = (V, E^s = e^s)$. Figure 8.3 shows the different types of connections and subgraphs.

The *topological graph* and *composite graph* are trivially non complete, since very few geometric objects are actually connected by *topological relations*, and atomic objects have no reason to be always part of a composite objects.

The *numeric graph* is not complete, which means that a vertex is connected to a limited set of vertices. The reason is that the number of edges in a complete graph of n vertices is equal to $\frac{n \cdot (n-1)}{2}$, so a complete graph would be possible to define in a small environment with very few landmarks, but it does not scale to large environment when the number of edges become large. Furthermore, many of the *numeric edges*, in a complete graph, would be meaningless, since the relative position of distant geometric objects would have a high uncertainty.

8.5 Geometric graph construction

In the previous section, we have presented the basic elements of the graph (vertices and edges) and how they are related to the geometric representations defined in section 8.2. But, using

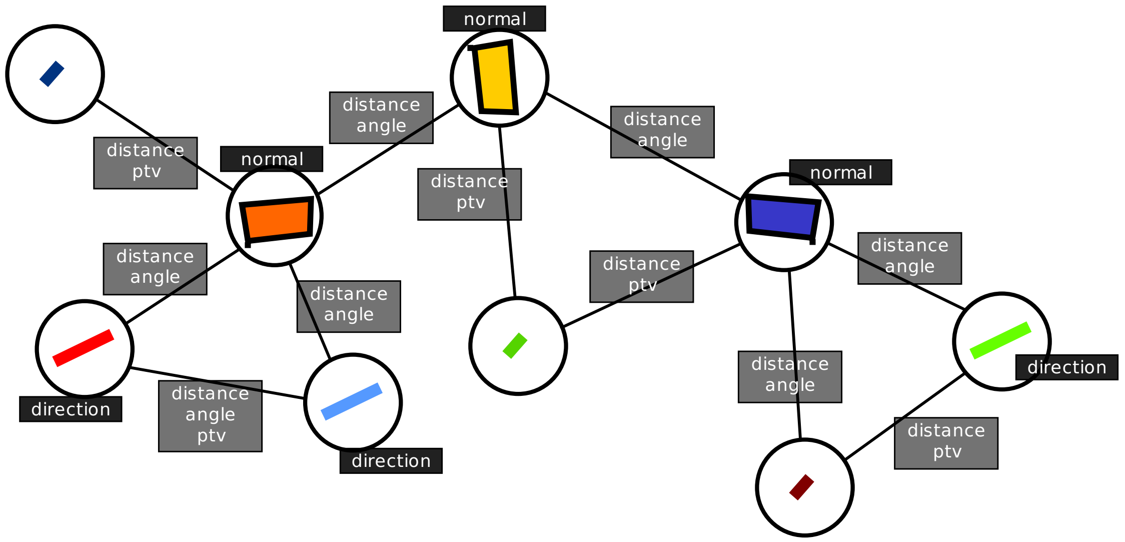


Figure 8.2: The graph with the information contained on the edges as well as on the node.

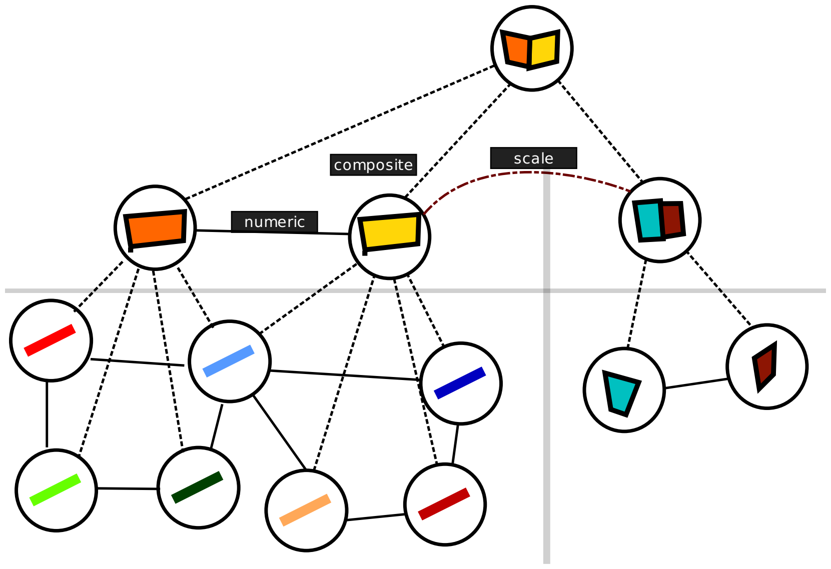


Figure 8.3: A graph with different types of edges

these elements, it is possible to construct many different graphs. Since we are interested in using the graph to solve the data association problem, the graph needs to be usable for matching purposes.

To use the graph for matching, it is important to ensure the repeatability of its construction: two observations of the same place in the environment needs to lead to two similar graphs. But it is also important that the graph contains meaningful information, for instance, if the parameters of the edge between two features are known with a big uncertainty, then this edge will appear similar to many other edges in the graph, which would increase the number of possible matches, and therefore the complexity of the matching process (it would also decrease the reliability of the result and increase the number of false positives).

In [Bailey, 2002], a complete graph (as defined in 9) of the environment is used. It offers the guarantee of the repeatability of the construction, but as mentioned in section 8.4.4, the number of edges in the graph is in $o(n^2)$, and many edges between distant features will have an important uncertainty.

But, since there is no way to guarantee the repeatability of the detection of features, the matching algorithm will need to be robust to occlusions, so it should be able to be resistant to small variations in the construction of the graph. This is why in this section we present how we construct a partial graph of the environment.

Section 8.5.1 covers the insertion in the graph of a model of the environment, known beforehand, while section 8.5.2 covers the case of the insertion and update of the graph from information coming from a multi-map environment (as described in section 1.3).

8.5.1 Global construction

Global construction is relatively straightforward, usually a model of the environment coming from a geographic information system, such as [IGN,], contains a list of features in a global frame, with an associated error on the parameters of the features.

The following algorithm will create the graph $G = (V, E)$ for such a model:

```
Given the set of objects $\mathcal{O} = \{o_i\}$ in the model:

  1. insert in the graph, a vertex $v_i$ for each object $o_i \in \mathcal{O}$

  2. for all vertex $v_i \in V$, associated to object $o_i$, and for all vertex $v_j \in V / j \neq i$, given $o_j$
     the object associated to $v_j$, if $d(o_i, o_j) + 3\sigma_{d(o_i,o_j)} < \mathcal{T}_d$ connect the two vertices with a
     *proximity edge*
```

Where $\sigma_{d(o_i,o_j)}$ is the uncertainty associated with the distance $d(o_i, o_j)$. For a given model, this algorithm defines a unique graph.

8.5.2 Incremental construction and update

The challenge of constructing a graph from a map built by a mobile robot is to ensure that the *numeric edges* are meaningful, in other words, that the relative relation between two features remains known with a high level of confidence. Assuming there is no loop closure, the problem with a SLAM approach is that the error on the position of the robot increases when the robot moves in the map, therefore the uncertainty on the position of the landmarks also increases, and since the relative relations are computed from these positions, it makes it problematic to compute meaningful information.

The problem of the increase of uncertainty can be kept under control using a multi-map approach like in [Estrada et al., 2005] (see section 1.2), since in multi-map, the error on the position of the robot, and the local error on the position of the landmarks, is kept small. This means that the incremental construction can only compute reliable relations between landmarks that belong to the same local map.

It is important to note that two objects o_1 and o_2 can be available in two different maps M_1 and M_2 ($o_1 \in M_1$, $o_1 \in M_2$, $o_2 \in M_1$ and $o_2 \in M_2$), in which case the map where the relative transformation is the smallest is used to label the *numeric edge* between o_1 and o_2.

The following algorithm corresponds to the SPAtial Feature rectangle in figure 1.6, it takes as input a finished map $\mathcal{M} = o_i$, and update the graph $G = (V, E)$.

---

1. $\forall o_i \in \mathcal{M}$, if there is no vertex in V corresponding to o_i, add a new vertex v_i in the graph G
2. $\forall (o_i, o_j) \in \mathcal{M}^2 / i \neq j$:
   - if there is already an edge between $e^n_{i,j} = (v_i, v_j)$, check if the transformation given by the new map is more accurate, and then update the edge $e^n_{i,j}$. Then check that $d(o_i, o_j) + 3\sigma_{d(o_i,o_j)} < \mathcal{T}_d$ is still true, if it is not then the edge $e^n_{i,j}$ is removed
   - if there is no existing edge, and if $d(o_i, o_j) + 3\sigma_{d(o_i,o_j)} < \mathcal{T}_d$, then add an edge $e^n_{i,j} = (v_i, v_j)$

Chapter 9

Graph matching

The previous chapter shows how the features in the environment can be connected together in a graph that exhibits the relations between the geometric features. As explained in [Bailey and Durrant-Whyte, 2006], it is important to use the neighborhood of features to get an accurate match. The geometric graph allow an easier manipulation of that neighborhood.

Section 9.1 contains a review of existing algorithm for graph matching, as well as an explanation of why we think they are inadequate for solving the problem of spatial graph matching. The algorithm we are proposing relies on finding an initial set of matches, and then extending this set. To find the initial matches, the use of a geometric descriptor is proposed in section 9.2, and an algorithm to select the seeds using the descriptors is described in section 9.3. Then the seed and growth approach that has been used for interest points in section 4.3 and facets in section 5.2 is extended to all types of features in section 9.4

9.1 Graph match metric

Inexact graph matching Graph matching is an active research field, since graphs are an efficient way to represent objects, and relations between them, being able to match graphs allow to match different objects. There are applications to graph matching in many fields, such as computer vision, scene and object recognition, chemistry and biology.

The problem of *exact graph matching* between two graphs $G^{obs} = (V^{obs}, E^{obs})$ and $G^{db} = (V^{db}, E^{db})$ is to find a mapping f between the vertices of V^{obs} and the ones of V^{db}, such as for all $(u, v) \in E^{obs}$, $(f(u), f(v)) \in E^{db}$. When such a mapping f is an isomorphism, G^{obs} and G^{db} are isomorph. But the mapping f can exist only if the two graphs have the same number of elements $card(V^{obs}) = card(V^{db})$, which does not happen for matching the graph of the environment with an observation graph. Assuming $card(V^{obs}) < card(V^{db})$, the *inexact graph matching* problem can then be transformed in an exact problem by finding a match between G^{obs} and a subgraph of G^{db}. But in the case of the SLAM problem, the model of the environment is not complete, in other words, the robot can observe new features, which means there are features in V^{obs} that are not available in V^{db}, the problem at hand is then to match a subgraph of G^{obs} to a subgraph of G^{db}.

In [Abdulkader, 1998], the inexact graph matching has been proven to be a NP-complex problem, which means that a solution to the matching can be verified in polynomial time, but there is no efficient algorithm to find a solution. [Skiena, 2008] lists the algorithm that gives the best answer to the inexact graph matching. But all these algorithms have an exponential computational cost, which makes them impractical for use to match graphs of the size of an environment graph, in a context of a robot that needs to get match results quickly.

Graph matching for SLAM In the context of the SLAM problem, graph matching has been applied to solve the loop closure problem in [Bailey, 2002]. As explained in chapter 3, this algorithm uses a complete graph of the environment, initial matches are found by comparing the attributes and labels of the nodes, then all the matches with similar edges are connected together in a match graph, finally the set of good matches is selected as the maximum clique of the match graph. This raises several problems, the use of complete graph does not scale to a large environment, as mentioned in section 8.5, and the search of the maximum clique problem is also a NP-complex problem.

Overview of our approach Compared to [Bailey, 2002], our algorithm for matching works on a partial graph, the first step is very similar and generates initial matches using the attributes of the node, and similarly we construct a graph of matches with similar edges. Since finding the maximum clique would not work on a partial graph, our algorithm makes an approximation. The last stage of the algorithm uses the graph structure to find other matches:

1. Find an initial list of seeds
   (a) compute an initial set of matches using a geometric descriptor
   (b) generate a joint compatibility graph (a graph connecting two matches with similar edges) and an exclusive graph (a graph that connect two matches that share at least one node)
   (c) compute a transformation $\mathcal{T}$ between the observed features and the data base, and use that transformation to find the list of seeds
2. Complete the initial set of matches using the transformation $\mathcal{T}$ and a seed-and-growth approach.

Numeric Edge comparison To compare two numeric edges, the Mahalanobis distance is used on the distance and cosinus. For two edges e_1 (with distance d_1 and cosinus c_1) and e_2 (with distance d_2 and cosinus c_2), σ_x is the standard deviation of the value x, the condition of similarity for the two edges is:

$$e_1 \sim e_2 \Leftrightarrow \frac{(d_2 - d_1)^2}{\sigma_{d_1}^2 + \sigma_{d_2}^2} + \frac{(c_2 - c_1)^2}{\sigma_{c_1}^2 + \sigma_{c_2}^2} < \tau^2 \tag{9.1}$$

With $\tau = 9$, corresponding to a probability of 90% that the two edges are equals.

9.2 A Geometric descriptor for fast seed selection

An exhaustive search with the a "seed and grow" algorithm forces to look for each potential seed with each other node in the graph. Since this would be very costly, it is important to find a method to have a limited number of seeds. For interest points (see Chapter 4), or for facets (see Chapter 5), we have been able to select the seed using a signal-based descriptor, comparing the texture of the detected features with the content of the database.

But it is not always possible to use such a descriptor, for instance, there is currently no good segments signal based descriptor (see in the perspective, section 2.1). And in a multiple sensors system, the signal information is not usable from one type of sensor to an other. It is also possible that the information might not be available, for instance if the robot uses a pre-existing model. An other case where signal information fails is when there is an important change of viewpoint, which makes it impossible to use signal information for mapping with a ground robot and aerial robot, for instance.

Hence the need for a descriptor that does not rely on signal information, but on information that is available all the time, and is of same nature for each source of information, such as geometric information.

We list the properties of a good geometric descriptor (section 9.2.1), then we present existing descriptors for a shape made of a 3D points cloud (section 9.2.2). Then we see how to extend these descriptors to other types of features, like segments and planes (section 9.2.3), and how to weight the vote for an histogram using uncertainties (section 9.2.4). In section 9.2.5, we put all the pieces together and show how to compute the descriptor, and section 9.2.7 features some results obtained with the descriptor.

9.2.1 Properties of a good geometric descriptor

Importance of the neighbourhood The geometric information of a single feature is not good enough, for instance, points are sizeless, so individually they have the same geometric appearance. This is also true for segments or facets, since because of the detection process and of occultation the boundary of segments and facets is not accurate.

But the surrounding of features carry interesting information, for instance, the descriptor has to be able to distinguish a segment that is isolated, from one where many segments were detected around it (Figure 9.1).

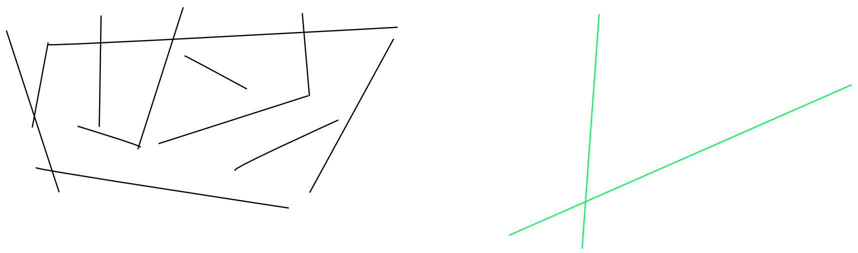


Figure 9.1: In this case the green segments are isolated, so if the robot is detecting two isolated segments, then it can deduce that it is not in the area with the black segments, and therefore it should not try to match the two isolated segment with the black segments.

Use of metric information So a geometric descriptor should count the number of each type of features. But this is not enough, since a given environment is bound to contain a lot of features with the same number of features types in the neighbourhood, for instance consider a set of three points on a line (first point at $x_1 = 0$, second at $x_2 = 2$ and last at $x_3 = 3$), all points have the same numbers of points in their neighbourhood.

That is why a good geometric descriptor must also take into account other geometric information such as distance, angles, or partial transformation vector (see section 8.3). For our example with three points and with a geometric descriptor that contains the list of distances, the descriptor of the first point is $d_1 = (2, 3)$, the second point is $d_2 = (1, 2)$ and for the last one $d_3 = (1, 3)$, all of them are different.

Uncertainty The geometric parameters of features are estimated with a known uncertainty, which means that the descriptor has to take this uncertainty into account.

Occultation Since it is not possible to guarantee a full observability of the feature, and of its surrounding at all time, the descriptor has to be able to rely on the parameters that are always observable, and to remain accurate when a feature is not visible.

Fast comparison Since the descriptor is supposed to help finding a seed, comparing descriptor, and finding the best set of candidates needs to be done quickly. This can be ensured if the size of the descriptor is fixed and the comparison is done by computing a distance. For efficient matching and retrieval, it should be possible to store it in a kd-tree [Beis and Lowe, 1997].

9.2.2 Shape descriptors for 3D Clouds

In case of a cloud of 3D points, the *Extended Gaussian Image* was introduced in [Horn, 1984], it relies on the use of a histogram (see Section 2.3.4) of the direction of each triangle weighted by the surface of the triangle, this histogram is invariant by translation, but rotates with the object. To make the *Extended Gaussian Image* invariant to rotation, in [Kang and Ikeuchi, 1991] Kang suggests to use a complex value for the voting, where the area of the triangle is the amplitude and the normal distance from the triangle to the center of mass is the phase.

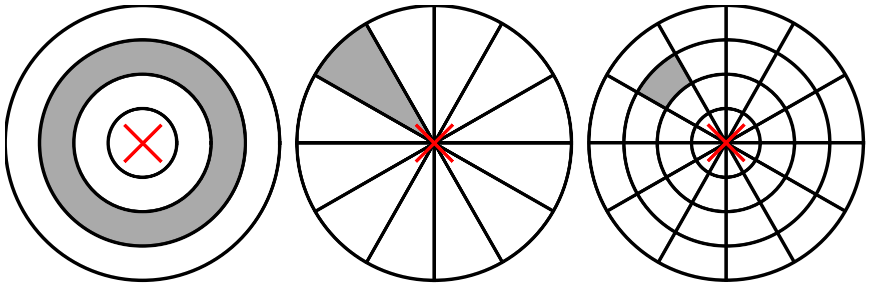


Figure 9.2: Possible division of space for a shape histogram, from left to right: shell, sectors and spiderweb.

