Thèse

préparée au

Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

en vue de l'obtention du

Doctorat de l'Ecole Nationale Supérieure de l'Aéronautique et de l'Espace

Spécialité : Informatique

par

Thomas Lemaire

Simultaneous Localisation And Mapping with Monocular Vision

Localisation et Cartographie Simultanées avec Vision Monoculaire

---

Soutenue le 20 décembre 2006 devant le jury composé de :

Juan Domingo Tardòs	Rapporteur
Roland Chapuis	Rapporteur
Raja Chatila	Examinateur
Guy Le Besnerais	Examinateur
Delphine Dufourd	Examinatrice
Francis Martinez	Examinateur
Manuel Samuelides	Examinateur
Simon Lacroix	Directeur de thèse

LAAS-CNRS
7, Avenue du Colonel Roche
31077 Toulouse Cedex 4

Contents

List of Notation

vector are denoted using lower case	x
matrix are denoted using uppercase	A
matrix and vector definition	$(.)$
matrix and vector block definition	$[.]$
matrix and vector size	$(.)_{(n\times m)}$ or $[.]_{(n\times m)}$
transposed vector and matrix	x^t, A^t
unit vector	$\underline{x}$
scalar (or dot) product between x and y	$x \cdot y$ or xy^t
cross product between x and y	$x \wedge y$
the quantity A in the frame F	$A_{/F}$
PDF of the variable x	$p(x)$

The matrix $[x]_\wedge$ is defined for the 3-vector $x_{(3)}$ such that $[x]_\wedge y = x \wedge y$.

$$[x]_\wedge = \begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix}$$

Chapter 1

Introduction

This chapter introduces the basics of the SLAM problem and motivates our work. The different processes which are involved in a SLAM algorithm are depicted. The main difficulties are underlined and the principal solutions found in the literature are presented. Our work is then outlined.

1.1 Context

A mobile robot is designed to carry out different tasks. Several kind of algorithms are typically required by such a system: localisation, trajectory control, wall following, face recognition, environment modelling... and many others depending on the actual mission. *Localisation* is one of the key blocks of such robotic systems, as many other functions depend on a reliable and accurate localisation of the robot.

Localisation. The localisation problem can be stated as follow: given an *a priori* map of the environment, given that the robot is endowed with a relevant sensor so as to comprehend real elements of the map, and given the input commands that drive the robots, a localisation algorithm computes the robot pose in the map frame. A GPS is also a very popular device to localise a platform, even if it does not give the orientation. The GPS receiver processes signals received from several satellites (the observations) and computes its distance to these satellites. Using an ephemeris to compute the absolute positions of the satellite (this is the *a priori* map), it can compute an estimate of its position on Earth.

In the real world, nothing is perfect: the *true* value of a variable v cannot be known. Mathematicians have developed the theory of *probabilities* in order to represent the uncertainty on the value of a variable. Rather than representing the knowledge on v with a single value, this knowledge is represented using a *Probability Density Function* (PDF) denoted $p(v)$. In the localisation problem the observations z, the input commands u, the map m, and the robot pose x are uncertain quantities. The problem, formulated using

probabilities, is to compute

$$p(x_t|x_{t-1}, u_t, z_t, m)$$

which reads "the PDF of the robot pose at time t, knowing its previous pose, the current command, the current observation, and the map". This problem is usually solved with recursive Bayes filters. For instance, $p(x)$ can be represented with a set of particles and updated with a particle filter. A good example of such methods can be found in [TFBD00].

Localisation And Mapping. In many applications a suitable map of the environment is not available. We are left with the command inputs $u_{0:t}$ and with the set of observations $z_{0:t}$, which are useless without a map. *Dead-reckoning*[1] techniques such as odometry compute a pose estimate using only $u_{0:t}$. They are integrative methods and accumulate errors over time: the growth of the pose estimate error is monotonic and unbounded.

But observations $z_{0:t}$ can be used to build a map. The kind of map which is built here is dependant on the type of sensors mounted on the robot. The first goal of the map is to enable robot self localisation and its representation can be very different from the map which could be produced by an architect. It is usually a set of simple elements of the environment, the *landmarks*, which positions are measured by the sensors. Once the environment has been visited and mapped by the robot, if the robot continues to move in the same area it benefits from the map it has just built. Indeed, the landmarks are *static* states and the uncertainty of their position never increases. Therefore while it is evolving in the same area, the robot pose estimate error is bounded.

There, localisation and mapping processes occur *simultaneously*. As a consequence the position estimate precision is highly dependant on the map errors, and similarly the map is sensitive to localisation errors. Both estimates are *correlated* and must be estimated jointly:

$$p(x_t, m|x_{t-1}, u_t, z_t)$$

In probabilistic terms, these correlations appears when the previous robot pose x_{t-1} is marginalised out.

This problem has been put in light in the seminal work by [MC89, SC87]. Simultaneous Localisation And Mapping, *SLAM* for short, was born...

1.2 Background: SLAM in a nutshell

1.2.1 Problem statement

A typical SLAM process, follows these steps to evolve from time $(t-1)$ to time t (figure 1.1):

(i) Prediction of the robot pose x_t from the previous pose x_{t-1} and using the input command u_t.

[1] The etymology of this expression is explained in Wikipedia http://en.wikipedia.org/wiki/Dead_reckoning

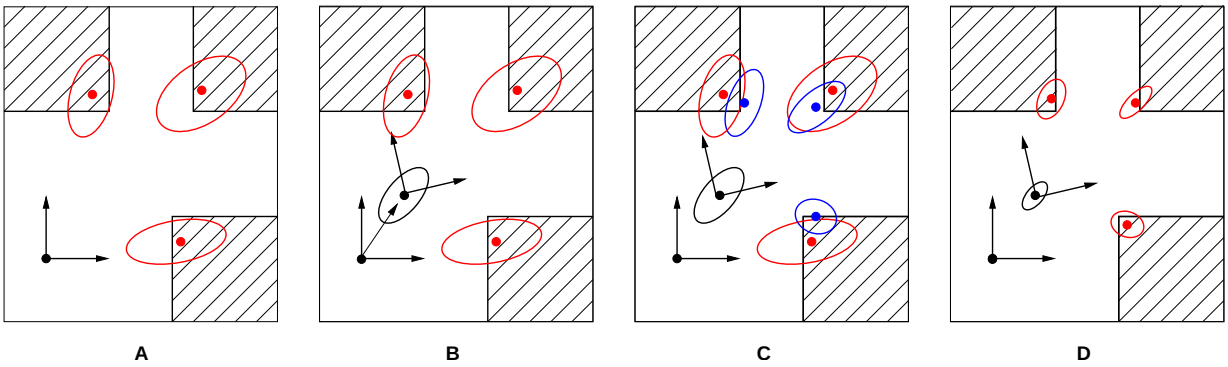

Figure 1.1: With the first observations, the robot builds a map of 3 points (A), then it moves and computes an estimate of its position (B), 3 new observations (blue) are matched with the current map (C), and are fused to update the map and robot pose (D).

(ii) Perception of the environment at time t, and data processing so as to extract relevant features observations z_t from raw sensor data.

(iii) Data association matches the features with landmarks in the map.

(iv) These observations are used to update the estimate of the robot pose and the map, new landmarks can be initialised.

(i) Prediction. In this step, a dynamic model of the vehicle and proprioceptive data are used to predict the robot position. In the classic case, simple 2D odometry is used, but for more complex systems such as Uninhabited Aerial Vehicles, an Inertial Measurement Unit can for instance provide the prediction inputs [KS03].

(ii) Perception. At the origin of the perception process, there is a sensor which produces raw data. These data can be metric distances, angular measurements, illumination intensities... Two different approaches exist to make use of these data:

- The raw data is used without any processing, and brute force algorithms such as ICP[2] are used.
- The raw data is processed in order to extract relevant features: this is *landmark based* SLAM. Note that a *feature* refers to an object in the observation space, and a *landmark* refers to its counterparts in the 3D map. A *good* feature has the following characteristics:
  - *Salient*, it is easily extracted from the raw data,
  - *Precise*, it can be accurately measured,

[2]Iterative Closest Point

– *Invariant*, it can be observed from different point of view,

Moreover the reconstructed landmarks can be *Meaningful*. They may represent something relevant for the robot or the mission (obstacle, target,...) so that the map can be used by higher-level algorithms such as reactive obstacle avoidance or path planning. But this is not the main goal of SLAM.

(iii) Data association. It is the process that actually finds in the set of the perceived features the ones which correspond to landmarks previously memorised in the map. The most direct approach consists in computing the *predicted* or *expected* feature observations using the current estimates of the robot pose and the landmark. Then, for each observation, the χ^2 test (see [BSL93]) is used to find the landmark which most likely produced this observation.

The data association process can be divided in two different tasks:

- *Tracking*: the feature is observed at time $(t-1)$ and also at time t, in this case the expected observation is close to the detected feature, and the view point has not changed too much: data association is easy.

- *Matching*: the feature has not been observed for a long time: the view point may have dramatically changed and the feature could even not be detected, also the predicted observation can be very far from the true feature.

In the general case, tracking and matching can be two different algorithms.

(iv) Estimation / Optimisation. Any SLAM algorithm requires an estimation (or optimisation) framework in order to fuse the ego-motion data and the observations data. Robotics needs an incremental algorithm which can be implemented for real-time operation on an embedded computer. The estimation process described above is very close to the Bayesian filtering paradigm. Indeed, the first fusion technique which have been applied is the Extended Kalman Filter (EKF) [Kal60, BSL93]. Nice convergence properties of the KF based SLAM which have been theoretically proved in [DNDW+01], and an easy implementation have made this solution very popular. The Kalman filter manipulates Gaussian PDFs which usually represent quite well the noise of the real data: $p(x_t, m)$ is represented with its two first moments, its mean and its variance.

1.2.2 Main difficulties

A SLAM algorithm which would be computationally efficient and consistent is one *Graal* of the robotic community: SLAM must run on-line on a robot moving in a large environment. This is one prerequisite for building a truly autonomous mobile robot.

Computational complexity. For any algorithm, one can categorise its *space* complexity and its *time* complexity with respect to the size of the problem. Here, the size of the problem is the number N of landmarks in the map. The well known EKF based SLAM is in $O(N^2)$ in space and time. With the computers currently available, the time complexity limits this approach to a few hundreds of landmarks.

Loop-closing and consistency. In the short-term, a benefit of SLAM is to reduce the drift in the pose estimate of any dead reckoning method, which accumulates errors. In the long-term, the most interesting feature of SLAM is certainly to *eliminate* this drift when the robot revisits a place already mapped: this is called a *loop-closing*. Loop-closing relies on two processes to be successful: data association and data fusion. Both processes are challenged by the loop-closing.

The data association algorithm must solve the *matching* problem, which is harder than the tracking problem. When the matching is successful, the resultant observations usually induce a large correction on the robot position and all the landmark estimates because of the correlations between these states. This has to be correctly handled by the fusion method.

If the loop is not closed correctly, the same walls mapped at the beginning and at the end of the loop are not aligned for instance, the map is then inconsistent. *Consistency* is also a concept of the stochastic estimation discipline: when the estimated value (mean and covariance in the case of a Kalman filter) is too far, in a probabilistic sense, from the true value, the estimate is said to be inconsistent. An inconsistent estimate can badly drive the matching process to do wrong data associations, and lead to dramatic divergence of the estimator.

The consistency of the EKF based SLAM is well understood [CNT04, BNG+06]: it is mainly caused by errors in the robot heading estimate which induce large linearization errors.

1.2.3 Main solutions

There have been several conferences offering numerous SLAM sessions which have produced a high number of articles. The SLAM community is very active, and already three SLAM Summer Schools have been organised in Stockholm, 2002 [SSSa], in Toulouse, 2004 [SSSb] and in Oxford, 2006 [SSSc]. Also, at the time of writing of this thesis, two SLAM tutorials were published in [DWB06, BDW06].

The purpose of this section is not to give an extensive description of all these works, but rather a short overview of the different solutions to the problems that have been identified in the previous section.

Estimation framework. The most popular approach is certainly the Kalman filter, several of its drawbacks have been mentioned just above. Other fusion systems have been proposed by the community.

The classic Kalman filter has been modified so as to reduce its complexity: the postponement technique [KDR01] consists in delaying the update of a part of the state vector until it is necessary, only the observed states need to be up-to-date. The exact complexity of the algorithm is not stated but is less than the usual Kalman filter.

The information filter is the dual of the Kalman filter, the information matrix being the inverse of the covariance matrix. From a theoretical point of view, both algorithms are equivalent. The Sparse Extended Information Filter (SEIF) presented in [TLK+04] takes advantage of the special structure of the SLAM problem to enforce sparsity on the information matrix, this approximation reduces the complexity down to $O(N)$.

A particle filter have been used to implement the *FastSLAM* algorithm [MTW03], its computational complexity is $O(M.log(N))$, where M is the number of particles. The Probability Distribution Function of the robot pose is represented with a set of particles, each particle encodes a full trajectory of the robot and carries a full map. The map is composed of a list of landmarks, each landmark being estimated with a single low dimension Kalman filter. The strength of this technique lies on the fact that a particle represents a robot pose which is *certain*, hence the landmarks are not correlated together.

Other works propose to use global optimisation tools. In [LM97], each pose of the robot trajectory is estimated. A graph representation of the problem is build: the poses are the vertex, an edge represents a constraint between two poses, these constraints are of two types. *Weak* constraints are the links between consecutive poses deduced from odometry readings. *Strong* constraints are the links between arbitrary poses obtained when processing sensor data. Then an energy function is defined: for each edge, this is the stochastic norm (or Mahalanobis distance) between the value of the constraint deduced from the current estimates of the two vertex D, and the value $\bar{D}$ and its covariance C given by the observation. Global optimisation methods can be applied to the problem of minimising the energy function of the graph:

$$\sum_i (\bar{D}_i - D_i)^t C_i^{-1} (\bar{D}_i - D_i)$$

In [ESL05], the same weak and strong constraints are used to compute an estimate of the full trajectory of the robot, the *delayed state*, using an Information Filter which is exactly sparse.

With these methods, the size of the problem to be solved continuously grows as the robot is moving, even if the robot stays in the same place. Also the processing of the raw observations so as to obtain the strong constraints is based on ICP with points acquired by a laser range finder, and is not easily extended to other sensors.

Map structure. Previous techniques rely on alternative estimation frameworks adapted for SLAM. Here, the Kalman filter is generally used and the focus is put on techniques which make use of different representations of the stochastic information, in different frames.

In [CMNT99] a SLAM framework based on the symmetries and perturbations map (SPMap) proposes a unified representation of any landmarks, and of observations from

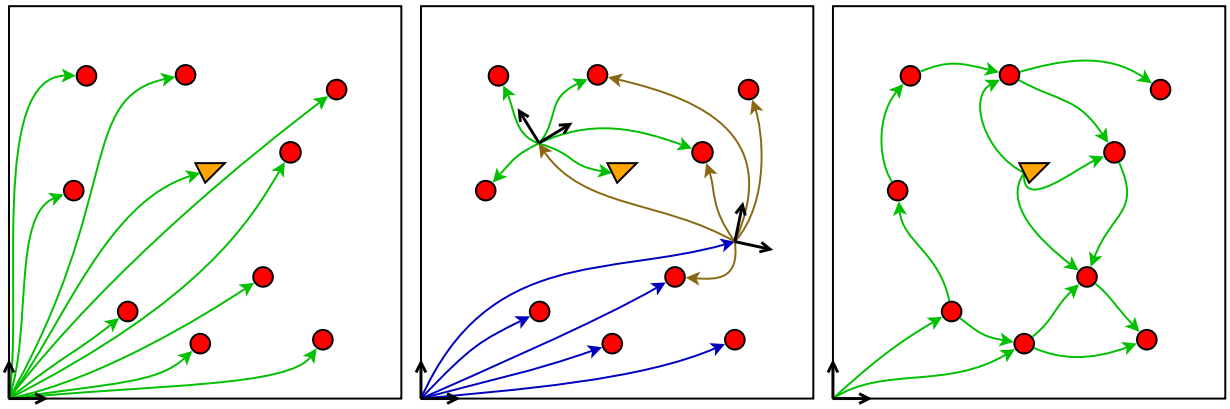

Figure 1.2: Stochastic vectors stored in different map formats are symbolised with arrows. Left: usual global map. Centre: 3 sub-maps (blue, green, brown). Right: relative map.

different sensors. This framework has been extended in [NLNT02] and very successful experiments are presented.

An original approach called D-SLAM [WHD05] (D stands for Decoupled) divides the SLAM problem into two concurrent yet separated estimation problems: the estimation of the map (static) and the low-dimensional estimation of the robot pose (dynamic). This algorithm requires a non trivial processing of the measurements for the update of the map. The map is represented in the information space, and the information matrix is exactly sparse thanks to the decoupling: the complexity of the update of the map is thus $O(1)$. As always when working in the information space, the estimate has to be recovered so as to compute the observation function Jacobian: this operation is $O(N)$. Also, the pose estimation step falls back on the Covariance Intersect [JU97] technique to fuse the robot pose estimate computed using the map and the current pose estimate – these estimates are correlated, but these correlations are unknown.

An interesting set of algorithms use different frames to represent the robot pose and the landmarks (see figure 1.2). As the previous algorithms they tackle the computational complexity problem, and the consistency problem is also addressed. 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 linearization errors. The main idea here is to adopt a representation where these errors can be bounded.

At the opposite of the monolithic absolute map approach, the relative map representation has been proposed in [New99]. 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 [New99] 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. The local maps are maintained using a 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 Constrained Local Sub-map Filter (CLSF) proposed in [SGH02] maintains a single global map but the robot builds a local map. Periodically the local map is fused into the global one, and a new local map is started.

In [NL03], a set of overlapping sub-maps is maintained, common landmarks in the maps are used to estimate relationships between map roots. In the ATLAS framework [BNL+03], a graph of the transformations which link the local maps frame forms a topological layer, the local maps being the metric layer. A Dijkstra algorithm is used to compute the coordinate of a local map in the world reference frame. The Hierarchical SLAM approach [ENT05] also adopts this two layer representation, and adds an optimisation step to the upper layer which takes into account loop-closing information.

Inshort, the computational complexity is reduced by approximately or exactly breaking correlations between the variables of the problem, and the consistency is achieved using more robust estimation techniques, possibly applying global optimisation methods.

Data associations. There are two approaches to solve this problem:

- make the estimation process robust to wrong matches,
- or improve the reliability of the matching process.

The work by [JNN03] falls in the first category: the natural multi-hypotheses ability of the particle filter is used here to maintain multiple data association hypotheses.

The second approach is more appropriate when using vision: an image contains a lot of information which can help to match some individual features. For example, *active search* [Dav05] relies on the predicted observation to define the zone where the feature is searched and on a *descriptor* of the feature to find it in this zone. It combines the strength of the estimation process and the robustness of the rich image data.

Also the matching process can be improved using a multi-sensors architecture as in [NCH06]. A laser range finder is combined with a camera: the LRF produces accurate metric data which are used to build a map, and the images are used to detect loop-closing and if detected, also to compute the transformation between the two images up to a scale factor, and the scale factor is found thanks to the associated metric laser data.

Figure 1.3: Perception of the same environment with a laser-range scanner (left) and a camera (centre). Right: depth image acquired with a Swissranger sensor.

1.3 SLAM with vision

For years in the 90's, most applications were indoor and based on a robot equipped with a laser range finder and evolving in a plane. The segments extracted from the laser data are the landmarks for SLAM, and, *by chance*, they correspond to walls *i.e.* the obstacles and the relevant elements of the map of a building. In this context, the map built by SLAM can also be used for obstacle avoidance for instance. This setup continues to deliver very good results for SLAM: the robot can be precisely localised in a building, while producing an accurate and meaningful map [NLNT02].

In this work we tackle the problem of 3D SLAM, also referred as 6 DoF[3] SLAM, in natural or semi-structured environments. There are not many sensors which offer perception of the 3D environment. In [JA04] a millimetre RADAR is used. This sensor is not too much affected by bad weather or varying lighting conditions, as a consequence it is more robust for outdoor operations than vision for instance. But it is difficult to embed a RADAR in a relatively small robot, or in a flying robot. Other sensors, not yet used in SLAM applications, may be of interest. For example the Swissranger[4] delivers intensity and depth images with a resolution of 176x144 (figure 1.3-right). This kind of time-of-flight range camera delivers distance data in a limited range, currently up to a few meters, and with a low resolution.

On the other hand, vision sensors are low-cost, small and power-saving and are therefore easy to embed on a robotic platform. Also they perceive data with a large field of view, at high frame rates and high resolutions. Moreover, the data produced are very rich and many algorithms from the computer vision community are ready to use and can be directly applied in robotic applications. Vision sensors bring together properties which are of main importance for SLAM.

One must differentiate *monocular* vision and *stereo* vision. A stereo head is composed of two cameras, with these two images 3D data of the surrounding environment can be recovered. But stereo-vision suffers two main problems:

[3] Degree of Freedom

[4] http://www.swissranger.ch

- Depending on the baseline of the stereo-bench, 3D information is obtained up to a given depth, this is an intrinsic limit of this sensor. For instance, it is very difficult to endow an aerial robot with a stereo-bench which baseline is large enough to get 3D information on the ground while flying at a reasonable altitude, to our knowledge, this has only been done in [JL03].

- A stereo-bench must be calibrated so as to produce a correct 3D information, and when it is used on a vehicle it is prone to lose its calibration.

For these reasons, monocular vision was chosen to develop our SLAM system. Of course monocular vision is not the ideal solution and other problems will have to be solved.

The type of features must also be chosen: salient points are very numerous in natural scenes and are the projection of simple geometric objects, 3D points. Also they are very popular in the vision community and many different algorithms exist to detect and march corner points between images. So this work was started with point features.

Moreover an optimisation framework is needed: the Kalman filter has many advantages, and its shortcomings can be overcome using a multiple maps approach, which is still based on the Kalman filter. So this work was started with the classic EKF-based SLAM approach.

This document is organised as follow:

1. Vision based SLAM raises the problem of landmark initialisation: a single feature observation is not sufficient to initialise it in the stochastic map since its depth is unknown. An initialisation method for the bearings-only EKF SLAM algorithm is developed in chapter 2.

2. Outdoor experiments are conducted and it becomes clear that a simple setup with a perspective camera does not performs well in this environment: an efficient architecture for outdoor 3D SLAM using a panoramic camera is proposed in chapter 3.

3. The semantic of size-less points is very poor, and they are hardly matched when a large viewpoint change occurs: the use of segment features is the next step. The initialisation method presented in chapter 2 is extended and a vision-based SLAM algorithm using 3D line-segments is proposed in chapter 4.

It can be noted that this work about *Vision* based SLAM does not directly address the problems of feature detection and matching for point features and neither for line segments. Rather state of the art algorithms from the computer vision community have been reused.

Chapter 2

Landmark initialisation in Bearings-Only SLAM

This chapter presents a landmark initialisation method for Bearings-Only SLAM within the Extended Kalman Filter framework. The method is based on a multiple Gaussian hypotheses selection paradigm. Then some extensions to the initial algorithm are proposed. Finally the influence of the parameters is discussed, based on results obtained in simulation.

2.1 Related work

Monocular vision based SLAM is a *partially observable* SLAM problem, in which the sensor does not give enough information to compute the full state of a landmark from a single observation, thus raising a landmark initialisation problem. An other instance of this problem, with sonar sensors, yield a *range-only* SLAM problem (a solution has been proposed in [LRNB02]): since a single observation is not enough to estimate all the parameters of a landmark, multiple observations are combined from multiple poses.

With vision sensors, we are in the *bearings-only* case. Initialisation algorithms can be divided into two groups:

- In the *delayed* algorithms, a feature observed at time t is added to the map at a subsequent time step $t + k$. This delay allows the angular baseline between observations of this landmark to grow, and the triangulation operation to become well conditioned.
- On the other hand, the *un-delayed* algorithms take advantage of the feature observations to localise the robot at time t. But the update of the stochastic map has to be computed carefully.

Several contributions propose different solutions for initial state estimation in bearings-only SLAM (see table 2.1).

– **TODO:** *naive approach single Gaussian [Sim05]* –

[JFKC06] maintains a buffer of a constant number of images and the filter is using the output of this buffer: here the filter itself is delayed. Feature points are tracked in this buffer and 3D point estimates are computed by triangulation. In [Bai03], an estimation is computed using observations from two robot poses, and is determined to be Gaussian using the Kullback distance. The complexity of the sampling method proposed to evaluate this distance is quite high. These two methods explicitly compute the intersection of the 3D lines defined by corner points observations: in the general case this computation is ill-conditioned and should be avoided.

In [DH00, SS03], a combination of a global optimisation (Bundle Adjustment) for feature initialisation and a Kalman filter is proposed. The BA is run on a limited number of camera poses and the associated feature observations. The complexity of the initialisation step is greater than a Kalman filter but theoretically gives more accurate results. Also these methods, as well as the one presented in [JFKC06], have a limitation on the baseline with which the features can be initialised. Depending on the camera motion and the landmark location, some feature cannot be initialised.

A multi-hypotheses method based on a particle filter to represent the initial depth of a feature is proposed in [Dav03, DCK04]. This work yielded to an effective real-time implementation which gives impressive results[1]. However its application in large environments is not straightforward, as it would require a huge number of particles.

A first *un-delayed* feature initialisation method was proposed in [KD04]. It is also a multi-hypotheses method: the initial state is approximated with a sum of Gaussians and is explicitly added to the state of the Kalman filter. The sum of Gaussians is not described and the convergence of the filter when updating a multi Gaussian feature is not proved. This algorithm has been extended in [KDH05] using a Gaussian Sum Filter, here the number of required filters can grow exponentially, this method is closed to what have been propose in [Pea95] for the bearings-only tracking. In [SDML05], a single Kalman filter is necessary: a rigorous method based on the Federate Kalman filtering technique is introduced. Also, the initial Probability Distribution Function (PDF) is defined using a geometric sum of Gaussians, an approximation which is also used in our work. The *un-delayed* techniques are especially well adapted when using a camera looking to the front of the robot: in this case, delayed methods require a long time before the features become observable, or may even fail to initialise some of them.

Very recently (in the year 2006) single hypothesis *un-delayed* methods have been published in [MCD06, ED06b]. These methods are very promising and are presented and discussed in more details in section 2.5.3.

Bearings-only SLAM using vision is also very similar to the well known Structure From Motion (SFM) problem. SFM methods do usually work only on images, on the contrary to SLAM methods which make use of additional sensors such as odometry. Also robotic applications require an incremental and computationally tractable solution whereas SFM algorithm can run in a time consuming batch process, Nevertheless recent work by [Nis03]

[1] The software is available at `http://www.doc.ic.ac.uk/~ajd` along with some videos.

	triangulation	multi-hypotheses	single hypothesis
delayed	[JFKC06, DH00, SS03]	[Dav03], our work	[ED06b]
un-delayed		[KDH05, SDML05]	[MCD06]
limitations	fixed size buffer	complexity	see section 2.5.3

Table 2.1: Overview of different initialisation methods for bearings-only SLAM.

shows real-time results on a small problem. Also work by [MLD+06] focuses on an incremental and real-time solution to the SFM problem. In SFM, the problem of initialisation is also of a great importance since it conditions the subsequent minimisation process. In [MLD+06], the initial 3D coordinates are obtained by triangulation on local part of the image sequence, which limits the opportunity of initialising distant points.

SFM relies on a global optimisation process: links between these non linear minimisation algorithms and the standard Extended Kalman Filter used in SLAM are studied in [Kon05].

2.2 Multi-hypotheses initialisation

2.2.1 Principle of the approach

The approach presented here is in the delayed multi-hypotheses category. This is a simple initialisation algorithm which does not involve any batch process, and the initialisation baseline is theoretically not limited. Figure 2.1 depicts it:

1. When a new feature is observed, a full Gaussian estimate of its state cannot be computed from the measure, since the bearing-only observation function cannot be inverted. The representation of this feature is initialised with a sum of Gaussians (section 2.2.3).

2. Then, a process updates this initial state representation, bad hypotheses are pruned until a single one remains (section 2.2.4).

3. Finally the landmark can be added to the stochastic map (section 2.2.5) which is managed by the usual EKF.

The main characteristics of our approach are the following:

- The initial probability density of a feature is approximated with a particular weighted sum of Gaussians.

- This initial state is expressed in the robot frame, and not in the global map frame, so that it is de-correlated from the stochastic map, until it is declared as a landmark and added to the map (the benefits of this are further explained).

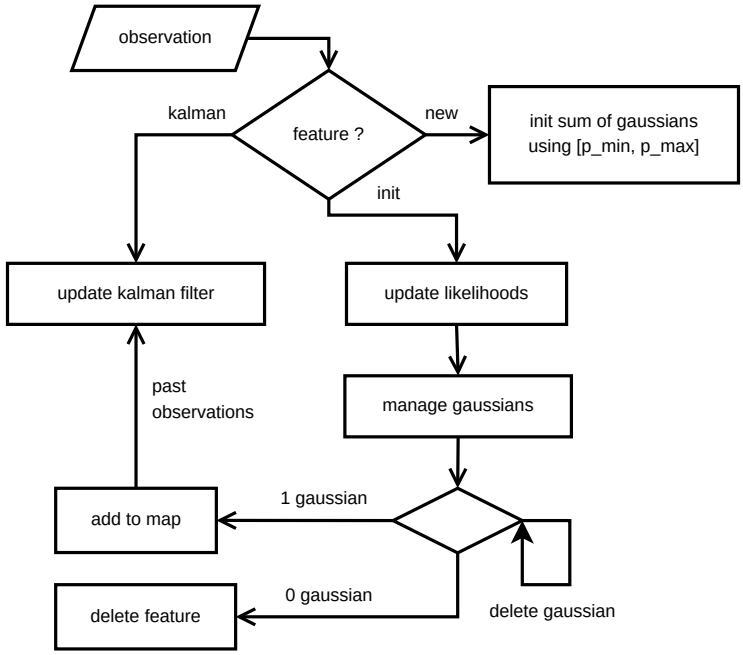

Figure 2.1: Our approach to the bearing-only SLAM problem.

- Many features can enter the initial estimation process at a low computational cost, and the delay can be used to select the best features.

In order to add the landmark to the map, and to compute its state in the map frame along with the correlations in a consistent way, the pose where the robot was when the feature was first seen has to be estimated in the filter. All observations of the feature are also stored along the corresponding robot poses estimates, so that all available information can be added to the filter when landmark initialisation occurs.

2.2.2 Structure of the Kalman filter

In this work, the Extended Kalman Filter (EKF) is used to fuse the robot ego-motion data and the observations.

The state of the EKF is composed of the landmarks estimates, the current robot pose, and as previously pointed out, some past poses of the robot. The state vector X and the covariance matrix P of the filter are the following:

$$X = \begin{pmatrix} X_r^0 \\ \vdots \\ X_r^k \\ X_l^1 \\ \vdots \\ X_l^n \end{pmatrix} \qquad P = \begin{pmatrix} P_{X_r^0} & \cdots & P_{X_r^0,X_r^k} & P_{X_r^0,X_l^1} & \cdots & P_{X_r^0,X_l^n} \\ \vdots & \ddots & & \vdots & & \vdots \\ P_{X_r^k,X_r^0} & & P_{X_r^k} & P_{X_r^k,X_l^1} & \cdots & P_{X_r^k,X_l^n} \\ P_{X_l^1,X_r^0} & \cdots & P_{X_l^1,X_r^k} & P_{X_l^1} & & P_{X_l^1,X_l^n} \\ \vdots & & \vdots & & \ddots & \vdots \\ P_{X_l^1,X_r^n} & \cdots & P_{X_l^n,X_r^k} & P_{X_l^n,X_l^1} & \cdots & P_{X_l^n} \end{pmatrix}$$

X_r^0 refers to the current robot pose and X_r^i to the i^{th} of the k old poses estimated in the filter. X_l^i refers to the i^{th} of the n landmarks in the map. In the associated covariance matrix P, P_{X_i} refers to covariances of sub-state X_i and P_{X_i,X_j} refers to cross covariance of sub-states X_i and X_j.

2.2.3 Feature initialisation

3D point landmarks represented by their Cartesian coordinates $X_l = (x, y, z)^t$ are now considered.

- the observation function is $z = \mathbf{h}(X_{l/R})$,
- the inverse observation function is $X_{b/R} = \mathbf{g}(z)$

$X_{l/R}$ is the state of the feature expressed in the robot frame, and $X_{b/R}$ is the bearing of the feature in the robot frame represented by a unit vector.