For shape retrieval in a spatial database, a *shape histogram* was introduced in [Ankerst et al., 1999], the space is divided around the center of mass of an object (either using a shell, sectors or spider pattern, see figure 9.2), the value of each bin is the number of points present in a given sector. For a shell division, the out-most shell is unbounded, this representation is invariant to rotation. *Shape distributions* were proposed as generalization of *shape histogram* in [Osada et al., 2001], instead of computing the distances with the center of mass, the histogram is computed using either the distance between two random points or the surface of

	Points	Segments	Facets
Points	d, ptv	d, ptv	d, ptv
Segments		d, θ, ptv	d (unreliable), θ
Facets			d (unreliable), θ

Table 9.1: This table show what kind of relations can be computed for each type of features, d (*distance*), θ (*angle*) and *ptv* (*partial transformation vector*).

the triangle made by three random points or the volume contained in the tetrahedron made of four random points.

In [Kazhdan et al., 2004], Kazhdan introduces a method to measure the level of reflective and rotational symmetry around an axis, the measures around different axes give a descriptor for the shape. The level of symmetry is given by the distance between the symmetric of an object and the closest real object.

9.2.3 Generalization to different types of features

All the existing shape descriptors [Tangelder and Veltkamp, 2004] are used to describe the shape of a single object, made of a cloud of points. This makes it straightforward to extend them to be a geometric descriptor for an environment model made of points, instead of using the center of mass as a reference, the point feature is the center of the descriptor, and the cloud of points is the set of neighbour features.

We are going to base our generalisation using the *shape descriptor* of [Ankerst et al., 1999], since this is a descriptor that relies on angles and distances, which as we have shown in section 8.2.4 can be computed for all types of features.

Table 9.1 show which kind of relations can be computed between each type of features. It is worth to note that the distance between segments and facets, and facets and facets is unreliable because it would rely on the boundary of each type of facets. There are two ways to compute a distance between two segments, a reliable one which is computed between the supporting lines of the segment, and an unreliable one, computed between the two segments, which is a better indication of proximity than infinite line, but requires a full observability of the segment extremities.

Characteristic vector and anchor It would also be interesting to include in the geometric feature descriptor the relation between the neighboring features. For a given atom $\mathring{a}_0$, with two neighbors atoms $\mathring{a}_1$ and $\mathring{a}_2$, the geometric descriptor of $\mathring{a}_0$ should also contain information of how $\mathring{a}_1$ relates to $\mathring{a}_2$, a distance can always be computed, but using the partial transformation vector for points (section 8.3), the vector director of segments and the normal of planes, we can also compute an angle and another distance.

As shown in section 8.3, the partial transformation vector is really well defined only for points, it can be extended to provide a meaning in case of line segments that do not intersect, but it is not possible to define a good one for a plane. That is why for segments and planes it is easier to use the vector director and the normal. The partial transformation vector is also associated to an anchor, for a point the anchor is the projection of the point on the other atom. In case of two segments S_1 and S_2, the anchor would be defined as the closest point of segment S_2 on segment S_1, and in case of parallel line, a point at the infinite, in case of a segment and plane, the anchor point can be defined in the same way. But for two planes Π_1 and Π_2, the anchor has to be a line, which is fine since we are going to compute distances between anchors to fill the histogram.

The following list sums up how to define the characteristic vector and its anchor point between an atom $\mathring{a}_0$ and $\mathring{a}_1$:

- when either $\mathring{a}_0$ or $\mathring{a}_1$ is a point, the partial transformation vector (section 8.3) can be used
- when either $\mathring{a}_0$ or $\mathring{a}_1$ is a segment (and none a point), the direction vector or the normal is used, the anchor is the closest point between the two atoms
- when both atoms are planes, the normal is used and the anchor is the intersection line

It is worth to note that if the feature for which we compute the descriptor is a point, then the distance between anchors is always equal to 0. And that if it is a plane, all the partial transformation vector of points are going to be parallel.

The *characteristic vector* is transformed into an histogram, by computing the angle between all the vectors, and the distance between the anchors.

Occupancy histogram An occupancy histogram is made by having each neighbour feature vote for the cell they are occupying. For instance a point vote for the cell at the distance from the object, while a segment or facet vote for multiple cells.

An atom $\mathring{a}$ vote for a cell $\mathcal{C}_k(\mathring{a}_0)$ of the occupancy histogram of the atom $\mathring{a}_0$ if

$$\mathring{a} \bigcap \mathcal{C}_k(\mathring{a}_0) \neq \emptyset \tag{9.2}$$

A cell for the occupancy histogram is defined by

$$\forall k \in [0; n-1[, \mathcal{C}_k(\mathring{a}_0) = \{P \mid k \cdot s < d(P, \mathring{a}_0) < (k+1) \cdot s\} \tag{9.3}$$
$$\mathcal{C}_n(\mathring{a}_0) = \{P \mid n \cdot s < d(P, \mathring{a}_0)\} \tag{9.4}$$

An other way to define the occupancy histogram is to compute the minimum and maximum distances between a point of $\mathring{a}$ and $\mathring{a}_0$, and then to use this interval to vote for each cell. These minimum and maximum distances define an interval called *the distance interval*: for an atom $\mathring{a}$ in respect of an atom $\mathring{a}_0$ is the interval $[d_1, d_2]$ such as:

$$d_1 = max(\delta \mid \forall P \in \mathring{a},\ \delta < d(P, \mathring{a}_0) \tag{9.5}$$
$$d_2 = min(\delta \mid \forall P \in \mathring{a},\ d(P, \mathring{a}_0) < \delta \tag{9.6}$$

This histogram allows to recover information when a feature is split, but it is too much dependent on the boundaries, and extremities of segments and planes are not reliably extracted. For points, the occupancy histogram is equal to the distance histogram.

List of possible histograms :

- distance
- support distance

- angle
- distance between neighbour features
- angle between characteristic vectors
- distance between anchor points
- occupancy histogram

9.2.4 Uncertain weighted histogram

In case of a dense cloud of 3D points the noise can be considered as negligible with respect to the number of points. But in our model of the environment, the number of features is very small, this increases the sensitivity to noise. And the discretisation of the histogram amplifies the problems: let's take the example of a two bins histogram $]-\infty, 0.0]$ and $[0.0, +\infty[$, the measurement $\mu = 0.05$ with a standard deviation of $\sigma = 0.1$ would contribute fully to the second bin, while there is a probability of 0.30854 that the actual measurement is inferior to 0.0 and belongs to the first bin. Using uncertainty to weight the histogram, the first bin would receive a vote of 0.300854 and the second one 0.699146 (see Figure 9.3 to see how a Gaussian covers an histogram).

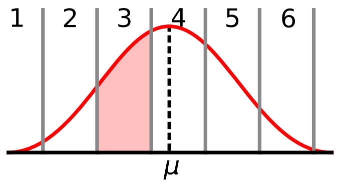


Figure 9.3: This shows the Gaussian distribution in (red) centred around the value μ. Using a "classical" histogram, only the fourth bin is incremented, with an "uncertain weighted" histogram, each bin is incremented by the corresponding surface of the Gaussian.

Gaussian weight The Gaussian distribution is given by:

$$E_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{9.7}$$

Which means that the probability that the measurement is inferior to x is given by the following equation:

$$\begin{aligned} P_{\mu,\sigma}(x) &= \int_{-\infty}^{x} E_{\mu,\sigma}(t)dt & (9.8) \\ &= \frac{1}{2}\left[1 + erf\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right] & (9.9) \\ \lim_{x\to\infty}(P_{\mu,\sigma}(x)) &= 1 \end{aligned}$$

As shown in equation (2.28) of section 2.3.4, the weight of the vote for a bin in the interval $I_{a,b} = [a, b]$, corresponds to the area below the Gaussian in the interval (Figure 9.3), this is given by the equation:

$$W_{\mu,\sigma}(a,b) = P_{\mu,\sigma}(b) - P_{\mu,\sigma}(a) \tag{9.10}$$

$$\sum_{I_{a,b}} W_{\mu,\sigma}(a,b) = 1 \tag{9.11}$$

Uncertain weighted occupancy histogram To compute the uncertain weighted occupancy histogram for a given feature å, the distance interval $[d_1, d_2]$ ((9.5) and (9.6)) is used, if σ_{d_1} and σ_{d_2} are the standard deviations of d_1 and d_2, then the vote function $f(\delta)$ is given by:

$$f_{\mu,\sigma}(\delta) = \begin{cases} \delta < \mu, & e^{-\frac{(x-\mu)^2}{2\sigma^2}} \\ \delta \geq \mu, & 2.0 - e^{-\frac{(x-\mu)^2}{2\sigma^2}} \end{cases} \tag{9.12}$$

$$f(\delta) = \frac{1}{(\sigma_{d_2} + \sigma_{d_1})\sqrt{2\pi}}(f_{d_1,\sigma_{d_1}} - f_{d_2,\sigma_{d_2}}) \tag{9.13}$$

This function $f(\delta)$ is continuous, since $\lim_{\delta \to d_1^-} f_{\mu,\sigma}(\delta) = 1.0 = \lim_{\delta \to d_1^+} f_{\mu,\sigma}(\delta)$ and an other interesting property is that when $\sigma_{d_1} = \sigma_{d_2}$, then $\int_{-\infty}^{\infty} f(t)dt = d_2 - d_1$, which means that for a segment the total vote to the occupancy histogram is equal to the length of the segment.

The figure 9.4 show the vote curve for the histogram.

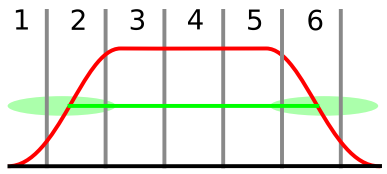


Figure 9.4: The uncertain weighted occupancy histogram means that the cell around the extremities do not get a full vote. The green line shows the two extreme distances with the uncertainty on the distance, while the red curve shows the value of the vote.

9.2.5 A Geometric Descriptor

The geometric descriptor is made of a selection of the histograms of section 9.2.3, computed using uncertain weighted histogram as explained in the previous section. The information needed to compute these histograms is available in the graph as shown on figure 8.2 of section 8.4.4. The choice of the histogram require experimentation and probably depend on the experiment and the type of matching needed.

9.2.6 Descriptor comparison

Since the histograms can be computed with a different number of neighbour features (due to occultation), to compute the comparison between descriptors we use the zero mean normalised cross correlation coefficient described in section 2.3.5.

9.2.7 Results

To validate the use of the descriptor, we made several experiments using synthetic segments. The goal is to show that the descriptor is robust to noise and to a certain extent to occultations, but also that the descriptor gives a good description of the neighborghood of the geometric objects.

For each experiment, we synthesize a set of features, we select one feature as the center of the descriptor, then we generate the edges between that feature and the other features, to compute many different descriptors. Then we can compute the correlation between all the descriptors and a selected descriptor.

When looking at the graph of the result, it is important to keep in mind that a score of 0.0 means that two histograms are uncorrelated, while a score of 1.0 means the histograms are equal (section 2.3.2).

The results usually show the result of histograms comparison for two standard deviations, $0.1m$ and $0.01m$, which corresponds to the range of error obtained in the slam process. The standard deviation is the error for the point or for the extremities of the segments.

Experiment #1: rotating segments The goal of this experiment is to demonstrate how the information on the angle of segments is represented by the histogram, this is simulating the corner of a building. So all segments are starting from the origin, and the end points are located on half of a sphere, the vector director of all segments is given by:

$$(\theta, \phi) \in [0, \pi] \times [0, \frac{\pi}{2}],\ (cos(\theta), sin(\theta), cos(\phi)) \tag{9.14}$$

For practical reason, the angles θ and ϕ are discretised:

$$i \in [0, 17] \cap \mathbb{N},\ \theta = \frac{i}{18} \cdot \pi \tag{9.15}$$

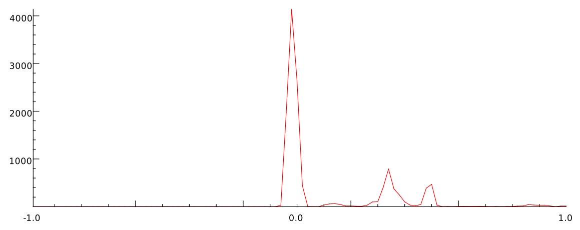
$$j \in [0, 8] \cap \mathbb{N},\ \phi = \frac{i}{9} \cdot \frac{\pi}{2} \tag{9.16}$$

Then we will consider all the possible corners using $S_{i=j=0}$ as the central segment. We create an edge between $S_{i=j=0}$ and two other segments S_{i_1,j_1} and S_{i_2,j_2}, then we compute the descriptor for $S_{i=j=0}$, since there are $18 * 9 = 162$ segments, we have 12880 possible corners and descriptors. Then a random corner is chosen as the reference, and the comparison of its histogram with all other descriptors is computed.

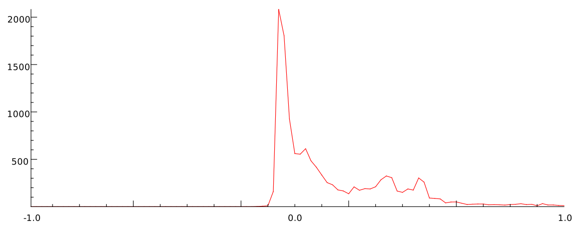
Figure 9.5 shows the curve of the correlation score, with a strong peak around 0.0 showing uncorrelated data, while on figure 9.6 the comparison score when only the angle information or rotation between characteristic vectors is used. This experiment shows that for segments, the descriptor is quite efficient at eliminating most of the possible configuration, and as shown on figure 9.7, the configuration with a high correlation score are very close to the reference configuration.

Experiment #2: points on a sphere We did the same experience as previously, but using points instead. The central point, that is used to compute the descriptor, is the origin $(0, 0, 0)$. And 162 points are equally dispatched on the quarter of a sphere. This time, three neighbour points are used to create the graph.

The results are shown on figure 9.8. Again, they show a peak of uncorrelated configuration around 0 and demonstrates the selectiveness of the descriptor.

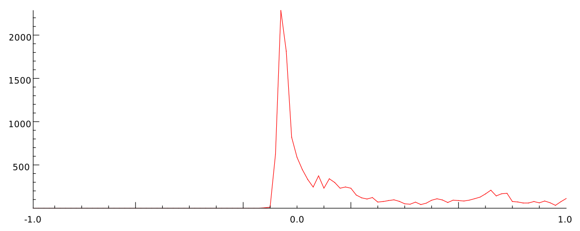


(a) $\sigma = 0.01m$

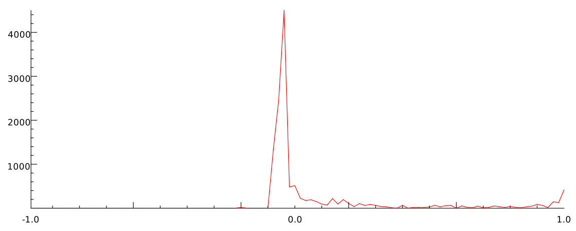


(b) $\sigma = 0.1m$

Figure 9.5: Histogram of the correlation score between the descriptor for a segment with other segments, using an angle and the characteristic vector histograms. The horizontal axis show the correlation score between $[-1, 1]$ and the vertical axis the number of features for a given score.



(a) Angle



(b) Characteristic Vector

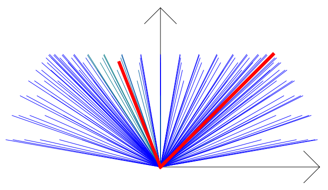
Figure 9.6: $\sigma = 0.01m$ Histogram of the correlation score between the descriptor for a segment with other segments, using an angle and characteristic vector histograms.

Experiment #3: points on multiple spheres This experiment demonstrates the use of the distance histogram, so we dispatch points on 5 spheres. 15931 configurations are generated for this experiment, figure 9.9 show the correlation results.

Experiment #4: Robustness to noise For this experience, we want to demonstrate that the descriptor is robust to the noise. To do this we use three segments, and using a normal distribution we add noise to those segments to generate 10000 configurations.

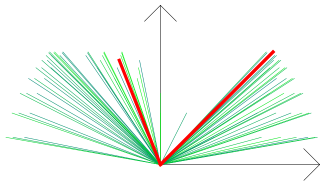
Figure 9.10 shows that most configurations have a score higher of 0.98 for a standard deviation of $0.01m$ and higher than 0.9 for a standard deviation of $0.1m$.

Experiment #5: occultation This experiment demonstrates how the descriptor behave when segments are removed from the configuration. For this experiment, we use the same set of segments as in the first experiment, except that this time the reference configuration contains all the 160 segments, and we generate 157 other configurations by removing one segment until only 3 segments are left. As shown on figure 9.11, the correlation score decreases slowly when the number of segments decreases, and the error on the extremities of a segment makes no difference for the occultation.



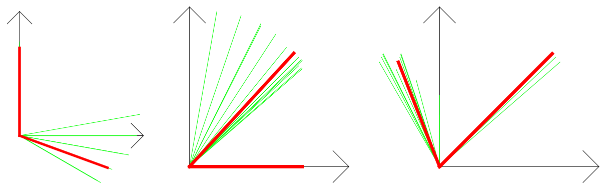
(a) XY (b) YZ (c) ZX

$c < 0.5$



(d) XY (e) YZ (f) ZX

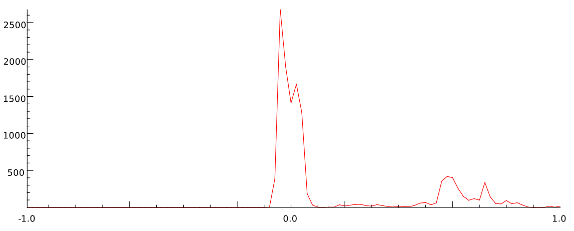
$0.5 < c < 0.95$



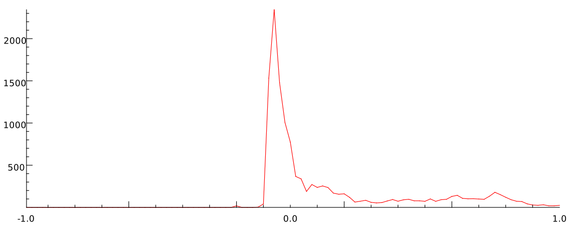
(g) XY (h) YZ (i) ZX

$0.95 < c$

Figure 9.7: Projection of segments on the XY, YZ and ZX planes, the red segments are the three segments of the reference configuration, the greener the segment is the higher correlation score c the configuration has with the reference one, the bluer the lower.

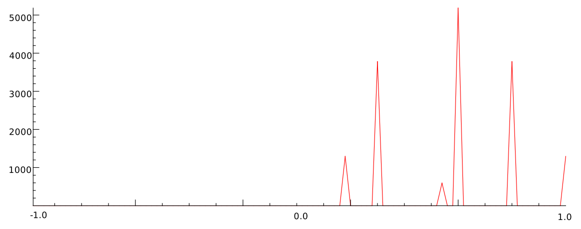


(a) $\sigma = 0.01m$

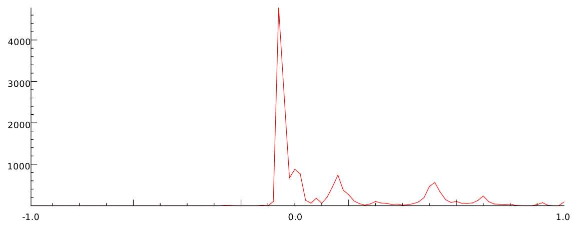


(b) $\sigma = 0.1m$

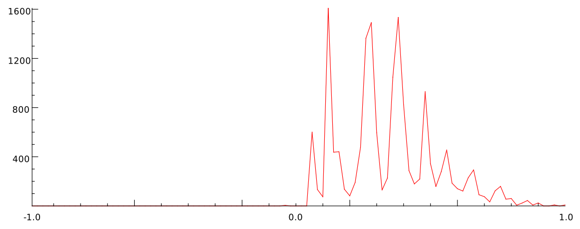
Figure 9.8: Points on a sphere, at equal distance of the origin, only the histogram of the characteristic vector is used.



(a) Distance

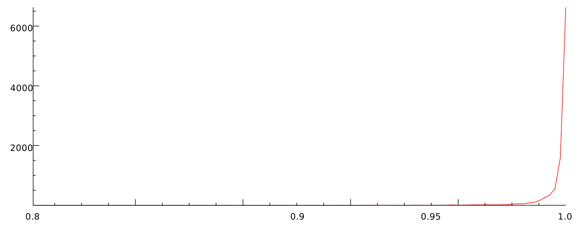


(b) Characteristic vector

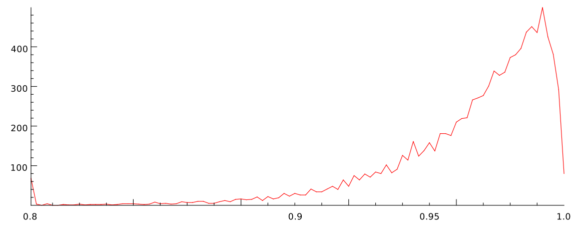


(c) Distance and PTV

Figure 9.9: Cloud of points at different distance using a standard deviation of $\sigma = 0.01m$

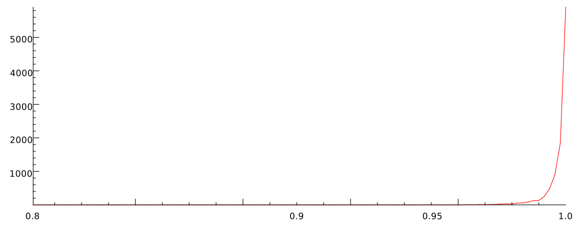


(a) $\sigma = 0.01m$

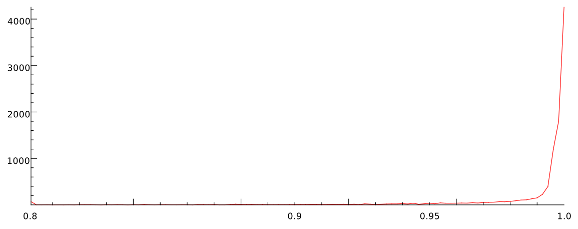


(b) $\sigma = 0.1m$

Distance, angle and characteristic vector

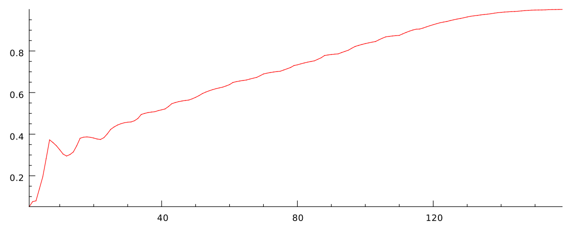


(c) Angle at $\sigma = 0.01m$

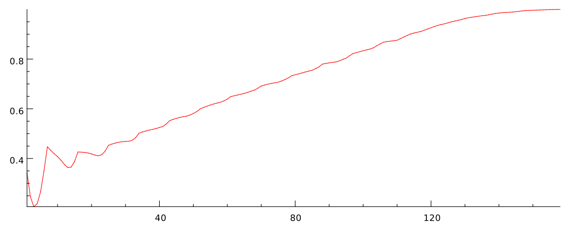


(d) Characteristic vector at $\sigma = 0.01m$

Figure 9.10: Robustness to noise, the interval for the correlation is $[0.8; 1.0]$



(a) angle and characteristic vector at $\sigma = 0.01m$

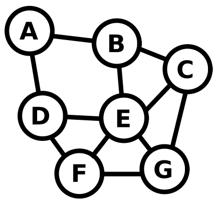


(b) angle and characteristic vector at $\sigma = 0.1m$

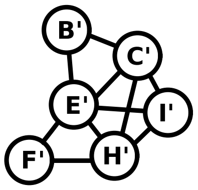
Figure 9.11: Occultation of segments. The horizontal axis show the number of segments in the configuration, while the vertical show the correlation

9.3 Seeds selection

Given the graph of observations G^{obs} and the graph of the features in the model G^{db} (also called reference graph), the first step of our algorithm requires to find a set of initial matches, commonly called *seeds*.



(a) G^{db}



(b) G^{obs}

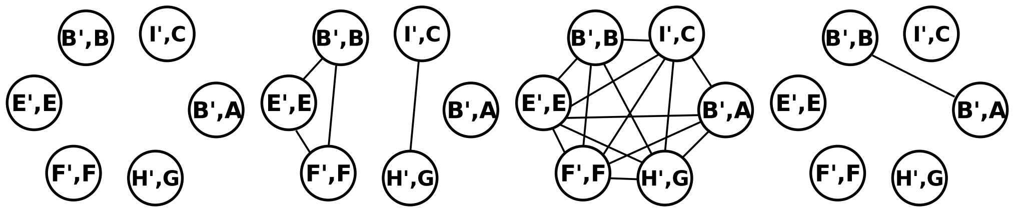
Figure 9.12: Example of graphs, on the left observation, on the right reference

The seeds selection algorithm uses the descriptor to find an initial set of matches. Then the next two steps of the algorithms are about generating two graphs that will be used to determine which nodes are good matches. The *joint compatibility graph* connect matches that are related to each other, matches with an edge in both the observed graph G^{obs} and in the reference graph G^{db}. While the *inclusive graph* connects matches that share the same vertex. The last step computes a transformation $\mathcal{T}$ between the observed features and the reference features.

Initial matches set Given two graphs of features $G^{obs} = (V^{obs}, E^{obs})$ and $G^{db} = (V^{db}, E^{db})$, we note $D(v)$ the descriptor of the vertex v. The initial set of matches $\mathcal{M}$ (Figure 9.13(a)) is given by:

$$\mathcal{M} = \left\{ m_{i,j} = (v_i^{obs}, v_j^{db}) \in V^{obs} \times V^{db} / \begin{array}{l} ZNCC(D(v_i^{obs}, v_j^{db}) > \mathcal{T}_D \\ type(v_i^{obs}) = type(v_j^{db}) \end{array} \right\} \tag{9.17}$$

$type(v)$ return the type of the feature, the condition $type(v_i^{obs}) = type(v_j^{db})$ ensures that the type of the match is consistent, and that a point is associated to a point, and not to other type of features.



(a) Initial matches set (b) Joint Compatibility Graph (c) Inclusive graph (d) Exclusive graph

Figure 9.13: Example of an initial set and of the induced graphes.

Joint compatibility graph The *joint compatibility graph* G^{common} (Figure 9.13(b)) is used to connect together matches whose vertex are connected together in the geometric graphs G^{obs} and G^{db}.

$G^{common} = (\mathcal{M}, E^{common})$, with an edge $e^{common}_{(i,j)\leftrightarrow(k,l)} = (m_{i,j}, m_{k,l})$ if and only if there is an edge $e^{obs}_{i,k} \in E^{obs}$ and an edge $e^{db}_{j,l} \in E^{db}$, and if the edge e^{obs} is compatible with e^{db}.

By definition of the graph, the vertices inside the *joint compatibility graph* are grouped by cluster of possible set of seeds, using a depth-first search algorithm [Cormen et al., 1990] those clusters are separated in a set of graphs $\{G^{candidates}_i\}$.

Inclusive graph Some of the matches of $\mathcal{M}$ contains features that appear more than once, for instance, it is possible to have $m_{i,j} = (v^{obs}_i, v^{db}_j) \in \mathcal{M}$ and $(v^{obs}_i, v^{db}_k) \in \mathcal{M}$. Selecting the match with the higher ZNCC is not a good solution, since because of occlusions, and imperfections of the descriptor, a good match can have a lower score than a wrong one.

To solve this problem a *inclusive graph* (Figure 9.13(c)) is built, it is defined such as $G^{comp}(\mathcal{M}, E^{comp})$, with an edge $e^{comp}_{(i,j)\leftrightarrow(k,l)} = (m_{i,j}, m_{k,l})$ if and only if $i \neq k$ and $j \neq l$. Since the lack of an edge between two matches indicates a conflict between these two matches, any clique of that graph G^{comp} ensure that the objects in the matches are used only once.

The complement graph of compatibility G^{comp} is the *exclusive graph* (Figure 9.13(d)) G^{incomp}, where two edges indicate a conflict.

Seeds selection A set $\mathcal{S}_i = \{m_{i,j} = (v_i, v_j) \in \mathcal{M}\}$ of possible seeds is needs to fulfil the following constraints:

$$\exists! \mathcal{T},\ v_i = \mathcal{T} \cdot v_j \tag{9.18}$$

$$\forall m_{k,l} \in \mathcal{S}_i i \neq k \text{ and } j \neq l \tag{9.19}$$

Equation (9.19) ensure that vertices of the graph of observation and of the reference graph are used only once, while equation (9.18) ensure the spatial compatibility of the matches.

To compute $\mathcal{T}$ and fill the set $\mathcal{S}_i$, a RANSAC [Fischler and Bolles, 1981] process is used, the graphs $G^{candidates}_i = (V^{candidates}_i, E^{candidates}_i)$ are used to focus the compatibility check:

The set $\mathcal{S}_max \in \{\mathcal{S}_i\}$ is the set with the maximum number of matches, and it is the set that is used for the seeds:

$$\mathcal{S}_{max} \in \{\mathcal{S}_i\} \text{ where } \forall i\ card(\mathcal{S}_i) \leq \mathcal{S}_{max} \tag{9.20}$$

1. Randomly take the minimal number of matches in $V_i^{candidates}$ to compute $\mathcal{T}$ (the number of features needed, depends on the type of features, see annexe B)
2. Count the number of matches $m_{i,j} = (v_i, v_j) \in V_i^{candidates}$ such as $v_i = \mathcal{T} \cdot v_j$
3. This process is repeated a given number of times, the transformation $\mathcal{T}$ is chosen as the transformation that validate the most matches

uncertainty on the observation	0.001	0.011	0.021	0.031	0.041	0.051	0.061	0.071	0.081	0.091
good matches	125	128	121	114	98	82	50	31	17	10
false matches	0	0	0	2	4	2	2	2	2	1

Table 9.2: Number of matches as a function of the uncertainty

9.4 Seed and grow

Between two graphs $G^1(V^1, E^1)$ and $G^2(V^2, E^2)$, once an initial set of matches $\mathcal{M} = (v_i^1, v_i^2)$ has been found, it is interesting to expand this set using the transformation $\mathcal{T}$ between the matches, $\forall (v_i^1, v_i^2) \in \mathcal{M}$, $v_i^1 = \mathcal{T} \cdot v_i^2$.

The idea of the seed and grow approach is to "walk" in the graph, jumping from node to node, following the edges:

Given an initial set of matches $\mathcal{M} = (v_i^1, v_i^2)$, and the transformation $\mathcal{T}$. As long as $\mathcal{M} \neq \emptyset$

1. Take a match $(v_i^1, v_i^2) \in \mathcal{M}$ and remove it from $\mathcal{M}$
2. Given E_i^1 the subset of edges from E^1 that contains the vertex v_i^1, and E_i^2 the subset of edges from E^2 that contains the vertex v_i^2.

   $\forall e_{i,j}^1 = (v_i^1, v_k^1) \in E_i^1$, $E_i^2(j)$ is the set of edges in E_i^2 that is compatible with $e_{i,j}^1$ (as defined in section 9.1), then if there is $e_{i,k}^2 = (v_i^1, v_k^1) \in E_i^2(j)$ such as $v_j^1 = \mathcal{T} \cdot v_k^2$, then (v_j^1, v_k^2) is a match and is added to $\mathcal{M}$.

9.5 Simulated test

To validate the algorithm, we used an existing dataset representing a small village, containing 11327 segments. The associated graph is constructed as in section 8.5.1, with an uncertainty on the extremity set to $\sigma = 0.01m$. Then observations are produced by applying a rotation and noise on subset of the segments, as shown on figure 9.14.

Table 9.2 contains the evolution of the number of matches as a function of the uncertainty on the observation set.

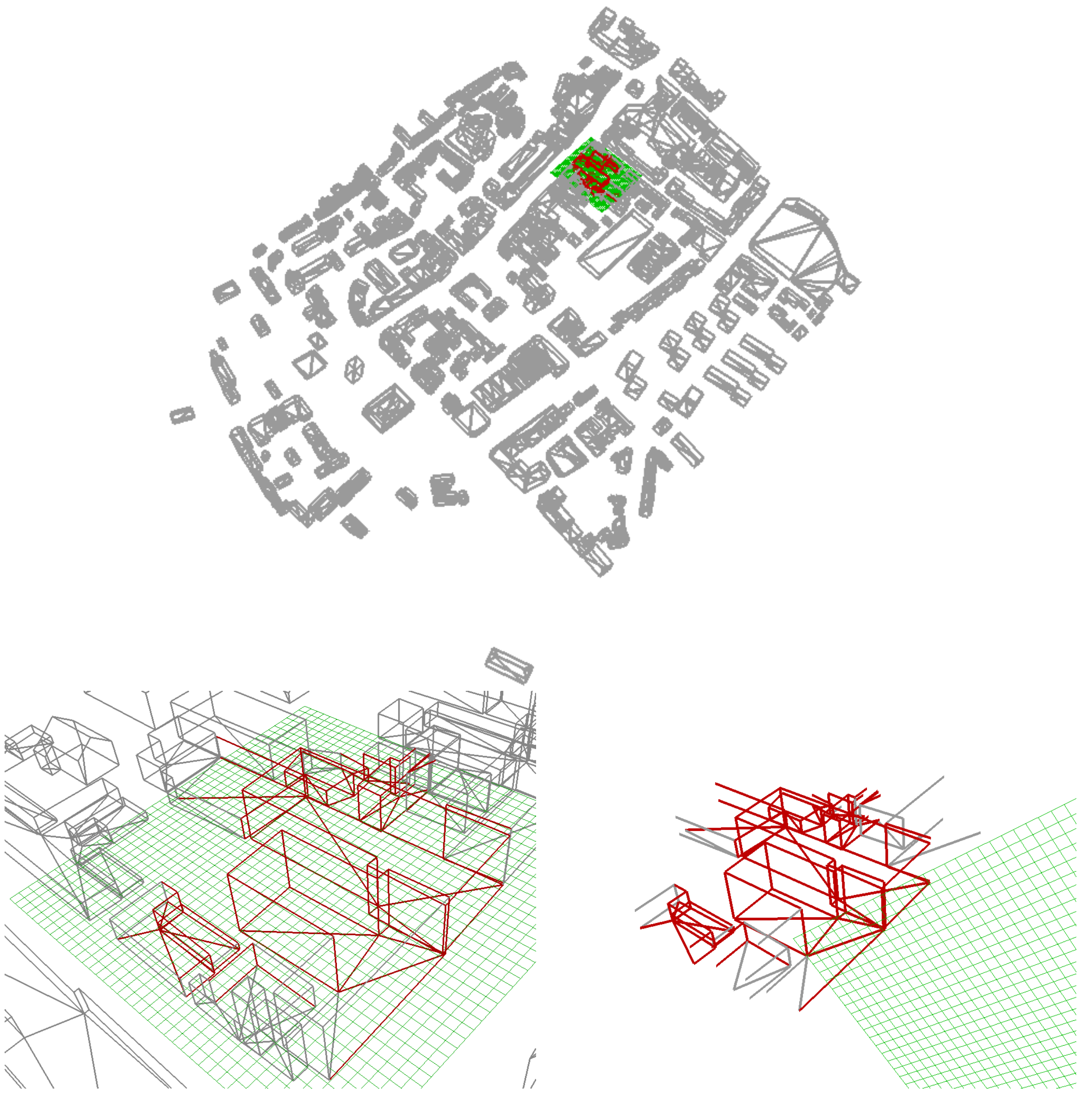


Figure 9.14: The top figure shows the initial environment model, the bottom left figure is a closeup of a given area of the village, and the bottom right shows the result of a simulated perception of this area. To simulate this perception, noise is added and the overall model is moved in a different reference frame – as shown by the difference with the grid. The red segments indicate successful matches before the “seed and growth” process

Conclusions

Augmenting the information contained in environment models requires to develop new algorithms to extract more information from the perceived data. In part II, we have presented two new landmarks detectors, for planar facets in chapter 5 and line segments in chapter 6.

But there is currently no efficient matching process for segments in the image frame, and that we are also interested in the possibility to match data coming from different sensors, as well as data coming from existing geographic information systems. Therefore, in part III, we have presented a method to define a structure over a environment model composed of heterogeneous landmarks, and a method to match geometric landmarks.

Future work would focus on further validation of this geometric model, first in the context of vision, then by extending it to other type of sensors (especially 3D Lidars), then for the use of geographic information system. But this raises a Lot of interesting challenges, both for the geometric model approach, as well as the need to develop and improve existing algorithms.

1 A multi source environment

Currently most of the work on mobile outdoor robot has focused on a self-sufficient robots operating alone in the environment. But we believe that in the future robots should be able to communicate with a large variety of other systems, that would be either other robots in the close vicinity, sensors arrays present in the environment, and other information database systems.

A multimap approach like in [Vidal-Calleja et al., 2009] allows for a system of robots to map the same environment. It has been designed to work in decentralized manner, but the algorithms have actually not been fully decentralized. Indeed in our experiments, while the robots build independent maps, they share the same graph of local maps. In a real world application, the robots will not be able to communicate which other in a permanent manner: each robot will need to maintain his copy of the graph of local map, and can only exchange information when it is possible. This exchange of information is essential, since it is the only way to keep the system consistent (by ensuring that no information is used twice), and of course for the robots to cooperate.