In our notation, the observation model $\mathbf{h}()$, as well as the inverse observation model $\mathbf{g}()$ do not include frame composition with the robot frame, instead these transformations are formalised in **to**() and **from**() functions: $\mathbf{to}(f, v)$ computes vector v in frame f, and $\mathbf{from}(f, v)$ computes vector v in frame f^{-1}. This eases the following developments, and is general with respect to the underlying representation of a 3D pose (using Euler angles, quaternions,...). This also makes the implementation more modular, and observation functions and their Jacobian matrix easier to write. Also, the same formalism can be used to take into account other frame transformations such as robot to sensor transformation. For the sake of clarity, the sensor frame is here supposed to be the same as the robot frame.

Given the first observation z of the feature with covariance P_z, the probability density of $X_{b/R}$ is already jointly Gaussian since the measure z is considered to be Gaussian.

The measure itself does not give any information about the *depth*, but we generally have an *a priori* knowledge. For indoor robots, the maximal depth can for instance be bounded to several meters. For outdoor robots the maximal range is theoretically infinity, but in general this *infinity* can be bounded. This gives us for the depth ρ an *a priori* uniform distribution in the range $[\rho_{min}, \rho_{max}]$.

This *a priori* PDF is approximated with a sum of n Gaussians Γ_i. On one hand, it is a convenient way to approximate a PDF, and on the other hand, once a single hypothesis is selected, it is straightforward to incorporate this Gaussian in the Kalman filter.

$$\begin{aligned} p(X_{b/R}, \rho) &\approx \Gamma(X_{b/R}, P_{X_{b/R}}).p(\rho) \\ &\approx \Gamma(X_{b/R}, P_{X_{b/R}}).\sum_{i=0}^{n-1} w_i \Gamma(\rho_i, \sigma_{\rho_i}) \end{aligned} \tag{2.1}$$

As shown in [Pea95], the stability of a Extended Kalman filter initialised with the Gaussian $\Gamma(\rho_i, \sigma_{\rho_i})$ is governed by the ratio $\alpha = \sigma_{\rho_i}/\rho_i$. If α is properly chosen the linearisation of a bearings-only observation function around ρ_i is valid for this Gaussian. This gives us the definition of σ_{ρ_i}.

The means ρ_i of the Gaussians are now defined. The distance between two consecutive means ρ_i and ρ_{i+1} is set to be proportional to $\sigma_{\rho_i} + \sigma_{\rho_{i+1}}$:

$$\rho_{i+1} - \rho_i = k_\sigma(\sigma_i + \sigma_{i+1})$$

It comes from the fact that each Gaussian fills up a depth interval which is proportional to the standard deviation of this Gaussian (see figure 2.2).

In fact these formula define a geometric progression for the means of the Gaussians, with the common ratio $\beta = \frac{\rho_{i+1}}{\rho_i}$:

$$\begin{aligned}
\rho_{i+1} - \rho_i &= k_\sigma(\sigma_{\rho_i} + \sigma_{\rho_{i+1}}) \\
\frac{\rho_{i+1}}{\rho_i} - 1 &= k_\sigma(\frac{\sigma_{\rho i}}{\rho_i} + \frac{\sigma_{\rho i+1}}{\rho_{i+1}}\frac{\rho_{i+1}}{\rho_i}) \\
\beta - 1 &= k_\sigma\alpha + k_\sigma\alpha\beta \\
\beta(1 - k_\sigma\alpha) &= 1 + k_\sigma\alpha \\
\beta &= \frac{1 + k_\sigma\alpha}{1 - k_\sigma\alpha}
\end{aligned}$$

The sum of Gaussians which approximates the PDF of the depth ρ is controlled by two parameters α and k_σ or equivalently by α and β:

$$\begin{aligned}
\rho_0 &= \frac{\rho_{min}}{(1 - k_\sigma\alpha)} \\
\rho_i = \beta^i.\rho_0 \qquad \sigma_{\rho_i} &= \alpha.\rho_i \qquad w_i = 1/n \\
\rho_{n-2} < \frac{\rho_{max}}{(1 + k_\sigma\alpha)} \qquad \rho_{n-1} &\geq \frac{\rho_{max}}{(1 + k_\sigma\alpha)}
\end{aligned}$$

Figure 2.2 shows a plot of individual Gaussian members and the resultant sum for $\alpha = 0.2$ and $k_\sigma = 1.0$. The numerical value of α and k_σ are discussed later in section 2.3.

Now, let's rewrite equation (2.1) under the form of a compact sum of Gaussians:

$$\begin{aligned}
p(X^r_{b/R}, \rho) &= \sum_i w_i \Gamma(X_{b/R}, P_{X_{b/R}}).\Gamma_i(\rho_i, \sigma_{\rho_i}) \\
&= \sum_i w_i \Gamma(X^i_{l/R}, P_{X^i_{l/R}})
\end{aligned}$$

where:

$$\begin{aligned}
X^i_{l/R} &= \rho_i X_{b/R} \\
P_{X^i_{l/R}} &= {\rho_i}^2 P_{X_{b/R}} + X_{b/R}\sigma_{\rho_i}{X_{b/R}}^T \\
P_{X_{b/R}} &= G P_z G^T \\
G &= \partial\mathbf{g}/\partial z|_z
\end{aligned}$$

Each $\Gamma(X^i_{l/R}, P_{X^i_{l/R}})$ represent a Gaussian hypothesis for the landmark state. Since it is kept in the robot frame, the distribution is uncorrelated with the current map. As a consequence the sum of Gaussians is not added to the state of the Kalman filter and this step of our algorithm is done at a low computational cost.

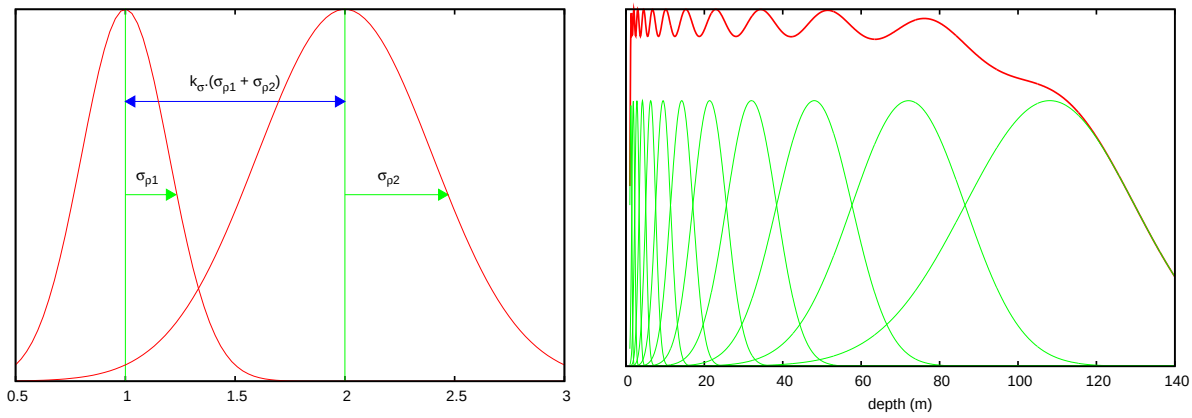

Figure 2.2: Left: definition of k_σ. Right: example of a Gaussian sum with $\alpha = 0.2$ and $k_\sigma = 1.0$.

2.2.4 Initial state update

The sequel of the initialisation step consists in choosing the Gaussian which best approximates the feature pose – the feature being thrown away if no consistent Gaussian is found. This process is illustrated in figure 2.3.

Hypothesis likelihood. Subsequent observations are used to compute the likelihood of each Gaussian Γ_i given observation z_t with covariance R_t at time t. The likelihood of Γ_i to be an estimation of the observed feature is:

$$L_i^t = \frac{1}{2\pi^{n_z/2}\sqrt{|S_i|}} \exp\left(-\frac{1}{2}(z_t - \hat{z}_i)^T S_i^{-1}(z_t - \hat{z}_i)\right)$$

where n_z is the dimension of the observation vector and S_i is the covariance of the innovation $z_t - \hat{z}_i$.

For each hypothesis, $\hat{z}_i$ and S_i have to be computed. Since the feature was first seen at time t_{ref}, the Gaussians are expressed in the past robot frame $X_r^{t_{ref}}$, they first need to be expressed into the map frame. For clarity, let $\mathcal{H}()$ be the full observation function, we have:

$$\begin{aligned}
\hat{z}_i &= \mathbf{h}(\mathbf{to}(\hat{X}_r^0, \mathbf{from}(\hat{X}_r^{t_{ref}}, \hat{X}_{l/R}^i))) \\
&= \mathcal{H}(\hat{X}_r^0, \hat{X}_r^{t_{ref}}, \hat{X}_{l/R}^i) \\
S_i &= H_1 P_{X_r^0} H_1^T + H_2 P_{X_r^{t_{ref}}} H_2^T \\
&\quad + H_1 P_{X_r^0, X_r^{t_{ref}}} H_2^T + H_2 P^T_{X_r^0, X_r^{t_{ref}}} H_1^T \\
&\quad + H_3 P_{X_{l/R}^i} H_3^T + R_t
\end{aligned}$$

where

$$H_1 = \partial\mathcal{H}/\partial X_r^0\big|_{\hat{X}_r^0, \hat{X}_r^{t_{ref}}, X_{l/R}^i} \qquad H_2 = \partial\mathcal{H}/\partial X_r^{t_f}\Big|_{\hat{X}_r^0, \hat{X}_r^{t_{ref}}, X_{l/R}^i} \qquad H_3 = \partial\mathcal{H}/\partial X_{l/R}^i\big|_{\hat{X}_r^0, \hat{X}_r^{t_{ref}}, X_{l/R}^i}$$

Figure 2.3: From an observed feature to a landmark in the map. From left to right: the sum of Gaussians is initialised in the robot frame; some Gaussians are pruned based on their likelihood after additional observations of the feature; when a single hypothesis remains, the feature is declared as a landmark and it is projected into the map frame; and finally past observations are used to update the landmark estimate.

A χ^2 test is performed on $(z_t - \hat{z}_i)^T S_i^{-1}(z_t - \hat{z}_i)$: bad hypotheses are thrown away as soon as possible .

Hypothesis selection. Then the bad hypotheses are selected and the associated Gaussian is pruned. Bad hypotheses are those whose likelihood Λ_i is low.

Two different tests can be used here:

- as in [Pea95, LLS05], bad hypotheses are pruned when their likelihood is below a given threshold,
- or, as in [KD04], the Sequential Probabilistic Ratio Test (SPRT) can be used.

The SPRT was chosen because this test is statistically well founded and avoids the choice of an arbitrary threshold. The SPRT is presented in [BSL93]: a likelihood ratio is computed between two hypotheses, the *null* hypothesis H_0 and the *alternate* hypothesis H_1. Such a test is computed for each Gaussian i:

$$SPRT_i = \prod_t \left[\frac{p(z_i|H_1)}{p(z_i|H_0)}\right]_t$$

- the *null* hypothesis being "this Gaussian is not an estimate of the landmark",
- the *alternate* hypothesis being "this Gaussian is an estimate of the landmark".

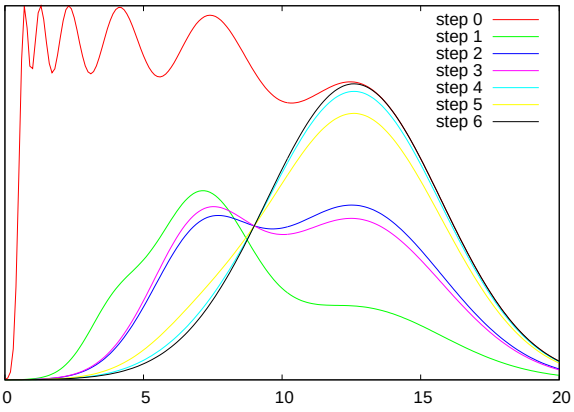

Figure 2.4: Evolution of the weighted sum of Gaussians through initialisation steps. Only the weights of the gaussians are modified, no correction is applied.

$p(z_i|H_1)$ is simply the likelihood L_i, and, as proposed in [KD04], $p(z_i|H_0)$ is set to the likelihood of the most likely Gaussian without considering the current one.

$$p(z_i|H_1) = L_i$$
$$p(z_i|H_0) = \max_{j,j\neq i} L_j$$

Then a bad hypothesis is pruned if its SPRT falls below a threshold defined by the *false alarm probability* P_{fa} and the *missed detection probability* P_{md}.

When only a single Gaussian remains, the feature is added to the map. An example of such a convergence is plotted step by step in figure 2.4.

2.2.5 Map augmentation

When a Gaussian $\Gamma(X^i_{l/R}, P_{X^i_{l/R}})$ is chosen, the corresponding feature j is declared as a landmark, and is added to the stochastic map:

$$X^+ = \begin{pmatrix} X^- \\ X^j_l \end{pmatrix} \quad P^+ = \begin{pmatrix} P^- & P_{X^-,X^j_l} \\ P_{X^j_l,X^-} & P_{X^j_l} \end{pmatrix}$$

$$\begin{aligned} \hat{X}^j_l &= \mathbf{from}(\hat{X}^{t_{ref}}_r, X^i_{l/R}) \\ P_{X^j_l} &= F_1 P_{X^{t_{ref}}_r} F_1^T + F_2 P_{X^i_{l/R}} F_2^T \\ P_{X^j_l,X^-} &= F_2 P^- \end{aligned}$$

where $F_1 = \partial\mathbf{from}/\partial f|_{\hat{X}^{t_{ref}}_r, X^i_{l/R}}$ and $F_2 = \partial\mathbf{from}/\partial v|_{\hat{X}^{t_{ref}}_r, X^i_{l/R}}$

Remember that for some steps since the feature was first seen, the feature observations were kept, and the corresponding poses of the robot have been estimated by the filter. Up

to now the observations were used only to compute the likelihood of the hypotheses, now this information is used to *update* the filter state: once a feature is added as a landmark, all available information regarding it is fused in the stochastic map. Section 2.4.1 explains that in practice some of this information can be discarded.

2.3 Parameters definition and evaluations in simulation

This algorithm requires several parameters to be properly set (table 2.2). The main problem is to isolate the influence of each individual parameter, this is a tricky problem which cannot completely be solved. Our approach is to use a simulation we have developed so as to study the influence of the parameters on the performance of the algorithm.

parameter	description	default value
α	ratio between mean and standard-deviation of each Gaussian	0.2
k_σ	defines the density of Gaussians	1.0
k	the maximum number of robot poses estimated in the filter	10

Table 2.2: This table presents the parameters evaluated in simulation.

2.3.1 Simulation environment

The simulation environment is a $80 \times 80 \times 2$ m parallelepiped. It contains 40 3D points randomly defined once and for all. The robot is following a circular trajectory with a radius of 10 m, centred in $(0, 0)$, in the plane $z = 0$. The robot moves with the constant speed of $\dot{s} = 0.2$ m/s and $\dot{\theta} = 0.2$ rad/s, the time-step of the simulation is 1 s.

The robot odometry gives classic $(ds, d\theta)$ measures, a Gaussian noise is added to these data with $(\sigma_{ds}/ds = 5\%, \sigma_{d\theta}/d\theta = 5\%)$. Also the robot is endowed with a bearings sensor which has a 360° field of view and a maximum range of 20 m. It provides (θ, ϕ) measures of the 3D points to which a Gaussian noise is added with $(\sigma_\theta = \sigma_\phi = 0.2°)$. With this setup, a total of 22 landmarks are visible along the trajectory. Figure 2.5 presents the robot trajectory and the environment.

Since the quality of the Gaussian noise is important to get sensible results, it is carefully generated using the *Boost Random Number Library*[2].

[2] http://www.boost.org/libs/random/index.html

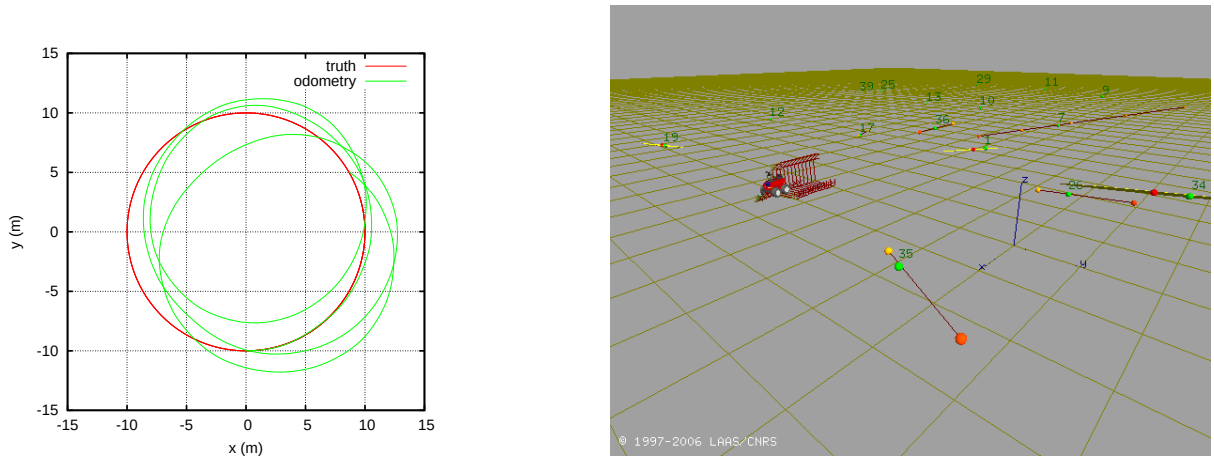

Figure 2.5: Left: true trajectory and trajectory obtained by integration of odometry. Right: 3D view of the simulated environment.

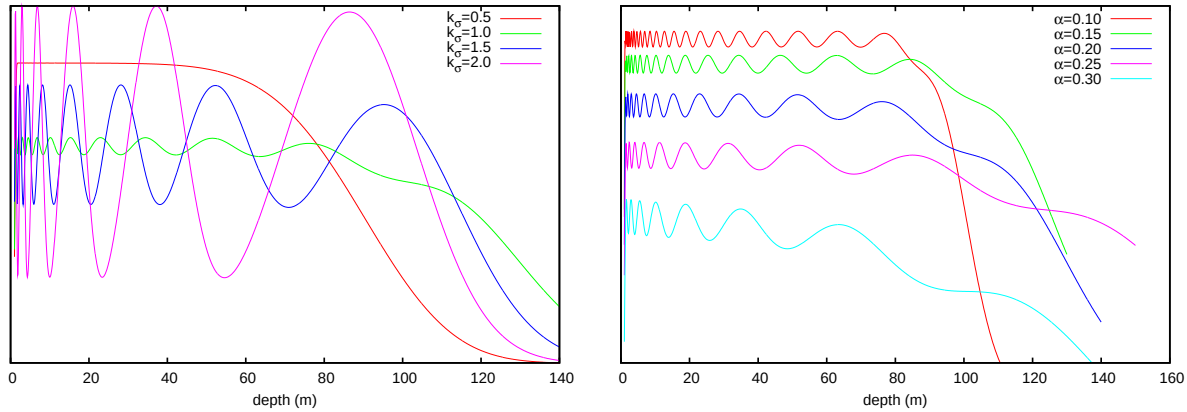

Figure 2.6: Left: $\alpha = 0.2$, PDFs for different values of k_σ. Right: $k_\sigma = 1.0$, PDFs for different values of α (for clarity, curves are shifted).

2.3.2 Initial PDF (α and k_σ)

The definition of the initial PDF is an important point since it greatly influences the consistency of the algorithm. The value of k_σ controls the distance between each Gaussian, and is set so as to reduce the "waves" of the PDF: it is a good approximation of the uniform distribution. Figure 2.6 also shows that α does not influence the wavy aspect of the distribution, the influence of the two parameters is not correlated. Even if $k_\sigma = 0.5$ could seem better at this point, $k_\sigma = 1.0$ is chosen for the following tests (see later in this section).

The ratio α must be chosen so that the linearisation errors of the observation function are small enough. In [Pea95], α is set to 0.2, and in [Dav03], α is set to 0.3. In both works, α is empirically chosen. Here, we have run several single run simulations with different

values of α, and the consistency of the robot pose is verified by computing its NEES[3]. Figure 2.7-*top* shows that $\alpha \geq 0.2$ gives inconsistent results.

Moreover, an intuitive result would have been to obtain a better robot localisation when α is smaller, since the initial feature state has less uncertainty. Figure 2.7-*bottom* shows that it is wrong. The reason is, when α is smaller, the initial state contains more smaller hypotheses, and the algorithm needs more information to select the good one. As a consequence, at the beginning of the simulation, the robot is moving with an empty map (process noise is added at each time-step), and landmarks are added later to the map, with, in the global reference frame, a larger uncertainty. But all in all, these differences are small and do not significantly modify the performance.

The parameter k_σ has a greater influence on the speed of landmark initialisation. When k_σ gets smaller, the overlap between the Gaussians increases, and the different hypotheses cannot be rapidly distinguished. Figure 2.8 shows the results of the simulations: for $k_\sigma = 0.3$, some landmarks are never initialised.

2.3.3 Influence of k

The maximum number of positions of the trajectory estimated in the filter (k) limits the number of past observations that can be used for landmark initialisation. Moreover, it limits the number of past poses which can be used to start new features initialisation. In order to bring to light the influence of k during simultaneous initialisation of multiple features, the environment described in section 2.3.1 is used, but the robot is moving along a longer circular trajectory with a radius of 20m.

Figure 2.9 shows that robot uncertainty is significantly reduced when k increases from 1 to 5, while increasing k further does improve the results significantly. One can see that the effective size of the trajectory is quite different for all k.

But, as shown on figure 2.9, most of the poses are used to store past observations and the difference between $k = 1$ and $k = 5$ comes from the higher number of features that can enter the initialisation process simultaneously with $k = 5$. As a conclusion, it is not very useful to increase k a lot, the information loss which occurs when some past observations are deleted is not crucial.

2.3.4 Comparison with fully observable SLAM

Pose uncertainty. We were naturally curious to know how the performance of the SLAM is affected when the range measure is not available. In simulation, it is possible to compare the result of boSLAM with range-and-bearings SLAM, fullSLAM for short. Using the simulation described in section 2.3.1, we modify the simulated sensor so that it returns also the range ρ of the points, a Gaussian noise with $\sigma_\rho/\rho^2 \in \{0.01, 0.02\}$ is added to this measure. Figure 2.10-left shows a plot of the robot pose uncertainty: as expected fullSLAM performs better than boSLAM but the difference is not so large.

[3]Normalised Estimation Error Squared, [BSL93]

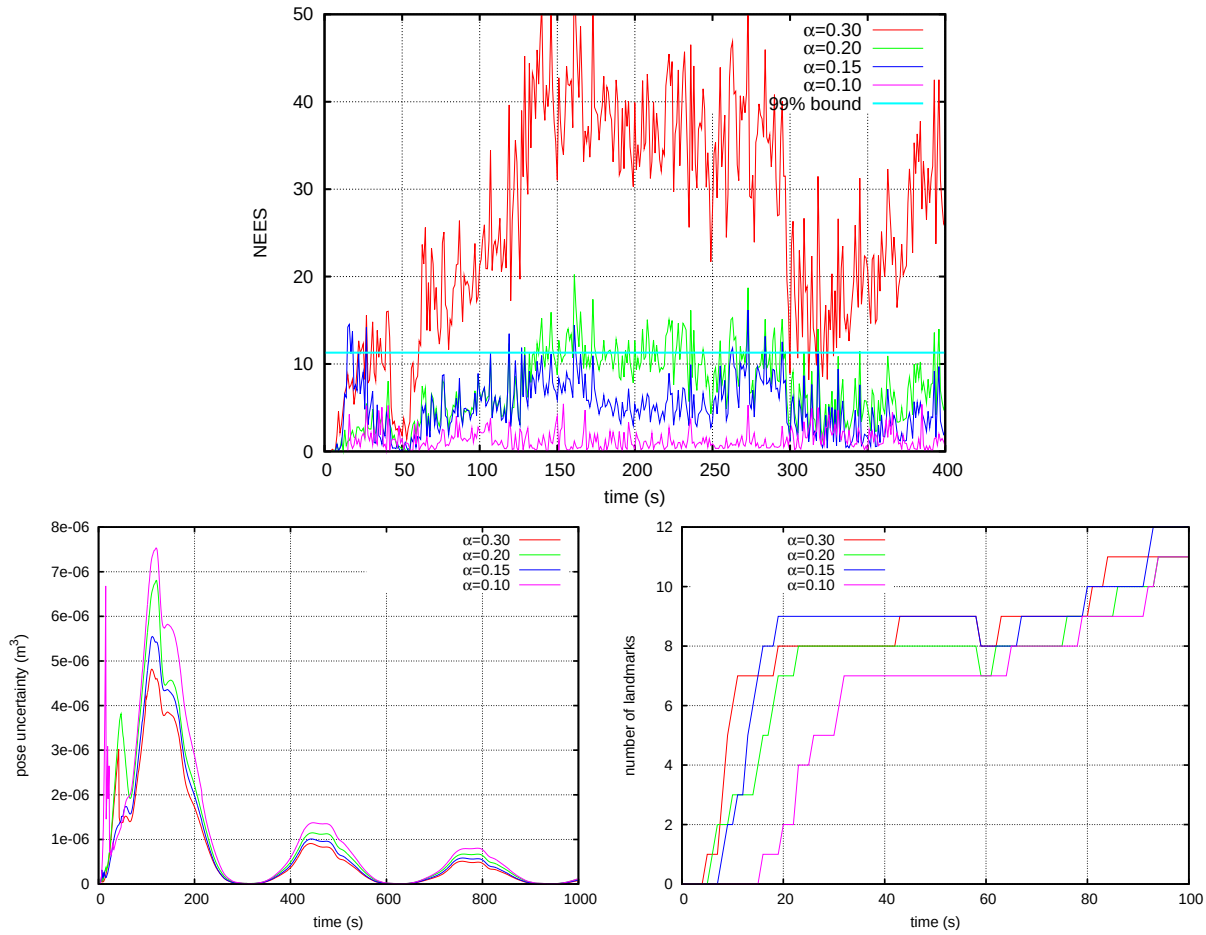

Figure 2.7: Top: evaluation of the robot pose NEES for several values of α, the 99% probability region bound is also plotted. Bottom: pose uncertainty and map size are plotted.

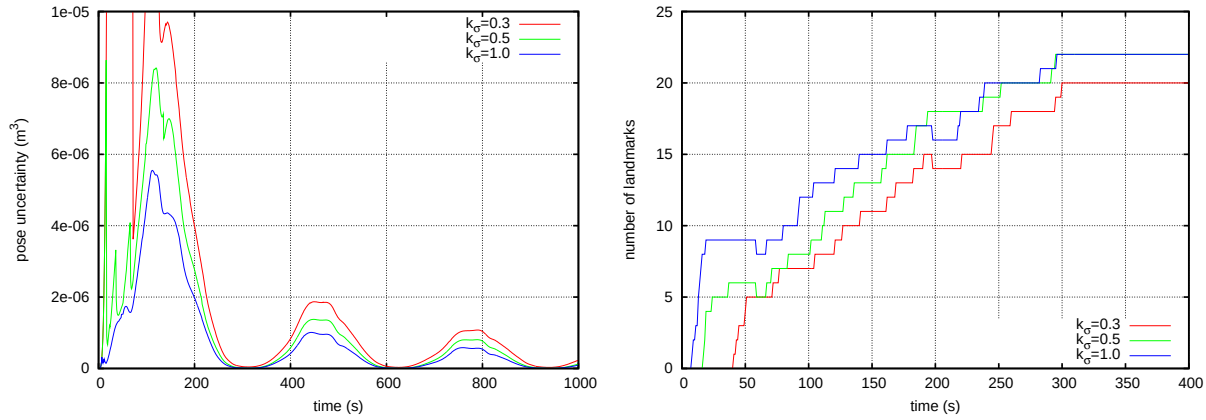

Figure 2.8: Robot pose uncertainty and map size are plotted for several values of k_σ.

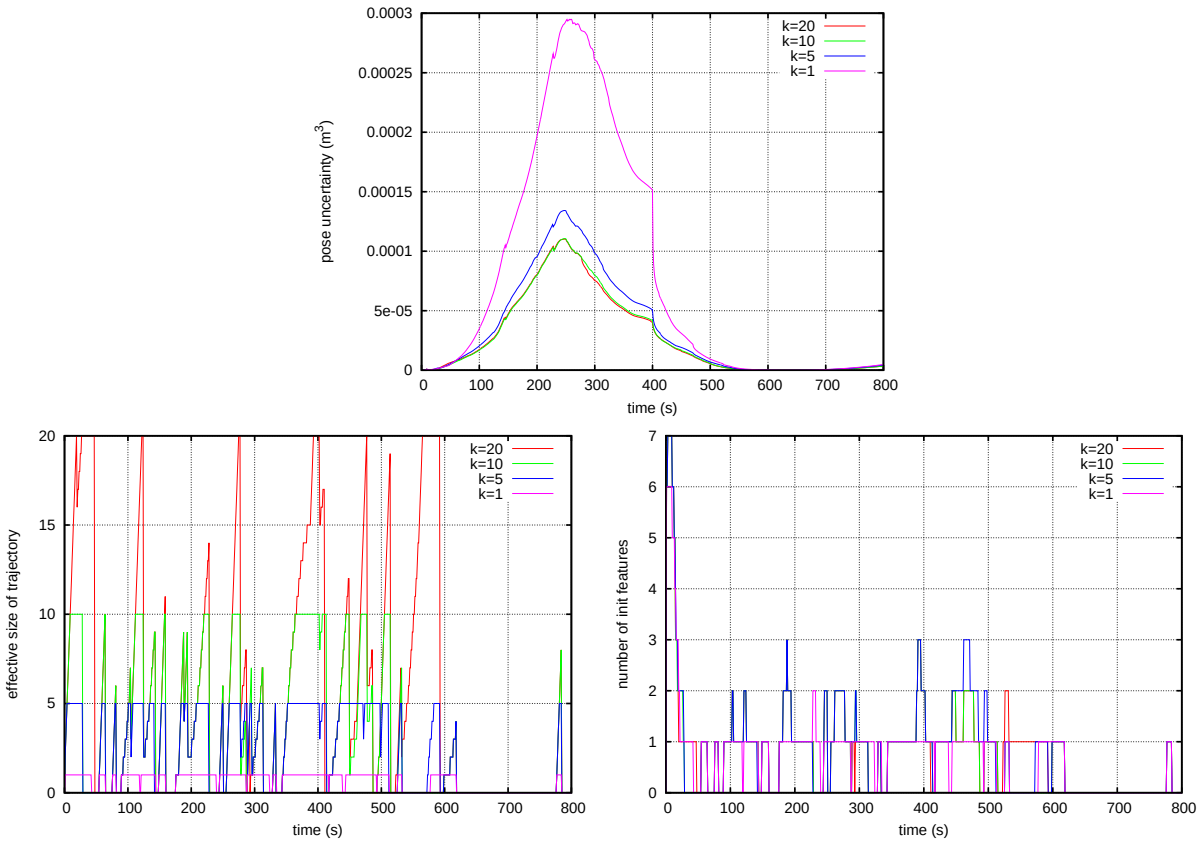

Figure 2.9: Influence of the value of k. Top: pose uncertainty. Bottom-left: number of utilised past robot poses. Bottom-right: number of features in the initialisation process.

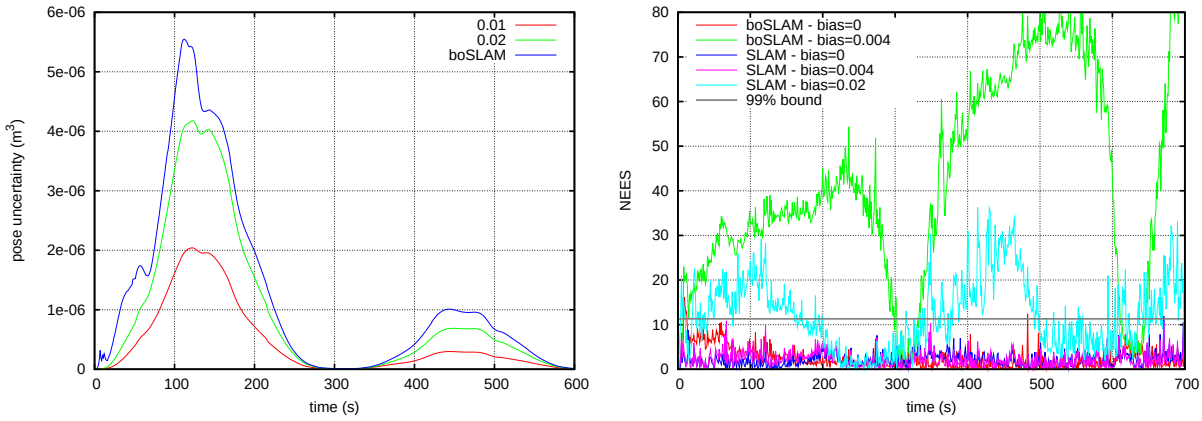

Figure 2.10: Left: Uncertainty on the robot pose for boSLAM and fullSLAM with different noise on the range measure. Right: consistency of boSLAM and fullSLAM with different odometry bias.