There is naturally a need to *select* what information has to be exchanged. Some information are indeed of no use for the map-matching process (*e.g.* a single non discriminative landmark). Grouping features is a way to structure and select relevant information to be exchanged – it also allows to reduce the information to store in the stochastic map with collapsing techniques [Gee et al., 2007].

2 Landmark detection

Of course, the more information are present in the landmark models, the better the resulting overall geometric model is. We draw here some work perspectives on the definition and extraction of landmarks.

2.1 Towards a line segment descriptor

Our line segment detectors are currently only able to extract geometric information from an image: there is no signal-based descriptor associated to segments. While a matching process as defined for points in chapter 4 is not necessarily needed because geometric matching can be sufficient, a signal-based descriptor is still useful to increase the differentiation between the different segments.

But defining a reliable signal-based descriptor for line segment is tricky. For points, an hypothesis of local planarity and signal invariance in the close vicinity can be used to use the image patch as a descriptor (or as a base for a gradient descriptor). But this assumption does not hold for segments, whose length in the image implies that it is not possible to ignore the distortion due to the projection. Also, many segments correspond to a 3D edge, and in many case, only one side of the segment can have a stable descriptor, which requires to have two descriptors for each side. It could be possible to use the depth information, recovered from the estimation of the 3D geometric parameters from a SLAM process, to compute a projection of the image patch around a segment like we did for facets in chapter 5.

2.2 Planes detection

Compared to points, segments and facets increase the density of the model of the environment. But the extraction of planes would yield a much better geometric model. Some ways to extract planes have been mentioned in chapter 7: an other one would be to rely on the detected facets and segments, that could be grouped to generate plane hypotheses. A further process could then confirm these hypotheses, *e.g.* by fitting homographies on the image area defined by the grouping process.

2.3 Composite object detection

Since individual landmarks are not discriminant enough and may lead to false matches, work has been devoted to do batch matching of multiple landmarks at the same time ([Neira and Tardos, 2001,Bailey, 2002], and the work depicted in part III of this manuscript). But we think it is interesting to extend the graph matching approach to directly match composite objects, because this would increase the discriminancy of individual objects. The main difficulty is to define a good grouping algorithm: for instance, in chapter 5, the grouping algorithm work by defining the center of the group as the center of gravity of a set of facets, which gives different results depending on whether a set of facets is completely or only partially detected. This does not prevent the matching algorithm to work, but this prevents some of the group to be used as a seed, and thus decreases the efficiency of the algorithm.

The geometric descriptor of chapter 9 could be used to check which features are likely to be part of a composite object. To do this it is necessary to use a model of the composite object, and then to compute the descriptor for each object. But since observed features are likely to be connected to features that are not part of the composite object, instead of using a

ZNCC score to check features, it is needed to compute whether a feature has a descriptor D_f that dominates the descriptor of the model D_m ($\forall i D_f(i) \geq D_m(i)$). Then a graph matching method could be used to check the compatibility between the set of features and the composite model.

3 Geometric graph

3.1 Extending the graph

It is important to reduce the number of geometric features to check in a graph match algorithm. This can be done using the current estimation of the position in the environment, but when the uncertainty on the position of the robot is large, other information can be used to reduce the number of candidates. As in chapter 5, each feature can be associated to a signal descriptor (for facets it is an image patch), and this signal descriptor is also used. It is also be interesting to associate a descriptor to the general surrounding of the environment, like mixing a topological map with a metric map, or use a fab maps approach [Cummins and Newman, 2009] to detect a loop closure.

Such information could be inserted in the graph by introducing a new type of vertex, called "location" that contains the information of the descriptor of the current place.

3.2 Graph construction

Improvements on the graph construction could relate on how links are established between features. Currently the features are connected using a distance measure, ensuring that the uncertainty on the distance remains small. Using a visibility map, or the information matrix (two features that have been perceived from the same position are highly correlated), it is possible to only connect two features that are likely to be visible together. Hopefully, this would decrease the likelihood that the two features are not found in the set of observed features.

One of the issue of the graph matching algorithm is that occlusions decrease the efficiency of the geometric descriptor, which prevents good matches to occur. An algorithm that detects the occurrence of occlusions could be exploited to trigger further observations before attempting the establishment of matches.

More generally, work on the data acquisition strategy is required to make the best use of a set of robots, and to increase the amount of information encoded in the maps.

Appendix A

Computing distances and projection

1 Projections

1.1 A point on a line

Given a line L of origin $\overrightarrow{o_L}$ and direction $\overrightarrow{d_L}$, with $M_{\overrightarrow{o_L}}$ and $M_{\overrightarrow{d_L}}$ the respective covariance, and given a point P of covariance M_P, the projection of the point P on L is given by:

$$\begin{aligned} proj(P,L) &= < \overrightarrow{d_L}, P - \overrightarrow{o_L} > \cdot \overrightarrow{d_L} + \overrightarrow{o_L} && \text{(A.1)} \\ cov_{proj(P,L)} &= Jp \cdot M \cdot Jp^t + Jo \cdot M_{\overrightarrow{o_L}} \cdot Jo^t + Jd \cdot M_{\overrightarrow{d_L}} \cdot Jd^t && \text{(A.2)} \\ \text{with } Jp &= \overrightarrow{d_L} \times \overrightarrow{d_L} && \text{(A.3)} \\ Jo &= -Jp + I_3 && \text{(A.4)} \\ Jd &= \overrightarrow{d_L} \times (P - \overrightarrow{o_L}) + < \overrightarrow{d_L}, P - \overrightarrow{o_L} > \cdot I_3 && \text{(A.5)} \end{aligned}$$

1.2 A point on a plane

The projection of a point P on a plane Π, with $O \in \Pi$ and $\overrightarrow{n}$ is given by:

$$P - < P - O, \overrightarrow{n} > \cdot \overrightarrow{n} \tag{A.6}$$

2 Distances

2.1 Closest points between two line

Given two lines L_1 (defined by its direction $\overrightarrow{d_1}$ and by a point $O_1 \in L_1$) and L_2 (defined by its direction $\overrightarrow{d_2}$ and by a point $O_2 \in L_2$), the closest point $P_1 \in L_1$ from L_2, and the closest point $P_2 \in L_2$ from L_1:

$$\begin{aligned} P_1 &= \frac{< \overrightarrow{d_1}, \overrightarrow{d_2} > \cdot < \overrightarrow{d_2}, \overrightarrow{O_1O_2} > - < \overrightarrow{d_1}, \overrightarrow{O_1O_2} >}{1 - < \overrightarrow{d_1}, \overrightarrow{d_2} >^2} \cdot \overrightarrow{d_1} + O_1 && \text{(A.7)} \\ P_2 &= \frac{< \overrightarrow{d_2}, \overrightarrow{O_1O_2} > - < \overrightarrow{d_1}, \overrightarrow{d_2} > \cdot < \overrightarrow{d_1}, \overrightarrow{O_1O_2} >}{1 - < \overrightarrow{d_1}, \overrightarrow{d_2} >^2} \cdot \overrightarrow{d_2} + O_2; && \text{(A.8)} \end{aligned}$$

2.2 Line to line

The distance between two lines is simply given by the distance between the closest points defined in section 2.1.

2.3 Line Segment to line

Given a line segment S_1 (with extremeties A and B) and line L_2, the closest point between the line support of S_1 and L_2 are P_1 and P_2, then the distance is given by:

$$t = \frac{< \overrightarrow{AP_1}, \overrightarrow{AB} >}{||\overrightarrow{AB}||^2} \tag{A.9}$$

$$d(S_1, L_1) = \begin{cases} ||\overrightarrow{AP_2}||, \ t < 0 \\ ||\overrightarrow{BP_2}||, \ t > 1 \\ ||\overrightarrow{P_1P_2}||, \ t \in [0,1] \end{cases} \tag{A.10}$$

2.4 Line Segment to Segment

Given two segment S_1 (with extremeties A_1 and B_1) and S_2 (with extremeties A_2 and B_2), the closest point between the line supports of S_1 and S_2 are P_1 and P_2, then the distance is given by:

$$t_1 = \frac{< \overrightarrow{A_1P_1}, \overrightarrow{A_1B_1} >}{||\overrightarrow{A_1B_1}||^2} \tag{A.11}$$

$$t_2 = \frac{< \overrightarrow{A_2P_2}, \overrightarrow{A_2B_2} >}{||\overrightarrow{A_2B_2}||^2} \tag{A.12}$$

$$d(S_1, L_1) = \begin{cases} ||\overrightarrow{A_1A_2}||, \ t1 < 0 \text{ and } t2 < 0 \\ ||\overrightarrow{A_1B_2}||, \ t1 < 0 \text{ and } t2 > 1 \\ ||\overrightarrow{A_1P_2}||, \ t1 < 0 \text{ and } t2 \in [0,1] \\ ||\overrightarrow{B_1A_2}||, \ t1 > 0 \text{ and } t2 < 0 \\ ||\overrightarrow{B_1B_2}||, \ t1 > 0 \text{ and } t2 > 1 \\ ||\overrightarrow{B_1P_2}||, \ t1 > 0 \text{ and } t2 \in [0,1] \\ ||\overrightarrow{P_1A_2}||, \ t1 \in [0,1] \text{ and } t2 < 0 \\ ||\overrightarrow{P_1B_2}||, \ t1 \in [0,1] \text{ and } t2 > 1 \\ ||\overrightarrow{P_1P_2}||, \ t1 \in [0,1] \text{ and } t2 \in [0,1] \end{cases} \tag{A.13}$$

Appendix B

Finding the transformation between geometric objects

1 Between two lines

A quaternion is an extension of the complex numbers, that introduces two other numbers, such as:

$$i^2 = j^2 = k^2 = ijk = 1 \tag{B.1}$$

Given two quaternions:

$$\mathbf{q} = q_1 + q_2 \cdot \mathbf{i} + q_3 \cdot \mathbf{j} + q_4 \cdot \mathbf{k} \tag{B.2}$$

$$\mathbf{p} = p_1 + p_2 \cdot \mathbf{i} + p_3 \cdot \mathbf{j} + p_4 \cdot \mathbf{k} \tag{B.3}$$

The multiplication between the two quaternion is given by:

$$\begin{aligned}\mathbf{q} \cdot \mathbf{p} =&(q_1p_1 - q_2p_2 - q_3p_3 - q_4p_4)\\ &+(q_1p_2 + q_2p_1 + q_3p_4 - q_4p_3)\mathbf{i}\\ &+(q_1p_3 + q_2p_4 + q_3p_1 + q_4p_2)\mathbf{j}\\ &+(q_1p_4 + q_2p_3 - q_3p_2 + q_4p_1)\mathbf{k}\end{aligned} \tag{B.4}$$

While a rotation of angle θ around the axis $\mathbf{u} = (x, y, z)$ is represented by the following quaternion:

$$\mathbf{q} = \left(cos\left(\frac{\theta}{2}\right), sin\left(\frac{\theta}{2}\right)\mathbf{u}\right) \tag{B.5}$$

The rank of the quaternion is 3, meaning that $\forall s,\ \mathbf{q}_s = s \cdot \mathbf{q}$ represent the same rotation. Given a vector $\overrightarrow{v}$ of quaternion $p_v = (0, v)$, its rotation $\overrightarrow{v'}$ is given by:

$$\mathbf{p_{v'}} = \mathbf{q} \cdot \mathbf{p_v} \cdot \mathbf{q}^-1 \tag{B.6}$$

Since a vector director of a line is of rank two, it is necesserary to use two non parallels lines to recover the rotation. So given two lines L_1 and L_2 of vector directors (x_1, y_1, z_1) and (x_2, y_2, z_2), and L_1' and L_2' the result of the rotation of L_1 and L_2 by the rotation q, to compute the value of q, we set $q_1 = 1$, and need to solve the following system:

$$(x'_1 - x_1) \cdot q_2 + (y'_1 - y_1) \cdot q_3 + (z'_1 - z_1) \cdot q_4 = 0 \tag{B.7}$$

$$-(z'_1 + z_1) \cdot q_3 + (y'_1 + y_1) \cdot q_4 = -(x'_1 - x_1) \tag{B.8}$$

$$(x'_2 - x_2) \cdot q_2 + (y'_2 - y_1) \cdot q_3 + (z'_1 - z_1) \cdot q_4 = 0 \tag{B.9}$$

Given a point P_1 of L_1, and P_2 of L_2. First, the rotation of the point P_1, $P_1^r = \mathbf{q} \cdot P_1 \cdot \mathbf{q}^- 1$ is projected on line L'_1, this give a first translation vector $\overrightarrow{t_1}$, then the rotation of P_2 translated by $\overrightarrow{t_1}$ is projected on L'_2 following the direction of L'_1, this gives a second translation vector $\overrightarrow{t_2}$, the full translation vector is then given by $\overrightarrow{t} = \overrightarrow{t_1} + \overrightarrow{t_2}$.

Appendix C

Résumée en Français

1 Introduction

L'objectif de la robotique mobile d'extérieur (figure 1) est de permettre aux robots d'accomplir une mission de manière autonome, tel que le transport, la surveillance ou l'exploration. De tel missions se décomposent en tâches élémentaires: déplacement, observations, saisie d'un objet, qui sont définies en fonction de l'environnement du robot, et elles nécessitent pour la planification et l'exécution des modèles de l'environnement de nature différentes: carte de traversabilité, carte d'amers... Pour exécuter ces tâches dans de bonne conditions, il est aussi important pour le robot de connaître, de manière précise sa position, qu'elle soit relative par rapport à un objet, ou absolue. Elle peut être obtenue soit par l'intégration du déplacement du robot (odométrie...), soit à l'aide de balises (GPS...) ou encore en utilisant des cartes de l'environnement. Chacune de ses méthodes présentent divers inconvénients, l'odométrie présente une dérive, le GPS souffre de masquage et de brouillage du signal, tandis que la localisation par carte soulève le problème de l'existence de ses cartes et de la capacité à en associer le contenu avec l'observation de l'environnement.

Localisation et cartographie simultanées Ainsi le lien entre modèles de l'environnement et localisation apparaît comme évident, d'une part les modèles peuvent fournir une solution au problème de la localisation, et d'autre part une localisation précise est nécessaire pour assurer la cohérence spatiale lors de la construction de nouveaux modèles par le robot.

Dans de nombreuses situations, il n'existe pas à priori de cartes de l'environnement, le robot doit donc construire sa propre carte, et se trouve confronté au problème de l'oeuf et de la poule, entre la nécessité d'une localisation précise pour construire le modèle qui est utilisé pour donner la position. En robotique, ce problème est appelé "localisation et cartographie simultanées" (Simultaneous Localisation and Mapping, SLAM) [Chatila and Laumond, 1985, Smith et al., 1987].

L'objectif de la thèse: vers des modèles riche en géométrie De la connaissance de la géométrie de l'environnement découle la construction des autres modèles de l'environnement, par exemple, la notion d'obstacle est une information géométrique. Mais elle est ni suffisante, une information de traversabilité est renforcé par la connaissance de la texture, ni nécessaire pour résoudre tous les problèmes, un suivi visuel est suffisant pour permettre au robot de saisir un objet. De même, la localisation absolue peut être effectuée sur des informations purement visuel [Cummins and Newman, 2008, Angeli et al., 2008].

Cependant la géométrie est une information caractéristique de l'environnement, et contrairement à une information visuel, elle est indépendante de l'illumination, du capteur utilisé pour les observations, du point de vue, et elle est compatible avec les modèles existant des systèmes d'information géographiques (Geographic Information System, GIS). Ceci fait de l'information géométrique un élément clé des systèmes multi robots, en effet la géométrie est la seule information commune entre un drone aérien et un robot terrestre.

L'approche Un des points forts des approches SLAM utilisant des amers est qu'elle résolve la localisation géométrique et le problème de la cartographie. Le modèle résultant est donc par essence géométrique, mais restreint à une collection d'amers. Les premières versions du SLAM utilisaient des segments 2D obtenus par des lasers, les approches 3D récentes utilisent un nuage de points. De tels représentations sont dédiées exclusivement à un usage de localisation, par exemple, il n'est pas possible d'en déduire une carte de traversabilité, et elles ne sont utilisables qu'avec un seul type de capteur.

Notre approche consiste à utiliser les mécanismes d'estimation géométrique du SLAM et des les utiliser avec des amers comportant une information géométrique plus importante, des plans et des segments. Cependant un tel ensemble d'amer reste une représentation creuse de l'environnement, c'est pourquoi nous avons aussi définie un graphe d'amers pour permettre d'exhiber la structure de l'environnement, et de faciliter la mise en correspondance.

Contributions Les contributions de cette thèse dans le cadre d'une modélisation de l'environnement sont les suivantes:

- Nous avons introduits l'usage de facettes et de segments pour des amers dans une approche de type SLAM.
- Ces amers ont été exploités dans une extension du filtre de Kalman pour une approche multi-robot.
- La définition d'un graphe d'amers et de son utilisation dans le problème d'association de données.

2 Le problème de la modélisation d'environnement

Pour être capable de construit un modèle de l'environnement, les robots doivent avoir la capacité de l'observer, de détecter des amers, de les suivre, de les associer et d'estimer leur position. La détection est effectué par un processus de *traitement du signal* qui produit une information métrique, utilisé pour mettre à jour les paramètres d'un amer, et un descripteur du signal. Tandis qu'un autre processus est chargé de mettre en correspondance les observations avec le modèle. Qui sont ensuite traitées dans un filtre de Kalman.

2.1 Cartographie et Localisation Simultanée

Une rapide introduction au problème du SLAM Un présentation de l'historique des principales contributions se trouve dans [Dissanayake et al., 2001], et une synthèse de l'état de l'art est disponible dans ces deux articles: [Durrant-Whyte and Bailey, 2006] et [Bailey and Durrant-Whyte, 2006].

La figure 1.1 présente la méthode incrémentale de mise à jour des amers. En quatre étapes:

- *Détection d'amers*, des éléments saillants dans l'environnement.
- *Estimation des mesures relatives* des amers par rapport au robot, et du déplacement du robot.
- *Association de données*: les amers permettent d'améliorer la position du robot, uniquement si le robot est capable de les observer plusieurs fois.
- L'*estimation* est la partie centrale des algorithme de SLAM, elle permet la fusion des mesures relatives et permet d'obtenir la position absolue du robot et des amers.

En plus de ces étapes, les algorithmes de SLAM sont confrontés à deux problèmes principaux:

- *La fermeture de boucle*, plus le robot se déplace, plus l'incertitude sur sa position augmente, mais ceci peut-être corrigé lorsque le robot revient dans une zone connue, le robot doit alors être capable de reconnaître des amers déjà vu, ce qui corrige l'ensemble des informations de la carte, la position du robot et des amers (Figure 1.2).
- La complexité algorithmique et les problèmes de consistance des processus d'estimation rend délicat la *gestion de la carte.*

Définition d'un amer : il est, au minimum, définit par ses paramètres géométriques, et doit avoir des caractéristiques qui permettent de le distinguer dans l'environnement. Bien que l'association de donnée puisse se faire uniquement sur les informations géométriques, elles sont généralement complétées par une information de signal, tel un descripteur visuel comme par exemple SIFT [Lowe, 2004]. Le SLAM 3D repose généralement sur l'usage de points, et plus récemment de segments et de plans.