Influence of odometry bias. Odometry data is often corrupted by additive noise. Odometry is the only metric information which fixes the scale of the estimated map and robot poses in boSLAM. It is interesting to understand how much bias can be tolerated by boSLAM. The simulated odometry is now written:

$$\begin{pmatrix} v \\ \omega \end{pmatrix} + \begin{pmatrix} w_v \\ w_\omega \end{pmatrix} + \begin{pmatrix} b_v \\ 0 \end{pmatrix}$$

where w is the Gaussian noise described in section 2.3.1, and b is an additive bias. Bias is added only to the linear speed component, since this is the only metric component (the rotation speed does not give a metric information). Simulations with $b_v \in \{0, 0.004, 0.02\}$ m/s were run on both boSLAM and fullSLAM ($v = 0.2$ m/s). Figure 2.10-right shows that boSLAM is already inconsistent with a very small $b_v = 0.004$ while fullSLAM is still consistent. This really underlines the sensitivity of boSLAM to the input command data.

2.4 Algorithm refinements

2.4.1 Past poses management

In the ideal case, all the robot past poses where an observation of an initialisation feature has been made should be kept in the filter state. This would lead to store an unbounded number of poses in the map and consequently increase the computation time. In our implementation, the number of sub-state of the filter containing past robot pose estimates is limited to k, as a consequence the problem of choosing the best k poses has to be solved.

For that purpose, each pose is given the following relevant attributes:

- $frameIndex$: the index of the frame attached to that pose,

- $nbInitFeatures$: the number of features added for initialisation at that frame,
- $nbInitObs$: the number of observations corresponding to features in initialisation,
- $distToPrev$: the distance to the previous pose in the trajectory, which can be a metric or an index distance.

Before applying the prediction step to the current robot pose (X_r^0), a procedure is applied to choose whether or not X_r^0 is memorised. The goal is to keep in the filter state the k most interesting poses. Algorithm 2.1 describes this process, and algorithm 2.2 defines how to compare two robot poses and choose the most important one. Figure 2.11 gives an example of some poses estimated in the filter.

The backup copy of the stochastic pose X_r^0 must be done carefully:

$$X_r^w \leftarrow X_r^0 \qquad P_{X_r^w} \leftarrow P_{X_r^0} \qquad \forall j \, P_{X_r^w, X_f^j} \leftarrow P_{X_r^0, X_f^j}$$
$$\forall i, i \neq 0, i \neq 1, P_{X_r^w, X_r^i} \leftarrow P_{X_r^0, X_r^i} \qquad P_{X_r^w, X_r^0} \leftarrow P_{X_r^0}$$

Discussion on operator $\prec$. Past poses which corresponds to a feature initial pose are really important: if such a pose is deleted, all the associated features must be thrown away. This is the reason why such poses are preferred. Also, the priority is given to older poses: if a feature has been observed during many frames, it means it is very stable and is a very good feature for SLAM. Finally the poses which distance to the previous one is larger are preferred in order to spread the estimated poses.

Updating features. When a past robot pose is thrown away, the features which were initialised at that frame are lost. Also, the observations of features made at that frame are now useless since it will not be possible to update the map at the time of feature initialisation. These observations are also thrown away.

Updating robot poses history. When a feature is initialised in the map, or when it is lost by the tracking system, the attributes of the robot poses are updated accordingly.

2.4.2 Hypotheses degeneration

The problem. During the initialisation process of a feature, the observations are used to update the weights of the different Gaussian hypotheses. Depending on the camera trajectory with respect to the feature position, a new observation does not necessarily brings more information on the depth of this feature. In this case, the likelihoods of the hypotheses are updated according to the observation noise, and their value may become completely erroneous[4]. This can lead to the selection of a wrong hypothesis (for example when a feature is detected near the image center and the robot moves along the camera optical axis). Such an inconsistent initialisation is illustrated figure 2.12.

These wrong initialisations can lead to filter divergence and must be eliminated.

[4] A theoretical demonstration can be found in [Sol07]

Algorithm 2.1 Current pose management

if $X_r^0.nbInitFeatures > 0$ or $X_r^0.nbInitObs > 0$ **then**
  if there is a free sub-state in the filter **then**
    backup X_r^0 in this sub-vector
  else
    $X_r^w \leftarrow \min_i^{\prec} \{X_r^i\}$ {find the worse pose}
    if $w = 0$ **then**
      do nothing
    else
      backup X_r^0 at X_r^w location in the filter state
    end if
  end if
else
  do nothing
end if

Algorithm 2.2 operator $\prec (X_r^i, X_r^j)$

if $X_r^i.nbInitFeatures = 0$ and $X_r^j.nbInitFeatures = 0$ **then**
  if $X_r^i.nbInitObs = X_r^j.nbInitObs$ **then**
    return $X_r^i.distToPrev < X_r^j.distToPrev$
  else
    return $X_r^i.nbInitObs < X_r^j.nbInitObs$
  end if
else if $X_r^i.nbInitFeatures > 0$ and $X_r^j.nbInitFeatures > 0$ **then**
  return $X_r^i.frameIndex > X_r^j.frameIndex$
else if $X_r^i.initFeatures = 0$ **then**
  return true
else
  return false
end if

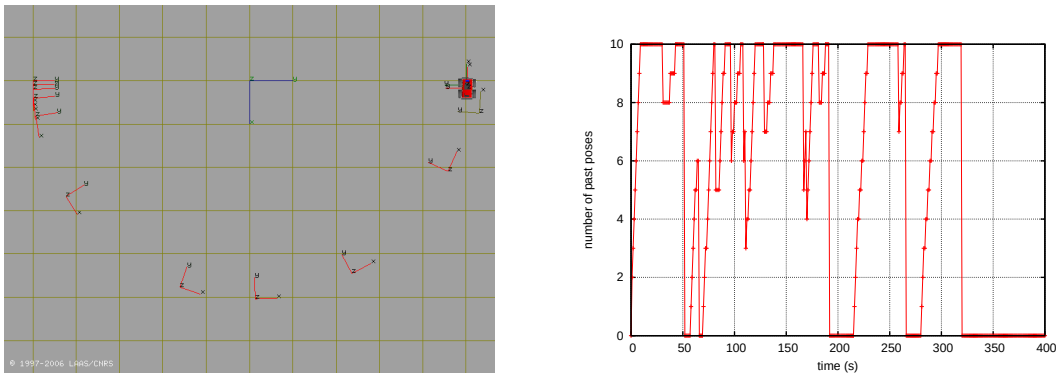

Figure 2.11: Left: the red frames show the ten past poses estimated in the filter. Right: the number of past poses estimated in the filter during a simulation run.

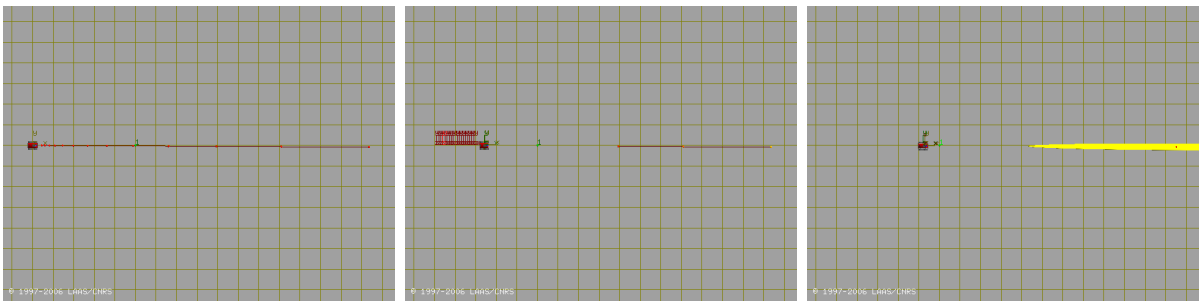

Figure 2.12: From left to right: feature initial hypotheses, hypotheses pruning, and landmark initialisation with a wrong hypothesis.

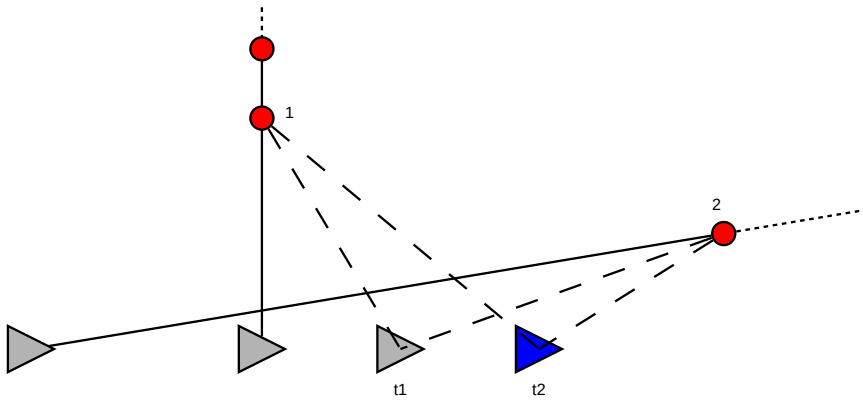

Figure 2.13: Predicted observation of the closest hypothesis for two different features.

Proposed solution. The solution consists in filtering the observations which will be used to update the feature depth hypotheses. Only observations which convey a new information on the depth of the landmark are to be considered.

Suppose that feature i is observed and its initial state is updated at time t_1. Feature i is observed again a time t_2, the decision of whether updating or not its initial state has to be taken. The information gained thanks to this new observation depends on *(i)* the translation of the robot between t_1 and t_2, *(ii)* the actual depth of the landmark, and *(iii)* the observation model and the noise covariance R. The observation of the closest remaining hypothesis X_0^i is computed for both t_1 and t_2:

$$z_1 = \mathcal{H}_1(X_0^i) \qquad z_2 = \mathcal{H}_2(X_0^i) \qquad d = (z_2 - z_1)R^{-1}(z_2 - z_1)^t$$

Where $\mathcal{H}_2$ only takes into account the translation between t_1 and t_2, not the rotation. This is illustrated figure 2.13.

If the Mahalanobis distance d is above a predetermined threshold, it means that from the robot pose at t_2 new information on the depth of feature i can be obtained with the considered sensor.

This method has the advantage of being generic with respect to the feature observation z: it can be a point but also a segment as presented in section 4.7 on page 80. Also the method properly takes into account the observation noise.

2.4.3 Landmarks at infinity

Motivations. Landmarks located at infinity are very interesting for SLAM: even if no information on the robot translation can be inferred, they provide a reliable measure of the orientation of the robot. Large errors on the orientation of the robot are precisely the main cause of linearisation errors of the observation model. In [MD06], a purely rotating camera is localised using a SLAM approach based on points at infinity. In the un-delayed single hypothesis bearings-only SLAM algorithm proposed in [MCD06] the unique Gaussian

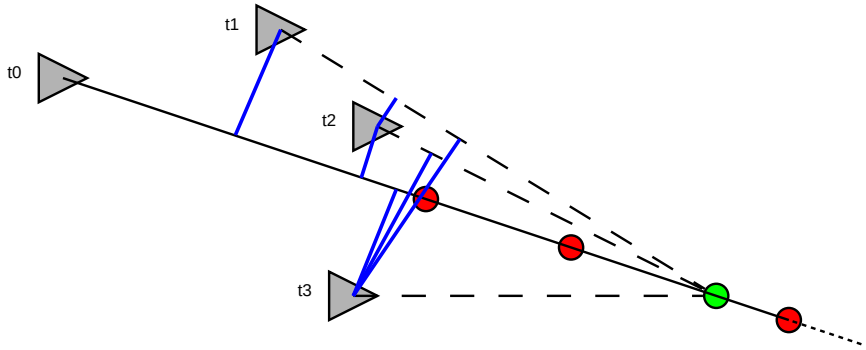

Figure 2.14: The feature is discovered at t_0, its initial state (in red) and the true landmark (in green) are represented. The baselines (in blue) are incrementally computed for the successive observations. Here the maximal baseline is achieved between t_1 and t_3.

theoretically represents landmarks possibly at infinity. Whereas in triangulation methods, the recovered depth is intrinsically limited.

With multi hypotheses algorithms, a special *infinity hypothesis* or a special decision process must be introduced to take into consideration landmarks at infinity.

Proposed approach. The concept of *infinity* is a relative concept: for a man travelling on Earth, the stars can be considered as being at infinity. It means that they define a fixed direction. For a robot moving in a given area, the infinity is not so far. How far is the infinity depends on two elements:

- The area where the robot can move, or in other words the maximum baseline with which the feature can be observed.
- And also the precision of the sensor which is used to measure the angular direction of the feature: the more accurate is the sensor, the farther is the infinity.

The size of the area where the robot moves can be defined according to its task, or more naturally according to the size of the local maps when a sub-maps based SLAM approach is implemented. The precision of the angular measures is already encoded in the error model of the sensor. With our algorithm, when a feature has been observed with a large enough baseline without being initialised, it means that the depth information for this feature cannot be retrieved. Then it is naturally added into the map as a feature at infinity which direction is represented by a 3D unit vector. Figure 2.14 illustrates the incremental computation of the maximal baseline with which a landmark has been observed.

Of course, it means that the robot has to travel at least this baseline before a remote feature could be initialised. In order to improve the process, the threshold on the baseline is set smaller than the maximal dimension of the area, and to balance this approximation, the observation noise of the landmarks at infinity is increased.

This process requires a bound on the maximal depth that can be recovered by the robot, this defines the ρ_{max} of the Gaussian Sum. It also needs a threshold on the baseline, along with an increased observation noise, the latter is set empirically.

2.5 Discussion

2.5.1 Complete list of parameters

parameter	description	default value
α	ratio between mean and standard-deviation of each Gaussian	0.2
k_σ	defines the density of Gaussians	1.0
k	the maximum number of robot poses estimated in the filter	10
$doUpdateTh$	threshold to decide whether the initial state is updated or not	10
$eraseHypothesisTh$	threshold to decide whether a hypothesis is deleted or not	20
P_{fa} P_{md}	SPRT false alarm probability SPRT missed detection probability	5% 5%
ρ_{max}	maximal depth of the initial hypotheses	in metres
$baselineTh$	threshold to decide whether a feature is at infinity	in metres

Table 2.3: This table presents all the parameters used in the boSLAM algorithm with their typical value.

Table 2.3 presents a complete list of the parameters of boSLAM. Some of them were not discussed because they correspond to statistically well founded threshold. $doUpdateTh$ and $eraseHypothesisTh$ are set according to the desired χ^2 test confidence. The same goes for P_{fa} and P_{md} which are used in the Sequential Probabilistic Ratio Test. ρ_{max} and $baselineTh$ are set according to the size of the area where the robot is moving.

2.5.2 About landmark parametrisation

In our algorithm, the landmark (or tentative landmark) states are manipulated in two different stages: *(i)* during the initialisation process and *(ii)* in the stochastic map. There is no specific problem for these two representations to use a different parametrisation. When a hypothesis is selected, its representation just has to be converted to the map parametrisation.

We have chosen the Cartesian representation of the 3D points for the initialisation procedure. This representation has the advantage of being the one which is used in the global

map. Other representations could have been used (see table 2.4). The *polar* representation is the natural representation for bearings sensors: the bearing measurements being directly mapped to $[\theta, \phi]$. The *modified polar* or *inverse depth* parametrisation is, for the same reason, very natural. It has also the advantage of reducing the non-linearity in the observation function. This representation has been already used in the past in algorithms to solve the bearings-only tracking (BOT) problem, for example in [Pea95, AH83]. Provided that the EKF linearisation constraints are met, all the representations must perform equally. Since the *polar* and the *modified polar* are meaningful only in the sensor reference frame, one must add the origin $[x_{ref}, y_{ref}, z_{ref}]$ of this frame to the representation.

Cartesian	$[x, y, z]^t$
polar	$[x_{ref}, y_{ref}, z_{ref}, \rho, \theta, \phi]^t$
modified polar	$[x_{ref}, y_{ref}, z_{ref}, 1/\rho, \theta, \phi]^t$

Table 2.4: Different representations for a 3D point.

With our algorithm, no update can occur during the initialisation steps, so there is no consistency problem here, and thus we do not see any advantage of using a different parametrisation for stage *(i)* and *(ii)*.

Nevertheless, the use of the *modified polar* parametrisation for both steps would have the following benefit: the linearisation of the observation function around the mean of each hypothesis would be valid in a larger interval, so the value of α could be increased. And, while keeping the algorithm consistent, the number of hypotheses could be reduced. The main drawback of this parametrisation is that it needs a 6-dimension vector. Since our algorithm does not suffer from a large number of initial hypotheses, we prefer to keep the Cartesian representation.

2.5.3 Bearings-Only SLAM using Inverse Depth parametrisation

Recent work. Very recently, the *inverse depth* parametrisation has been used to solve the bearings-only SLAM problem with un-delayed algorithms [MCD06, ED06b]. The estimation process benefits from the quasi-linearity of the observation function under the condition that the motion of the camera along the depth axis is small relatively to the depth of the point [ED06b]. Using this property, [MCD06, ED06b] can use a standard Kalman filter to estimate a landmark represented with the inverse depth parametrisation.

In [MCD06] the standard EKF based SLAM is applied, each landmark being represented by $[x_{ref}, y_{ref}, z_{ref}, 1/\rho, \theta, \phi]^t$. A new landmark is immediately initialised in the stochastic map with $1/\rho = 0.5$ and $\sigma_{1/\rho} = 0.5$.

In [ED06b] a FastSLAM2.0 based approach is proposed. Here, the landmark is not immediately added to the map when it is detected for the first time, it enters a "partially initialised" state, and is stored using the inverse-depth parametrisation. While in this state, new observations of this feature are used to update *(i)* its depth estimate, and *(ii)* the current estimate of the camera pose using the epipolar constraint. Once the uncertainty

over the depth is small enough, an unscented transformation is applied to transform the landmark to the Cartesian representation and to add it in the filter. This method is not a pure un-delayed method, but has the advantage that the observations are directly used to update the camera pose estimate.

Comparing with our algorithm. Since our *boSLAM* algorithm is based on the Kalman filter, it was easy to implement a version of [MCD06], here referred as *idSLAM*. Comparative simulations with the setup described in section 2.3.1 have been run. The results are shown in figure 2.15.

The uncertainty on the robot pose is nearly the same for both approaches, with a noticeable exception at the beginning of the run. Because of the delay, boSLAM does not benefit from the map to improve the robot pose estimate, until some landmarks are initialised. Then the previous observations are fused and the robot pose estimate updated accordingly. One can also verify that the landmarks are initialised faster with idSLAM. The consistency on the robot pose is also very similar: in our simulation setup the observation function linearity hypothesis of idSLAM is verified, even with a low frame rate of one every second which corresponds to an observation every 0.2 meter.

In order to evaluate the limit of idSLAM when the frame rate decreases, simulations with different frame rate were run. Figure 2.15-bottom-left shows that with observations every 0.4 meter, idSLAM is already inconsistent, whereas boSLAM is consistent. Of course it also depends on the real depth of the landmarks: if the landmarks are closer to the camera, the bound on the frame rate will be lower. We were quite surprised to observe that idSLAM works also with low frame rates, it makes this algorithm a good candidate for a bearings-only SLAM implementation for robotics.

Comparative simulations can be viewed on the following video: `http://www.laas.fr/~tlemaire/download/boIdSlam.mp4`.

Small issue with idSLAM. Nevertheless we encountered an issue with the idSLAM algorithm. With our simulation setup it often happens that the inverse depth becomes negative after a Kalman update. In this case, the next predicted observation for this landmark is 180° off the observation and is completely inconsistent. It causes the divergence of the Kalman filter. This difficulty is not mentioned in the original paper [MCD06]. In our implementation, the Kalman filter is protected against such wrong updates by a χ^2 test: the observation is ignored. Therefore good results could be obtained.

The origin of the problem is to update the depth of the landmark with an observation which does not carry any information on this dimension. Then the observation noise predominates in the update of the depth: the landmark can get behind the camera. Such observations are numerous when the camera points to the direction of the trajectory, which is unavoidable with a panoramic sensor for instance.

We had to cope with the same problem, see section 2.4.2. This solution could be adapted here, but the rejected observations would be definitely lost. A sanity check can also be applied on the inverse depth and set it to an arbitrary positive value when it gets

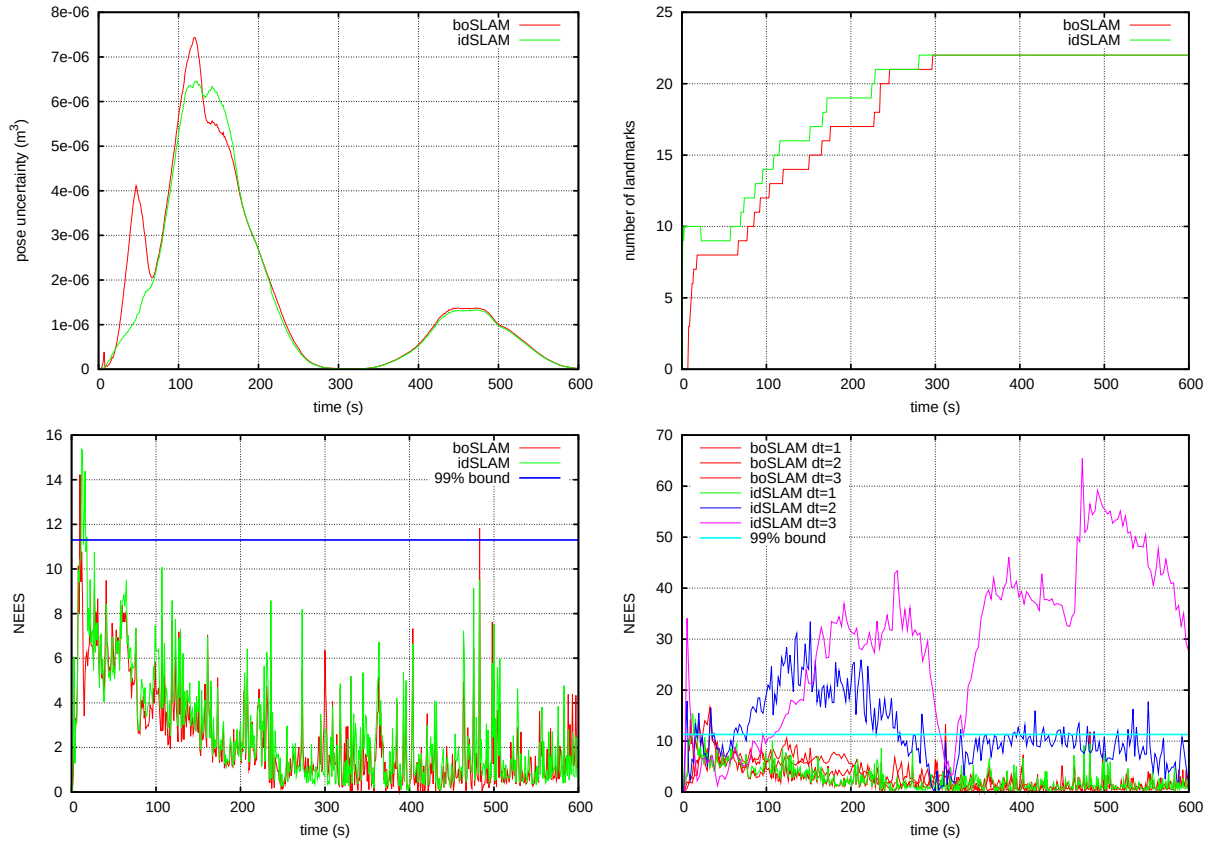

Figure 2.15: Comparing of our Bearings-Only SLAM algorithm (boSLAM) and the Inverse-Depth SLAM algorithm (idSLAM). Top-left: uncertainty of the robot pose estimate. Top-right: size of the map. Bottom-left: consistency of the robot pose estimate. Bottom-right: consistency of idSLAM for different frame rate.

negative.

The same issue should also appear with the algorithm presented in [ED06b]. In this case the two relevant components of the observation are well separated: the epipolar update can always be applied and the update on the depth of the feature could be filtered without loss of information.

2.6 Conclusion

In this chapter, a method for landmark initialisation in Bearings-Only SLAM has been proposed. This algorithm has a low complexity with respect to the initial state update. In practice this is of a great advantage: many features can enter the initialisation, and if during the delay a feature is found to be of poor quality or is lost by the perception algorithm, it is simply erased and has not consumed computational resources.

Also our algorithm has a low complexity with respect to the number of hypotheses

at initialisation. More complex landmark states, with more unobservable parameters can easily be considered: chapter 4 presents an extension of this method for line segments.

Only a few parameters are used by this method, simulations were run so as to understand their influence, and also to evaluate the performance of the method. In the next chapter, experiments with real images data are analysed.

A video illustrating the boSLAM algorithm and presenting some results on real images can be downloaded at http://www.laas.fr/~tlemaire/publications/lemaireBoSLAMVideo.mp4

Chapter 3

Results with perspective and panoramic cameras

In this chapter, the boSLAM algorithm is run on real-data acquired with our robots: the obtained results are presented. These real world experiments bring up some problems which were not addressed in chapter 2 but are nevertheless of main importance. This chapter describes how these issues were solved, and under the light of the results obtained with a perspective camera, an original setup using a panoramic camera is proposed.

3.1 Introduction

Experiments on real data are of essential importance in robotics. While simulations are useful to check if the implementation is correct and to analyse some properties of the algorithms, experiments on real data do usually raise unexpected problems and really demonstrate the feasibility of the approach.

The SLAM process described in chapter 2 requires several kinds of input data for the experiments:

- Perception produces feature observations:
  - features are detected in the images,
  - some features are tracked in consecutive images,
  - the observation model is defined: $z = h(X_l) + w$.
- The robot pose is predicted:
  - ego-motion data are available,

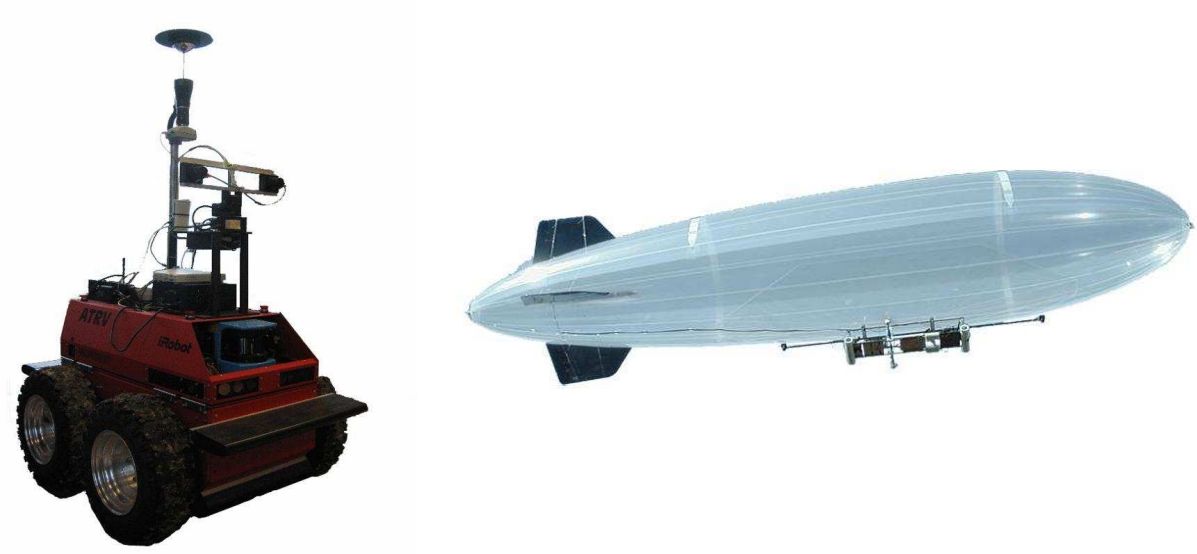

Figure 3.1: The ATRV rover Dala and the $10m$ long blimp Karma.

  - the dynamic model of the robot is defined: $X_r^{t+1} = f(X_r^t, u_t) + w$.

- Loop-closing mechanism:

  - possible loop-closing are detected,

  - current detected features are matched with landmarks in the map.

In particular, the loop-closing mechanism is very important and is the main goal of SLAM.

Section 3.2 proposes such a setup for a rover equipped with a single perspective camera and odometry, the results which were obtained are presented and discussed section 3.3. Then a new setup for a rover endowed with a stereo bench and a panoramic camera is proposed section 3.4 and results are given section 3.5. Finally some preliminary results obtained with a blimp are shown in section 3.6.

These experiments have been performed with the robots Dala and Karma (figure 3.1). Dala is an iRobot ATRV model, equipped with a 0.35 metre wide stereo-vision bench mounted on a pan-tilt unit (the images are down-sampled to a resolution of 512×384), a panoramic camera and a 6-axis inertial measurement unit. Karma is an airship equipped with a 2.4 meter wide stereo-vision bench, a differential GPS that provides position and speed information, a magnetic compass and a 2-axis inclinometre.

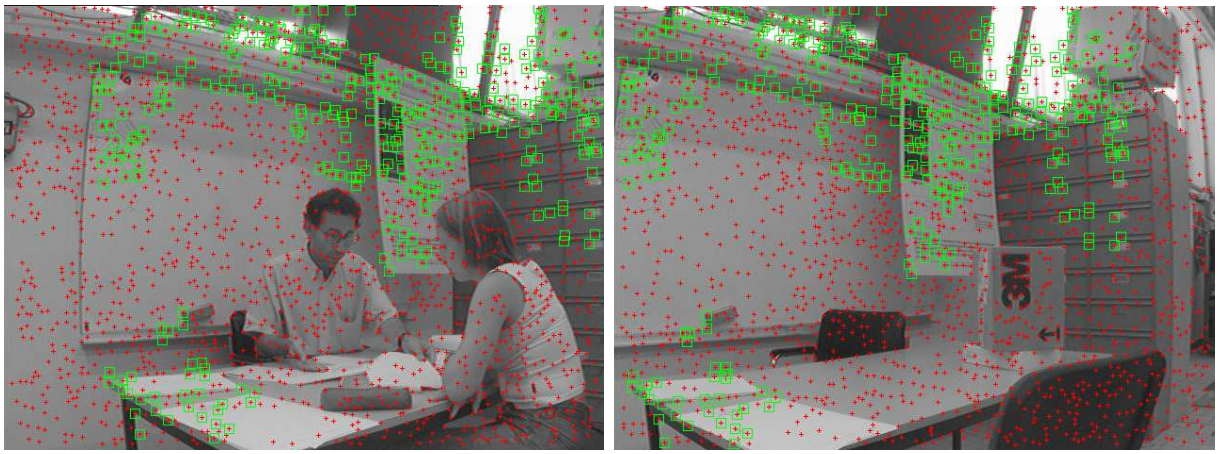

Figure 3.2: Results of the matching algorithm. The red crosses shows the interest points, the green squares indicate successful matches. There was 194 matches for a total computanional time of 201ms (including 116ms for the Harris points detection and 85ms for the matching) from [LBJL06].

3.2 SLAM for a rover equipped with a perspective camera

3.2.1 Camera model

The classic pinhole camera model is used here. A 3D point $(x, y, z)^t$ is projected in the image plane according to the following equation:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \end{pmatrix} \frac{1}{\sqrt{x^2+y^2+z^2}} \begin{pmatrix} x \\ y \end{pmatrix}$$

The cameras of Dala are equipped with short focal lens and therefore the raw images present distortion. A second order radial distortion model is used to compute images without distortion.

The parameters $u_0, v_0, \alpha_u, \alpha_v$ as well as the two distortion parameters are obtained after a calibration process. A tool developed in the laboratory was used to do the calibration work, as well as a Matlab calibration toolbox written by Jean-Yves Bouguet[1].

3.2.2 Perception: corner point features

Point features must first be detected in the images, and then *tracked* in consecutive images. Also, during the loop-closing, features have to be *matched*. Feature *tracking* is easier than feature *matching*: the viewpoint has not changed too much, and also in SLAM a good predicted observation is available to guide the search.

[1] http://www.vision.caltech.edu/bouguetj/calib_doc/

In all our experiments, the algorithm developed at LAAS in [JL01] and recently modified in [LBJL06] is used. It is based on the modified Harris detector to extract features. Then the features are matched based on a combination of the signal information around the point and of the geometric constraints between detected points.

Given two images, the algorithm returns the numerous detected corners and the matches between the two images (see figure 3.2). For SLAM, this algorithm is used for the *tracking* with input images acquired at time t and $t+1$, and also for the *matching* with arbitrary images.