L'association de donnée consiste à mettre en correspondance des amers observées depuis différents points de vue successifs (suivie de points) ou non (appariement).

L'estimation des paramètres géométriques est effectué par un processus de filtrage, la méthode classique pour le SLAM est l'usage d'un filtre de Kalman étendu. La figure 1.3 montre les différentes méthodes possible pour gérer les cartes. Une possibilité introduite dans [Estrada et al., 2005] est l'usage d'un ensemble de cartes locales et indépendantes, reliées entre elle dans un niveau global.

Multi-robots: dans un contexte multi-robots, chaque robot doit pouvoir être capable d'évoluer dans l'environnement de manière indépendante, mais il doit être capable d'utiliser une information qui provient des autres robots. [Vidal-Calleja et al., 2009] montre l'intérêt de l'approche multi-cartes dans le cas d'un systèmes de robots, avec deux types d'évènements, rendez-vous entre les robots et reconnaissances d'amers. Ces évènements permettent des fermetures de boucle entre des cartes qui ont été construites par des robots différents, comme sur la figure 1.4.

Notre approche pour la construction de cartes est composée de quatre processus comme sur la figure 1.6, la *localisation* utilise les capteurs de déplacement pour produire une estimation du déplacement qui est utilisée par le *SLAM* lors de la construction de carte locale. Le *gestionnaire de carte* est chargé des mises à jour des liaisons entre les cartes, tandis que notre algorithme de structuration de carte *SPAF* (pour SPAtial Features) détecte des fermetures de boucles.

2.2 Détection

La détection est la phase qui permet au robot de détecter des objets dans son environnement à l'aide de capteurs.

Capteurs actifs et passifs Il existe deux types de capteurs, les *capteurs actifs* qui émettent un signal (infrarouge, radar, laser...) et qui mesurent le retour du signal. Ils offrent l'avantage d'une grande facilitée de mise en oeuvre, puisqu'ils fournissent directement l'information de distance avec les objets. Par contre ils ont une portée limitée, et sont soumis à des problèmes d'interférence, et nuise à la discrétion du robot. Il existe aussi des *capteurs passifs* qui captent les émissions de radiations naturelles, par exemple des caméras. Elles offrent l'avantage d'une portée théorique infinie, de ne pas émettre de signal, mais sont plus complexe de mise en oeuvre, puisqu'elles ne permettent que d'obtenir un positionnement angulaire de l'amer. Cependant, l'utilisation de deux caméras en stéréovision permet à l'aide de deux mesures angulaires d'obtenir la profondeur de l'objet, mais avec une portée utile limitée.

Extraction d'information Les bon amers marins sont ceux visibles de loin, et qui ont une forme unique pour être facilement reconnu et apparier, les amers détectés par un robot devront avoir les même caractéristiques. Par exemple, dans une image, le maximum local de gradient permet d'obtenir la position de points d'intérêts.

2.3 Association de données

Une fois des objets détectés dans l'environnement, il est nécessaire de le suivre entre deux instants, afin de mettre à jour le filtre, il est aussi nécessaire de détecter les fermetures de boucle. La principale difficulté de l'association de donnée est d'éviter les faux appariements, qui ont des effets néfastes sur la qualité des modèles construits, tout en gardant un maximum de bons appariements.

Appariement d'amers en utilisant les informations du signal, comme le descripteur, et une fonction de corrélation. Dans une première étape, les amers sont extraites, chaque amer est comparé avec le contenu de la base de données. Mais des appariements individuels sont insuffisant, étant donné que l'information est souvent répétitive, il est nécessaire de tenir compte des appariements voisins, plusieurs techniques sont envisageables : des techniques statistiques (RANSAC [Fischler and Bolles, 1981]), l'utilisation d'un modèle du capteur [Hartley and Zisserman, 2000], ou des contraintes géométriques et un regroupement des amers [Jung and Lacroix, 2001].

Fermeture de boucles Il s'agit d'un problème spécifique au SLAM, il s'agit de détecter si l'endroit courant a déjà été cartographié. Plusieurs types de techniques ont été développées pour résoudre ce problème:

- Une première possibilité est d'utiliser des techniques d'appariement d'amers, ou de comparaison d'image à image [Mikolajczyk and Schmid, 2001], ou encore d'utiliser un dictionnaire de points [Cummins and Newman, 2009].
- Il est aussi possible de projeter un modèle dans les données, pour retrouver les éléments.
- Ou encore d'utiliser les informations géométriques pour un appariement carte à carte [Neira and Tardos, 2001]. Ces méthodes consistent à apparier les amers par groupe d'amers d'appariement compatible.
- Le dernier type consiste à apparier les données extraites avec le contenu de la carte [Williams et al., 2008].

Suivie Lorsqu'une bonne estimé de la position des amers est connu, des techniques de suivie (tracking) sont envisageables pour résoudre le problème de l'association de données. Elles ont l'avantage d'être plus performante, et de fournir des résultats plus sûr. Elles consistent à optimiser l'estimée initial de la position des amers, avec des techniques d'optimisation, par exemple, pour des points d'intérêts la méthode KLT [Shi and Tomasi, 1994b].

3 Modèles d'amers

Le modèle d'amer est un élément critique dans la définition d'un solution au problème du SLAM, un bon amer est une donnée saillante, facilement détectable, suivable et appariable depuis différents points de vues. Tous ces processus doivent permettre d'estimer la position relative des estimées, associé à un modèle d'erreur.

Toute solution au problème d'appariements d'amers nécessitent trois étapes, définition des amers, définition d'une mesure de similarité, et une stratégie d'appariement. De la définition du type d'amers découle les algorithmes utilisables pour l'appariement. Un amer peut être un élément de signal, un coin, un contour, une ligne, une région...

3.1 Points d'intérêt

La solution classique au problème du SLAM 3D en vision repose sur la détection de points dans une image, différentes méthodes sont possibles, comme le point de Harris [Harris and Stephens, 1988b], ils sont centrés sur des maximums locaux de gradients dans toutes les directions. Cette méthode de détection souffre d'une faible répétabilité au changement d'échelle et de point de vue, ce qui a conduit au D'autre méthodes développement d'autres méthode comme les points SIFT [Lowe, 2004] ou SURF [Bay et al., 2008] qui offre une meilleur répétabilité.

3.2 Facettes

L'utilisation de points d'intérêts offre un modèle creux de l'environnement, c'est pourquoi nous avons proposés l'extension des points aux facettes dans [Berger and Lacroix, 2008]. Le modèle de facette repose sur six paramètres géométriques et une image. Ce modèle offre une meilleur observabilité de la position du robot, et facilite le processus d'appariement.

Le modèle de facette La figure 5.1 montre un exemple d'extraction de facettes. Une facette est définie par les trois coordonnées de son centre, et par trois angles d'Euler. Ainsi que par une image.

Les étapes de l'extraction de facettes Pour extraire les facettes, l'utilisation d'un ban stéréovision est nécessaire. Les facettes sont centrées sur un point d'intérêt, à l'aide de la stéréovision et d'un algorithme d'appariement de points d'intérêt, les coordonnées en 3D sont obtenues. Ensuite pour établir si le voisinage du point est planaire, une homographie H est calculée entre l'image de la caméra gauche et celle de la caméra droite, en utilisant une méthode d'optimisation avec déformation de l'homographie [Baker and Matthews, 2001,Malis, 2004]. A partir de cette homographie, il est possible de calculer le vecteur normal de la facette, et le calcul du gradient de l'image autour du point d'intérêt donne le troisième angle d'Euler.

En utilisant le vecteur normal et l'orientation du gradient l'image de la facette est extraites et sa perspective est corrigée afin de favoriser les appariements.

Modèle d'erreur L'extraction des coordonnées 3D des points d'intérêt et des informations angulaires sont indépendantes, la matrice de covariance d'une facette (équation 5.5) est donc diagonale par bloque. Le premier bloque correspondant à un modèle classique de stéréovision, et le second bloque a une matrice diagonale avec les variances sur les angles.

Suivie de facettes L'un des avantages d'utiliser des facettes est la possibilité de prédire leur apparence quand la transformation est connue. Lors d'un faible déplacement, entre deux acquisitions d'images, une bonne estimation du déplacement du robot est connu, ce qui permet d'avoir une bonne estimation de la projection des facettes dans l'image et de leur apparence. Ainsi, pour chaque instant $t + 1$, la liste des points d'intérêts est extraite, et pour chaque facette de l'instant t on compare l'image de la facette avec le voisinage des points proches de l'estimation, ceci permet de trouver une nouvelle estimation de la position. Un tel processus est appliquée sur l'image de la caméra gauche et de la caméra droite, ce qui permet de recalculer l'ensemble des paramètres et d'obtenir une nouvelle observation de l'amer.

Appariement de facettes Dans le cas générale, lorsqu'aucune transformation n'est connu, par exemple dans le cas d'un fermeture de boucle. Il est nécessaire de recourir à un algorithme général d'appariement. La première étape est d'effectuer une détection de facettes dans les images. Ensuite, en comparant les images de chaque facettes de la base de donnée avec l'une des facettes dernièrement détectées, on obtient un premier appariement. Ce premier appariement permet de calculer une transformation entre les facettes observées et la base de donnée. Ensuite en utilisant cette transformation, il est possible de focaliser la recherche d'appariements pour les autres facettes détectées.

3.3 Segments de lignes

Les segments de lignes représentent une part importante de la structure d'une image, en effet, dans en environnement semi-urbain, ils constituent les limites de nombreuses structures artificielles. Parmi les avantages des segments, on trouve une plus grande expressivité, comme montré dans la figure 6.1, mais aussi une plus grande robustesse par rapport aux changements de point de vue, d'échelle et de luminosité. Mais cela est vrai, uniquement si l'algorithme d'extraction est lui même robuste et stable, ce qui n'est pas le cas des algorithmes classiques de chaînage [Etemadi, 1992], ou de Hough probabiliste [Hough, 1962, Guo et al., 2008].

Direct Segment Detection Nous utilisons un modèle paramétrique pour représenter les lignes en 2D :

$$x(t) = a \cdot t + x_0 \tag{C.1}$$

$$y(t) = b \cdot t + y_0 \tag{C.2}$$

(a, b) est le vecteur directeur de la ligne, et (x_0, y_0) correspond à l'origine. Bien que ce modèle soit une représentation surparamétrée pour des lignes 2D, il a l'avantage de ne présenter aucune singularité et que tous les points de la ligne sont obtenue linéairement en fonction de t. Et comme aucune contrainte existe entre les paramètres, le modèle peut être utilisé sans aucune difficulté dans un processus d'estimation.

La première étape de notre algorithme consiste à trouver un point de départ de la ligne, pour se faire nous utilisons un filtre de Canny [Canny, 1986] qui donne les maximums de gradients dans la direction du gradient. Cette première étape permet d'initialiser l'origine (x_0, y_0) du segment ainsi que le vecteur directeur (a, b) de la ligne qui est perpendiculaire à la direction du gradient.

Extension de segments Ensuite nous avons développées un algorithme d'extension de segments, figure 6.2:

1. $t = 1$
2. $t \leftarrow t \pm \delta_t$, où δ_t est la distance curviligne à laquelle un nouveau point est recherché
3. Prédiction: en utilisant l'estimation courante des paramètres de la ligne et leurs variances, une estimation des coordonnées du nouveau point (x_t, y_t) de recherche est obtenu, ainsi que l'erreur e associée à ses coordonnées
4. Observation: un ensemble de mesures sont effectuées le long de la normal de la ligne qui intersecte en (x_t, y_t) l'estimation courante du segment, la meilleur observation est sélectionnée comme étant le maximum de gradient le plus proche de (x_t, y_t) et ayant le gradient le plus orthogonal au segment

L'algorithme s'arrête lorsque qu'aucune observation n'est trouvé. Cependant, le bruit et la discrétisation peuvent causer des trous dans le segment, pour remédier à ce problème, l'algorithme peut regarder un cran plus loin si il continue à trouver des observations.

Estimation des paramètres de la ligne Un filtre de Kalman est utilisée pour mettre à jour les paramètres de la ligne. Le vecteur d'état contient les paramètres de la ligne (a, x_0, b, y_0), étant donné l'absence de bruits de processus et que le modèle est stationnaire, l'état reste constant comme le montre l'équation 6.14. Le modèle d'observation est linéaire, équation 6.15. A l'initialisation la matrice de covariance de l'état est linéaire, équation 6.17.

Fusion de lignes A la fin du processus de détection, il arrive qu'un segment chevauche un autre segment. Dans ce cas un test du Chi^2 est utilisé pour vérifier si les paramètres des deux segments sont compatibles, si c'est le cas les paramètres de l'un des segments sont mis à jour en utilisant les paramètres de l'autre comme observation.

Paramètres Le processus de détection nécessite la définition de quelques paramètres et seuils, mais l'utilisation de valeurs importantes et conservatrices est suffisante, et la figure 6.4 montre que les paramètres ont une faible influence sur les résultats.

Extension multi-échelle La réduction de l'image permet de filtrer le bruit, elle permet aussi de simuler l'éloignement, et donc de détecter des segments de meilleur qualité et qui devrait pouvoir être suivi sur une plus longue distance. Mais le prix à payer est une perte de précision sur la position métrique des segments, en effet les pixels de l'image réduite corresponde à une surface plus importante sur le capteur que ceux de l'image originelle.

Pour obtenir de segments de meilleur qualité tout en conservant la précision sur la localisation, nous avons proposé une approche multi-échelle. Les segments détectée dans l'image réduite serve de graine pour trouver des segments dans l'image non-réduite, comme montré sur la figure 6.5. Pour une image donnée, une pyramide à n niveau est construite, en divisant la taille de l'image par deux entre chaque niveau, le niveau $level_{i=0}$ correspond au plus bas étage de la pyramide et à l'image pleine résolution, tandis que $level_{i=n-1}$ correspond à l'image la plus petite.

1. L'algorithme DSeg est utilisé sur le niveau $level_{i=n-1}$ pour détecter les premiers segments

2. Pour un segment seg_j^i détecté au niveau $level_i$, ces paramètres pour le niveau $level_{i-1}$ sont obtenus en multipliant par deux les paramètres au niveau $level_i$: $seg_j^{i-1} = 2 \cdot seg_j^i$.

   Soit $O_j^i(x_{i-1}, y_{i-1})$ les coordonnées de l'une des extrémités, $\mathbf{u}$ le vecteur directeur du segment, $\mathbf{v}$ est le vecteur orthogonal à $\mathbf{u}$, et l la longueur du segment, une liste d'intervalles $L = \{[0, l]\}$ est initialisée.

   (a) Soit un $I = [a, b] \in L$, un maximum de gradient est recherché dans la direction de v, autour du point:

   $$c = \frac{a+b}{2} \tag{C.3}$$

   $$(x_I, y_I) = (x_{i-1}, y_{i-1}) + c \cdot u \tag{C.4}$$

   (b) Si il y a un maximum de gradient dans (x_I, y_I), $(x_I, y_I) + v$ ou $(x_I, y_I) - v$, il est utilisé comme point de départ pour l'algorithme d'extension de lignes.

   (c) Pour chaque segment détecté au niveau $level_{i-1}$, la projection $P(ext1)$ et $P(ext2)$ de ces extrémités $ext1$ et $ext2$ est calculé sur $2 \cdot seg_j^i$. Ensuite, en utilisant $idx1 =< O_j^i P(ext1), \mathbf{u} >$ et $idx2 =< O_j^i P(ext2), \mathbf{u} >$, on peut retirer des intervalles de L, l'air où le segment a déjà été détectée, comme expliqué dans la figure 6.6.

   (d) Si aucun segment n'est trouvé l'intervalle $I = [a, b]$ est coupé en deux intervalles $[a, c]$, $[c, b]$, qui sont inséré dans L.

   (e) Retour à l'étape **(a)** tant L n'est pas vide.

   Un intervalle est inséré dans L seulement si sa longueur est supérieur à la longueur minimal d'un segment.

Résultats La figure 6.7 montre des résultats d'extraction de segments avec *DSeg* et *Hierarchical DSeg*, il apparaît que le nombre de segments extraits avec *Hierarchical DSeg* est plus faible qu'avec *DSeg*, et que de long segments sont extraits sur les images avec d'important changements d'échelle et d'orientation.

Analyse de la sensibilité Nous avons testé la sensibilité des deux algorithmes en fonction de l'augmentation du bruit et des changements de luminosités. Pour se faire nous avons utilisé des séquences d'images avec un bruit et une luminosité changeante, mais le point de vue reste identique. Pour la séquence de bruit, un bruit gaussien est ajouté à l'image, pour le changement de luminosité une caméra a été posée sur un pied et a acquis des images à intervalle régulier. Nous avons définit une métrique 6.23 pour déterminer si deux segments détecté dans deux images prises correspondent au même segment, ceci permet de calculer la répétabilité.

La figure 6.11 montre l'influence du bruit sur la répétabilité, les résultats sont présentés pour différentes valeur des paramètres, et montre que le changement de paramètres a une faible influence sur le résultat. Les figures 6.13 et 6.14 montre la sensibilité au changement de luminosité pour différentes longueur minimale de segments.

Comparaison Nous avons comparé nos deux algorithmes *DSeg* et *Hierarchical DSeg*, avec d'autres algorithmes: *Probabilistic Hough* ([Guo et al., 2008] implémentation d'OpenCV), *Split and merge* (chaînage sur Canny, implémentation du LAAS), et l'approche *LSD* ([R. Grompone von Gioi and Randall, 2008], avec l'implémentation de ses auteurs). Des figures 6.8 et 6.9, il apparaît que *LSD* et *DSeg* sont plus lent que les deux autres algorithmes, mais *DSeg* produit des segments plutôt plus long, et aussi les deux algorithmes n'ont pas nécessité de réglage lors des expériences, alors que pour *Probabilistic Hough* et *Split and merge*, il a fallu ajuster les paramètres aux différentes condition d'acquisition des images.

Nous avons aussi effectué l'analyse de sensibilité sur les cinq algorithmes, ainsi qu'en utilisant *DSeg* sur une image divisé par quatre. Les résultats sont visibles sur les figures 6.12 et 6.15. Ils montrent un avantage de nos méthodes sur les autres en terme de répétabilité. Le faible succès de *LSD* lorsque le bruit augmente s'explique par un algorithme volontairement conservatif afin d'éviter au maximum les faux segments qui peuvent apparaître dans des images bruités. Il est intéressant de remarquer que l'approche *Hierarchical DSeg* présente une meilleur répétabilité que *DSeg*, proche de celle de *DSeg* sur une image divisé par quatre.

Suivi de segments Le processus de suivi de segments d'une image à l'autre repose sur l'usage d'un filtre de Kalman, comme dans [Deriche and Faugeras, 1990], qui fusionne la prédiction fournie par un modèle avec des observations, de manière similaire à la méthode d'extension de lignes.

La principale difficulté du suivi de segments vient des segments proches et parallèles. En effet, même en utilisant un modèle de mouvement précis, la prédiction peut être fausse de quelques pixels, ce qui peut conduire à des erreurs d'associations avec des segments proches et parallèles (comme sur la figure 6.16). Pour résoudre ce problème nous avons utilisé un processus de sélection qui élimine les segments qui pourraient conduire à de mauvaises associations, et nous avons aussi utilisé les informations de gradients le long des segments.

Sélection de segments Le processus de sélection élimine les segments qui sont proche et dont le gradient de ses pixels est dans la même direction. La deuxième condition implique que les segments sont parallèles, mais il est possible de faire la différence deux segments proches et parallèles mais dont les gradients sont en direction opposées. Cependant si l'un des deux segments est plus long, il est possible de le conserver, et de n'éliminer que le petit segment, en effet, en utilisant la partie du segment long qui n'est pas doubler par le segment court on peut faire la différence dans l'appariement.

Première hypothèse La première étape du suivie consiste à initialiser les paramètres qui serviront au processus d'extension de ligne. Dans un premier temps, la position du segment est prédite, soit avec un modèle constant, à accélération constante ou du SLAM. A partir de ce segment, un certain nombre de points de contrôles sont utilisés pour trouver des maximums de gradients (équation 6.31), qui vont servir de support à la ligne.

Une fois la première hypothèse trouvée, l'algorithme d'extension de ligne est appliqué pour ajuster les paramètres du segment.

Résultat Pour évaluer les résultats, nous avons fait une évaluation statique en utilisant la même image, mais en appliquant une translation sur la prédiction des segments, ceci montre l'effet de la sélection sur la figure 6.17. La figure 6.18 montre le résultat d'un suivi de segments lorsque le robot se déplace.

3.4 Discussion

Au cours de la thèse nous avons utilisé principalement trois types d'amers, des points d'intérêt, des facettes et des segments.

Les points d'intérêt Le principal intérêt des points d'intérêts est leur facilité de détection, et la présence dans la littérature de nombreux algorithmes matures d'extraction, ainsi que pour un usage dans le SLAM. Mais les points fournissent un modèle de l'environnement avec une faible densité, et détaché de sa structure géométrique. Et l'extraction de points dans les images n'est pas robuste lors d'important changement de points de vue.

Facettes Nous avons proposé une extension des points d'intérêt à des facettes localement planes. Facettes qui contiennent une information de la structure géométrique plus importante, et qui présente une amélioration de la localisation du robot dans une approche SLAM, ceci vient du ait que chaque facette contient suffisamment d'information géométrique pour donner une observation complète de la position du robot, alors qu'il est nécessaire d'observer un minimum de trois points pour obtenir la même information. En revanche, les facettes nécessitent un temps de calcul plus important pour le calcul de l'homographie et de la normale. Étant centrées autour d'un point d'intérêt, elles soufrent du même problème de robustesse lors de changement important de point de vue.

Segments de ligne Nous avons proposé deux algorithmes d'extraction de segments, ainsi qu'un algorithme de suivie. Le choix entre *Hierarchical DSeg* et *DSeg* dépends de l'utilisation que l'on veut faire des segments, entre plus de segments ou des segments plus robustes. Nos expériences ont montrés que l'extraction de segments était aussi rapide que celle des points d'intérêts, et ils fournissent une information géométrique plus riche. Cependant, ils n'existent pas de bonne méthode d'appariement des segments dans une image.

La paramétrisation des segments est un aspect important dans le contexte du SLAM. Dans [Gee and Mayol, 2006] et [Smith et al., 2006] les lignes sont paramétrées dans le filtre de Kalman en utilisant deux points en "inverse depth", alors que nous avons utilisé une représentation de Plücker décrite dans [Solà et al., 2009].

4 Modèle géométrique

Une des limitations de nos expériences multi-robots et multi-cartes est l'absence de détection automatique des fermetures de boucle entre des robots aériens et terrestres. Les informations de signal (comme une petite image) est trop différente d'un robot à l'autre, sans compter sur les changements d'illumination.

Cependant, la géométrie de l'environnement est invariante, il est donc nécessaire d'utiliser les amers géométriques : mais elles sont très similaires les unes par rapport aux autres, il est donc nécessaire d'utiliser le voisinage de chaque amer, et de calculer plusieurs appariements en même temps. Pour ce faire nous avons introduit une structuration de l'environnement dans un graphe, structure qui est utilisée pour résoudre le problème de l'association de données au niveau géométrique.

4.1 Structuration d'un nuage d'amers géométriques dans un graphe

Les objets géométriques considérés font partie du système de géométrie Euclidien, et leurs paramètres sont exprimés dans l'espace Cartésien. Chaque objet peut avoir différente représentation, en fonction de l'utilisation, en effet, la représentation sera différente pour une utilisation dans le filtre de Kalman ou dans un processus d'appariement ou pour exprimer des relations entre les objets.

Objets géométriques Il existe deux types d'objets géométriques, des objets qui sont auto définies (par exemple avec un ensemble d'équations), appelé *objets composites* , ou des objets qui sont définies comme un ensemble d'autres objets, appelé *objets atomiques*.

Un point (P), une ligne (L), un segment de ligne (S), un cercle (Ci), ou encore un plan (Π) sont des objets atomiques. Tandis qu'un polygone (Po, ensemble de segments de ligne connectés les uns aux autres), une facette (F, plan limité par un polygone), un cylindre (Cy, volume délimité par deux cercles), une boîte (Bo volume délimité par un ensemble de plans) sont des exemples d'objets composites.

Pour définir des relations entre les objets, un certain nombre d'opérateurs sont utilisés : distance ($d(o_1, o_2)$), angle ($\theta(o_1, o_2)$), longueur ($l(S)$), périmètre ($p(Po)$), surface ($s(F)$), volume ($v(\mathring{a})$), projections ($proj(o_1, o_2)$)...

Relations entre objets géométriques Les objets géométriques sont reliés entre eux par des relations numériques, topologiques, constructives ou bien d'échelle.

Une *relation topologique* entre un ensemble d'objets géométriques est une relation qui réduit le degré liberté d'au moins l'un des objets.

Il existe une *relation constructive* entre un objet o et un ensemble d'objets $\{o_i\}$ lorsque touts les paramètres de o sont définis par les paramètres des objets de $\{o_i\}$.

Une *relation d'échelle* entre deux ensembles d'objets est établie lorsque les objets des deux ensembles correspondent au même objet réel.

Relations topologiques Les différentes relations topologiques définissent l'appartenance, l'intersection, le parallélisme ou si les objets sont perpendiculaires. Par exemple un point sur une ligne définit une relation topologique, de même pour deux lignes parallèles ou perpendiculaires.

Relations constructives et objets composites Les *objets composites* sont définis comme un ensemble d'objets, il existe donc une relation constructive entre l'objet composite o et tous les objets o_i qui le composent. Les objets o_i étant soit des objets atomiques, soit d'autres objets composites.

Dans un environnement, il n'est pas toujours possible d'observer tous les éléments d'un objet composite, par exemple la face du bas d'un cube posé sur le sol est caché, dans ce cas il est possible de créer des objets géométriques virtuels, non observables, mais qui viennent compléter l'information des objets réels.

Parmi les objets atomiques, il en est un qui n'a pas d'existence physique, il s'agit de la base. Par contre, on peut définir une base avec une combinaison d'objets atomiques : trois points, un point et une ligne...

Relations numériques Les objets atomiques sont exprimés à l'aide de paramètres numériques, tandis que les objets composites sont directement, ou indirectement, un groupe d'objets atomiques, et donc peuvent être exprimés avec des paramètres numériques. Il est donc possible de calculer un certains nombre de mesures entre les objets : *angle*, *distance*, *coplanarité*, *vecteur de transformation partielle*.

Un *angle* est défini entre deux objets définis par un vecteur, l'angle entre les objets est l'angle entre les vecteurs. La *distance* entre deux objets o_1 et o_2 est définie comme la distance minimum entre tous les points de o_1 et tous les points de o_2.

Pour les *objets composites*, plusieurs valeurs peuvent être calculé pour les relations numériques, mais en fait, pour la plus part des objets composites il est possible de définir une unique base qui sert alors à calculer les relations numériques.

Les relations topologiques peuvent être exprimés à l'aide des relations numériques.

Vecteur de transformation partielle L'angle et la distance permettent d'obtenir le lieu des positions de l'amer, les uns par rapport aux autres. Par exemple, considérons deux points P_1 et P_2, si les coordonnées de P_1 et la distance $d = |\overrightarrow{P_1P_2}|$ sont connues, alors le point P_2 appartient à la sphère centrée sur P_1 et de rayon d, mais l'information sur la position exacte de P_2 reste inconnue.

Si l'on considère quatre points, P_1, P_2, P_3 et P_4, avec P_2, P_3 et P_4 équidistant de P_1 avec une distance $d = |\overrightarrow{P_1P_2}| = |\overrightarrow{P_1P_3}| = |\overrightarrow{P_1P_4}|$. Si la distance entre les autres points est aussi connu, il est possible de reconstituer la géométrie (à une rotation près). Mais si cette distance n'est pas connu, par exemple, parce que les points ne sont pas toujours observés en même temps. Mais à un instant donné, on fait une observation des quatre points en même temps, et on veut être capable d'apparier les points. La seule connaissance de d ne permet de distinguer entre P_2, P_3 et P_4. Mais avec la connaissance des angles $\widehat{\overrightarrow{P_1P_2}, \overrightarrow{P_1P_3}}$, $\widehat{\overrightarrow{P_1P_2}, \overrightarrow{P_1P_4}}$

et $\widehat{\overrightarrow{P_1P_3}, \overrightarrow{P_1P_4}}$ (ce qui est possible avec la connaissance des vecteurs $\overrightarrow{P_1P_2}$, $\overrightarrow{P_1P_3}$ et $\overrightarrow{P_1P_4}$), il est alors possible d'obtenir la position relative des objets (voir Figure 8.1).

Nous appelons ce vecteur le *vecteur de transformation partielle.* La section 9.2.3 montre l'intérêt de ce vecteur, et la raison pour laquelle il est utile d'avoir une information de voisinage centrée sur l'amer plutôt que distribué.

Definition 12 *Le* ***vecteur de transformation partiel*** *est un vecteur unique $\overrightarrow{ptv}(o_1, o_2)$, entre deux objets géométriques o_1 et o_2, qui est ancré sur les points de deux objets: $\exists(P_1, P_2) \in (o_1, o_2)$ tel que $\overrightarrow{ptv}(o_1, o_2) = P_2 - P_1$.*

Les points d'ancrage du vecteur sont stable lors d'une transformation : soit une transformation $\mathcal{T}$ (rotation et/ou translation), si P_1 et P_2 sont deux points d'ancrage de $\overrightarrow{ptv}(o_1, o_2)$, alors $\mathcal{T}(P_1)$ et $\mathcal{T}(P_2)$ sont deux points d'ancrage de $\overrightarrow{ptv}(\mathcal{T}(o_1), \mathcal{T}(o_2))$:

$$\overrightarrow{ptv}(o_1, o_2) = P_2 - P_1 \Leftrightarrow \overrightarrow{ptv}(\mathcal{T}(o_1), \mathcal{T}(o_2)) = \mathcal{T}(P_2) - \mathcal{T}(P_1) \tag{C.5}$$

La valeur du *vecteur de transformation partiel* est dépendante de l'orientation de chaque amer, mais l'angle et la distance des points d'ancrage entre deux vecteurs sont indépendant.

Le *vecteur de transformation partiel* entre deux points est définie par le vecteur d'un point à l'autre, et entre un point et un autre objet il s'agit du vecteur de projection. Entre deux lignes, il s'agit du vecteur entre les deux points les plus proches. Il n'existe pas de vecteur entre deux plans, et entre un plan et une ligne, mais cela n'est pas un problème, le *vecteur de transformation partiel* peut être remplacé par le vecteur directeur ou normal pour calculer des angles entre amers.

4.2 Graphe géométrique

Les noeuds du graphe sont les amers de l'environnement, tandis que les arcs correspondent aux relations entre les amers. Il s'agit d'un graphe non complet, même le sous-graphe qui contient uniquement les arcs de relation numériques, en effet, bien qu'il soit théoriquement possible de connecter tous les noeuds avec une relation numérique, cela conduirait une importante complexité algorithmique, sans compter que plus les noeuds sont éloignés, plus l'incertitude sur les mesures numériques devient importantes.

Construction d'un graphe géométrique Par nature, il n'y a pas unicité du graphe, à partir du même jeu de données il est possible de créer une infinité de graphe différent. Cependant, pour la phase d'appariement, il est nécessaire de s'assurer d'une certaine répétabilité dans la construction du graphe, deux observations du même endroit doivent conduire à la création de deux graphes similaires. Tout en assurant que le graphe contienne un information utile, une trop grande incertitude sur un arc conduit un trop grand nombre d'appariements potentiels, ce qui augmente le temps de calcul pour les appariements, et diminue la fiabilité du résultat.

Construction globale Dans le cas d'un modèle préexistant de l'environnement, l'ensemble des amers est connu à la construction du graphe. Un noeud est créer pour chaque amer, et un arc avec les informations numériques est crée entre les amers qui sont proches les unes des autres.

Construction incrémentale La principale difficulté de la construction d'un graphe d'une carte construite par un robot mobile est de s'assurer que les *arcs numériques* contiennent une information utile, en d'autres mots, que la relation relative entre deux amers est connu avec un haut niveau de confiance. En supposant, qu'il n'y a pas de fermeture de boucle, le problème avec une approche de SLAM est que l'erreur sur la position augmente avec le déplacement du robot dans la carte, ce qui conduit à une augmentation de l'incertitude sur la position des amers, et donc à une augmentation de l'incertitude sur les mesures relatives qui sont obtenues à partir de ces positions, ce qui réduit l'utilité de l'information.

Le problème de l'augmentation de l'incertitude peut être résolu en utilisant une approche multi-cartes comme dans [Estrada et al., 2005] (voir section 1.2), en effet, avec une approche multi-cartes, l'erreur sur la position du robot et l'erreur local sur la position des amers sont maintenu réduite. Ce qui signifie que la construction incrémental du graphe ne peut obtenir des informations de relations entre amers fiables qu'entre des amers qui se trouve dans la même carte.

Il est important de remarquer que deux objets o_1 et o_2 peuvent apparaître simultanément dans deux cartes différentes M_1 et M_2 ($o_1 \in M_1$, $o_1 \in M_2$, $o_2 \in M_1$ et $o_2 \in M_2$), au quel cas la carte qui donne la transformation relative la plus faible est utilisé pour fournir l'information à l'*arc numérique* entre o_1 et o_2.

L'algorithme ci-dessous corresponds au rectangle SPAtial Feature de la figure 1.6, il prend en entré une carte finie $\mathcal{M} = o_i$, et procède à la mise à jour du graphe $G = (V, E)$.

1. $\forall o_i \in \mathcal{M}$, si il n'y a pas de noeud V correspondant à o_i, ajoute un nouveau noeud v_i dans le graphe G
2. $\forall (o_i, o_j) \in \mathcal{M}^2 / i \neq j$:
   - si il y a déjà un arc entre $e^n_{i,j} = (v_i, v_j)$, si la transformation dans la nouvelle carte est plus précise, alors l'arc $e^n_{i,j}$ est mise à jour. Ensuite si la condition $d(o_i, o_j) + 3\sigma_{d(o_i,o_j)} < \mathcal{T}_d$ devient fausse, l'arc est retiré $e^n_{i,j}$
   - si il n'y a pas d'arc et si $d(o_i, o_j) + 3\sigma_{d(o_i,o_j)} < \mathcal{T}_d$, alors un arc $e^n_{i,j} = (v_i, v_j)$ est ajouté

4.3 Descripteur géométrique

La représentation des objets sous forme de graphe est une méthode classique pour la reconnaissance d'objets, ce qui a conduit à de nombreux développement d'algorithmes d'appariements de graphe, malheureusement ces algorithmes ne s'appliquent pas bien à notre cas, pour des raisons de performances, mais aussi parce que généralement il est supposé que le graphe de l'objet est complet, ce qui n'est pas le cas pour un graphe de l'environnement.

Dans [Bailey, 2002], une approche utilisant un graphe complet de l'environnement est proposée pour résoudre le problème d'association de données dans le cadre du SLAM. Cette méthode est efficace pour des petits environnements, mais ne peut s'appliquer à un environnement de taille importante, c'est pourquoi nous avons développé une méthode similaire mais qui fonctionne en utilisant un graphe partiel, et s'applique donc à des environnements plus large :

1. Recherche d'une liste initiale de graines :
   (a) détermination d'un ensemble d'appariements initiaux en utilisant un descripteur géométrique
   (b) génération d'un graphe de compatibilité jointe (un graphe connectant deux appariements ayant un arc similaire) et un graphe d'exclusion (un graphe qui connecte deux appariements d'un même noeud)
   (c) calcul d'une transformation $\mathcal{T}$ entre les amers observés et ceux de la base de donnée, cette transformation est utilisée pour raffiner la liste des graines
2. L'ensemble d'appariements initiaux est complété en utilisant la transformation $\mathcal{T}$ et une approche de *propagation* à partir d'une graine.

La comparaison entre deux arcs est effectuée en utilisant la distance de Mahalanobis (équation 9.1).

Descripteur géométrique Une recherche d'appariement exhaustive avec une approche de *propagation* nécessite d'effectuer la recherche pour toutes les graines potentielles avec tous les noeuds du graphe. Comme cela serait extrêmement coûteux en temps de calcul, il est important de trouver une méthode qui utilise un nombre limité de graines. Pour les points d'intérêt ou les facettes, l'utilisation d'un descripteur de signal, comme la comparaison d'une image avec le contenu de la base de donnée permet d'effectuer cette sélection de graine. Mais pour de nombreuses raisons évoqués précédemment, il n'est pas toujours possible d'utiliser un descripteur utilisant une information de signal. Il est donc nécessaire d'utiliser un descripteur basé sur l'information géométrique.