3.2.3 Ego-motion estimation

3D odometry. With a robot equipped with an inertial measurement unit, an estimate of the 3D elementary motions $\mathbf{u}(k+1)$ can be provided by integrating the odometry data on the plane defined by the pitch and roll angles of the robot. An actual error model on odometry is difficult to establish: since the rover Dala experiences important slippage and is equipped with a cheap inertial measurement unit, we defined the following conservative error model:

- The standard deviation on the translation parameters $\Delta t_x, \Delta t_y, \Delta t_z$ is set to 8% of the travelled distance,
- The standard deviation on $\Delta\Phi$ (yaw) is set to 1.0 degree per travelled meter, and to 1.0 degree for each measure on $\Delta\Theta, \Delta\Psi$ (pitch and roll).

3.2.4 Features selection and map management

One of the advantages of using interest points as features is that they are very numerous. However, keeping many landmarks in the map is costly, the filter update stage having a quadratic complexity with the size of the state vector. It is therefore desirable to actively select among all the interest points the ones that will be kept as landmarks.

Feature selection. A good landmark should easily be observable (matched), and landmarks should be regularly dispatched in the environment. The strategy to select the landmarks is the following: the image is regularly sampled in cells (figure 3.3). If there is at least one mapped landmark in a cell, no new landmark is selected; if not, the point that has the highest Harris low eigenvalue λ_2 is selected as a landmark. This ensures a quite good regularity in the observation space (the image plane).

Map management. In a first stage, all the landmarks are integrated in the map. The corresponding observations are helpful to estimate the pose of the robot in the short-term. Once a feature is not matched anymore (either because it is out of the field of view or is not tracked), the associated landmark is only valuable for a future loop closing: in the meantime it only consumes computational resources. A successful loop closing does not

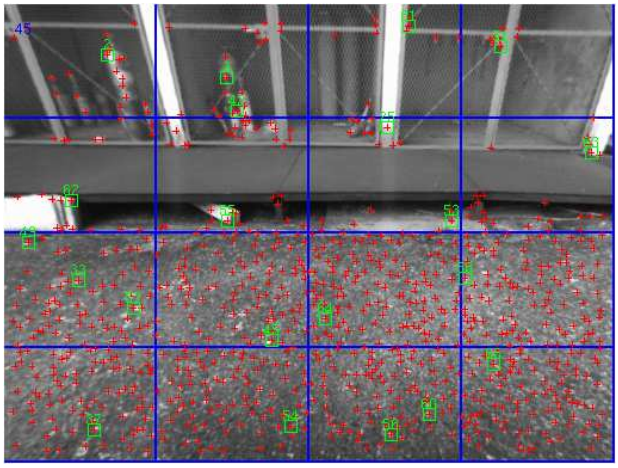

Figure 3.3: Selection of the points that will be kept as landmarks (green squares)

require many landmarks, only a few good ones are necessary: our strategy is to keep the landmark density under a given threshold (one landmark per 0.8^3 m^3 in the experiments), the least observed landmarks being simply thrown away.

3.2.5 Loop closing.

Since no appearance model of the landmarks is memorised, an image database is built to achieve loop-closings: single images are stored every time the robot travels a given distance, along with the corresponding estimated robot pose and feature IDs. This database is periodically searched for a possible loop-closing on the basis of the current estimate of the robot position. For that purpose, a loose test between the current robot pose and the stored images positions is sufficient: one of the strengths of the matching algorithm is that it is able to match points with no knowledge of the relative transformation between the images. When a loop-closing detection occurs, the corresponding image found in the database and the current image are fed to the matching algorithm, and successful matches are used to update the stochastic map.

3.3 Results with a perspective camera

This section presents results obtained using data acquired with Dala. For the bearings-only SLAM, only the left images of the stereo pairs are used.

3.3.1 On a small trajectory

Data acquired along a simple 3-loops circular trajectory with a diameter of 6 meters provide insights on the behaviour of the algorithms. Two sets of data are compared here: one with the camera heading forward, and one with the camera heading sidewards.

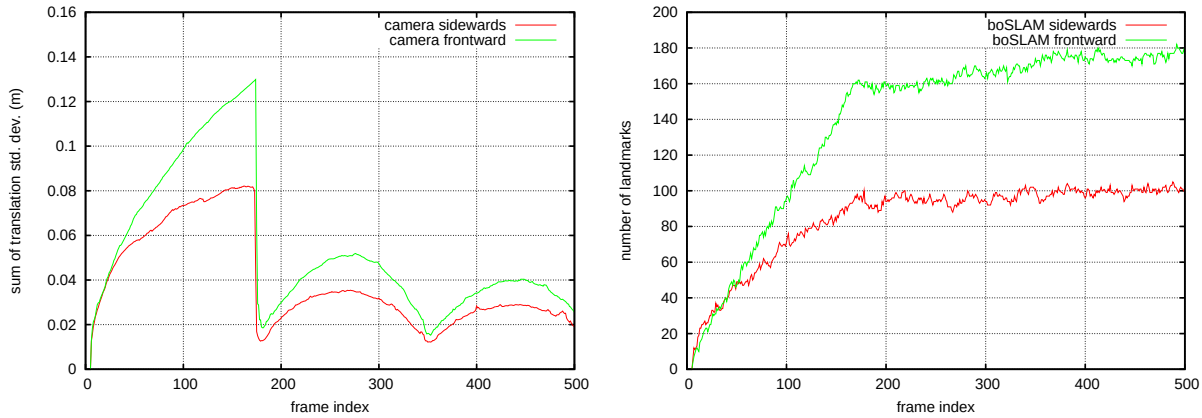

Figure 3.4: Comparison of the estimated uncertainty on the robot pose (left) and on the map size (right) with the camera looking forward and sidewards.

Figure 3.4 illustrate the evolution of the estimated robot pose standard deviations and number of mapped landmarks. They exhibit the usual loop-closing effects of SLAM: the uncertainty on the robot pose dramatically drops after the first lap, and the number of mapped landmarks stops growing once the first loop is closed. Figure 3.5-left shows the correct behaviour of our simple loop-closing procedure: the loop-closing is correctly detected, and landmarks from the map are matched against the current image.

The plots also exhibit a better performance with the sidewards setup than with the forward setup. Indeed, when the cameras are looking sidewards, the features are tracked on more frames: the mapped landmarks are more often observed, and fewer landmarks are selected. And in the bearings-only case, the sidewards setup yields a greater baseline between consecutive images, the landmarks are therefore initialised faster.

Finally, figure 3.5-right illustrates the good performance of the interest point matching algorithm: more than 700 points are tracked in consecutive images. For the loop-closing, the algorithm also performs very well, but this is an easy case since images are acquired from very close positions.

3.3.2 On a longer trajectory

Results are now presented on data acquired on a 100 meters long trajectory during which two loop closures occur, the cameras are oriented sidewards. The trajectory of the robot is presented on figure 3.6-left. The results of boSLAM on this dataset are included in the following video: `http://www.laas.fr/~tlemaire/publications/lemaireBoSLAMVideo.mp4`

Qualitative results. Figure 3.8 presents an overview of the map obtained with bearings-only SLAM at different points of the trajectory. The landmarks uncertainties obviously drop after the loop-closing process has successfully matched already mapped features:

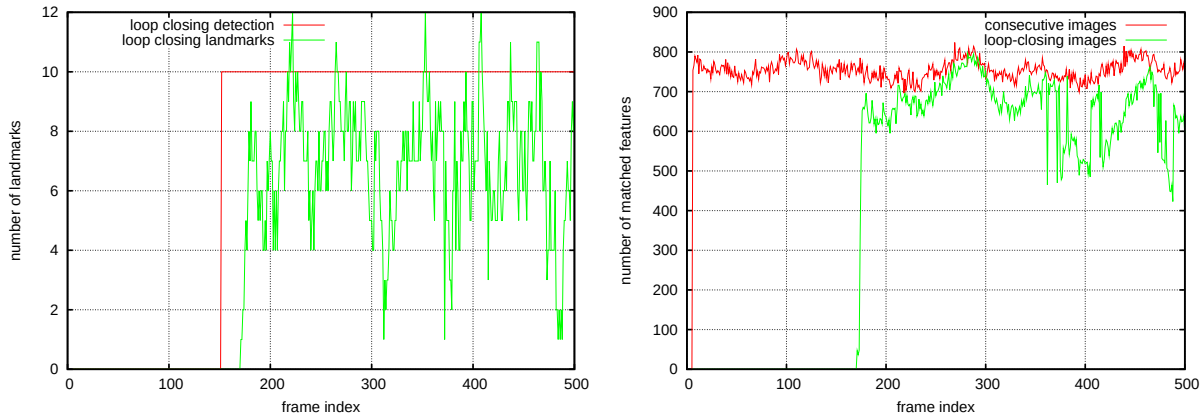

Figure 3.5: Left: loop-closing detection and number of landmarks in the map detected in the database image. Right: performance of the interest point matching algorithm on consecutive images and on loop-closing pairs.

figure 3.6-right shows when a loop-closing event is detected and the number of landmarks successfully matched. Figure 3.7 shows in images the features used by boSLAM, after the second loop has been closed. Blue features are the ones that have been mapped during the beginning of the trajectory and that are "recovered" by the loop-closing process: note that the predicted observations are consistent with the real observations, the detected features lie in the ellipses that represent their 3σ prediction.

Loop-closing errors. The ground truth of the pose of the natural landmarks which are introduced in our map is difficult to obtain. No ground truth of the robot pose is available either. Instead, a precise measure of the robot pose can be obtained when the robot closes a loop, applying a visual motion estimate algorithm on stereo images. The technique is presented later in section 3.4.2. Table 3.1 gives the SLAM errors on the robot pose estimate. The translation error of $\sqrt{0.25^2 + 0.14^2 + 0.03^2} = 0.29$ meter corresponds to an error of less than 0.5% of the travelled distance for this 100 meters trajectory.

Nevertheless, one can notice that these errors are not consistent with the covariance estimated by the filter, some errors exceed the 3σ bound. As shown in section 2.3.4, bearings-only SLAM is very sensitive to the prediction input: this is the only metric data in the system which therefore sets the scale of the robot trajectory and of the estimated map. On the ATRV Dala, the non-suspensed non-orientable wheels are often slipping, which yields poor odometry motion estimates (figure 3.6). A Gaussian error model for this odometry is not well-adapted: this is certainly the main source of inconsistency here.

3.4 Vision-based SLAM using a panoramic camera

Thanks to their 360° field of view, panoramic cameras have various advantages over perspective cameras for the SLAM problem. Features can be tracked over long distances and

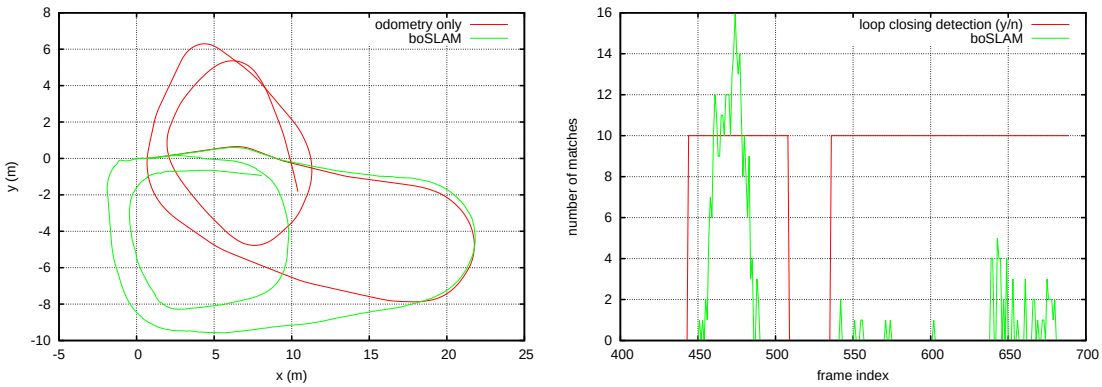

Figure 3.6: Left: trajectories obtained by odometry integration and estimated with boSLAM. Right: loop-closing detection and number of landmarks in the map detected in the database image.

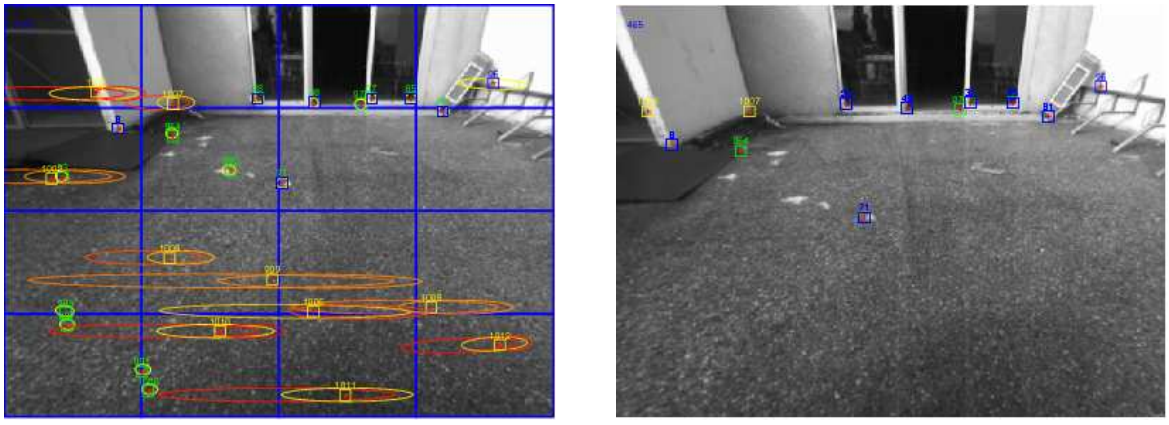

Figure 3.7: Loop closing: matched features between the current image (left) and the image from database (right) are indicated in blue.

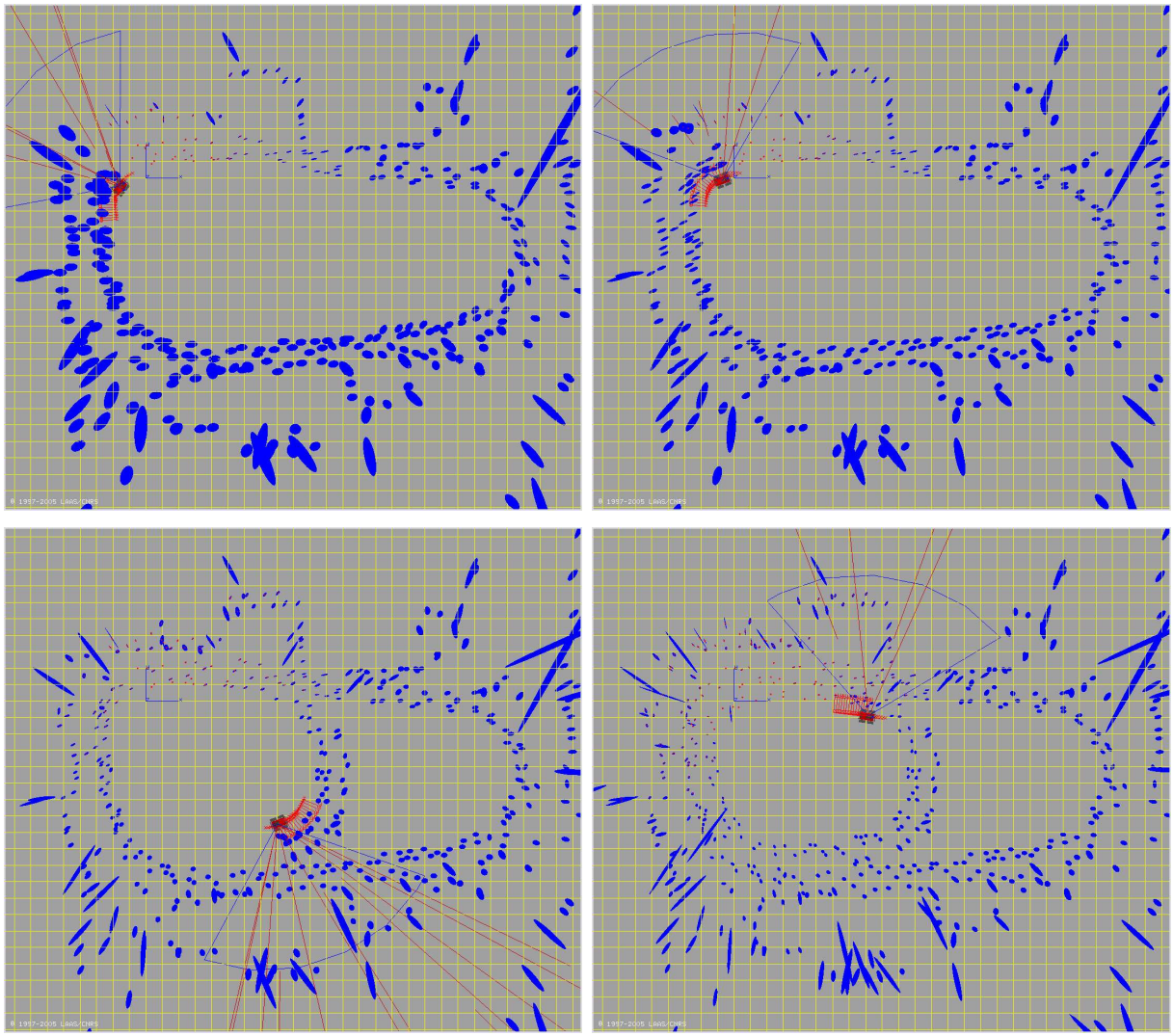

Figure 3.8: Orthogonal projection of the landmarks (3σ ellipses) just before closing the first loop (top-left), just after the first loop (top-right), in the middle of the second loop (bottom-left) and at the end of the trajectory (bottom-right). The yellow grid has a step of 1 meter.

Loop 70/463	odometry error	boSLAM error	boSLAM std. dev.
x	3.71	0.25	0.01
y	1.55	0.14	0.03
z	0.16	0.03	0.02
yaw	43.2	2.3	0.2
pitch	1.7	0.4	0.3
roll	0.03	0.08	0.2

Table 3.1: Errors made by boSLAM between frames 70 and 463 (distances in meter, angles in degree).

therefore the estimation process is very well conditioned by numerous observations of the same landmark. It also gives the possibility to map distant landmarks as they can be observed from distant robot positions.

In spite of these advantages, very few bearings-only SLAM algorithms have been applied with a panoramic camera. The initialisation method based on a bundle adjustment presented in [DH00] has been tested with panoramic images: results on a short sequence (91 images) acquired with a robot moving on a plane are presented in [Dea02].

Some articles on metric localisation using panoramic images have been published. In [CSS04] a visual odometry algorithm is developed, making the assumption that the ground is flat. Nice results with a rover moving in a desert are presented. A global localisation algorithm based on a particle filter is proposed in [ATD05]. A map of SIFT features extracted from panoramic images is first build based on a standard 2D SLAM which exploits a laser range finder. Then successful localisation is demonstrated using panoramic images only.

The domain where panoramic images are the most popular in robotic is certainly qualitative localisation. In the vision community, a large number of well founded approaches to image indexing have recently been proposed[2]. Many of them have been exploited with panoramic images in robotics to tackle the place recognition problem, a problem often associated with qualitative or topological localisation, for example in [GL02, LTG+03, CLIM05].

Some elements from the previous section are used almost unchanged:

- The same Harris corners with the same tracking and matching algorithm is used. Although it has been developed for perspective images, it works in a satisfactory way on panoramic images (see figure 3.9).
- The feature management process and the map management process described in 3.2.4 are also used here. The geometry of image cells is adapted to panoramic images as illustrated in figure 3.9.

The other parts of the whole system are described in the following sections:

[2]Progress in this area is mainly driven by image database management applications.

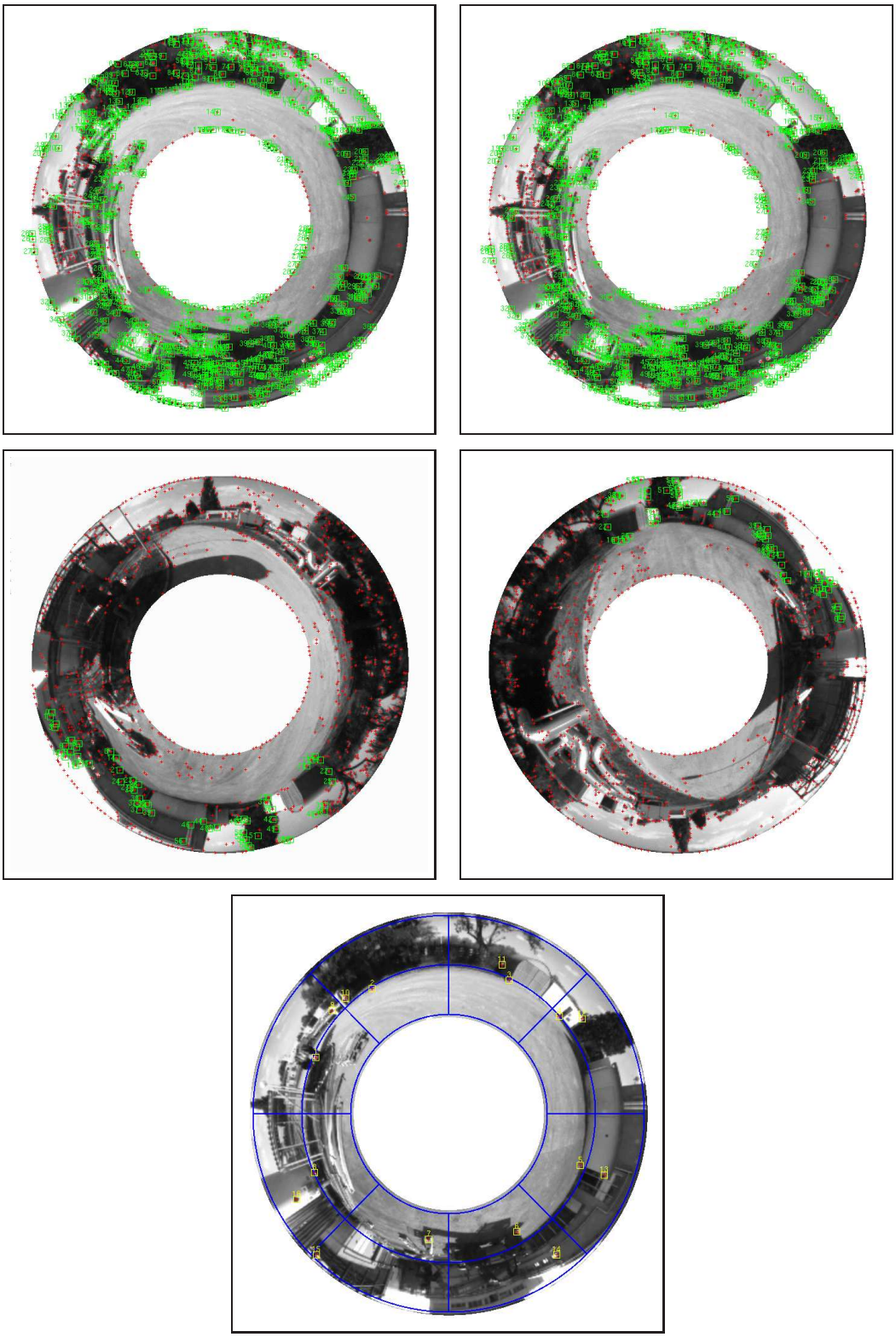

Figure 3.9: Results of the interest point matching algorithm applied on panoramic images in the case of consecutive images (top), and in the case of significantly different points of view (middle). Bottom: Selection of the features that enters the boSLAM process.

- The analytical model we used for the panoramic camera is given in section 3.4.1.
- Section 3.4.2 describes the stereo-vision based ego-motion estimation algorithm which have been implemented.
- The crucial loop closing mechanism adopted here is described in section 3.4.3.
- Calibration issues which are often taken for granted are discussed in section 3.4.4.

3.4.1 Panoramic camera model

EKF based SLAM requires the observation model $\mathbf{h}()$ and its inverse $\mathbf{g}()$ of the panoramic camera, as well as the Jacobian matrix of these functions. Only Analytical expressions of these functions are given in the following, the Jacobian matrix being computed using classic calculus methods.

The "general central projection systems model" proposed by Barreto in [Bar03a] is used in this work. A non-linear transformation depending on the mirror of the catadioptric system is first applied on the incoming rays, followed by a standard perspective projection[3]:

$$\mathbf{h}\begin{pmatrix}x\\y\\z\end{pmatrix} = PM_c\begin{pmatrix}\frac{x}{\sqrt{x^2+y^2+z^2}}\\ \frac{y}{\sqrt{x^2+y^2+z^2}}\\ \frac{z}{\sqrt{x^2+y^2+z^2}}-\xi\end{pmatrix} \qquad \mathbf{g}\begin{pmatrix}u\\v\end{pmatrix} = \begin{pmatrix}\frac{-\bar{z}\xi-\sqrt{\bar{z}^2+(1-\xi^2)(\bar{x}^2+\bar{y}^2)}}{\bar{x}^2+\bar{y}^2+\bar{z}^2}\bar{x}\\ \frac{-\bar{z}\xi-\sqrt{\bar{z}^2+(1-\xi^2)(\bar{x}^2+\bar{y}^2)}}{\bar{x}^2+\bar{y}^2+\bar{z}^2}\bar{y}\\ \frac{-\bar{z}\xi-\sqrt{\bar{z}^2+(1-\xi^2)(\bar{x}^2+\bar{y}^2)}}{\bar{x}^2+\bar{y}^2+\bar{z}^2}\bar{z}+\xi\end{pmatrix}$$

with $(x,y,z)^T$ a 3D point in the camera frame, $(u,v)^T$ a 2D point in the image plane, and:

$$P = \begin{pmatrix}\alpha_u & 0 & u_0\\ 0 & \alpha_v & v_0\\ 0 & 0 & 1\end{pmatrix} \qquad M_c = \begin{pmatrix}\xi-\phi & 0 & 0\\ 0 & \xi-\phi & 0\\ 0 & 0 & 1\end{pmatrix} \qquad \begin{pmatrix}\bar{x}\\ \bar{y}\\ \bar{z}\end{pmatrix} = M_c^{-1}P^{-1}\begin{pmatrix}u\\ v\\ 1\end{pmatrix}$$

Our mirror is a parabola with equation $\sqrt{x^2+y^2+z^2} = z+2p$, in this case the Barreto model gives $\xi = 1$ and $\phi = 1+2p$. The parameters p, u_0, v_0, α_u, α_v are obtained after a calibration process (See section 3.4.4).

3.4.2 Ego-motion estimation

Visual Motion Estimation. With a stereo-vision bench, the motion between two consecutive frames can easily be estimated using the interest point matching algorithm [MLG00, OMSM00]. Indeed, the interest points matched between the image provided by one camera at times t and $t+1$ can also be matched with the points detected in the other image (figure 3.10): this produces a set of 3D point matches between time t and $t+1$, from which an estimate of the 6 displacement parameters can be obtained (we use the least square fitting technique presented in [AHB87] for that purpose). A more detailed description of the mathematical basis can be found in appendix B.

[3] Camera lens distortions are omitted here.

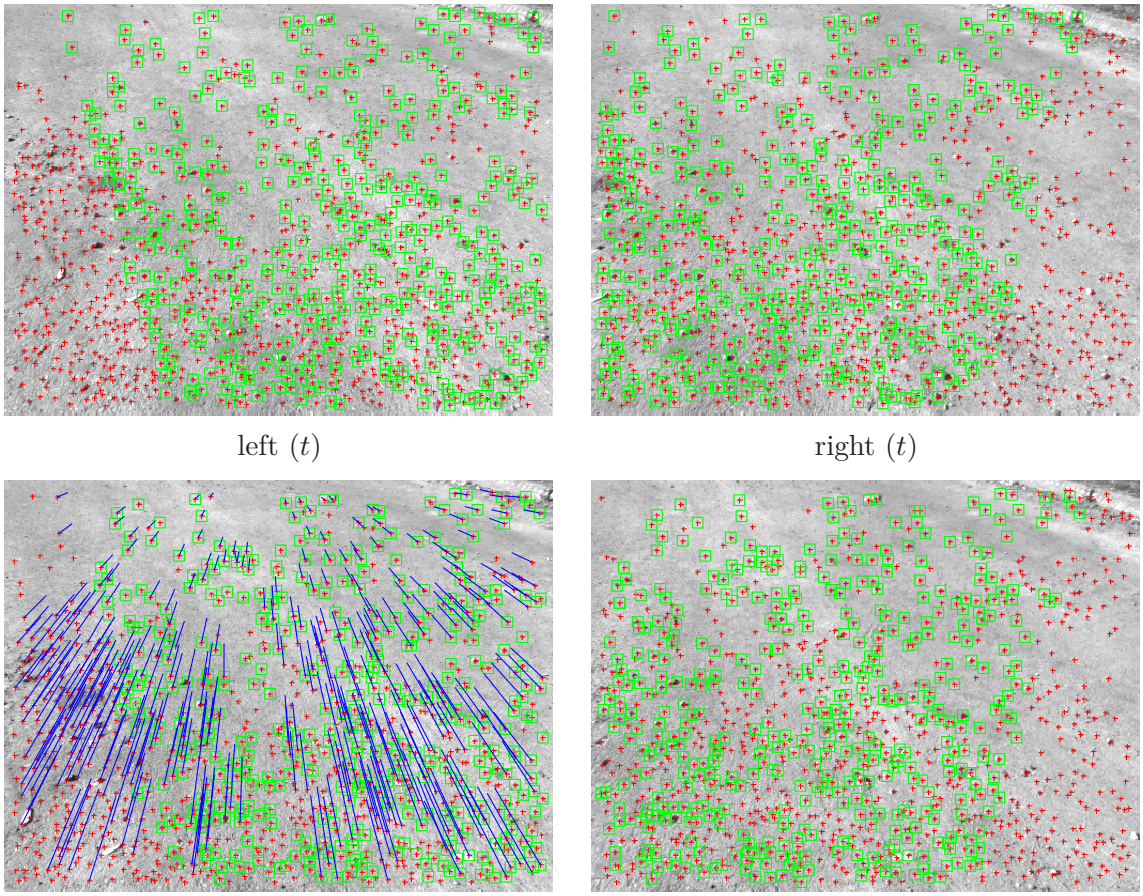

Figure 3.10: Illustration of the matches used to estimate robot motion with stereo-vision images, on a parking lot scene. The points matched between the stereo images are surrounded by green rectangles. The robot moved about half a meter forward between t and $t+1$, and the matches obtained between the two left images are represented by blue segments.

The important point here is to get rid of the wrong matches, as they considerably corrupt the minimisation result. Since the interest point matching algorithm generates only very scarce false matches, we do not need to use a robust statistic approach, and the outliers are therefore simply eliminated as follows:

1. A 3D transformation is determined by least-square minimisation. The mean and standard deviation of the residual errors are computed.
2. A threshold is defined as k times the residual error standard deviation (k should be at least greater than 3).
3. The 3D matches whose error is over the threshold are eliminated.
4. k is set to $k-1$ and the procedure is re-iterated until $k=3$.

This approach to estimate motions yields precise results: with the rover Dala, the mean estimated standard deviations on the rotations and translations are of the order of $0.3°$ and $0.01m$ for about half-meter motions (the error covariance on the computed motion parameters is determined using a first order approximation of the Jacobian of the minimised function [HJL+89]).

In the scarce cases where VME fails to provide a motion estimate (*e.g.* when the perceived area is not textured enough to provide enough point matches), the 3D odometry estimate is used.

Ground truth. In the absence of precise devices that provide a reference position (such as a centimetre accuracy differential GPS), it is difficult to obtain a ground truth for the robot position along a whole trajectory. However, one can have a fairly precise estimate of the true rover position with respect to the starting position when it comes back near the starting position at time t, using the VME technique applied to the stereo pairs acquired at the starting and current positions. VME then provides the current position with respect to the starting point, *independently from the achieved trajectory*: this position estimate can be used as a "ground truth" to estimate SLAM errors at time t.

3.4.3 Loop closing

The loop closing is one of the weakness of our previous approach: it is detected on the basis of the current robot pose estimate. Because large scale SLAM is prone to yield inconsistent position estimates, it is of crucial importance to have means to detect loop closing situations that is independent from the position estimates. In [NCH06] a similarity measure between two images is defined. A sequence of monocular images acquired by the robot is registered and a similarity matrix is built to detect loop closures.

Panoramic images are very well suited for this detection, as they easily allow the application of any image indexing technique to solve this problem. In this work, we use an indexing technique which was previously developed at LAAS in [GL02]. The principle of

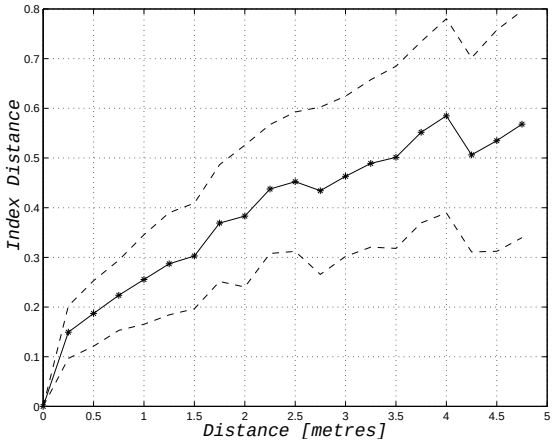

Figure 3.11: Evolution of the distance in the images index space as a function of the Euclidean distance between their viewpoints (mean and standard deviation, computed on thousands of panoramic image pairs) in [GL02].

this approach is the following: as it navigates, the rover continuously collects panoramic views of its environment and builds a database of *image indexes*, *i.e.* a set of characteristics computed on the images (learning phase). The characteristics computed are statistics (sets of histograms) of the local characteristics defined by the first and second Gaussian derivatives responses of the images, that prove to have good invariance properties. After a long traverse, when the rover arrives nearby an already crossed area, a newly perceived image is matched with the stored ones using a distance in the index space (query phase – the distance is based on χ^2 statistics between the sets of histograms, considered as probability density functions).

To reduce both the storage memory and the recognition computation time, it is worthwhile to reduce the number of stored images: a simple way to achieve this is to discard the images that are similar, using the distance between image index as a criterion. Also, when used in a SLAM context, the search process can be focused on the basis of the current robot pose estimate – even if it is a coarse one.

The algorithm can efficiently find among a database of hundreds of images the closest image, "closest" being defined here in the index space. It happens that for Euclidean distances on the order of a few meters, there is a quite strong correlation between the index distance and the point of view distance. Figure 3.11 presents statistics obtained with thousands of image pairs that exhibit this fact, which is exploited here to determine whether a loop has been closed or not.

Figure 3.12 summarises the processes that are applied to each panoramic image every time they are acquired: the Harris interest points are extracted, and the histograms that index the image database are computed. Of course, the fact that both computations rely on the determination of Gaussian derivatives computed on the image allows efficient processing of the images.

The rest of the algorithm is presented in figure 3.13: first, feature points detected at

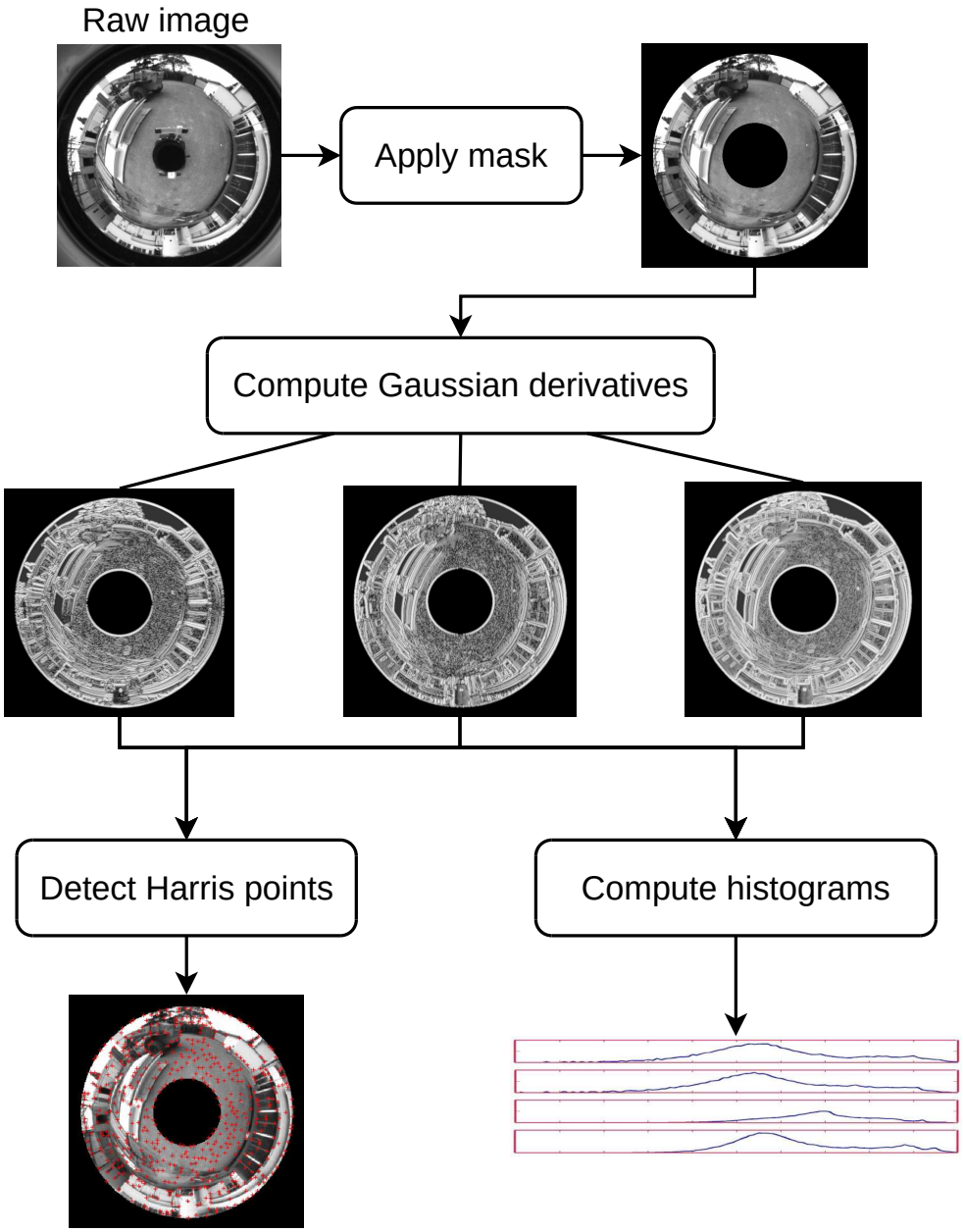

Figure 3.12: Processes involved every time a panoramic image is acquired.

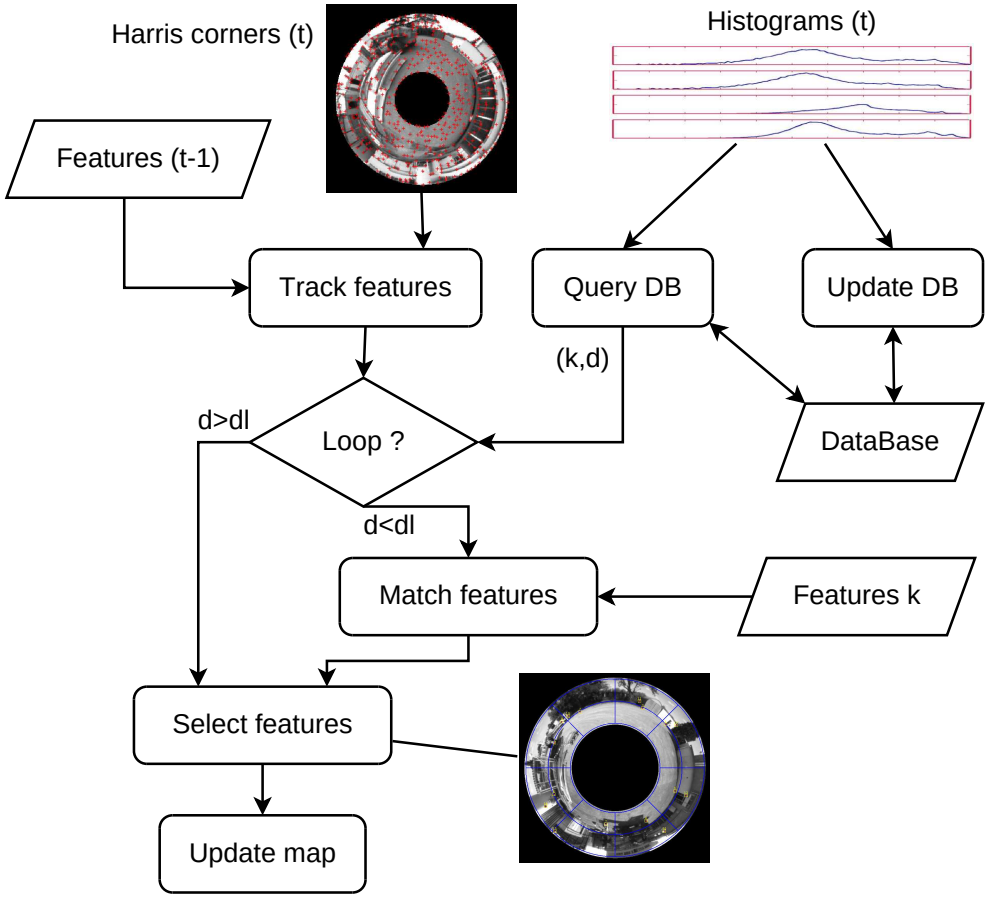

Figure 3.13: Panoramic data flow for SLAM: feature tracking, matching and loop closing.

time $t-1$ are tracked in the new set of points t. The histogram database is updated with the new sets of histograms, and is queried: it returns the index k and the index distance d of the most similar panoramic image in the database, omitting a fixed number of recent frames k_{min}. $d < d_l$ triggers a potential loop detection: the points detected in image k and the current points are sent to the point matching algorithm. The matched points between images t and k yield observations that are used to update the map, and new features are selected in the current image areas where no corner has been tracked or matched.

There is an important point here. The threshold d_l on the image index distance can be defined thanks to the relation between the image index distance and the Euclidean distance presented in figure 3.11: a small value of d_l would cause the process to try to establish matches only with memorised images that are supposed to be very close to the current position, thus trying to only detect *actual* loop closures. But higher values of d_l cause the detection of matches with images that are not so close to the actual position (most recent images being omitted by the threshold k_{min}). We actually use a rather large value for d_l, in order to re-establish matches with landmarks (*i.e.* re-observe them) that are not currently being tracked. This redefines the notion of "loop closure": a loop-closure occurs

here when a landmark that is not currently being tracked is re-observed. Loop-closure is then a "perceptual event" that does not necessarily correspond to a topological situation. Section 3.5 shows a plot of such "events".

3.4.4 Calibration

Panoramic camera. The Panoramic camera is calibrated thanks to the Matlab toolbox developed by Christopher Mei [MR06, Mei] (we have used the *biased calibration* procedure). The calibration process is very similar to the one of a simple perspective camera: images of a known planar chessboard pattern are first acquired, and a semi-automated process extracts the corners of the chessboards from the set of images. The parameter p that defines the mirror is given by the manufacturer and is usually known very precisely, and the detection of the panoramic image circular boundary gives a first estimate of u_0, v_0 and of the focal length. Then a minimisation procedure refines their estimation.

Inter-frames calibration. The motion predictions being provided by one sensor (the stereo-vision bench) and the landmarks being observed by another one (the panoramic camera), it is of essential importance to have a precise estimate of the transformation between the associated frames: errors in this transformation would indeed bias the SLAM estimation process. The precise knowledge of this transformation is often taken for granted, although most SLAM applications do rely on different sensors for the prediction and the observation.

Thanks to camera calibration tools, this transformation can be estimated with good precision. Images of the chessboard are acquired simultaneously by the stereo-vision bench and the panoramic camera (figure 3.14). Knowing the intrinsic parameters of both cameras, the chessboard to camera transformations are computed, from which the transformation from the stereo-vision cameras to the panoramic cameras is estimated (we actually estimate the transformation between the *left* camera of the stereo bench and the panoramic camera, as VME produces motion estimates in this camera frame).

Figure 3.14 presents the frames used: the *ref* frame position is the one estimated by the SLAM process. In our case, the *ref* frame is not the usual *robot* frame but the *sensor* frame (the panoramic camera). As a consequence, there is no need for a *robot to sensor* frame composition at each landmark observation. The required frame composition is only applied to the predictions: since there are fewer prediction steps than observation ones, this is computationally more efficient.

3.5 Results with a panoramic camera

3.5.1 On a small loop

Figure 3.15 shows results obtained while the rover is following a path of about 25 meters. The top-right figure indicates the occurrence of loop-closing detection events: mapped

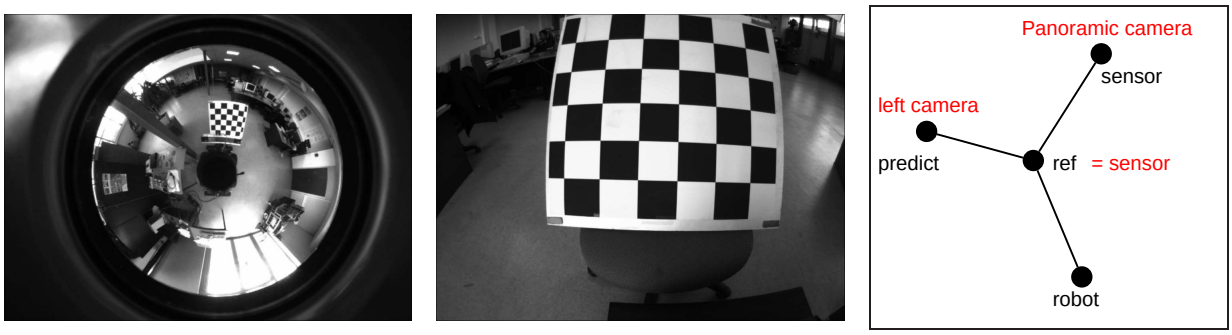

Figure 3.14: Images used for the inter-sensor calibration. Right: in black the frames usually involved in SLAM, in red the frames we use.

landmarks are re-observed before the rover actually comes back to a previously visited position. As a consequence, the robot position estimate is smoothly corrected by these observations of "loop-closing" features. Here the drop in the uncertainty of the robot pose is less abrupt than with a sensor with a limited field of view (compare *e.g.* with figure 3.4 on page 42). More abrupt loop closing causes huge Kalman gain and large innovation, which is one of the causes of divergence.

3.5.2 On a longer trajectory

Figure 3.16 shows an about 200m long trajectory, that exhibits three different loops, and during which the robot is driven in a rough area, in which the robot's pitch and roll angles reach values of the order of 5 to 10 degrees. The complete run can be viewed on the following video: http://www.laas.fr/~tlemaire/download/boSlamPano.mp4.

Figure 3.17 gathers various data illustrating the behavior of the SLAM process. The top figures compare the trajectory estimated by integration of the VME local motion estimates: the $x - y$ plot shows a particular good behavior of VME, that has been "lucky" here, its errors being well compensated. However, the robot elevation estimate (top right) shows that VME is actually drifting. The middle row figures exhibit the behavior of the loop detection process. The three loops can be seen on the left curve that plots the robot pose "uncertainty volume" (the determinant of the covariance matrix): topological loop closing occur around frames 150, 320 and 1070. The right plot indicates the loop closure detections and the number of landmarks that are re-observed: this number exactly looks as expected. Indeed, the path followed by the robot revisits many known places on its way back from the rough area (around frame 1000), but no loop closing feature is matched there. The database search can sometimes give a wrong answer (*i.e.* the returned image has not been acquired near the robot current position), and sometimes the feature matching algorithm does not find enough matches on quite similar images. The hesitating behavior of the database search is illustrated on the figure 3.17-bottom-left. Nevertheless, landmarks mapped at the beginning of the trajectory are eventually re-observed, and the robot uncertainty is then considerably reduced – still in a rather smooth manner. The bottom right plot shows

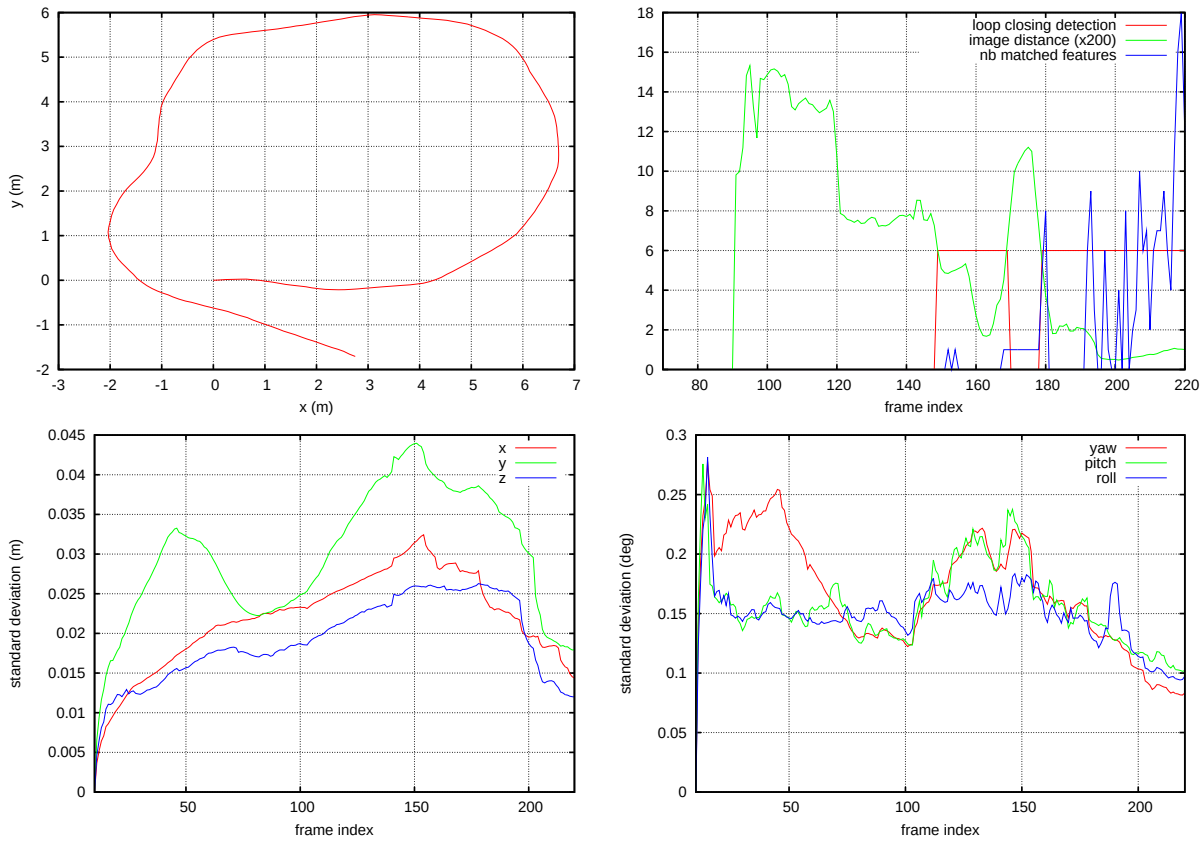

Figure 3.15: Top-left: the estimated small loop trajectory, right: loop-closing detection events. Bottom: evolution of the standard deviations of (x, y, z) and (θ, ϕ, ψ).

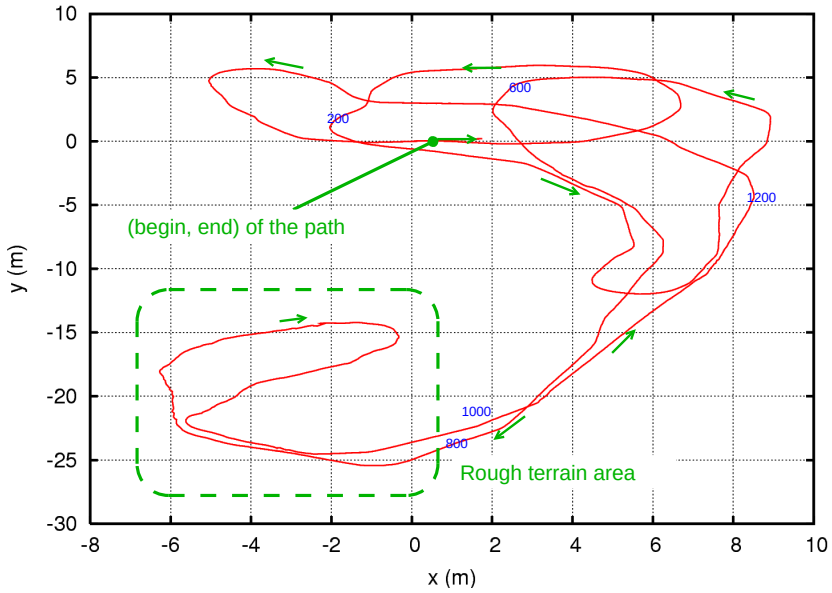

Figure 3.16: An about 200 m long trajectory, approximate frame index are indicated in blue.

the evolution of the robot attitude (note the rough terrain traversed between frames 800 and 1000).

Estimation errors

Although no ground truth is available along the whole trajectory, the estimated robot pose errors can be computed at some points of the trajectory: the 3D transformation between two close robot positions after an actual loop closure can be computed using the "ground thuth" technique described in 3.4.2. Table 3.2 demonstrates the precision of our SLAM algorithm:

- After an about 200 meter long trajectory, the robot pose estimate has a translation error of less 0.15m, and orientation of errors of less than a degree.
- Nevertheless, this estimate is not strictly consistent since some translation and angular errors exceed the $3-\sigma$ bound.

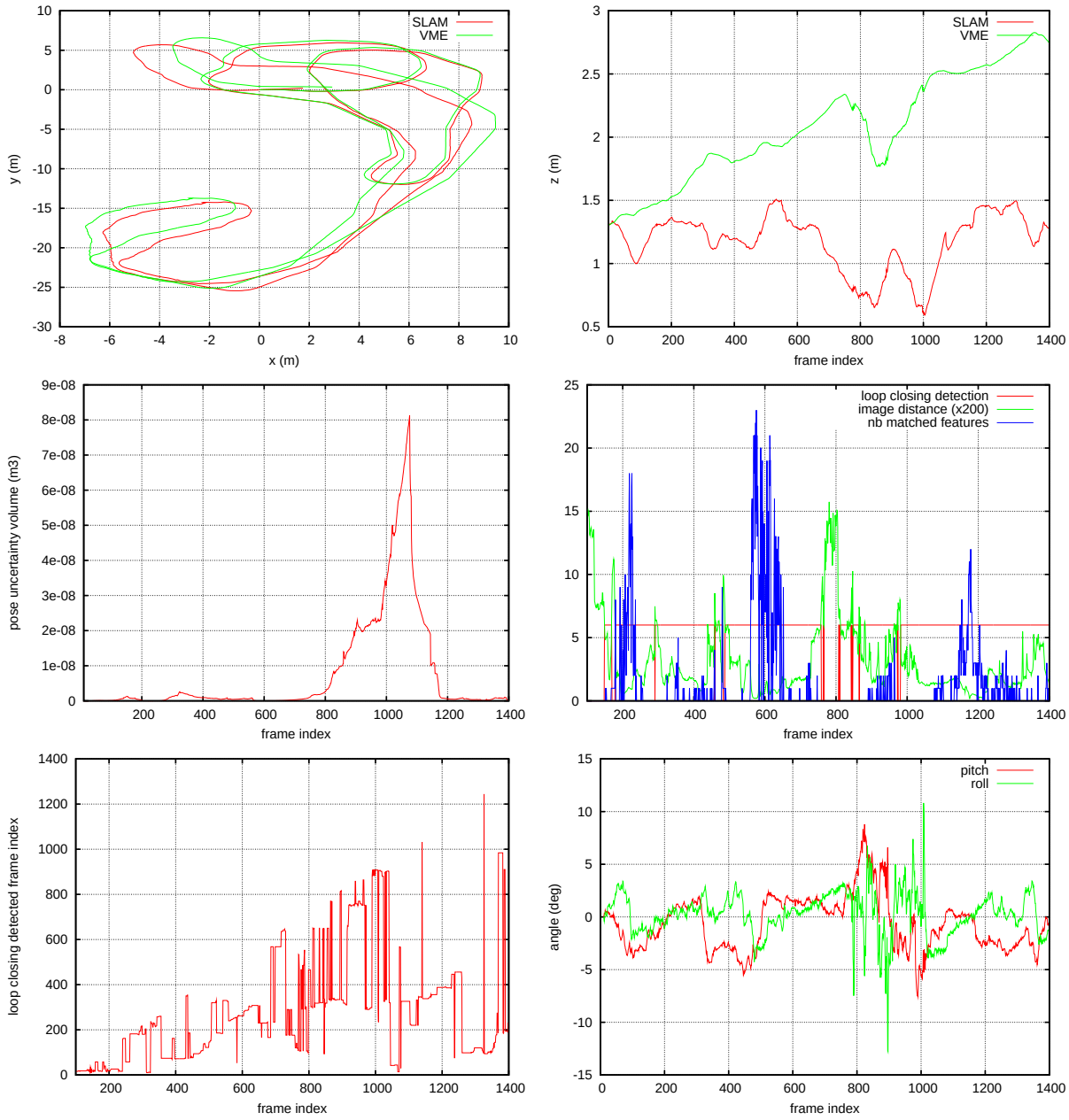

Figure 3.17: Top: comparison of SLAM pose estimate versus pose estimate obtained with VME integration, left: on the $x - y$ plane, right: on the z axis. Middle-left: the loop closing effect can be seen in the evolution of the robot pose "uncertainty volume", right: number of re-perceived landmarks. Bottom-left: index of the memorised images matched as a function of the current image index, right: evolution of the pitch and roll angles along the trajectory.

	VME 10→1430	σ	SLAM 10→1430	σ	SLAM error
x	-0.033	0.005	-0.015	0.016	0.018
y	0.126	0.003	-0.016	0.016	0.142
z	-0.011	0.004	0.050	0.011	0.061
θ	5.1	0.1	4.8	0.1	0.3
ϕ	0.1	0.1	0.3	0.1	0.2
ψ	0.1	0.1	0.6	0.1	0.5

Table 3.2: Position estimated by the SLAM process between the first and last positions of figure 3.16 (frames 10 and 1430), compared to the "ground truth" provided by VME applied between the corresponding stereo-vision images.

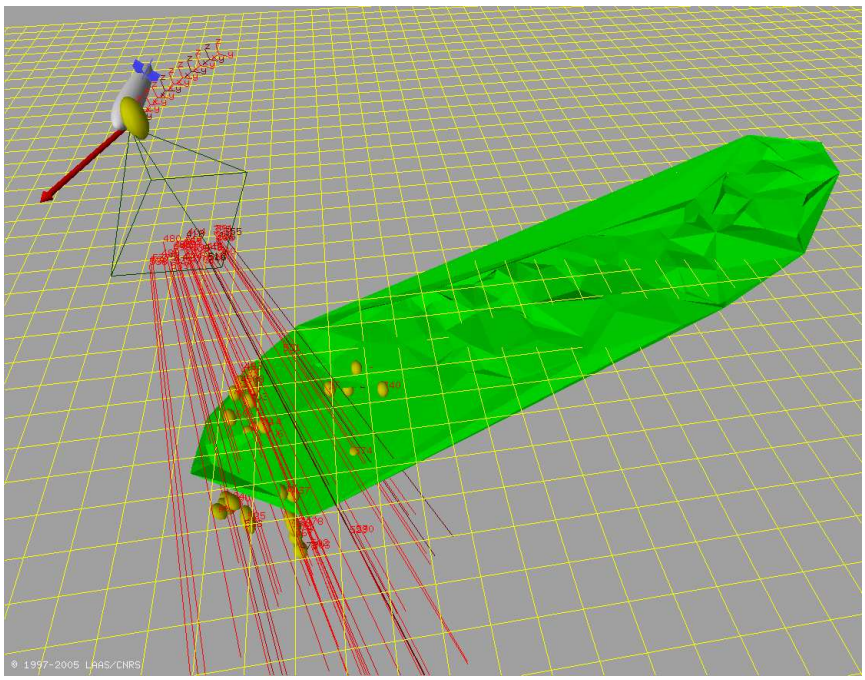

Figure 3.18: Sparse elevation map of a flat area reconstructed using low altitude aerial images, the grid step is 5 meters.

3.6 Results with aerial images

3.6.1 Building a sparse digital elevation map

The blimp Karma was used for these experiments. Karma is equipped with a single camera looking downward to the ground. The same feature detection and matching algorithm as in previous experiments is used. But the prediction has to be conducted with the on board available sensors: a 3D compass with inclinometers and a metric GPS. The pose estimate given by the GPS is not usable for the purpose of SLAM since it often "jumps", especially the altitude estimate. Nevertheless, the GPS also gives a measure of the linear speed which is much more reliable. Finally the state vector of the robot is composed of the position, the orientation and the linear speed. The 3D compass data are used to update the orientation, the GPS speed the linear speed of the robot, and a "constant linear speed, constant orientation" prediction model is applied.

Because of the poor quality of this model and of the input data, results could only be obtained on about 100 images along a nearly straight trajectory of 120 meters at an altitude of 40 meters. Here, no loop closing was attempted, rather a technique of "short memory SLAM" was applied: as soon as a feature is lost by the perception system it is removed from the stochastic map. Then it is added to a sparse digital elevation map which is built using an incremental Delaunay triangulation algorithm. Figure 3.18 shows the map obtained at the end of the considered data set. The boSLAM algorithm and the construction of the elevation map are illustrated in the following video: http://www.laas.fr/~tlemaire/download/karmaSparseDEM.mp4

3.6.2 Closing the loop

The previous results did not demonstrate loop closing because it was not possible to find a loop in our data set with prediction data of good quality. The prediction sensors are especially erroneous when Karma is turning.

In this section results on a full loop are presented. But the setup is not fully monocular: the prediction data is now obtained with the VME algorithm presented in section 3.4.2 which is based on stereo-vision.

The simple loop-closing procedure presented in 3.2.5 is applied here. Figure 3.20 illustrates the loop detected in this data set: the current image (left) shows in blue the features which are matched with the image in the data base (right).

The trajectory of Karma during this experiments is presented figure 3.19. The trajectory obtained by integration of VME and the trajectory estimated with boSLAM are compared: the drift in the VME trajectory is reduced by boSLAM, especially on the height estimate.

The loop is closed around frame 1951. A "ground truth" can be computed for frame 1961: figure 3.21 presents the two left images used to compute this 3D transformation.

The errors of VME only and boSLAM pose estimates are presented in table 3.3: boSLAM exhibits a serious error in the x coordinate. Also the boSLAM estimates are

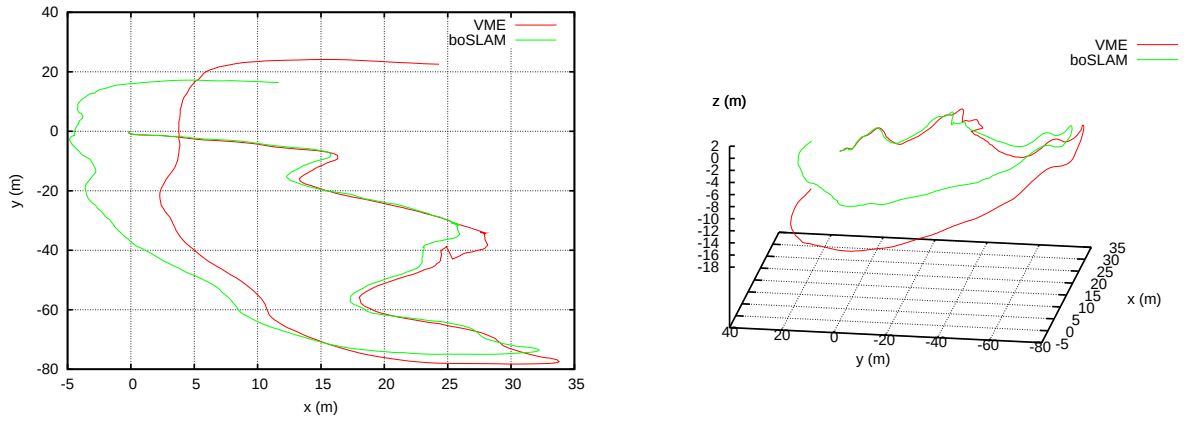

Figure 3.19: Trajectory of Karma obtained with integration of VME and with boSLAM (left projected on the (x, y) plane, right in 3D)

Loop 1650/1961	VME error	boSLAM error	boSLAM std. dev.
x	4.11	9.11	0.27
y	1.01	1.87	0.16
z	13.97	1.68	0.20
yaw	4.26	4.49	0.26
pitch	5.70	1.24	0.47
roll	7.66	0.59	0.24

Table 3.3: Loop closing results with the blimp (distances in meter, angles in degree).

not consistent: this underlines the difficulties to achieve bearings only SLAM with an aerial robot. In this sequence, the setups of the altitude of the blimp, the field of view and the frame rate of the camera are not ideal. Once perceived, the features rapidly leave the image, therefore their estimate is of poor quality and the robot pose estimation does not benefit a lot from the map.