Caractéristique importante d'un bon descripteur géométrique Comme l'information géométrique d'un seul amer n'est pas suffisante, le descripteur doit être capable de représenter l'information de **voisinage**. Mais un simple comptage du nombre d'amers de chaque type dans le voisinage n'est pas suffisant, il faut aussi rajouter des informations **métrique**, et leur **incertitude** associé, il faudra veiller à être le plus robuste possible aux **occultations**, et en effet, comme il va falloir effectuer de nombreuses comparaisons entre différent descripteurs, il faut pouvoir effectuer des **comparaisons rapides**.

Descripteur de formes pour un nuage de points 3D Dans [Ankerst et al., 1999], l'espace est divisé autour du centre de masse d'un objet, et la valeur de chaque élément de l'histogramme est le nombre de point contenu dans un secteur donné de l'espace.

Généralisation pour des types variés d'amers Ce descripteur de nuage de points 3D s'applique directement pour un modèle d'environnement composé uniquement de points, en effet, au lieu d'utiliser comme le centre de masse comme référence du descripteur, il suffit d'utiliser l'amer pour laquelle on cherche à calculer un descripteur. De la même manière, puisque l'on peut calculer une distance et un angle entre tous types d'objets géométriques, le descripteur s'applique aussi bien à des points, qu'à des segments, plans, cubes...

A l'histogramme sur les distances et angles, on en rajoute un qui utilise l'information fournie par le *vecteur caractéristique*, qui est soit le *vecteur de transformation partielle*, soit la normal soit le vecteur directeur. En effet, ce *vecteur caractéristique* permet de calculer un angle entre tous les amers voisins d'un amer donné.

Des histogrammes pondérés par l'incertitude Dans le cas d'un nuage dense de points 3D, le bruit est considéré comme négligeable par rapport à la quantité de points. Mais cette hypothèse n'est pas valide dans le cas d'un modèle de l'environnement contenant un faible nombre d'amers, il est donc nécessaire de prendre en compte l'information d'incertitude dans le calcul de l'histogramme. Ce qui est fait en pondérant le vote de chaque mesure dans l'histogramme par son incertitude. Par exemple dans le cas d'un histogramme à deux éléments $]-\infty, 0.0]$ et $[0.0, +\infty[$, la mesure $\mu = 0.05$ avec un écart type de $\sigma = 0.1$ contribue entièrement au deuxième élément, alors qu'il y a une probabilité de 0.30854 que la mesure réel soit inférieur à 0.0 et corresponde au premier élément de l'histogramme. En utilisant un histogramme pondéré par l'incertitude, le premier élément reçoit un vote de 0.300854 et le second de 0.699146.

Comparaison de descripteurs Le score de corrélation ZNCC est utilisé pour comparer deux descripteurs.

Résultats Pour valider le descripteur, nous avons synthétisé un certain nombre de configuration et étudier le comportement de la comparaison. En particulier, nous avons voulu vérifier que le descripteur était suffisamment distinctif, comme montré sur la figure 9.7 pour des segments ou la figure 9.8 pour des points. Il a fallu aussi s'assurer de la résistance au bruit, figure 9.10 et aux occultations, figure 9.11.

4.4 Appariement de graphe

L'appariement de graphe se déroule en deux étapes, une première étape de sélection de graines, et une étape de propagation.

Sélection de graines Soit le graphe G^{obs} des amers observés et le graphe G^{db} des amers du modèle d'environnement. Pour la sélection de graines, la première étape consiste à trouver l'ensemble des *appariements possibles* en utilisant uniquement le descripteur. Ensuite le graphe de *compatibilité jointe* et le graphe d'*exclusion* sont utilisés pour sélectionner les appariements possibles. En effet, l'appariement sur le descripteur génère plusieurs hypothèses d'appariements pour un amer donné, et il est nécessaire de n'en sélectionner qu'un seul, et de s'assurer qu'il est compatible avec d'autres appariements dans le voisinage.

A partir de l'ensemble d'*appariements possible* et deux graphes, il existe plusieurs ensemble d'hypothèse, il s'agit donc de les départager, pour se faire un algorithme de RANSAC est utilisé pour calculer la transformation entre l'observation et le modèle, cette transformation $\mathcal{T}$ est utilisé pour trouver l'ensemble avec le plus grand nombre d'appariement qui sont compatibles avec cette transformation.

Propagation La transformation $\mathcal{T}$ est ensuite utilisé pour tenté de trouver d'autres appariements en se déplaçant dans le graphe.

Tests en simulation La figure 9.14 montre la capacité de l'algorithme a trouver des appariements dans un environnement de grande taille sans connaissance à priori de la position.

5 Conclusion et perspective

Enrichir l'information contenu dans les modèles d'environnement nécessite de développer de nouveaux algorithmes pour extraire plus d'information des données perçues. Dans un premier temps nous avons proposés deux nouveaux algorithmes de détection d'amers, pour des facettes et des segments de lignes.

Mais en l'absence d'un processus efficace d'appariement de segments dans une image, et comme nous souhaitons pouvoir être capable d'apparier des données qui proviennent de différents capteurs, ainsi que des données provenant de systèmes d'information géographique, nous avons aussi proposé une représentation de la géométrie de l'environnement dans un modèle hétérogène d'amer, avec une méthode d'appariement des amers.

Les travaux futurs devront se focaliser sur une validation plus approfondie de ce modèle, dans un premier temps dans un contexte de vision, et ensuite en étendant les tests à d'autres types de capteurs (en particulier, les lasers 3D), et à l'utilisation de systèmes d'informations géographiques. Mais ceci soulève un grand nombre de challenges, autant sur le modèle géométrique, que la nécessité de développer de nouveaux algorithmes, et d'améliorer les algorithmes existants.

5.1 Un environnement multi source

Jusqu'à présent, l'essentiel des travaux sur la robotique mobile d'extérieur s'est concentré sur des robots autonomes et solitaires dans l'environnement. Mais nous pensons que les futurs robots devront être capable de communiquer avec une grande variété de systèmes, soit d'autres robots, des réseaux de capteurs ou des base de données d'information.

Une approche multi-cartes, comme présentée dans [Vidal-Calleja et al., 2009], permet à un système de robots de cartographier le même environnement et de partager l'information. Bien que conçu pour fonctionner de manière décentralisée, dans l'implémentation actuelle certains aspects demeurent centralisé. Ainsi, les robots construisent des cartes localement indépendantes, mais ils partagent le même graphe de cartes. Dans une application réel, les robots ne seront pas capables de communiquer en permanence, ainsi chaque robot devras conserver sa copie du graphe de cartes, et échanger l'information lorsque c'est possible. Il est nécessaire d'être prudent lors de l'échange d'information pour s'assurer que le système reste consistant (en particulier, d'éviter que l'information soit utilisée deux fois.

Il y a cependant un besoin de *sélectionner* quel information doit être échangée. Regrouper des amers est aussi une possibilité qui permet de structurer l'information et de réduire l'information à échanger, elle permet aussi de réduire l'information utilisé dans les cartes stochastiques ([Gee et al., 2007]).

5.2 Détection d'amers

Il est évident que plus il y a d'information dans les modèles d'amer, meilleur est le modèle géométrique résultant. Nous allons présenter quelques perspectives sur la définition et l'extraction d'amers.

Vers un descripteur de segment de lignes Nos détecteurs de segments ne sont pour le moment capables d'extraire que l'information géométrique d'une image, il n'y pas de descripteur du signal associé aux segments. Bien qu'un procédure d'appariement de segments similaire à celle utilisée pour les points ne soit pas nécessaire, un descripteur du signal pourrait être utile pour mieux faire la différence entre les segments.

Cependant définir un descripteur fiable du signal pour les segments de ligne n'est pas évident. Pour les points, une hypothèse de planéité local et d'invariance du signal dans le voisinage du point est utilisée pour définir le descripteur comme étant une portion de l'image (ou comme base à un descripteur de gradients). Mais cette hypothèse n'est pas valide pour les segments, dont la longueur implique qu'il n'est pas possible d'ignorer la distorsion provoquée par la projection. Dans une image, les segments correspondent souvent à un coin 3D, ce qui signifie que seul un côté du segment a un descripteur stable, ce qui nécessite l'utilisation de deux descripteurs, un pour chaque côté du segment. Il est envisageable d'utiliser l'information de profondeur, obtenu par l'estimation des paramètres géométrique 3D provenant d'un processus de SLAM, pour calculer la projection de l'image autour du segment, d'une manière similaire à ce que nous avons fait pour les facettes.

Détection de plans Par rapport aux points, les segments et facettes améliorent la densité du modèle de l'environnement. Mais l'extraction de plans permettraient un bien meilleur modèle géométrique. Quelques méthodes d'extraction de plans ont été mentionnées dans le chapitre 7: une autre possibilité serait d'utiliser les facettes et segments détectés, et de les regrouper dans des hypothèses de plans. Ensuite un algorithme devra être développé pour confirmer ses hypothèses, par exemple en effectuant une recherche d'homographie dans le voisinage des amers de l'hypothèse.

Objets composites: regrouper les amers Puisque des amers individuels ne sont pas suffisamment discriminantes, et peuvent conduire à de faux appariements, de nombreux travaux ont été consacrés à l'appariement de plusieurs amers en même temps ([Neira and Tardos, 2001, Bailey, 2002], ainsi que les travaux décrits dans III). Mais nous pensons qu'il serait intéressant d'étendre la procédure d'appariement de graphe pour travailler directement avec des groupes d'objets, pour augmenter la discriminance des objets individuels. La principale difficulté est de définir un bon algorithme de regroupement d'amers, par exemple au chapitre 5, l'algorithme de regroupement repose sur la définition d'un centre de gravité d'un ensemble de facettes, mais l'on obtient un résultat différent selon que l'ensemble d'amers est détecté partiellement ou non. Ceci n'empêche pas la procédure d'appariement de fonctionner, mais empêche certains groupes d'être utilisé comme point de départ, et diminue l'efficacité de l'algorithme.

Le descripteur géométrique du chapitre 9 pourrait être utilisé pour déterminer quels amers font probablement partie d'un objet composite. Pour ce faire, il est nécessaire de définir un modèle de l'objet composite (par exemple, quatre segments perpendiculaires), et ensuite de calculer le descripteur correspondant à chaque sous-objet du modèle. Mais comme les objets observés seront vraisemblablement connectés à des amers extérieur à l'objet composite, il n'est pas possible d'utiliser un score de corrélation, il sera nécessaire de vérifier si l'objet a un descripteur D_f qui domine le descripteur du modèle D_m $(\forall i D_f(i) \geq D_m(i))$. Ensuite une procédure d'appariement de graphe peut être utilisée pour vérifier la compatibilité entre un ensemble d'objet et le modèle composite.

5.3 Graphe géométrique

Extension du graphe Il est important de réduire le nombre d'objets géométriques testé dans un algorithme d'appariement de graphe. L'estimation de la position courante du robot dans l'environnement peut être utilisée, mais quand l'incertitude sur la position du robot est trop importante, d'autres informations peuvent être utilisées pour réduire le nombre de candidats. Comme dans le chapitre 5, chaque amer peut être associée à un descripteur du signal qui est alors utilisé pour réduire le nombre de candidats. Il est aussi intéressant d'associer les objets à un descripteur plus général de l'environnement, mélangeant cartes topologiques et cartes métriques, ou d'utiliser une approche de type FabMaps pour détecter les fermetures de boucles.

Une telle information pourrait être insérée dans le graphe par l'introduction d'un nouveau type de vertex, appelé "lieu" qui contient l'information du descripteur de lieu.

Construction du graphe Des améliorations à la construction du graphe pourrait être apporté sur le choix de quel amer est connecté à quel autre amer. Pour le moment, les amers sont connectés en utilisant une mesure de distance, pour s'assurer que l'incertitude sur les distances reste faible. En utilisant une carte de visibilité, ou la matrice d'information (deux amers ont été perçus de la même position lorsque leur corrélation dans la matrice d'information est importante), ainsi il serait possible de ne connecter deux amers que si il est probable de les observer ensemble. Ceci devrait augmenter la probabilité d'observer les amers connectés en même temps.

Un des problèmes de l'algorithme d'appariement de graphe est que les occlusions diminuent l'efficacité du descripteur géométrique, ce qui peut empêcher de bons appariements de se produire. Un algorithme qui détecte les occlusions pourraient être utilisé pour déclencher de nouvelles observations avant d'essayer de trouver des appariements.

Plus généralement, un important travail est nécessaire sur les stratégies d'acquisition de données, afin de faire une meilleur utilisation d'un groupe de robots, et d'accroître la quantité d'information contenue dans les cartes.

Bibliography

[Abdulkader, 1998] Abdulkader, A. M. (1998). *Parallel Algorithms for Labelled Graph Matching*. PhD thesis, Colorado School of Mines.

[Adelson et al., 1984] Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., and Ogden, J. M. (1984). Pyramid methods in image processing. *RCA Engineer*, pages 33–41.

[AG,] AG, M. I. Swissranger 3d camera. Available from: `http://www.mesa-imaging.ch`.

[Angeli et al., 2008] Angeli, A., Filliat, D., Doncieux, S., and Meyer, J.-A. (2008). A fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions On Robotics*.

[Ankerst et al., 1999] Ankerst, M., Kastenmüller, G., Kriegel, H.-P., and Seidl, T. (1999). 3d shape histograms for similarity search and classification in spatial databases. In *Advances in Spatial Databases*, pages 207–226.

[Bailey, 2002] Bailey, T. (2002). *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. PhD thesis, University of Sydney.

[Bailey and Durrant-Whyte, 2006] Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous Localisation and Mapping (SLAM): Part II - State of the Art. *Robotics and Automation Magazine*.

[Baker and Matthews, 2001] Baker, S. and Matthews, I. (2001). Equivalence and efficiency of image alignment algorithms. In *Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition*.

[Baker and Matthews, 2004] Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255.

[Ballard, 1981] Ballard, D. (1981). Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*.

[Ballard, 1987] Ballard, D. H. (1987). Generalizing the hough transform to detect arbitrary shapes. *Readings in computer vision: issues, problems, principles, and paradigms*, pages 714 – 725.

[Baltzakis et al., 2003] Baltzakis, H., Argyros, A., and Trahanias, P. (2003). Fusion of laser and visual data for robot motion planning and collision avoidance. *Machine Vision Applications*, 15(2):92–100.

[Barreto and Araujo., 2001] Barreto, J. P. and Araujo., H. (2001). Issues on the geometry of central catadioptric imaging. In *Computer Vision and Pattern Recognition.*

[Baumstarck et al., 2006] Baumstarck, P. G., Brudevold, B. D., and Reynolds, P. D. (2006). Learning planar geometric scene context using stereo vision. Master's thesis, Stanford. Available from: `http://www.stanford.edu/class/cs229/proj2006/BaumstarckBrudevoldReynolds-LearningPlanarGeometricSceneContextUsingStereoVision.pdf`.

[Bay et al., 2008] Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Surf: Speeded up robust features. *mputer Vision and Image Understanding*, 110:346–359.

[Beis and Lowe, 1997] Beis, J. S. and Lowe, D. G. (1997). Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Conference on Computer Vision and Pattern Recognition*, pages 1000–1006.

[Berger and Lacroix, 2008] Berger, C. and Lacroix, S. (2008). Using planar facets for stereo-vision slam. In *IEEE/RSJ International Conference on Intelligent Robots and Systems.*

[Bouguet,] Bouguet, J.-Y. Camera calibration toolbox for matlab. Available from: `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[Brown, 1992] Brown, L. (1992). A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376.

[Burns et al., 1986] Burns, J. B., Hanson, A. R., and Riseman, E. M. (1986). Extracting straight lines. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 8(4):425–455.

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 8, pages 679–714.

[Castle et al., 2007] Castle, R., Gawley, D., Klein, G., and Murray, D. (2007). Towards simultaneous recognition, localization, and mapping for hand-held and wearable cameras. In *Proceedings of the 2007 IEEE Conference on Robotics and Automation, Roma, Italy.*

[Cemagref et al.,] Cemagref, Lasmea, and Thales. High definition laser. Available from: `https://impala.cemagref.fr/`.

[Chatila and Laumond, 1985] Chatila, R. and Laumond, J.-P. (1985). Position referencing and consistent world modeling for mobile robots. In *IEEE International Conference on Robotics and Automation, St Louis (USA)*, pages 138–145.

[Chekhlov et al., 2007] Chekhlov, D., Gee, A., Calway, A., and Mayol-Cuevas, W. (2007). Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual slam. In *International Symposium on Mixed and Augmented Reality (ISMAR).*

[Chen and al, 2007] Chen, L. and al (2007). A simple tracking approach for line extraction. In *Proceedings of the International Conference on Wavelet Analysis and Pattern Recognition.*

[Cormen et al., 1990] Cormen, T., Leiserson, C., and Rivest, R. (1990). *Introduction to Algorithms.* McGraw-Hill.

[Cummins and Newman, 2008] Cummins, M. and Newman, P. (2008). Fab-map: Probabilistic localization and mapping in the space of appearance. *International Journal of Robotics Research*, 27(6):647–665.

[Cummins and Newman, 2009] Cummins, M. and Newman, P. (2009). Highly scalable appearance-only slam - fab-map 2.0. In *Robotics: Science and Systems*.

[David and DeMenthon, 2005] David, P. and DeMenthon, D. (2005). Object recognition in high clutter images using line features. In *Tenth IEEE International Conference on Computer Vision, 2005.*, volume 2, pages 1581–1588.

[Davison, 2003] Davison, A. (2003). Simultaneous localisation and mapping with a single camera. In *9th ICCV, Nice (France)*.

[Deriche and Faugeras, 1990] Deriche, R. and Faugeras, O. D. (1990). Tracking line segments. In *1th European Conference on Computer vision*, page 259–268. Springer-Verlag New York, Inc.

[Desolneux et al., 2000] Desolneux, A., Moisan, L., and Morel, J. (2000). Meaningful alignments. *International Journal of Computer Vision*.

[Dhond and Aggarwal, 1989] Dhond, U. and Aggarwal, J. (1989). Structure from stereo-a review. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1489–1510.

[Dissanayake et al., 2001] Dissanayake, G., Newman, P. M., Durrant-Whyte, H.-F., Clark, S., and Csorba, M. (2001). A solution to the simultaneous localization and map building (slam) problem. *IEEE Transaction on Robotic and Automation*, 17(3):229–241.

[Duda and Hart, 1972] Duda, R. O. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*.

[Dufournaud et al., 2004] Dufournaud, Y., Schmid, C., and Horaud, R. (2004). Image matching with scale adjustment. *Computer Vision and Image Understanding*, 93(2):175–194.

[Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous Localisation and Mapping (SLAM): Part I - The Essential Algorithms. *Robotics and Automation Magazine*.

[Eade and Drummond, 2006] Eade, E. and Drummond, T. (2006). Edge landmarks in monocular slam. In *British Machine Vision Conference, Edinburgh (UK)*.