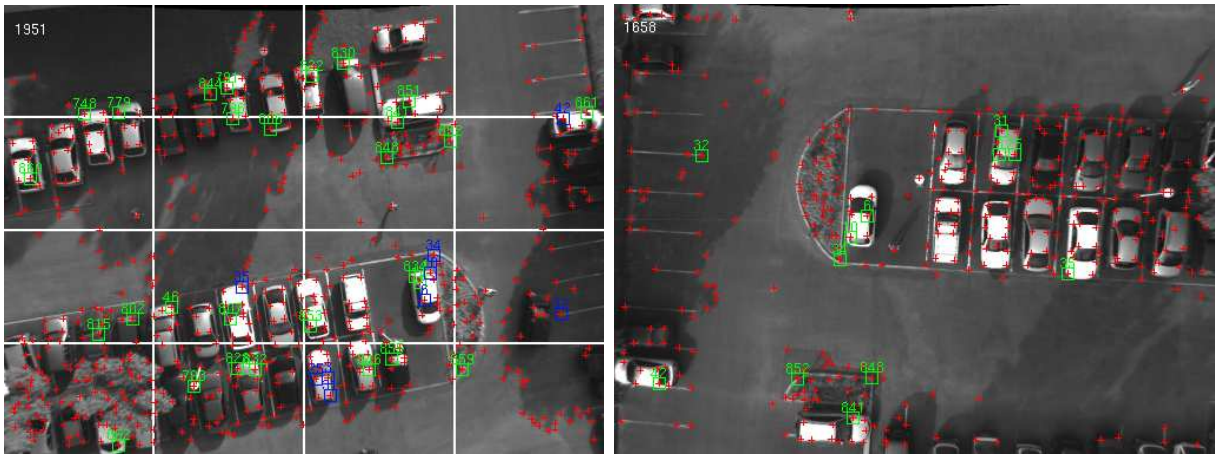

Figure 3.20: Landmarks matched during a loop closing. On the left the current processed image, with the landmarks currently tracked (green squares), and the ones that have been perceived again and matched (blue squares) with the stored image shown on the right.

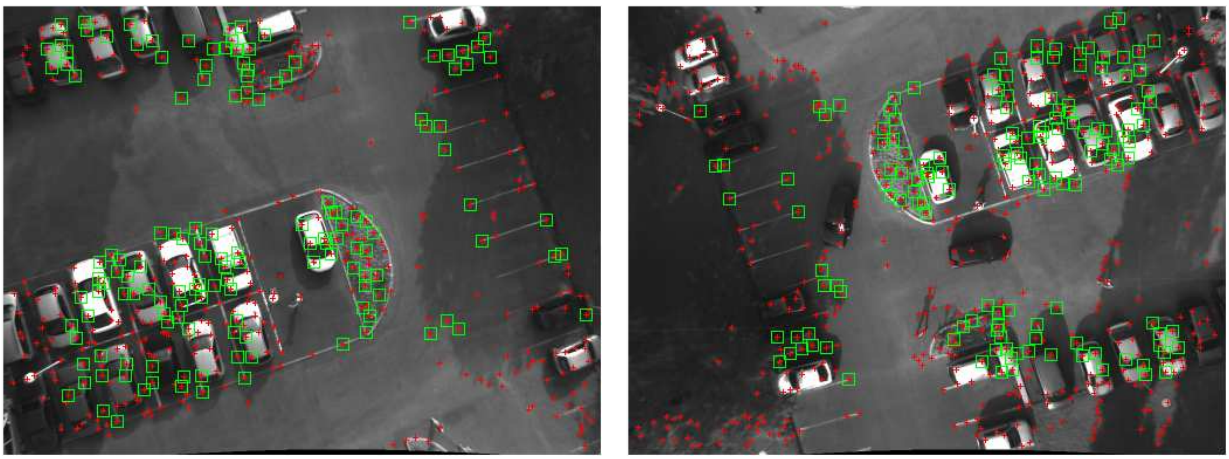

Figure 3.21: images used to establish the ground truth by applying VME between the reference frame 1650 and the frame 1961. Green squares are the points matched between the two frames.

3.7 Discussion and conclusion

About the Interest Point Matching algorithm. In our current implementation, the IPM algorithm does not benefit from the predicted observations provided by SLAM. This information is very reliable in the tracking case and could easily be used by IPM. In a first stage of IPM the Harris corners are extracted and grouped in both images (only in the new image in the tracking case), in a second stage an initial group match is randomly searched among groups from both images. Here it is straightforward to guide the search with the predicted observations.

Conclusion. In this chapter, our bearings-only SLAM algorithm is demonstrated on real images. Algorithm validation on real data corrupted with real life noise is of main importance in robotics. A complete SLAM solution based on the use of a panoramic camera for features perception and on a stereo-bench for robot pose prediction is proposed. Also, a loop-closing mechanism only based on panoramic images is described. The same data is used to build two different database: a higher level data base for the loop closing detection, and a feature data base for feature matching. This architecture proves to be robust and efficient in real situations.

A real-time implementation of this SLAM solution is currently being developed, more details about the software architecture are given in appendix C.

Chapter 4

Vision based SLAM with Line Segments

This chapter is an extension of chapter 2 to incorporate any 3D line segments in the stochastic map. It successively addresses the problems of line representation, landmark initialisation, and Kalman estimation under constraints. Finally the algorithm is validated in simulation, and results on real data are presented.

4.1 Introduction

In this chapter, the use of landmarks made of 3D lines segments derived from edges detected in images is proposed. There are various advantages to use 3D lines: first, such primitives are very numerous in structured environments (indoor or urban outdoor), second, contrary to sparse map of points which are only useful for localisation purposes, a map of segments gives relevant information on the structure of the environment: it is a sound basis to extract planes for instance. Finally, edge matching can be achieved even when important viewpoint changes occur, like in loop closing, or when matching aerial and ground data.

Work by Folkesson *et al.* published in [FJC05] addresses the problem of vision based SLAM with segments. In this work, an innovative estimation process based on a Kalman filter is developed, it takes advantage of partially uninitialised landmarks. An illustrative example is given in which horizontal lines are extracted from an image acquired by a camera looking up to the ceiling. The lines are supposed horizontal, so the first observation gives an estimate of their direction, but the depth is left unknown until it can be computed by triangulation. This work does not address the estimation of plain 3D lines.

It is only recently that 3D segment vision based SLAM algorithm have been published. These algorithms are based on the inverse depth parametrisation presented in [ED06b, MCD06] for points and were extended in [ED06a, SRD06] to incorporate any 3D segments in the map. This inverse-depth parametrisation has already been discussed for points in section 2.5.3 on page 32. In [SRD06], the line segments are represented by two points

which are chosen to be close to the extremities of the real segment. A segment is initialised immediately in the map, each end point being initialised using the inverse-depth technique presented in [MCD06]. In [SRD06] a 3D line segment is represented by its middle point and a unit vector for its orientation. For the middle point, the same initialisation method as in [ED06b] is applied, but the article does not mention anything special about the initialisation of the orientation. These representations are not minimal and present gauge freedoms: neither the extremities not the middle point of the segments are observed. While it does not seem to introduce any problem, this point is not discussed.

Several other initialisation methods for points have been reviewed in chapter 2 (see table 2.1 on page 13). The state of a 3D line is defined in a space of higher dimension than a 3D point, and moreover two dimensions of this space are not measured by a single image segment observation: depth and orientation. Therefore, multi-hypotheses delayed and moreover un-delayed methods are challenged by the high number of hypotheses which are required to approximate the initial state of a 3D line. The algorithm presented in chapter 2 offers a delayed method well adapted to a huge number of hypotheses and is a good candidate for developing a vision based SLAM algorithm using line segments.

As for points, the vision based SLAM problem and the Structure From Motion problem with line segments do have much in common. Strong results were obtained in the computer vision community, for example in [JJL00, BS05], but often on a small data-set and with computationally expensive techniques.

In this chapter, the focus is put on the representation and estimation problem for SLAM using image segments extracted from images. Section 4.2 justifies the choice of the Plücker coordinates to represent 3D lines. Section 4.3 describes how the initial sum of Gaussians representation of a 3D line is computed, and section 4.4 presents how the satisfaction of the Plücker constraint can be considered within a classic EKF-based SLAM approach. Section 4.5 then presents the validation of the algorithm conducted in simulation, and finally section 4.6 shows results obtained with real data.

4.2 3D segments for SLAM

4.2.1 3D line representation

Several sets of parameters can be used to represent a 3D line L in the euclidean space. The minimal representation consists of 4 scalars: such a minimal representation is (P_1, P_2) where $P_1 = (x_1, y_1, 0)^t$ is the intersection of L with the plane $\Pi_1(z = 0)$ and $P_2 = (x_2, y_2, 1)^t$ is the intersection of L with the plane $\Pi_2(z = 1)$. Several conventions for (Π_1, Π_2) coexist so as to represent all possible lines with a satisfactory numerical precision.

A more intuitive but non minimal representation of L is $(A, \underline{u})$, where A is any point of L, and $\underline{u}$ is a direction vector of L[1]. In this representation, the choice of A is arbitrary and A is not observable since it cannot be distinguished on the line.

[1]Notation: $\underline{x}$ means that vector x is a unit vector.

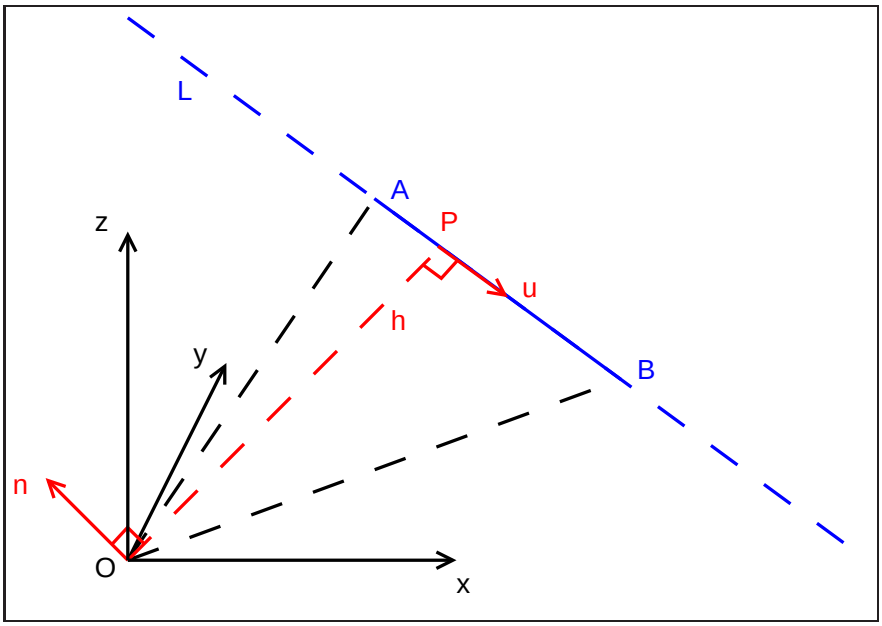

Figure 4.1: Presentation of the Plücker coordinate $(h\underline{n}, \underline{u})^t$ of the 3D line L. $\underline{n}$ is the normal vector to the plane supported by the points A,B and O.

An other representation often used in the vision community is the Plücker coordinates, because it is well adapted to the projection through a pinhole camera (an in-depth presentation of the Plücker coordinates can be found in [HZ04]). In this work, we use the Euclidean Plücker coordinates, represented by the following 6-vector (see details in appendix D):

$$L_{(6\times1)} = \begin{pmatrix} n = h\underline{n} \\ \underline{u} \end{pmatrix} \tag{4.1}$$

$\underline{n}$ is the normal to the plane containing the line and the origin O of the reference frame, h is the distance between O and the line and $\underline{u}$ is a unit vector which represents the direction of the line. This representation is presented on figure 4.1.

The Plücker constraint has to be satisfied[2]:

$$n \cdot \underline{u} = 0$$

This ensures that the representation is geometrically consistent. Any point P on the line satisfies the relation:

$$P \wedge \underline{u} = n \tag{4.2}$$

It is also interesting to note that the closest point to the origin is given by:

$$P_O = \underline{u} \wedge n$$

The state vector of a line feature being defined by its Euclidean Plücker coordinates, the observation function and the reference frame transformation need to be defined so as to tackle the SLAM problem.

[2]Notation: $x \cdot y$ is the dot product between x and y

Observation function. The projection of a 3D line L in an image is a 2D line l which is defined by the intersection of the image plane and the plane defined by n: the canonical representation of l ($ax + by + c = 0$) is exactly n expressed in image coordinates:

$$l = P_{l(3\times3)} \begin{bmatrix} 1_{(3\times3)} & 0_{(3\times3)} \end{bmatrix} \begin{pmatrix} n \\ u \end{pmatrix}_{(6\times1)} \qquad l = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \tag{4.3}$$

P_l is the camera projection matrix for a Plücker line, it is defined on the basis of the camera intrinsic calibration parameters:

$$P_l = \begin{pmatrix} \alpha_v & 0 & 0 \\ 0 & \alpha_u & 0 \\ -\alpha_v u_0 & -\alpha_u v_0 & \alpha_u \alpha_v \end{pmatrix}$$

This canonical representation of l is not unique and has to be normalised so as to compute the observation innovation (see section 4.4.1). A common normalised parametrisation for 2D lines is (ρ, θ). It is trivially related to the canonical representation $l = (a, b, c)^t$, by the following non-linear expression:

$$\frac{1}{\sqrt{a^2 + b^2}} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ -\rho \end{pmatrix}$$

Frame transformation. This operation is very common in a SLAM algorithm. Given a reference frame transformation $(R, t)_{1\to2}$, the Plücker coordinates of L in the two frames are related by:

$$L_1 = \begin{bmatrix} R & [t]_\wedge R \\ 0_{(3\times3)} & R \end{bmatrix}_{(6\times6)} L_2 \tag{4.4}$$

where $[x]_\wedge$ denotes the (3×3) anti-symmetric matrix that corresponds to the cross product: $\forall y, x \wedge y = [x]_\wedge . y$.

4.2.2 About segment extremities

Only the representation of the supporting line of a segment has been presented so far. The extremities of a segment cannot be used in the stochastic map: their extraction from images is not stable and depends on the viewpoint because of occlusions (see figure 4.9 on page 79). Nevertheless extremities of a segment are very useful: they can be used to find visible segments so that only relevant features are tested during the matching process, and segments are more informative than infinity lines on the structure of the environment. The two extremities can be stored as two abscissas (s_1, s_2) on the 3D line, the frame of the line being defined by the closest point to the origin and the direction of the line:

$$(P_O = \underline{u} \wedge n, \underline{u})$$

With each observation of the segment, an estimate of (s_1, s_2) can be computed. Since it is not guaranteed that true segment extremities are observed, this value cannot be *fused* with the current estimate. We choose to update (s_1, s_2) each time their new values increase the length of the segment. That way, the estimated segment is not sensitive to occlusions or false detection of the segment extremities.

4.3 Line segment initialisation

4.3.1 Gaussian hypotheses generation

In this section, the construction of the initial Gaussian hypotheses is presented step by step. Let's consider a line observation $z = (\rho, \theta)^t$ extracted from an image. This observation of the 3D line L constraints L to lie on the plane Π supported by the focal point of the camera and the (ρ, θ) line of the image plane. This observation is not perfect and is modelled using a centred Gaussian noise. The problem is to find an analytical approximation of the PDF $p(L_{/C})$, where $L_{/C} = (h\underline{n}, \underline{u})$ is the 6-vector Plücker coordinates of L expressed in the camera frame. Because of the projection function of the camera, $p(L_{/C})$ cannot be approximated using a single Gaussian.

Nevertheless, the unit vector $\underline{n}$ which is the normal vector of Π is readily estimated and its PDF is approximated by a single Gaussian. The projection equation of the Plücker coordinates (4.3) gives[3]:

$$\underline{n} = \frac{l'}{\|l'\|} \quad \text{where} \quad l' = P_l^{-1} \begin{pmatrix} \cos\theta \\ \sin\theta \\ -\rho \end{pmatrix}$$

The result of the retro-projection needs to be normalised to recover $\underline{n}$. Using usual uncertainties propagation calculus, the Gaussian PDF $p(\underline{n})$ is obtained.

There are two quantities which prevent $p(L_{/C})$ from being a Gaussian: depth and direction of the line in the plane Π. In other words, depth and orientation cannot be approximated with a single Gaussian, and are approximated with a Sum of Gaussians.

In order to sample $p(L_{/C})$, in a relevant way, we proceed as follows:

- a "generate" vector $\underline{g}$ is defined,
- the depth d of the line is defined along $\underline{g}$, $p(d)$ is sampled,
- the direction ϕ of the line is defined with respect to $\underline{g}$, $p(\phi)$ is sampled.

The steps of this process are shown figure 4.2.

[3] see appendix D

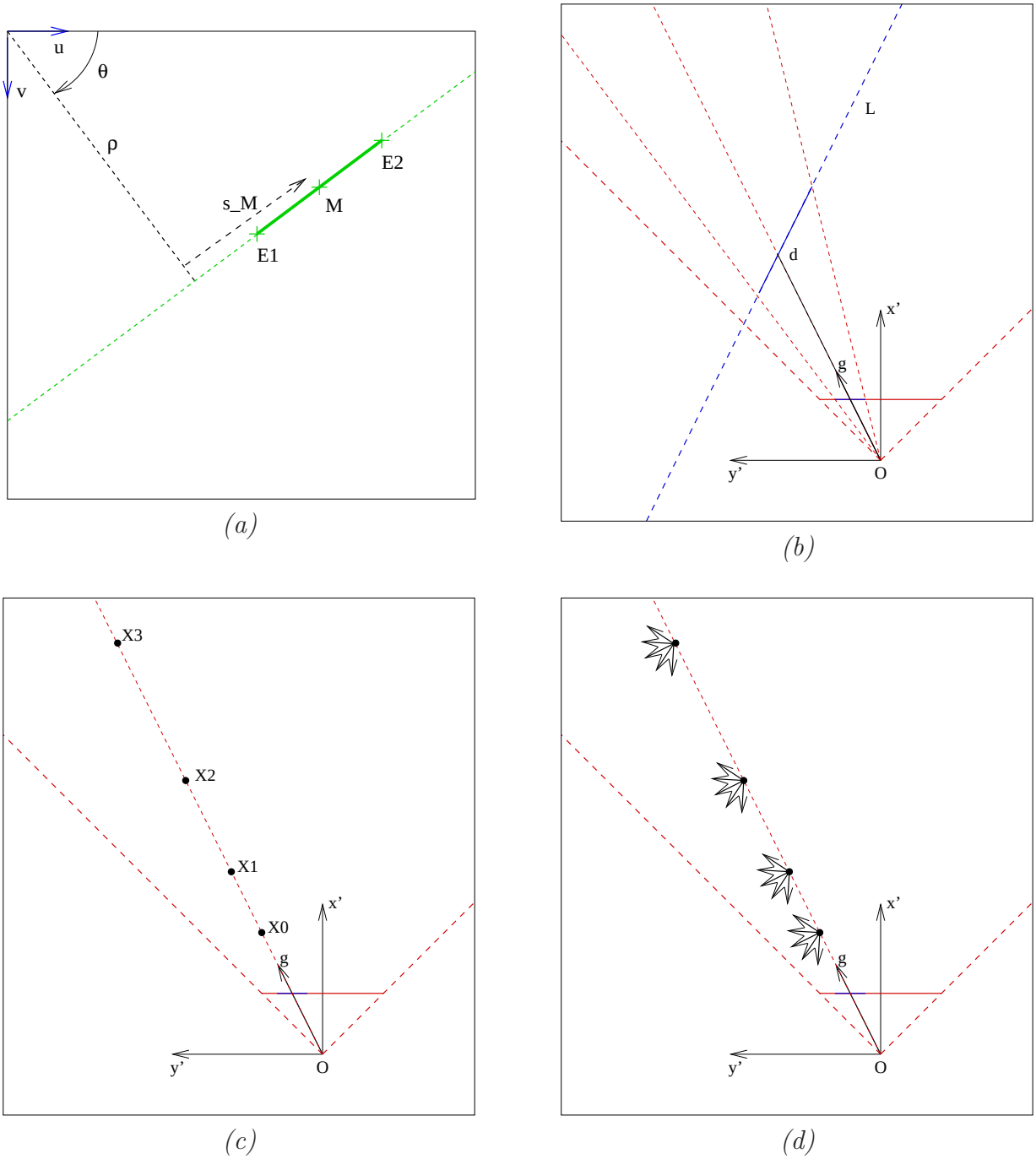

Figure 4.2: *(a)* in the image plane, definition of the point M with $(\rho, \theta, s_M)^t$. *(b)* Projection of line L and definition of vector $\underline{g}$. (Ox'y') corresponds to the plane Π, in red the image plane projection in Π and the camera aperture. *(c)* The set of points X_i. *(d)* The set of lines $L^{i,j}$.

Depth. A relevant "depth" for the line L is not trivial to define. The natural depth is the distance of the line to the origin O of the camera frame, this distance is represented by h (4.1). But depending on the direction of the line, the distance to the segment (the visible part of the line) can be very different from h. As a consequence, sampling directly the parameter h is not relevant to represent the hypotheses on the depth of the object from which the segment is extracted: we rather consider as the "depth" the distance d from the camera to the real object, in the direction defined by $\underline{g}$.

To define the generate vector $\underline{g}$, an arbitrary point M is chosen on the segment in the image, for example the middle point of the visible part of the segment. M is partially stochastic: it lies on the stochastic line (ρ, θ) going through the segment, at the abscissa s_M (figure 4.2). s_M is an arbitrary value, so it is not stochastic. $\underline{g}$ is the stochastic unit vector pointing to M, it is approximated by the Gaussian $\Gamma(\underline{g}, P_{\underline{g}})$.

$p(d)$ is supported by $\underline{g}$ and can now be expressed. $p(d)$ is a uniform distribution, its range is limited to $[d_{min}, d_{max}]$ by an *a priori* knowledge of the environment. As in chapter 2, a Gaussian sum defined by a geometric series is a good approximation of $p(d)$. The sum is composed of n_d Gaussians:

$$d_0 = d_{min}/(1-\alpha_d)$$
$$d_i = \beta_d^i d_0 \qquad \sigma_{d_i} = \alpha_d d_i \qquad w_i = 1/n_d$$
$$d_{n_d-2} < d_{max}/(1-\alpha_d) \qquad d_{n_d-1} \geq d_{max}/(1-\alpha_d)$$
$$\beta_d = \frac{1+k_{\sigma_d}\alpha}{1-k_{\sigma_d}\alpha_d}$$

Figure 4.3-left illustrates this sum.

The above sampling defines a set of points X_i which is the basis for the sampling of $p(L_{/C})$:

$$X_i = d_i \underline{g}$$

Direction. For each point X_i, the set of possible lines going through this point is sampled considering the direction ϕ of the line L. Again, $\underline{g}$ is used as a reference. Any possible direction $\underline{u_\phi}$ for the line L can be obtained with $\underline{g}$ by a simple rotation of ϕ radians around $\underline{n}$:

$$\underline{u_\phi} = Rot(-\phi \underline{n})\underline{g}$$

where $Rot(x)$ is the (3×3) rotation matrix associated to the rotation vector x. Any direction is equally likely, which means that $p(\phi)$ is a uniform PDF. Let's chose $\phi \in [0, \pi[$, the following Gaussian sum is proposed to approximate $p(\phi)$: each Gaussian has a standard-deviation σ_ϕ and meets its two neighbours at $-k_{\sigma_\phi}\sigma_\phi$ and $+k_{\sigma_\phi}\sigma_\phi$ (0 and π are also considered as neighbours).

$$p(\phi) \approx \sum_{0 \leq j < n_\phi} w_j \Gamma(\phi_j, \sigma_\phi)$$
$$\phi_j = (1+2j)k_{\sigma_\phi}\sigma_\phi \qquad w_j = 1/n_\phi$$

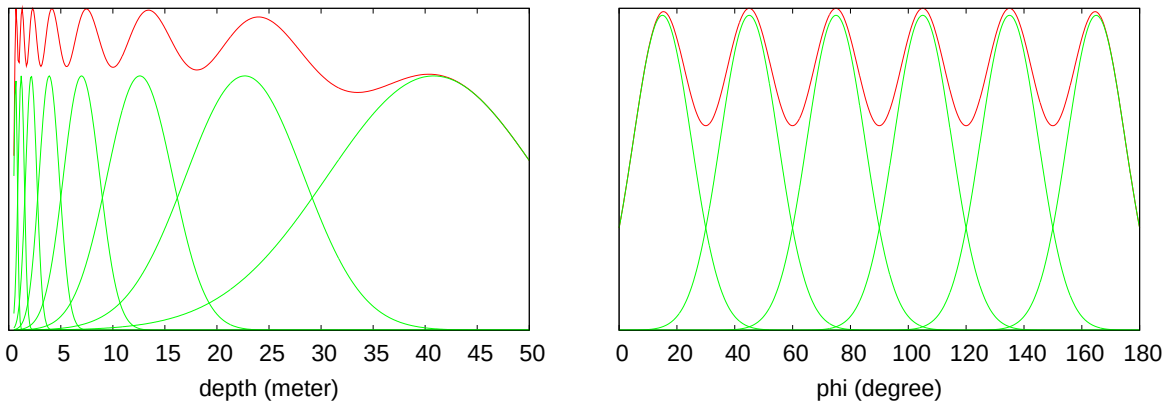

Figure 4.3: Left: geometric sum of Gaussian in the range $[0.5, 50]$ with $\alpha_d = 0.25$ and $\beta_d = 1.8$. Right: uniform sum of Gaussians in the range $[0, 180]^\circ$ with $\sigma_\phi = 10^\circ$ and $k_{\sigma_\phi} = 1.5^\circ$

The sum is composed of n_ϕ Gaussians:

$$n_\phi = \pi/(2.k_{\sigma_\phi}.\sigma_\phi)$$

Figure 4.3-right illustrates this sum.

Approximation of $p(L_{/C})$. The Plücker coordinates of a sample ${L_{/C}}^{i,j}$ are now clarified: the lower part $\underline{u}^{i,j}$ is trivially obtained from $\underline{u}_{\phi_j}$. The upper part is obtained with equation (4.2):

$$n^{i,j} = h^{i,j}\underline{n} = (d_i\underline{g}) \wedge \underline{u}_{\phi_j}$$

The following relation is true by definition of $\underline{g}$:

$$\underline{n} = \underline{g} \wedge \underline{u}_{\phi_j}$$

So we are left with the following equation derived from the cross product:

$$h^{i,j} = d_i \sin\phi_j$$

Which gives for $L^{i,j}_{/C}$:

$$L^{i,j}_{/C} = \begin{pmatrix} (d_i \sin\phi_j).\underline{n} \\ Rot(-\phi_j.\underline{n}).\underline{g} \end{pmatrix}$$

The original stochastic variables are the observation $(\rho, \theta)^t$ and the sampling variables over depth and direction. The following function can be formulated:

$$L^{i,j}_{/C} = pluckerInit(\rho, \theta, d_i, \phi_j)$$

Its Jacobian is also computed so that the usual uncertainties propagation equation can be applied to compute the covariance matrix $P_{L^{i,j}_{/C}}$

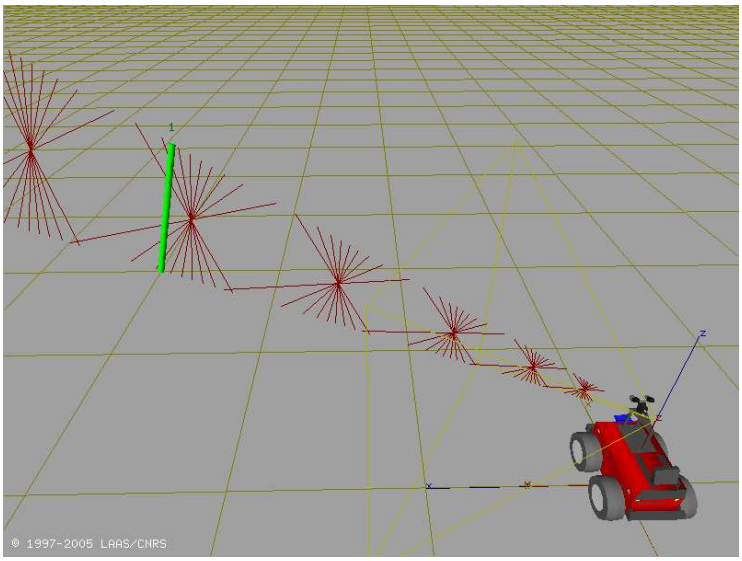

Figure 4.4: 3D view of the hypotheses in red, and the real segment in green.

Finally, $p(L_{/C})$ is approximated with a Gaussian sum of $n_d \times n_\phi$ members.

$$p(L_{/C}) \approx \sum_{\substack{0 \leq i < n_d \\ 0 \leq j < n_\phi}} w_{i,j} \Gamma(L^{i,j}_{/C}, P_{L^{i,j}_{/C}}) \qquad w_{i,j} = \frac{1}{n_d n_\phi}$$

Figure 4.4 illustrates the set of generated hypotheses in 3D.

Once the set of Gaussian hypotheses is defined, the computation of the likelihoods and the selection process is similar to the one presented in chapter 2. The best hypothesis is converted to the map frame (equation (4.4)) and added to the stochastic map. Past observations are then used to update the map.

4.4 Estimation process

As in chapter 2, the core of the estimation process is the Kalman filter. This section focuses on two additional problems which have to be solved:

- computation of the innovation in the (ρ, θ) space,
- consideration of the Plücker representation constraints.

4.4.1 Innovation

It is a key value in the Kalman update process It is given by: $y = z - \hat{z}$ where z is the actual observation and $\hat{z}$ is the observation predicted with the current robot pose and map estimate. In the case of 2D lines observations (ρ, θ), a problem arises when the observed

$$z = \begin{pmatrix} r_1 \\ a \end{pmatrix} \quad \hat{z} = \begin{pmatrix} r_2 \\ b \end{pmatrix} \quad z^* = \begin{pmatrix} -r_1 \\ c \end{pmatrix}$$

$$z - \hat{z} = \begin{pmatrix} r_1 - r_2 \\ a - b \end{pmatrix}$$

$$z^* - \hat{z} = \begin{pmatrix} -(r_1 + r_2) \\ a - c \end{pmatrix}$$

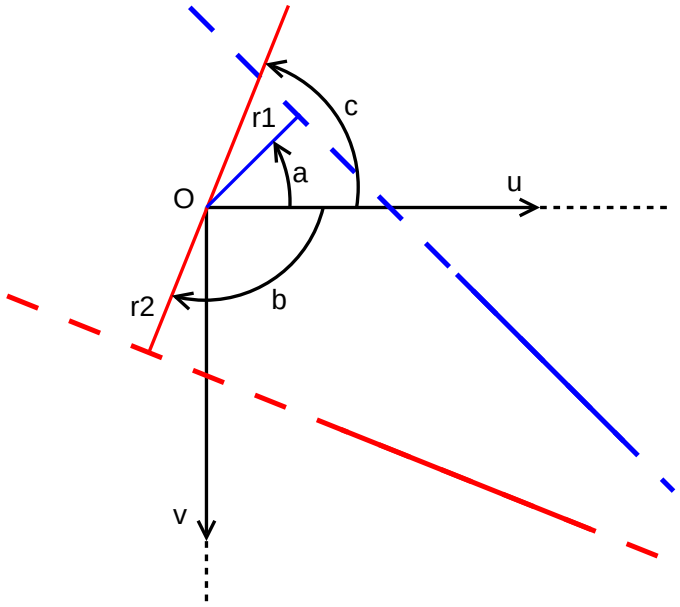

Figure 4.5: In the image plane (O, u, v), the observation z (in blue) and the expected observation $\hat{z}$ (in red).

line and the predicted line are on both sides of the origin of the image frame. In that case, the observation is artificially modified (z^*) and expressed with a negative ρ value so that the innovation reflects the correct error and the angle θ is modified accordingly:

$$y = z^* - \hat{z} \qquad z^* = (-\rho, \theta \pm \pi)^t \qquad (\theta \pm \pi) \in [-\pi, \pi]$$

Without taking care of this problem, the filter would dramatically diverge since corrections are applied in the opposite direction !

4.4.2 Constraints

A 3D line represented with the 6-dimension Plücker vector $L = (n, \underline{u})^t$ must respect two constraints so that it represents a valid line (section 4.2.1):

$$\begin{cases} ||u|| = 1 & \text{(normalisation)} \\ n \cdot \underline{u} = 0 & \text{(Plücker constraint)} \end{cases}$$

All the hypotheses obtained with the procedure of section 4.3 do meet these two constraints by construction. But once an hypothesis has been chosen and added to the stochastic map, there is no guarantee that the Kalman updates will not break the constraints.

An interesting work on that topic can be found in [Bar03b], it is called the "Plücker correction". Given a 6-dimension vector which does not satisfies the Plücker constraint, the algorithm finds the vector which fulfils this constraint and minimises a special correction criterion. This method is quite complex and is not adapted to the stochastic representation.

More simply, u can be normalised at each step of the filter, or after each update of the line. This method has already been successfully applied for orientation quaternion in [Dav03]. But it is impossible to enforce the Plücker constraint that way.

In the case of a linear Kalman filter, with a linear constraint C applying to the state vector x ($C.x = c$), the solution is trivial. The constraint is exactly enforced when updating the filter once with observation c, null observation noise and observation matrix C. The state gets correlated so that subsequent updates do not break the constraint C.

But in the case of an extended Kalman filter, with non-linear constraints, the problem is much more tedious. The simple approach has been applied using linearised constraints, for example in [New99] within the Geometric Projection Filter (GPF) filter. But poorly linearised constraints can add a large base-point error and lead to divergence of the filter. The work by De Geeter *et al.* in [GBSD97] presents a *smoothly constrained Kalman filter*. This work covers strong and weak nonlinear constraints. The Plücker constraint is a strong constraint since it has to be enforced *exactly*. The constraint is *smoothly* applied to the state vector: instead of applying the constraint once with null noise (case of a strong constraint), the constraint is applied several times with an added *weakening* noise.

When the line L is added to the stochastic map, its estimate $\hat{L}$ respects the Plücker constraint. Then, past observations are used to update the map. After these updates, it is likely that $\hat{L}$ breaks the Plücker constraint. At this point, L is considered to be sufficiently updated so as to start the constraint update process (this is our *start criteria* as defined in [GBSD97]). The initial weakening value ξ_0^w is given by:

$$\xi_0^w = \alpha_c . C \hat{P}_L C^t$$

Where C is the (1×6) Jacobian of the Plücker constraint computed at $\hat{L}$, and $\hat{P}_L$ is the covariance matrix of $\hat{L}$. The value of α_c is empirically chosen according to simulation tests.

The constraint updates are interlaced with the usual observation updates. In the case of a strong constraint, constraint updates are triggered by the test $s^c < th_c$ where s^c is the *relative strength* defined in [GBSD97]. s^c measures how well $\hat{P}_L$ respects the correlations induced by the Plücker constraint:

$$s^c = \frac{\max_i C_{0i} . \hat{P}_{Lii} . C_{0i}}{C \hat{P}_L C^t}$$

The weakening values of the consecutive updates are obtained with the following formula, as suggested in [GBSD97]:

$$\xi_i^w = \xi_0^w \exp^{-nc}$$

Where nc is the number of times the constraint has already been applied.

4.5 Simulation tests

We ran the algorithm in simulation in order to tune the parameters, evaluate the constraints application and check the filter consistency. The simulation environment contains eight

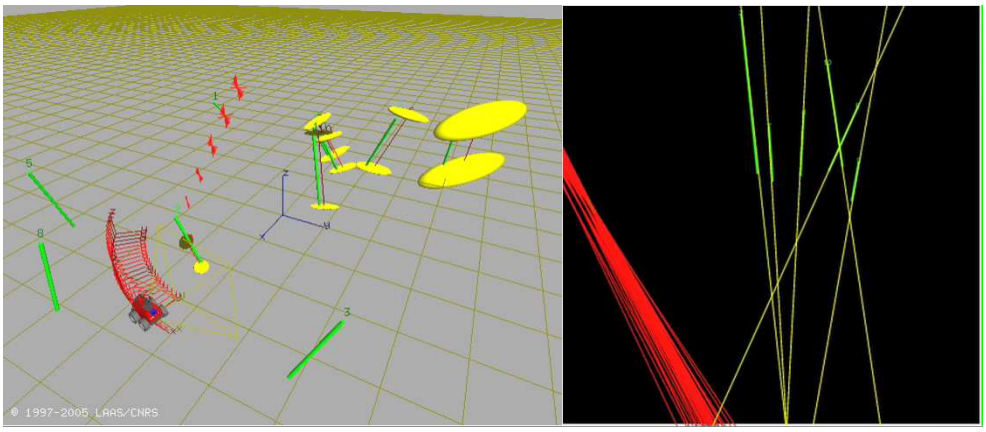

Figure 4.6: Illustration of the simulation environment.

segments and the robot moves along a circle with a diameter of 10 meters. A Gaussian noise is added to the odometry $(d_s, d_\theta)^t$ with $\sigma_{d_s} = 2.5$ cm.m^{-1} and $\sigma_{d_\theta} = 1°$.m^{-1}. A Gaussian noise with $\sigma = 0.5$ pixel is also added to the observations. This noise is added to the detected extremities $(u_1, v_1)^t$ and $(u_2, v_2)^t$ of the segment and then propagated to the line observation $(\rho, \theta)^t$. The algorithms are implemented in full 3D, but in the simulation the robot is moving on a plane. Figure 4.6 illustrates the simulation environment and a full run can be viewed on the following video: http://www.laas.fr/~tlemaire/download/boSlamSegments.mp4.

4.5.1 Parameters definition

The parameters have all an intuitive meaning, but their effect is not really independent. (β_d, α_d) and $(\sigma_\phi, k_{\sigma_\phi})$ define respectively the Gaussian sums over depth d and direction ϕ of the approximation of the initial PDF of the 3D line:

- α_d and σ_ϕ define the size of each Gaussian: the subsequent linearisation of the observation function must be valid around each member,
- β_d and k_{σ_ϕ} defines the density of Gaussians: each member must not overlap too much with its neighbours so that a single hypothesis remains after a small number of observations.

α_c is the ratio which sets the initial value of the weakening variance for the strong nonlinear Plücker constraint. This ratio is adjusted so that constraint application has no strong effect on overall consistency of the Kalman filter. The threshold th_c which triggers the application of the constraint is set to 100. We found this value is enough, as advised in [GBSD97].

The set of parameters is summarised in table 4.1, their value have been empirically set in simulation.

parameter	description	value
β_d	rate of the geometric series	1.3
α_d	ratio between mean and standard-deviation	0.2
σ_ϕ	standard-deviation of each hypothesis	4°
k_{σ_ϕ}	where 2 consecutive Gaussians meet in a fraction of σ_ϕ	1.3
τ	threshold to prune bad hypothesis	10^{-2}
α_c	initial constraint noise factor	0.1
th_c	threshold on relative strength to trigger constraint application	100

Table 4.1: Summary of the algorithm parameters.

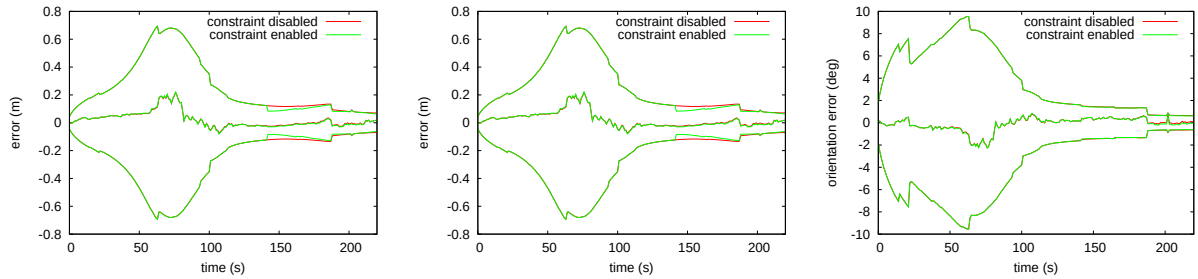

Figure 4.7: Errors and 3-σ bounds of the robot pose (x, y, θ) in a simulation run, comparison with constraint disabled (red) and enabled (green).

4.5.2 Consistency check

Figure 4.7 presents the errors and the 3-σ bounds on the robot pose during a simulation run. This estimate is consistent all along the loop and also when the loop is closed, which is the main source of consistency violation in SLAM. Using the same random seed, in order to obtain the same sequence of noise values, the simulation was run with constraints disabled and enabled. The three plots of figure 4.7 show that no significant difference appears: the application of the Plücker constraint does not affect the filter consistency.

Figure 4.8 illustrates the effect of the constraint update process for feature number 4. The plot of the dot product of the Plücker constraint (left hand-side) shows that it is closer to zero when soft constraint update is applied. The plot of the relative strength (right hand-side) exhibits when the constraint is applied. Just after the landmark is initialised, the relative strength of the Plücker constraint is quite high: this is due to our initialisation method which properly propagates the correlations.

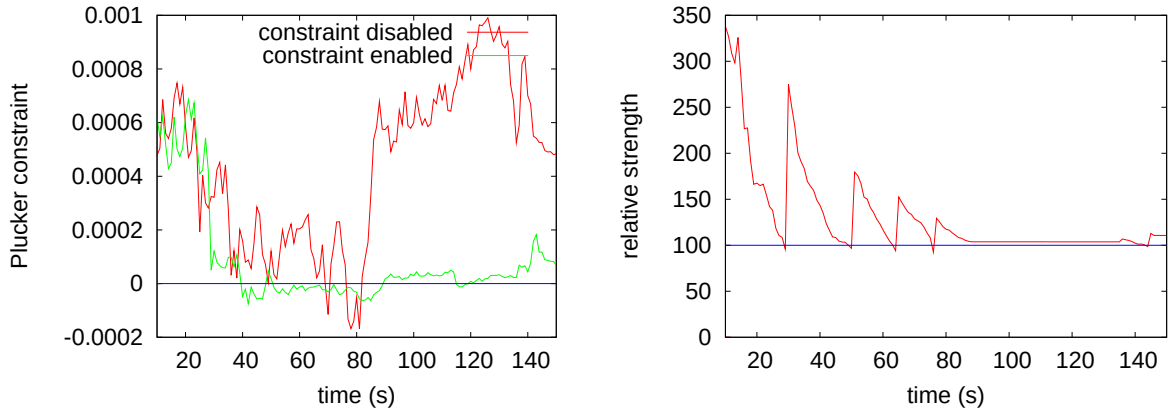

Figure 4.8: Landmark number 4. Left: evaluation of $n.\underline{u}$, comparison with constraint disabled (red) and enabled (green). Right: relative strength of the constraint.

4.6 Experiments with real images

4.6.1 Image segments matching

The segments are extracted in the images according to a classical procedure: first a gradient filter is applied, then the gradient is thresholded and the resulting binary image is structured into *contours*, that links neighbouring high gradient pixels. A line fitting process is then applied, yielding an image of line segments (figure 4.9).

As can be seen on figure 4.9, the image noise strongly influences the segment extraction process: even for images acquired from very close positions, some segments are not repeated, and some are extracted with very different extremities – not to mention long segments that are split in two shorter segments. As a consequence, segments can hardly be matched on the basis of the coordinates of their extremities. To ensure robust and reliable segment matches, we rely the same Harris interest points matching algorithm as in chapter 3: to each segment are associated the closest matched interest points, with a distance threshold very easy to specify. Segment matches are then established according to a hypothesis generation and confirmation paradigm. This simple process has proved to yield outlier-free matches (figure 4.10), even for large viewpoint changes, which is very helpful to associate landmarks when closing loops.

4.6.2 Results

We present results on an image sequence acquired with our ATRV rover. The robot odometry is used to feed the prediction step of SLAM. The robot moves along a circular trajectory with a diameter of 5 meters. The camera is looking sidewards to the centre of the circle where two boxes have been put. In order to reconstruct the boxes edges, only segments within the blue rectangular are considered (figure 4.11-right). After half a circle, the 3D model contains 5 segments which are shown on figure 4.11-right. The figure indicates the

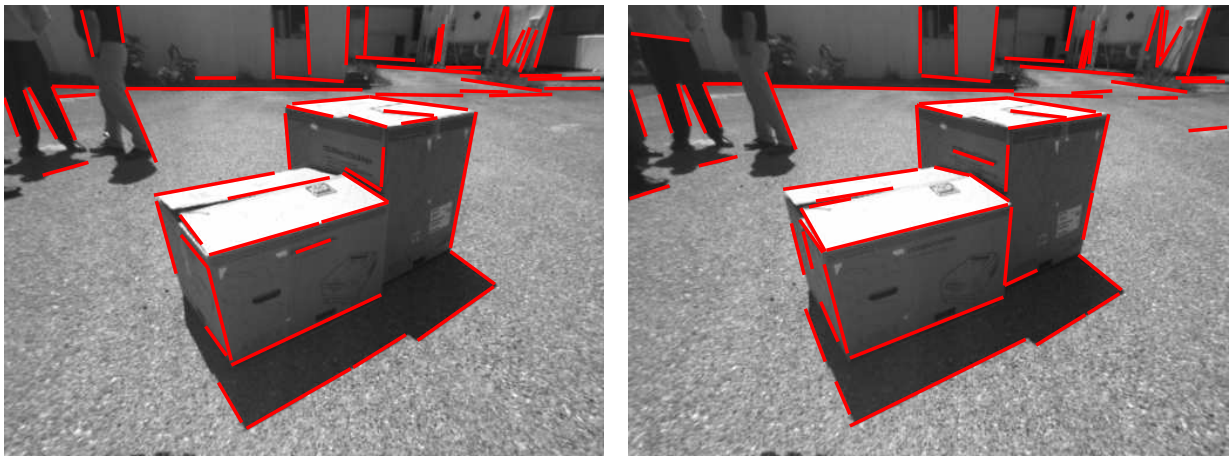

Figure 4.9: Segments extracted in two consecutive images. Even though the viewpoints are close, some segments are not detected in both images, and some are detected with varying length.

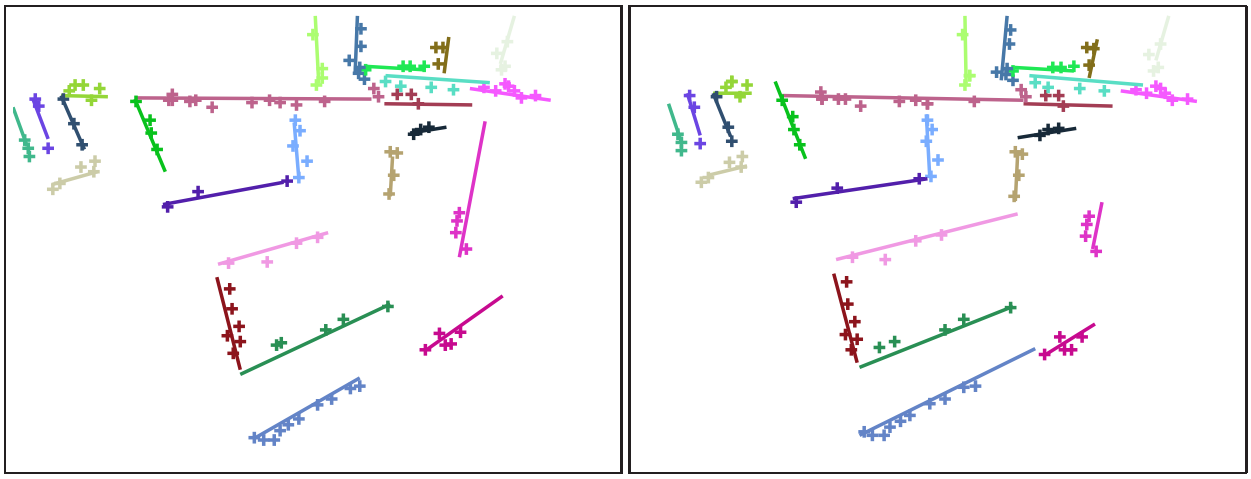

Figure 4.10: Segments matches established for the segments shown figure 4.9. The corresponding interest points are displayed, and matching segments are drawn with the same color.

id	n_x	n_y	n_z	u_x	u_y	u_z
1	1.597	-0.266	-0.083	0.051	-0.004	0.999
2	1.790	-0.117	0.059	-0.031	0.034	0.999
3	2.189	-0.563	-0.020	0.014	0.019	0.999
4	-0.268	-0.228	1.890	-0.662	0.749	-0.004
4'	-0.226	-0.246	1.877	-0.659	0.752	0.019

Table 4.2: Plücker coordinates of the supporting line of the 5 landmarks

edges to which they corresponds. Segment 1,2 and 3 are vertical edges, and segments 4 and 4' actually model the same horizontal edge. It can be verified on table 4.2 that segments 1,2 and 3 are close to vertical ($u_z \approx 1.0$), also segments 4 and 4' are close to horizontal ($u_z \approx 0.0$), and that they have nearly the same supporting line since their Plücker coordinates are very close. Moreover, the angle between segment 3 and 4 which forms a corner of the right box is 88.61° with a standard deviation of 1.75°, which is consistent with the expected value of 90°.

An other 3D model is shown figure 4.12. Here the robot moved along a circular trajectory around bigger boxes.

4.7 Discussion

Constraints application. In the light of simulations, it appears that the application of the Plücker constraint is not absolutely necessary. Numerical values of figure 4.8 shows that the constraint is naturally verified with an acceptable precision. This is due to the initial state cross correlations which are computed in the initialisation procedure. The Jacobian of the $plockerInit(\rho, \theta, d_i, \phi_j)$ function does create dependencies among the Plücker vector variables. It helps to hold the constraint during the Kalman updates which are driven by these correlations. Nevertheless, this has to be verified in long terms experiments.

3D lines observability. Depending on the trajectory of the robot, some landmarks are never initialised because the observations do not bring enough information on the depth of the landmark. This is the case of points in located in front of the robot when using point features. When using segment features, the space of uninitialised landmarks has an additional dimension. When the camera is moving within a plane, which is mostly the case in our simulations and experiments, all the 3D segments lying in that particular plane cannot be initialised. This is not a drawback of our algorithm but a geometric fact. Notice that the algorithm presented in section 2.4.2 on page 26 to cope with low observability cases is general with respect to the kind of feature and works equally for points and segments.

Segments at infinity. The approach for points at infinity presented in 2.4.3 on page 29 is extended to line segments. The *baseline* of a segment observation and the segment at infinity representation are to be defined.

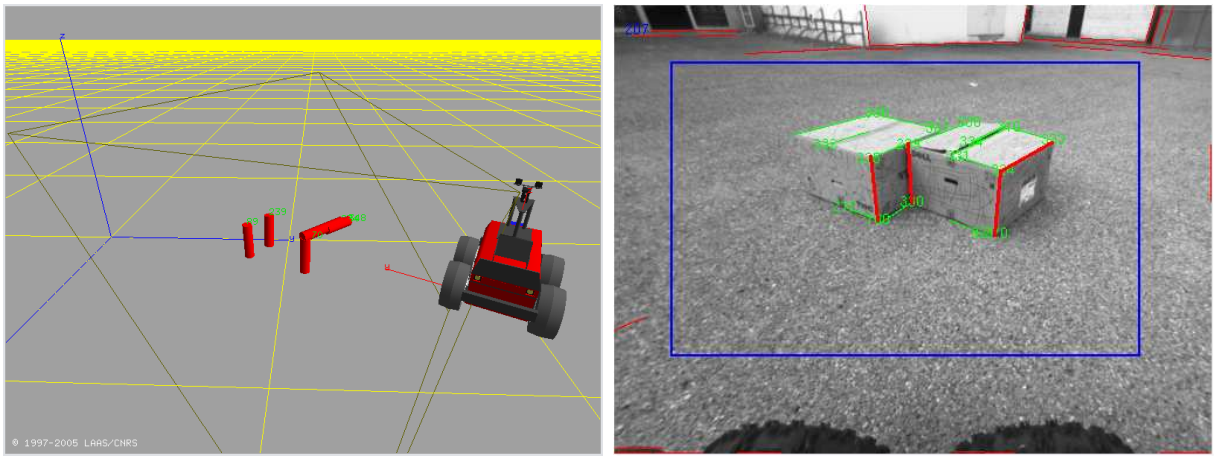

Figure 4.11: Left: 3D model with 5 segments. Right: one image taken by the robot, in green the matched segments used for SLAM, in thick red the corresponding segments of the map. The segments are numbered from the left 1,2,3,4 and 4'

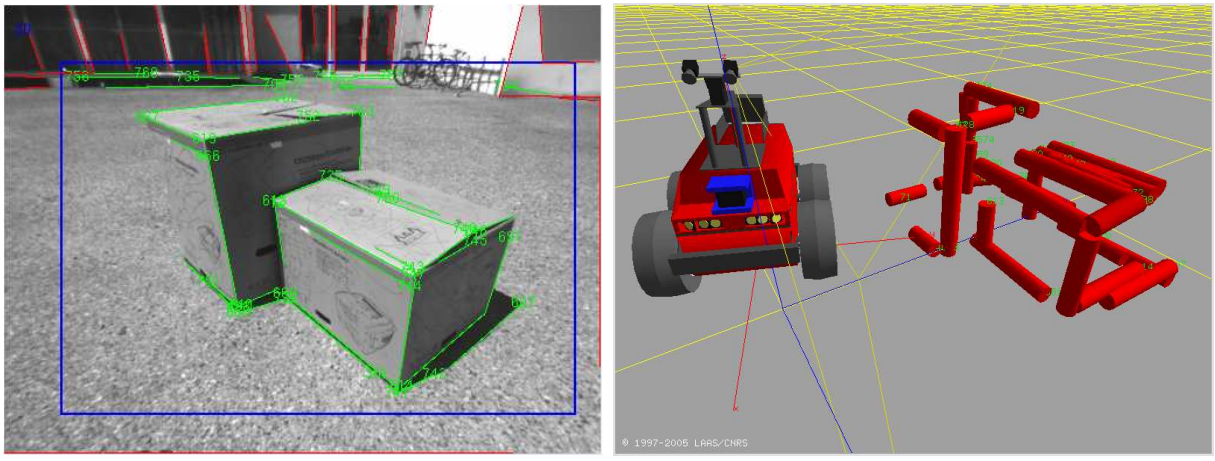

Figure 4.12: Left: an image of the sequence. Right: the 3D model from approximately the same view point.

The line observation z_1 at position x_1 defines a plane with normal n_1. The baseline b which is gained at position x_2 is simply the geometric distance between x_2 and the plane (x_1, n_1):

$$b = |(x_2 - x_1) \cdot n_1|$$

When enough baseline is gained and the landmark is not initialised, the line segment can be declared as being at infinity.

A line at infinity defines only the normal vector of a plane. This is the $\underline{n}$ of the Plücker coordinates $(h\underline{n}, \underline{u})^t$. The distance h is infinity and the direction $\underline{u}$ is undefined. One can notice that the extremities of the segment cannot be computed here. Since the distance to the segment is not known, the size in pixels of the segment in the image plane does not give any information on its real size in the metric map. The extremities can be represented by two angles a_1, a_2 giving the direction of these extremities in the plane defined by $\underline{n}$. Here, as for usual segments, a_1 and a_2 should not be added to the stochastic map since they are usually not reliably detected in the images.

Panoramic vision. An implementation of a visual SLAM algorithm based on segments extracted from panoramic images would be of great interest. In the general case, the projection of a 3D line in a central panoramic sensor is a conic curve [Bar03a], from which the normal of the plane Π could be recovered using the panoramic camera model: our approach would then easily be adapted.

Note that in the particular case of a robot evolving on a plane in man made environments, vertical segments are radial lines in the panoramic image, and can therefore easily be detected (work by [FB02] propose to use a Hough transformation to detect horizontal and vertical segments in panoramic images). The problem here becomes a simple 2D bearings only problem.

4.8 Conclusion

In this chapter, a vision based SLAM algorithm for line segments has been proposed. The Plücker coordinates are used to represent a 3D line in the stochastic map. This parametrisation is very well adapted to vision since it is associated to a simple observation model. Results on real images are presented to demonstrate the correctness of our approach. Nevertheless the data set is not very large and more work has to be done to implement a more robust segments tracker and integrate it within the SLAM algorithm. Also results with aerial images where the structure of buildings could be recovered would be very interesting.

Chapter 5

Conclusion

5.1 Contributions

In this thesis, several algorithms to tackle the monocular vision SLAM problem have been developed. The focus has been put on building a practical SLAM architecture running on a real robot. The contributions of this thesis are as follows:

(i) A new method for landmark initialisation in bearings-only SLAM is proposed. The performance of boSLAM and the definition of its parameters are analysed in simulations. The issues raised by distant landmarks and poor depth observability are analysed and solved. These problems are often ignored in the literature.

(ii) boSLAM is run on real images acquired with a perspective camera mounted on a ground rover. A simple setup allows the SLAM algorithm to successfully close the loop.

(iii) A more efficient SLAM architecture based on a panoramic camera for feature perception, and on stereo-vision for robot pose prediction is proposed. We take advantage of the panoramic images to propose a two level database: the first level to detect loop-closing, and the second one to match features with landmarks in the map. This setup is successfully demonstrated on real data. Moreover some solutions are proposed to practical issues such as map management which highly impact on the performance of the overall system.

(iv) In order to incorporate segments in the stochastic map, the boSLAM method is extended to 3D lines. The Plücker coordinates are proposed to represent line segments. The algorithm is tested in simulation and results on real images are presented.

5.2 Discussion

The perfect bearings only SLAM algorithm ? Monocular vision based SLAM has been an active research topic in robotics during the last three years. In the light of the

numerous contributions in the domain, here are the most important elements:

- With un-delayed methods [KDH05, SDML05, MCD06] the robot pose is updated as soon as the features are detected. This is of main importance especially when working with a single camera where no prediction command is available.
- Decoupling observation information between a depth measure and an epipolar constraint [ED06b] is definitely a nice idea. Advantages of both delayed and un-delayed algorithms can be exploited. But this cannot be generalised to line segments.
- The physical setup (the orientation of the camera, its field of view) is essential to obtain successful results. This is obvious in our work and also in [DCK04].
- Observability issues must absolutely be tackled. If not, hypotheses selection or map updates are conducted with measures where the observation noise predominates. This can lead to a dramatic divergence of the underlying filter.

Mixing points and segments. Points and segments are complementary features. In fully unstructured environments, mostly points are detected. At the opposite, man made environments often contains untextured objects with only a few corner points, but many sharp edges can be detected in the images. Also both kinds of primitives are complementary for the feature matching and [RD05] proposes a robust points and segments tracker. Generally speaking, the use of different features, and possibly different sensors, would improve the robustness of the overall system.

Large scale SLAM. The vision based SLAM method proposed in chapter 3 is large scale from the perception point of view. The space complexity of the landmarks data base and of the panoramic images database is linear with the size of the environment. Moreover the time complexity of the loop closing detection method is constant when the set of image indexes to be searched is restricted by a rough robot pose estimate.

But from an estimation point of view, the approach is not large scale for two reasons. First, the space and time complexity of the Kalman filter are in $O(N^2)$ where N is the number of estimated landmarks. Second, the important estimation errors inherent to large loop closing are not acceptable for the Kalman filter.

These two limitations can be overcome with the use of a sub-maps approach (for example [ENT05]) where multiple Kalman filters are used to estimate each local map. The size of each local map can be bounded so that the approach is constant time. Also, the space complexity of the algorithm is linear.

5.3 Future research

Vision has only been considered recently in the SLAM community, and most of the contributions tackle the problems raised by the depth information recovery. But the use of vision

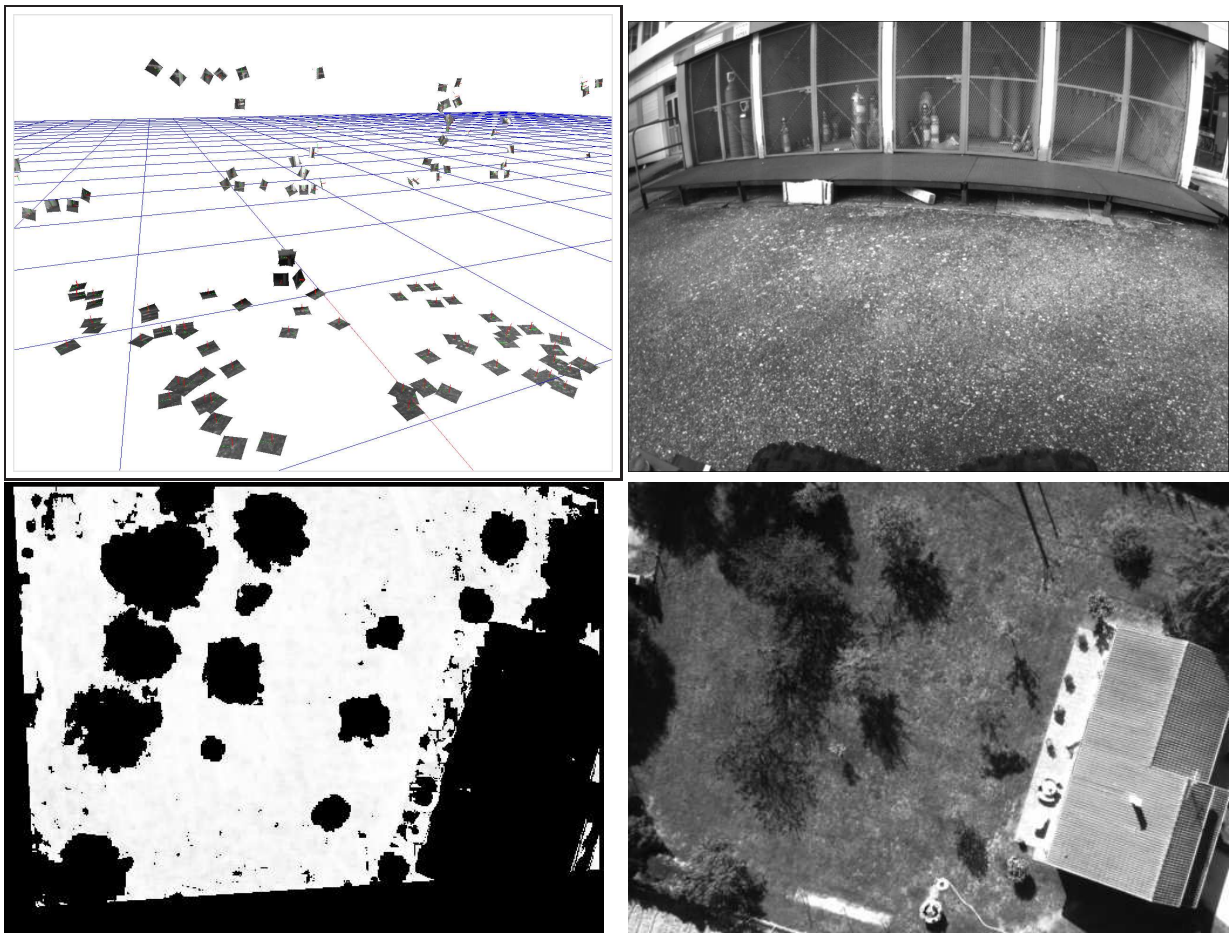

Figure 5.1: Top: Example of facets obtained with stereo-vision, the normals are in red (thanks to Cyrille Berger [Ber05]). Bottom: detection of the predominant plane (in white) from a sequence of monocular images acquired with the blimp Karma (thanks to Sebastien Bosch [BLC06]).

opens several possibilities yet poorly explored. In particular, SLAM systems in which the reconstructed map convey much more information on the structure of the environment than a sparse 3D points map can be developed.

Planar landmarks. A stochastic map which incorporates planar regions of the environment would exhibit meaningful information, and in particular ease the data association process and make it more robust.

A first idea is to use locally planar regions. In [MDR04], the monocular SLAM algorithm presented in [Dav03] is extended for that purpose. A small patch around the salient points is stored and the normal to this surface is estimated, supposing that this surface is planar. In recent work [Ber05] a technique to extract facets from a pair of stereo images is developed. The facets correspond to small regions of the environment which are detected to be planar

by locally fitting an homography around matched interest points in the image pair (see figure 5.1-top). Such facets are potentially good features for SLAM, their normal adding two useful orientation parameters. Similar features could also be extracted with monocular vision, as in [RLSP03].

A different approach is to incorporate large planar regions detected in monocular vision sequences. In [BLC06] the predominant planar region perceived in a sequence of monocular images can be detected (see figure 5.1-bottom). This method is based on homography computations and is independent of any external pose estimate. This information is very valuable for a SLAM algorithm:

- Some features (points or segments) of the plane can be chosen so as to represent it in the stochastic map and to define a local reference frame.
- The region which is detected to be planar is represented in that reference frame (with a set of polygons for instance).
- If the same plane grows, new features can be added to define new visible reference frames and to continue to register planar regions.
- Also the texture of the planar regions can be extracted from the images and added to the map, possibly using "super resolution techniques" [CKK+96] to increase the resolution of the texture.