[Egenhofer, 1991] Egenhofer, M. J. (1991). Reasoning about binary topological relations. In *International Symposium on Advances in Spatial Databases*, pages 143–160.

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

[Etemadi, 1992] Etemadi, A. (1992). Robust segmentation of edge data. In *Int. Conf. on Image Processing and its Applications*, page 311–314.

[Everett, 1995] Everett, H. (1995). *Sensors for Mobile Robots*. A. K. Peters Ltd.

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the Association of Computer Machinery.*

[Folkesson and Christensen, 2004] Folkesson, J. and Christensen, H. I. (2004). Graphical slam - a self-correcting map. In *IEEE International Conference on Robotics and Automation.*

[Freeman and Adelson, 1991] Freeman, W. T. and Adelson, E. H. (1991). The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906.

[Gee et al., 2007] Gee, A., Chekhlov, D., Mayol-Cuevas, W., and Calway, A. (2007). Discovering planes and collapsing the state space in visual slam. In *British Machine Vision Conference.*

[Gee and Mayol, 2006] Gee, A. P. and Mayol, W. (2006). Real-time model-based slam using line segments. In *International Symposium on Visual Computing.*

[Geyer and Daniilidis, 1999] Geyer, C. and Daniilidis, K. (1999). Catadioptric camera calibration. In *International Conference on Computer Vision*, pages 398–404.

[Gonzalez and Woods, 2002] Gonzalez, R. C. and Woods, R. E. (2002). *Digital Image Processing.* Prentice Hall.

[Gonzalez-Barbosa and Lacroix, 2002] Gonzalez-Barbosa, J.-J. and Lacroix, S. (2002). Rover localization in natural environments by indexing panoramic images. In *IEEE International Conference on Robotics and Automation.*

[Google,] Google. Maps. Available from: `http://maps.google.com`.

[Guivant and Nebot, 2001] Guivant, J. and Nebot, E. (2001). Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotics and Automation*, 17(3):242–257.

[Guo et al., 2008] Guo, S.-Y., Kong, Y.-G., Tang, Q., and Zhang, F. (2008). Probabilistic hough transform for line detection utilizing surround suppression. In *International Conference on Machine Learning and Cybernetics*, volume 5, pages 2993–2998.

[Harris, 1992] Harris, C. (1992). *Tracking with Rigid Objects.* MIT Press.

[Harris and Stephens, 1988a] Harris, C. and Stephens, M. (1988a). A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151.

[Harris and Stephens, 1988b] Harris, C. and Stephens, M. (1988b). A combined corner and edge detector. In *4th Alvey Vision Conference, Manchester (UK)*, pages 147–151.

[Hartley and Zisserman, 2000] Hartley, R. and Zisserman, A. (2000). *Multiple View Geometry in Computer Vision.* Cambridge University Press.

[Horn, 1984] Horn, B. K. P. (1984). Extended gaussian images. *Proceedings of the IEEE*, 72(2):1671–1686.

[Hough, 1962] Hough, P. (1962). Method and means for recognizing complex patterns. US Patent 3069654.

[IGN,] IGN. Géoportail. Available from: `http://www.geoportail.fr`.

[Jung and Lacroix, 2001] Jung, I.-K. and Lacroix, S. (2001). A robust interest point matching algorithm. In *8th ICCV, Vancouver (Canada)*.

[Kahn et al., 1990] Kahn, P., Kitchen, L., and Riseman, E. M. (1990). A fast line finder for vision-guide robot navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence archive*, 12(11):1098–1102.

[Kang and Ikeuchi, 1991] Kang, S. and Ikeuchi, K. (1991). Determining 3-d object pose using the complex extended gaussian image. In *Computer Vision and Pattern Recognition*, page 580–585.

[Kazhdan et al., 2004] Kazhdan, M., Funkhouser, T., and Rusinkiewicz, S. (2004). Symmetry descriptors and 3d shape matching. In *Geometry Processing*.

[Kieffer et al., 2000] Kieffer, M., Jaulin, L., Walter, E., and Meizel, D. (2000). Robust autonomous robot localization using interval analysis. *Reliable Computing*, 6(3):337–362.

[Kiryati et al., 1991] Kiryati, N., Eldar, Y., and Bruckstein, A. M. (1991). A probabilistic hough transform. *Pattern Recognition*, pages 303–316.

[Knight et al., 2001] Knight, J., Davison, A., and Reid, I. (2001). Towards constant time SLAM using postponement. In *Proc. IEEE/RSJ Conf. on Intelligent Robots and Systems, Maui, HI*, volume 1, pages 406–412. IEEE Computer Society Press.

[Lacroix et al., 1999] Lacroix, S., Mallet, A., and Chatila, R. (1999). Rover self localization in planetary-like environments. In *International Symposium on Artificial Intelligence*, pages 433–440.

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

[Lemaire and Lacroix, 2007a] Lemaire, T. and Lacroix, S. (2007a). Monocular-vision based SLAM using line segments. In *IEEE International Conference on Robotics and Automation, Roma (Italy)*.

[Lemaire and Lacroix, 2007b] Lemaire, T. and Lacroix, S. (2007b). Slam with panoramic vision. *Journal of Field Robotics*, pages 91–111.

[Lepetit and Fua, 2006] Lepetit, V. and Fua, P. (2006). Keypoint recognition using randomized trees. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1465–1479.

[Lhuillier and Quan, 2003] Lhuillier, M. and Quan, L. (2003). Match propagation for image-based modeling and rendering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1140–1146.

[Lindeberg, 1991] Lindeberg, T. (1991). *Discrete Scale-Space Theory and the Scale-Space Primal Sketch.* PhD thesis, Department of Numerical Analysis and Computer Science, KTH.

[Lindeberg, 1994] Lindeberg, T. (1994). Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, pages 224–270.

[Lindeberg, 1998] Lindeberg, T. (1998). Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30:77–116.

[Lindeberg, 2009] Lindeberg, T. (2009). Scale-space. *Encyclopedia of Computer Science and Engineering*, IV:2495–2504.

[Lowe, 1999] Lowe, D. (1999). Object recognition from local scale-invariant features. In *7th International Conference on Computer Vision, Kerkyra, Corfu (Greece)*, pages 1150–1157.

[Lowe, 2004] Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[Lu and Milios, 1997] Lu, F. and Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349.

[Malis, 2004] Malis, E. (2004). Improving vision-based control using efficient second-order minimization techniques. In *IEEE ICRA*.

[Mansouri et al., 1987] Mansouri, A.-R., Malowany, A., and Levine, M. (1987). Line detection in digital pictures: a hypothesis prediction / verification paradigm. *Computer Vision, Graphics and Image Processing*, 40(1):95–114.

[Martin and Crowley, 1995] Martin, J. and Crowley, J. (1995). Comparison of correlation techniques. In *International Conference on Intelligent Autonmous Systems, Karlsruhe (Germany)*, pages 86–93.

[Mikolajczyk and Schmid, 2001] Mikolajczyk, K. and Schmid, C. (2001). Indexing based on scale invariant interest points. In *International Conference on Computer Vision*, pages 525–531.

[Mikolajczyk and Schmid, 2002] Mikolajczyk, K. and Schmid, C. (2002). An affine invariant interest point detector. In *European Conference on Computer Vision.*

[Mikolajczyk and Schmid, 2004] Mikolajczyk, K. and Schmid, C. (2004). Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60:63–86.

[Mikolajczyk and Schmid, 2005] Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1615–1630.

[Molton et al., 2004] Molton, N., Davison, A., and Reid, I. (2004). Locally planar patch features for real-time structure from motion. In *BMVC.*

[Montemerlo et al., 2003] Montemerlo, M., Thrun, S., and Wegbreit, B. (2003). Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *International Conference on Artificial Intelligence (AAAI)*. Available from: `http://www-2.cs.cmu.edu/~mmde/mmdeijcai2003.pdf`.

[Montiel et al., 2006] Montiel, J. M. M., Civera, J., and Davison, A. J. (2006). Unified inverse depth parametrization for monocular SLAM. In *RSS 2006*.

[Moon and Stirling, 2000] Moon, T. K. and Stirling, W. C. (2000). *Mathematical Methods and Algorithms for Signal Processing*. Prentice Hall.

[Moravec, 1996] Moravec, H. (1996). Robot spatial perception by stereoscopic vision and 3d evidence grids. Technical Report CMU-RI-TR-96-34, The Robotics Institute - CMU.

[Moravec and Elfes, 1985] Moravec, H. and Elfes, A. (1985). High resolution maps from wide angle sonar. In *IEEE Conference on Robotics and Automation*, pages 116–121.

[Murray and Little, 2005a] Murray, D. and Little, J. (2005a). Patchlets: representing stereo vision data with surface elements. In *WACV, Colorado*.

[Murray and Little, 2005b] Murray, D. and Little, J. J. (2005b). Patchlets: representing stereo vision data with surface elements. In *Workshop on Application of Computer Vision*.

[Neira and Tardos, 2001] Neira, J. and Tardos, J. (2001). Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*.

[Neubert et al., 2008] Neubert, P., Protzel, P., Vidal-Calleja, T., and Lacroix, S. (2008). A fast visual line segment tracker. In *13th IEEE International Conference on Emerging Technologies and Factory Automation, Hamburg (Germany)*.

[Newman, 1999] Newman, P. (1999). *On the Structure and Solution of the Simultaneous Localisation and Map Building Problem*. PhD thesis, Australian Centre for Field Robotics - The University of Sydney. Available from: `http://oceanai.mit.edu/pnewman/papers/pmnthesis.pdf`.

[Nuske et al., 2009] Nuske, S., Roberts, J., and Wyeth, G. (2009). Robust outdoor visual localization using a three-dimensional-edge map. *Journal of Field Robotics*, 26(9):728–756.

[Olson et al., 2007] Olson, E., Leonard, J., and Teller, S. (2007). Spatially-adaptive learning rates for online incremental slam. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA.

[Osada et al., 2001] Osada, R., Funkhouser, T., Chazelle, B., and Dobkin, D. (2001). Matching 3d models with shape distributions. In *International Conference on Shape Modeling and Applications*.

[R. Grompone von Gioi and Randall, 2008] R. Grompone von Gioi, J. Jakubowicz, J.-M. M. and Randall, G. (2008). Lsd: A fast line segment detector with a false detection control. *IEEE Transaction on Pattern Analysis and Machine Intelligence*.

[Schmid and Mohr, 1997] Schmid, C. and Mohr, R. (1997). Local greyvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5).

[Schmid et al., 1998] Schmid, C., Mohr, R., and Bauckhage, C. (1998). Comparing and evaluating interest points. In *International Conference on Computer Vision*.

[Schmid and Zisserman, 2000] Schmid, C. and Zisserman, A. (2000). The geometry and matching of lines and curves over multiple views. *International Journal on Computer Vision*, pages 199–234.

[Shannon, 1949] Shannon, C. E. (1949). Communication in the presence of noise. *Institute of Radio Engineers*.

[Shi and Tomasi, 1994a] Shi, J. and Tomasi, C. (1994a). Good features to track. In *IEEE CVPR*, Seattle (USA).

[Shi and Tomasi, 1994b] Shi, J. and Tomasi, C. (1994b). Good features to track. In *IEEE International Conference on Computer Vision and Pattern Recognition, Seattle (USA)*, pages 593–600.

[Silveira et al., 2007] Silveira, G., Malis, E., and Rives, P. (2007). An efficient direct method for improving visual SLAM. In *IEEE International Conference on Robotics and Automation, Rome, Italy*.

[Skiena, 2008] Skiena, S. S. (2008). *The Algorithm Design Manual*. Springer.

[Smith et al., 2006] Smith, P., Reid, I., and Davison, A. (2006). Real-time monocular slam with straight lines. In *British Machine Vision Conference*.

[Smith et al., 1987] Smith, R., Self, M., and Cheeseman, P. (1987). A stochastic map for uncertain spatial relationships. In *Robotics Research: The Fourth International Symposium, Santa Cruz (USA)*, pages 468–474.

[Sobel and Feldman, 1968] Sobel, I. and Feldman, G. (1968). A 3x3 isotropic gradient operator for image processing. presented at a talk at the Stanford Artificial Project.

[Sola et al., 2005] Sola, J., Lemaire, T., Devy, M., Lacroix, S., and Monin, A. (2005). Delayed vs undelayed landmark initialization for bearing only SLAM. In *SLAM Workshop, IEEE International Conference on Robotics and Automation, Barcelona (Spain)*.

[Solà et al., 2009] Solà, J., Vidal-Calleja, T., and Devy, M. (2009). Undelayed initialization of line segments in monocular slam. In *International Conference on Intelligent RObots and Systems*.

[Steger, 1998] Steger, C. (1998). An unbiased detector of curvilinear structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence archive*, 20(2):113–125.

[Stricker and Swain, 1994] Stricker, M. and Swain, M. (1994). The capacity of color histogram indexing. In *In Proceedings 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 704–708.

[Swain and Ballard, 1991] Swain, M. J. and Ballard, D. H. (1991). Color indexing. *International Journal of Computer Vision*.

[Tangelder and Veltkamp, 2004] Tangelder, J. W. H. and Veltkamp, R. C. (2004). A survey of content based 3d shape retrieval methods. In *Shape Modeling International*, pages 145–156.

[Thrun, 2002] Thrun, S. (2002). Robotic mapping: A survey. In Lakemeyer, G. and Nebel, B., editors, *Exploring Artificial Intelligence in the New Millenium.* Morgan Kaufmann.

[Thrun et al., 2000] Thrun, S., Burgard, W., and Fox, D. (2000). A real-time algorithm for mobile robot with applications to multi-robot and 3d mapping. In *IEEE International Conference on Robotics and Automation, San Francisco, CA (USA).*

[Thrun et al., 2004] Thrun, S., Liu, Y., Koller, D., Ng, A., Ghahramani, Z., and Durrant-Whyte, H. (2004). Simultaneous Localization and Mapping With Sparse Extended Information Filters. *International Journal of Robotics Research.* Submitted for journal publication.

[Thrun and Montemerlo, 2005] Thrun, S. and Montemerlo, M. (2005). The GraphSLAM algorithm with applications to large-scale mapping of urban structures. *International Journal on Robotics Research*, 25(5/6):403–430.

[Velodyne,] Velodyne. High definition laser. Available from: `http://www.velodyne.com/lidar/vision/default.aspx`.

[Vidal-Calleja et al., 2009] Vidal-Calleja, T. A., Berger, C., and Lacroix, S. (2009). Event-driven loop closure in multi-robot mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems.*

[Viejo and Cazorla, 2006] Viejo, D. and Cazorla, M. (2006). Plane extraction and error modeling of 3d data. In *International Symposium on Rototics and Automation.*

[Vosselman and Dijkman, 2001] Vosselman, G. and Dijkman, S. (2001). 3d building model reconstruction from point clouds and ground plans. In *International Archives on Photogrammetry and Remote Sensing.*

[Weingarten and Siegwart, 2006] Weingarten, J. and Siegwart, R. (2006). 3d slam using planar segments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems.*

[Wijk and Christensen, 2000] Wijk, O. and Christensen, H. (2000). Triangulation based fusion of sonar data for robust robot pose tracking. *IEEE Transactions on Robotics and Automation*, 16(6):740–752.

[Williams et al., 2008] Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., and Tardos, J. (2008). An image–to–map loop closing method for monocular slam. In *IEEERSJ International Conference on Intelligent Robots and Systems.*

[Witkin, 1983] Witkin, A. (1983). Scale-space filtering. In *8th International Conference of Artificial Intelligence*, pages 1019–1022.

[Wolf et al., 2005] Wolf, D., Howard, A., and Sukhatme, G. (2005). Towards geometric 3d mapping of outdoor environments using mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems.*

[Xiong and Matthies, 1997] Xiong, Y. and Matthies, L. (1997). Error analysis of a real time stereo system. In *IEEE CVPR*, pages 1087–1093.

[Xu and Velastin, 1994] Xu, C. and Velastin, S. (1994). The mahalanobis distance hough transform with extended kalman filter refinement. In *Int. Conf. on Circuits and Systems.*

[Yacoub and Jolion, 1995] Yacoub, S. B. and Jolion, J.-M. (1995). Hierarchical line extraction. In *IEEE Proceedings Vision Image Signal Processing*, volume 142.

[Zabih and Woodfill, 1994] Zabih, R. and Woodfill, J. (1994). Non-parametric local transforms for computing visual correspondence. In *Third European Conference on Computer Vision, Stockholm (Sweden)*.

AUTHOR: Cyrille Berger

TITLE: Perception of the environment geometry for autonomous navigation

ENGLISH SUMMARY:

The goal of the mobile robotic research is to give robots the capability to accomplish missions in an environment that might be unknown. To accomplish his mission, the robot need to execute a given set of elementary actions (movement, manipulation of objects...) which require an accurate localisation of the robot, as well as a the construction of good geometric model of the environment.

Thus, a robot will need to take the most out of his own sensors, of external sensors, of information coming from an other robot and of existing model coming from a Geographic Information System. The common information is the geometry of the environment.

The first part of the presentation will be about the different methods to extract geometric information. The second part will be about the creation of the geometric model using a graph structure, along with a method to retrieve information in the graph to allow the robot to localise itself in the environment.

KEYWORDS: mobile robotic, SLAM, environment modelling

AUTEUR: Cyrille Berger

TITRE: Perception de la géométrie de l'environment pour la navigation autonome
DIRECTEUR DE THÈSE: Simon Lacroix
LIEU ET DATE DE SOUTENANCE: LAAS/CNRS le 15 Décembre 2009

RÉSUMÉ EN FRANÇAIS:

Le but de de la recherche en robotique mobile est de donner aux robots la capacité d'accomplir des missions dans un environnement qui n'est pas parfaitement connu. Mission, qui consiste en l'exécution d'un certain nombre d'actions élémentaires (déplacement, manipulation d'objets...) et qui nécessite une localisation précise, ainsi que la construction d'un bon modèle géométrique de l'environnement, a partir de l'exploitation de ses propres capteurs, des capteurs externes, de l'information provenant d'autres robots et de modèle existant, par exemple d'un système d'information géographique. L'information commune est la géométrie de l'environnement.

La première partie du manuscrit couvre les différents méthodes d'extraction de l'information géométrique. La seconde partie présente la création d'un modèle géométrique en utilisant un graphe, ainsi qu'une méthode pour extraire de l'information du graphe et permettre au robot de se localiser dans l'environnement.

MOTS-CLÉS: robotique mobile, SLAM, modélisation d'environnement

DISCIPLINE ADMINISTRATIVE: Systèmes embarquées

INTITULÉ ET ADRESSE DE L'U.F.R OU DU LABORATOIRE:
LAAS-CNRS
7, avenue du Colonel Roche
31077 Toulouse Cedex 4