With facets or planar regions memorised in the map, the data association process could be greatly improved. The textured planar regions of the map can be projected in the predicted image plane: not only the feature coordinates are predicted but also dense parts of the image are computed at a relevant resolution to help the matching algorithm. Also it can avoid to store many images in a data base.

Objects based on segments. A given segment can be detected from many different points of view, but the matching algorithm is challenged by the depth discontinuities which usually occurs near segments: the background on which a segment is perceived may vary a lot. A single segment is difficult to match with the potentially numerous segments extracted in an image.

Several segments together can give a good description of many human-made objects such that buildings. The matching of rigid objects modelled as a set of line segments with edges detected in an image is much more robust. Such techniques have been developed in the vision community ([Low85]). In order to build such maps, segments pertaining to the same object could be grouped together in a single rigid landmark of the stochastic map. This would reduce the size of the map and would help the matching process, allowing the algorithm to match full objects rather than single segments. Note also that segment based models can be a good support to define and represent textured planar regions.

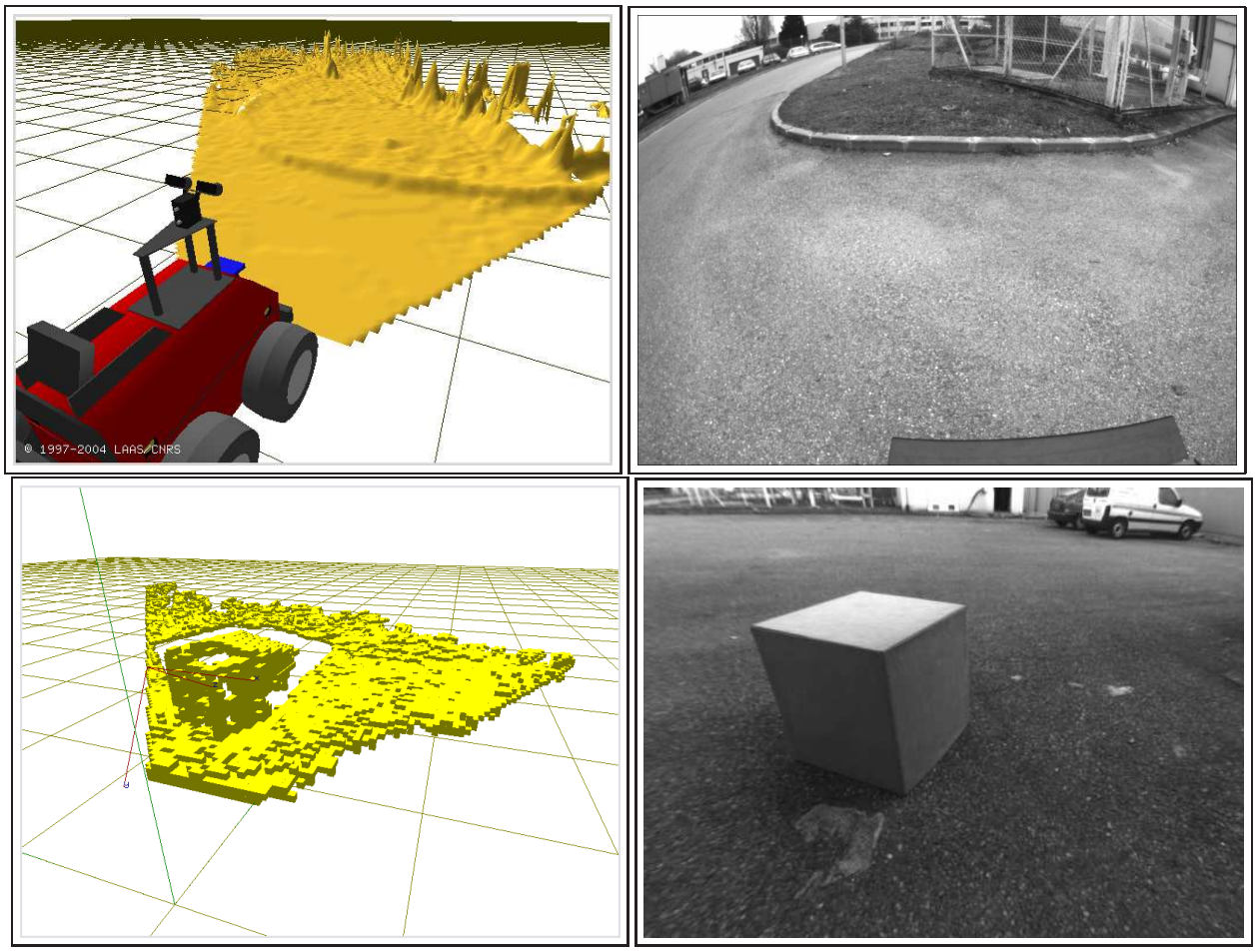

Figure 5.2: Example of dense environment mapping using stereo images. Top: digital Elevation Map (thanks to Thierry Peynot). Bottom: 3D occupancy grid.

Dense mapping. The map produced by a landmark based SLAM algorithm is inherently sparse: it contains a set of points, segments, or any other *ad hoc* objects (pink golf balls. . .). With the noticeable exception of laser based 2D SLAM, these elements are meaningful only for robot localisation. A dense representation of the environment such that a digital elevation terrain is often necessary (see figure 5.2).

Local trajectory generation algorithms (for example [MOM04, BLS01]) only require a map of the close surroundings: a SLAM technique is not absolutely required here. But for long range path planning algorithms (see *e.g.* [GL03a, SH95]) a large scale *spatially consistent* map build thanks to a SLAM algorithm is essential.

In the general case, the stochastic map produced by a SLAM algorithm is not suitable for these tasks. But it provides a sound basis for building relevant representations of the environment. A simple approach consists to add in the stochastic map a reference frame (a robot pose for instance). Then any dense representation can be built with respect to this frame. We end up with a set of frames estimated in a consistent way, and a set of

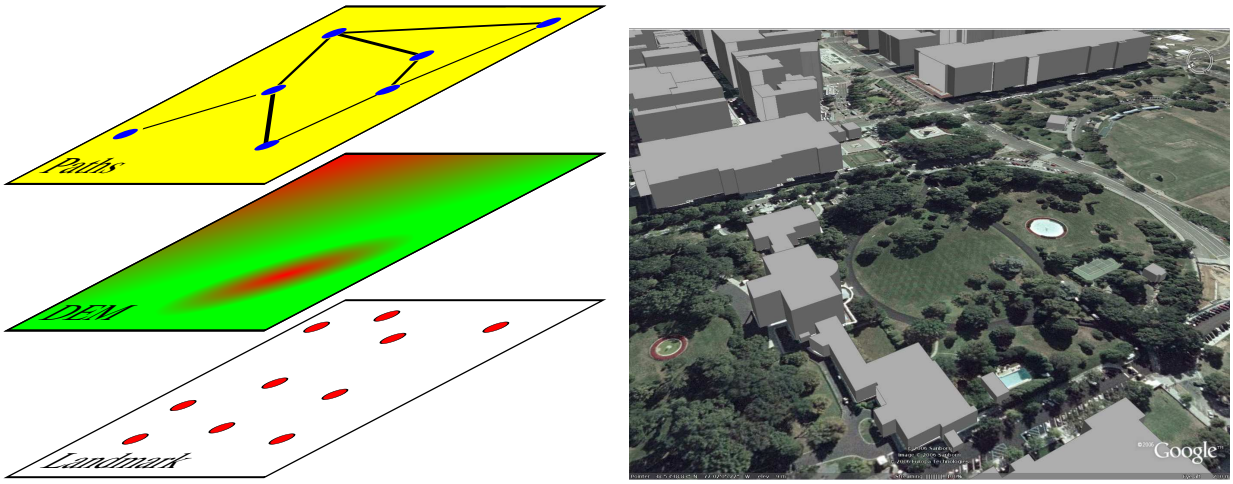

Figure 5.3: Left: representation of a multi-layers map. Right: current Geographical Information Systems already contain high level geometric primitives (here a 3D building model provided by Google Earth http://earth.google.com/).

local dense maps attached to each frame. At any time a partial or global dense map can be obtained by projecting some of these local dense map in a global one. An important issue with this approach is to *fuse* the overlapping regions of the local maps.

A solution to this problem has been proposed in [NGN04]. The landmarks of the stochastic map are used to build "Local Triangular Regions". Each LTR defines a local reference frame used to store dense data. These frames are not orthogonal and are warped according to landmarks corrections. Also the dense data must be obtained with the sensor used to observe the landmarks, or at least landmarks must be detected in the dense data. This approach has been demonstrated in 2D with a laser range finder and is not easily extended to 3D dense data representation.

SLAM with multiple robots. The problem of multi robots SLAM is well understood on the estimation/filtering side [TL03, NTDW03]. State of the art solutions take advantage of the information filter which can efficiently fuse many observations. The problem of the fusion of delayed data has also been tackled [NDW01]. However, to our knowledge, these techniques have not yet been demonstrated in large experiments.

The principal challenges lie on the perception side, the data association is still lacking robust and efficient algorithm to match strong features detected by different sensors, mounted on different robots and with large viewpoint changes. We believe that the mapping of planar regions or rigid objects suggested in the previous paragraph would be very efficient here.

SLAM and Geographical Information Systems (GIS). Considering the various environment models previously sketched (planar regions, segment-based object descriptions, dense models), we end up with an environment model that has the same layered structure

of a usual GIS. The bottom layer is made of the set of landmarks which are consistently estimated by SLAM. The upper layers are the maps containing dense data, or possibly other sparse information relevant for the robots or the mission (see figure 5.3). The only difference is that the layers of a GIS are defined in a single Earth centred reference frame, whereas the layers of the SLAM maps are made of local maps anchored in the bottom stochastic layer.

More and more information are collected about the Earth (and other planets) and stored in Geographical Information Systems. An operational robotic system should not ignore such information: by matching them with acquired data, it can observe its positions, and even refine these information. The problem here is essentially on the data association side. We believe that the development of higher environment models such as planar regions or segment-based descriptions in a SLAM context is a prerequisite to tackle such problems.

Closing the control loop. Finally, if SLAM is required because neither a precise map nor an absolute localisation mean is available, one must not forget that it is integrated in a whole architecture developed for autonomous robots.

Even if the robot mission is restrained to the exploration of its environment, strategies must be developed to autonomously achieve this task. Such a strategy has two goals:

- First, the robot trajectory should try to minimise both its pose estimate uncertainty, and the stochastic map uncertainty. A multi-step trajectory planning is proposed in [HKD+05]: the result is that the robot goes back and force so as to maintain low uncertainty on its pose while exploring new areas and adding landmarks in the map.

- Second, the meaningful dense map must be build as fast as possible. For this task, navigation strategies which maximise the information gain must be developed [GL03b, FAS+02].

A global strategy which would meet these two requirements has to be developed. Moreover, when a sub-maps approach is used to tackle the large scale SLAM problem, the decision of creating a new local map must be taken carefully. Intuitively, this is better if the robot creates a new map when the current one has a low uncertainty. A rule of thumb is to stop doing SLAM when the map is fully correlated. The robot can switch to a localisation algorithm using the freshly built map. This decision can be taken on a per sub-map basis. However, the stochastic map must not be thrown away because it is necessary in case of loop closing occurring at the upper topologic level.

Appendix A

Jafar

http://www.laas.fr/~tlemaire/jafar/

I am one of the main designers of an interactive development framework currently used at LAAS in the robotic groups. Its name is *Jafar*.

Motivations. This kind of tool is of essential importance in a research team. Many different people produce software to test and demonstrate their algorithms. Jafar is the basis which supports the collaborative efforts of several researchers to build software for autonomous robots. It is a solid development framework which provides standard data types and algorithms focused on image processing and robotics. It also integrates an up-to-date documentation to help new users, and developers can easily integrate documentation for their own module.

Overview. The libraries are written in C/C++ and are automatically made available to an interactive shell *ala* Matlab. Currently Tcl/Tk (http://www.tcl.tk/) and Ruby (http://www.ruby-lang.org) interactive languages are supported. The two layers approach adopted by Jafar allows the development of libraries in C/C++ and the rapid testing of their functionalities. The glue between the libraries and the interactive shell is generated using swig (http://www.swig.org). The full documentation integrates both documentations for the framework itself and the modules. It is generated by Doxygen (http://www.doxygen.org) which provides hyperlinks and a useful "search" functionality. Figure A.1 shows the mechanism of a Jafar module. Jafar is the descendant of *Calife* which was also built with this two layers approach.

Jafar does not reinvent the wheel and takes advantage of many Open Source tools. It is itself published under a BSD-like licence. It is part of the *Open Robots* project and is integrated with the other LAAS tools (http://www.openrobots.net).

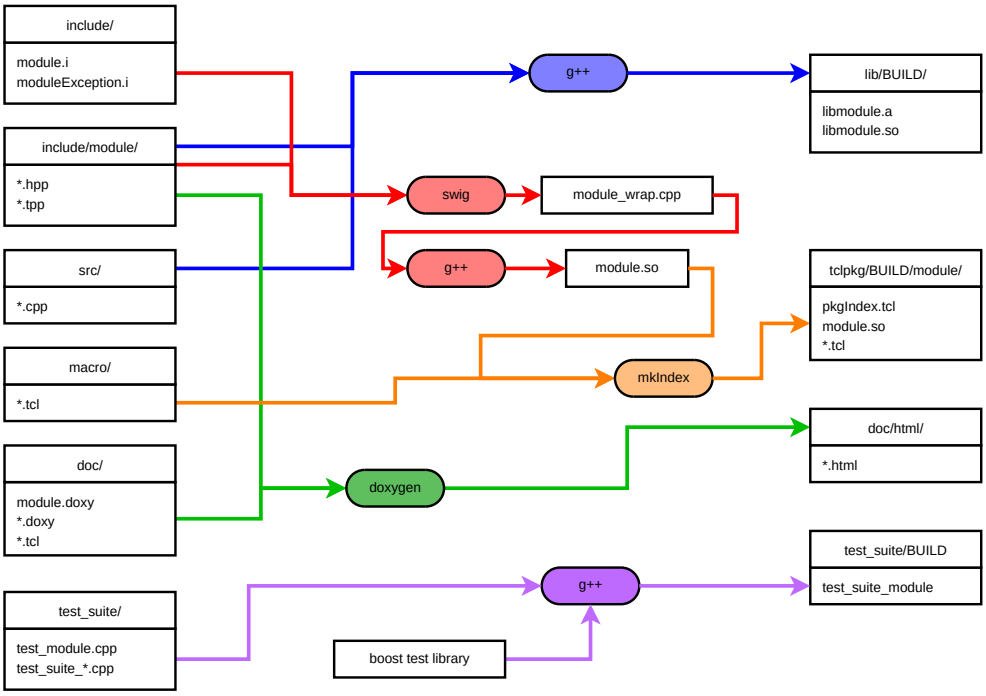

Figure A.1: Structure of a Jafar module. Left: the developers can edit these files. Right: the elements produced by Jafar thanks to several tools.

Jafar is now 2 years old, 27 modules have been contributed by more than 10 Ph-D students, post-doc or master students, the subversion repository contains more than 2000 revisions.

http://www.laas.fr/~tlemaire/jafar/

Appendix B

Visual Motion Estimator

B.1 Problem statement and notation

Given a set of 3D points p_i acquired at time k and a set p'_i acquired at time $k+1$ and given a set of pairs of corresponding points $q_i = [p_i, p'_i]$, the 3D transformation (R, t) between frame k and frame $k+1$ is estimated. The following least-squares problem is considered to find optimat rotation matrix R and translation vector t:

$$\epsilon^2 = \frac{1}{\sum w_i} \sum_i w_i \left\| p_i - (R.p'_i + t) \right\|^2$$

w_i are the weights of each 3D point pair.

The etimated transformation (R, t) is usually given in a more compact and convenient vectorial representation[1], noted T. Pariance matrix of 3D points p, p' is noted $P_p, P_{p'}$, and covariance matrix of T is P_T.

B.2 Least-squares minimisation

The Least-squares problem is solved using the solution described in [HJL+89] and extended in [Ume91]. In [HJL+89], each 3D point pair is weighted but the solution for R can be a reflection and not a rotation, in [Ume91] this singularity is solved but the solution is presented whithout weighted points.

Rotation and translation estimation are decoupled by computing the 3D points cloud barycenters:

$$\bar{p} = \frac{1}{\sum w_i} \sum w_i.p_i \qquad \bar{p}' = \frac{1}{\sum w_i} \sum w_i.p'_i \qquad t = \bar{p} - R.\bar{p}'$$

The p' p covariance matrix is:

$$P_{p'p} = \frac{1}{\sum w_i} \sum w_i (p_i - \bar{p})(p'_i - \bar{p}')^T$$

[1] Euler, Quaternion,...

The Singular Value Decomposition gives:

$$P_{p'p} = UDV^T$$

and

$$R = USV^T \qquad S = \begin{cases} I & \text{if } det(U)det(v) = 1 \\ diag(1,1,-1) & \text{if } det(U)det(v) = -1 \end{cases}$$

B.3 Uncertainties computation

The work in [Har94] gives the generic mathematical for first order uncertainty propagation in least-squares problems. With the function to be minimized:

$$F(Q,T) = \sum_i f_i(q_i,T)$$
$$\frac{\partial F}{\partial T}(Q,T) = G(Q,T) = \sum_i g_i(q_i,T)$$

It comes:

$$\begin{aligned} P_T &= \frac{\partial G}{\partial T}^{-1} \frac{\partial G}{\partial Q} P_Q \frac{\partial G}{\partial Q}^T \frac{\partial G}{\partial T}^{-1} (\hat{Q},\hat{T}) \\ &= \frac{\partial G}{\partial T}^{-1} \left[\sum_i \bar{w_i} \left(\frac{\partial g_i}{\partial q} P_{q_i} \frac{\partial g_i}{\partial q}^T \right) \right] \frac{\partial G}{\partial T}^{-1} (\hat{Q},\hat{T}) \\ &= \frac{\partial G}{\partial T}^{-1} \left[\sum_i \bar{w_i} \left(\frac{\partial g_i}{\partial p} P_{p_i} \frac{\partial g_i}{\partial p}^T + \frac{\partial g_i}{\partial p'} P_{p'_i} \frac{\partial g_i}{\partial p'}^T \right) \right] \frac{\partial G}{\partial T}^{-1} (\hat{Q},\hat{T}) \end{aligned}$$

with

$$\frac{\partial G}{\partial T} = \sum_i \bar{w_i} \frac{\partial g_i}{\partial T}(\hat{q_i},\hat{T}) \qquad \bar{w_i} = \frac{w_i}{\sum w_i}$$

Appendix C

Real-time implementation

Overview. An on-board demonstration of the whole boSLAM machinery using a panoramic camera described section 3.4 on page 43 is being developed. The robot will be able to execute a basic "goto" task, the idea is to demonstrate the ability of SLAM to localise the robot in a natural environment.

The functional modules. The on-board software components are developed with the GenoM framework. GenoM stands for "Generator of Modules", the developer must provide:

- a description of the services (the *requests*),
- a description of the output data (the *posters*),
- and an implementation of the elementary functions (the *codels*) of the module.

GenoM generates a set of libraries and executables. A module is executed in its own process and then the requests can be started. The set of modules used in this demonstration is presented figure C.1, only VME and SLAM modules are being developed by ourselves.

GenoM is freely available at `http://softs.laas.fr/openrobots/`.

The remote station. The robot can be controlled and supervised from a remote station. A preliminary view of the graphical user interface is shown figure C.2. Several elements are displayed:

- the current panoramic image, with an overlay of tracked and matched features,
- the current left stereo image, with an overlay of tracked points,
- a 3D view of the current map with landmarks, robot trajectory,...

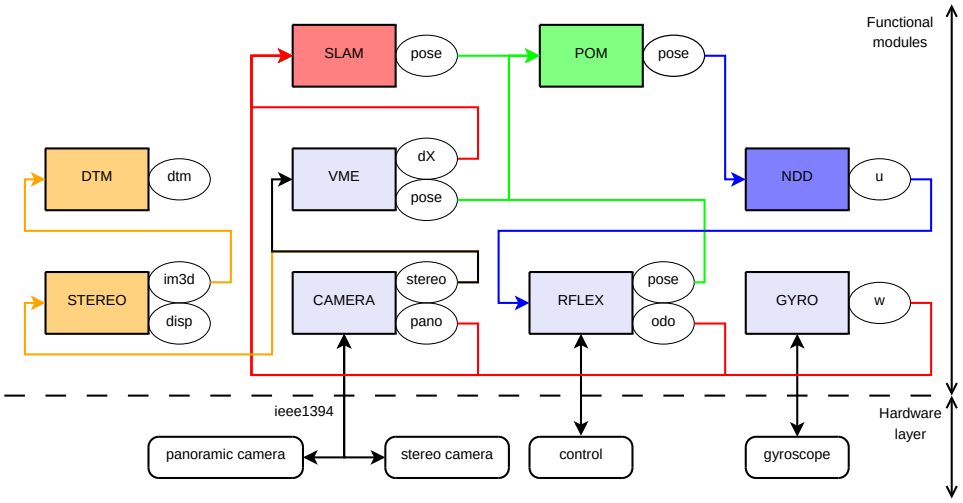

Figure C.1: Functional modules architecture.

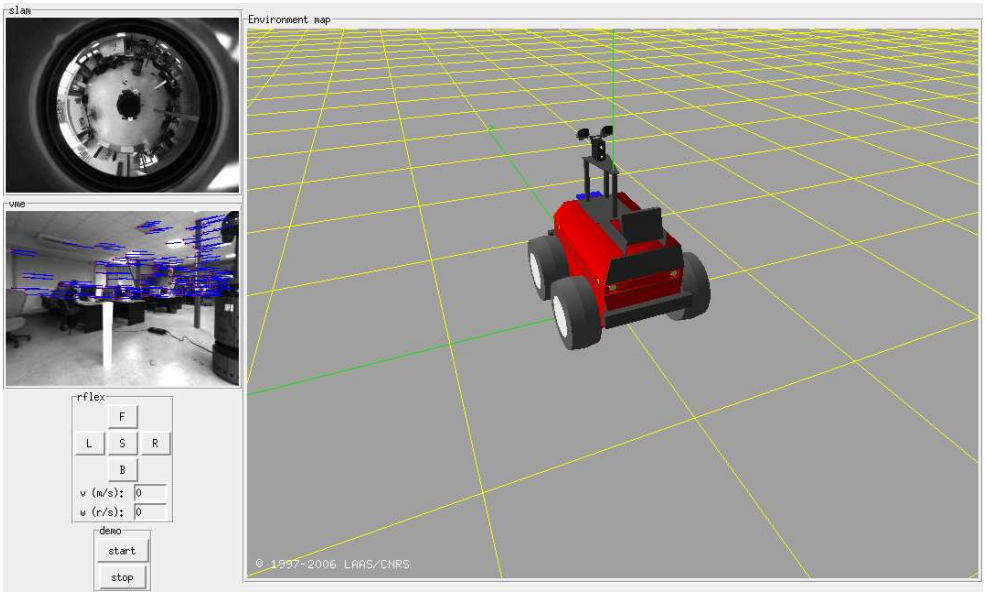

Figure C.2: Overview of the graphical user interface.

Appendix D

Plücker line representations

D.1 3D line representation using euclidian Plücker parameters

$$L = \begin{pmatrix} cn = h.\underline{n} \\ \underline{u} \end{pmatrix}$$

$\underline{n}$ is the normal to the plane containing the line and the origin, h is the distance between the origin and the line, $\underline{u}$ represents the direction of the line. Any point P on the line satifies the relation:

$$P \wedge \underline{u} = n$$

P_o, the closest point to the origin, is given by:

$$P_o = \underline{u} \wedge n$$

D.2 Applying transformation (R, t)

Matrix M_l is defined such that the representation of line L is $L_1 = M_l L_0$ after transformation (R, t).

$$\begin{pmatrix} cn_1 \\ \underline{u}_1 \end{pmatrix} = M_l \begin{pmatrix} cn_0 \\ \underline{u}_0 \end{pmatrix}$$

$\underline{u}$ is a simple direction, so the transformation is obvious:

$$\underline{u}_1 = R\underline{u}_0$$

Given P a point on the line, we have the relation:

$$\begin{aligned} n_1 &= P_1 \wedge \underline{u}_1 \\ &= (RP_0 + t) \wedge (R\underline{u}_0) \\ &= (RP_0) \wedge (R\underline{u}_0) + t \wedge (R\underline{u}_0) \\ &= Rn_0 + [t]_\wedge R\underline{u}_0 \end{aligned}$$

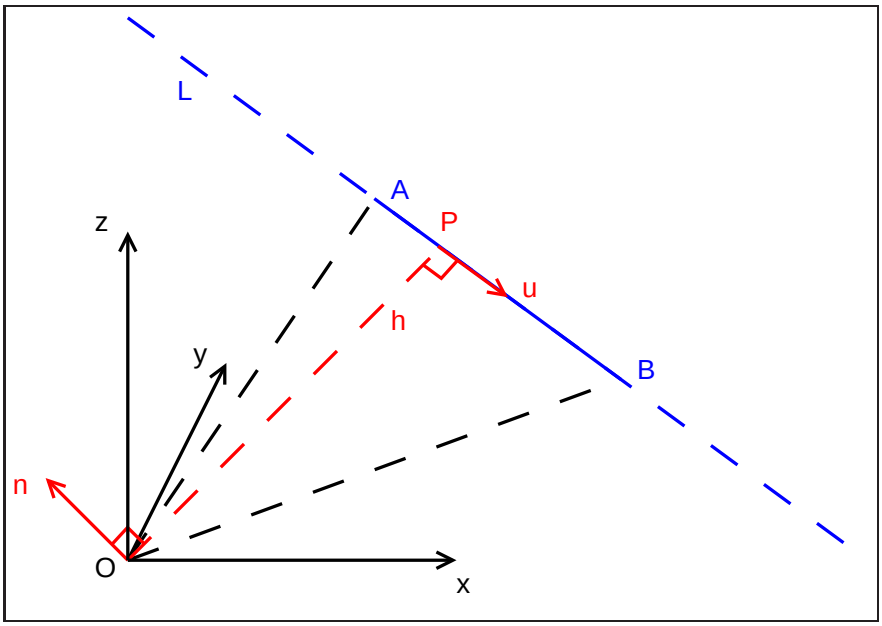

Figure D.1: Presentation of the Plücker coordinate $(h.\underline{n}, \underline{u})^t$ of the 3D line L

Putting it all together in the matrix M_l leads to:

$$M_l = \begin{bmatrix} R & [t]_\wedge R \\ 0_{(3\times3)} & R \end{bmatrix}$$

D.3 Projection through a pinhole camera model

Given the line L $(n, \underline{u})^t$ in the camera frame, a point P $(x, y, z)^t$ of line l, projection of L in the image plane satisfies: (i) to be on the image plane, (ii) to be on the plane defined by the line L and the origin of the camera frame:

$$\begin{cases} z = 1 \\ P \cdot n = 0 \end{cases}$$

Which leads to the following canonical line equation for l:

$$n_1 x + n_2 y + n_3 = 0$$

To compute the line equation in the image frame, we use the following from the camera pinhole model:

$$\begin{cases} x = \frac{(u-u_0)}{\alpha_u} \\ y = \frac{(v-v_0)}{\alpha_v} \end{cases}$$

Making the substitution of this in the previous line equation leads to:

$$\alpha_v n_1 u + \alpha_u n_2 v - \alpha_v u_0 n_1 - \alpha_u v_0 n_2 + \alpha_u \alpha_v n_3 = 0$$

And the following matrix expression can be written:

$$l = \begin{pmatrix} \alpha_v & 0 & 0 \\ 0 & \alpha_u & 0 \\ -\alpha_v u_0 & -\alpha_u v_0 & \alpha_u \alpha_v \end{pmatrix} \begin{bmatrix} 1_{(3\times3)} & 0_{(3\times3)} \end{bmatrix} L$$

D.4 2D line representation

The 3D lines are observed using a camera, an observation is then a line in 2D. The canonical representation of a 2D line is:

$$ax + by + c = 0 \quad l = (a, b, c)^t$$

Vector $(-b, a)$ is colinear to the line. We can normalize this representation in different way, for example:

$$\text{with } a^2 + b^2 = 1 \quad l = \frac{1}{\sqrt{a^2 + b^2}} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ -\rho \end{pmatrix}$$

A minimal representation can be deduced, $(\rho, \theta)^t$, ρ is the distance from the line to the origin, and θ gives the orientation of a perpendicular to the line.

Bibliography

[AH83] V.J. Aidala and S.E. Hammel. Utilization of modified polar coordinates for bearing-only tracking. *IEEE Trans. Automatic Control*, 28(3):283?294, March 1983. 2.5.2

[AHB87] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3D point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698–700, 1987. 3.4.2

[ATD05] Henrik Andreasson, André Treptow, and Tom Duckett. Localization for mobile robots using panoramic vision, local features and particle filter. In *IEEE International Conference on Robotics and Automation (ICRA 2005)*, 2005. Available from: http://aass.oru.se/~han/papers/ha_at_td_icra05.pdf. 3.4

[Bai03] Tim Bailey. Constrained initialisation for Bearing-Only SLAM. In *IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, September 2003. Available from: http://www.acfr.usyd.edu.au/publications/downloads/2003/Bailey206/bearing_only_constrained.pdf. 2.1

[Bar03a] Joao Pedro de Almeida Barreto. *General Central Projection Systems Modeling, Calibration And Visual Servoing.* PhD thesis, University of Coimbra, 2003. Available from: http://www.isr.uc.pt/~jpbar/Publication_Source/phd_thesis.pdf. 3.4.1, 4.7

[Bar03b] Adrien Bartoli. *Reconstruction et alignement en vision 3D : points, droites, plans et caméras.* PhD thesis, GRAVIR, 2003. Available from: http://www.lasmea.univ-bpclermont.fr/ftp/pub/bartoli/Thesis.pdf. 4.4.2

[BDW06] Tim Bailey and Hugh Durrant-Whyte. Simultaneous Localisation and Mapping (SLAM): Part II - State of the Art. *Robotics and Automation Magazine*, September 2006. Available from: http://www.acfr.usyd.edu.au/homepages/academic/tbailey/papers/slamtute2.pdf. 1.2.3

[Ber05] Cyrille Berger. Construction d'un modèle de l'environnement pour la localisation d'un robot mobile. Master's thesis, Université Pierre et Marie Curie, 2005. 5.1, 5.3

[BLC06] Sébastien Bosch, Simon Lacroix, and Fernando Caballero. Autonomous detection of safe landing areas for an uav from monocular images. In *IROS*, 2006. Available from: http://www.laas.fr/~sbosch/others/boschLandingArea.pdf. 5.1, 5.3

[BLS01] D. Bonnafous, S. Lacroix, and T. Siméon. Motion generation for a rover on rough terrains. In *International Conference on Intelligent Robotics and Systems*, 2001. Available from: http://www.laas.fr/~simon/publis/BONNAFOUS-IROS-2001.pdf. 5.3

[BNG+06] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens, and Eduardo Nebot. Consistency of the EKF-SLAM Algorithm. In *EEE/RSJ International Conference on Intelligent Robots and Systems*, 2006. Available from: http://www.acfr.usyd.edu.au/homepages/academic/tbailey/papers/ekfslam.pdf. 1.2.2

[BNL+03] Michael Bosse, Paul Newman, John Leonard, Martin Soika, Wendelin Feiten, and Seth Teller. An Atlas Framework for Scalable Mapping. In *IEEE International Conference on Robotics and Automation*, Taiwan, September 2003. Available from: http://graphics.lcs.mit.edu/~seth/pubs/bosse_etal_icra2003.pdf. 1.2.3

[BS05] Adrien Bartoli and Peter Sturm. Structure from motion using lines: Representation, triangulation and bundle adjustment. *Computer Vision and Image Understanding*, 100(3):416–441, dec 2005. Available from: http://perception.inrialpes.fr/Publications/2005/BS05. 4.1

[BSL93] Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation and Tracking: Principles, Techniques, and Software*. Artech House, 1993. 1.2.1, 1.2.1, 2.2.4, 3

[CKK+96] Peter Cheeseman, Bob Kanefsky, Richard Kraft, John Stutz, and Robin Hanson. Super-resolved surface reconstruction from multiple images. In *Maximum Entropy and Bayesian Methods*, pages 293–308. Kluwer Academic Publishers, 1996. Available from: citeseer.ist.psu.edu/cheeseman96superresolved.html. 5.3

[CLIM05] Cyril Charron, Ouiddad Labbani-Igbida, and El Mustapha Mouaddib. Qualitative localization using omnidirectional images and invariant features. In *IROS 2005*, 2005. 3.4

[CMNT99] J.A. Castellanos, J.M.M. Montiel, J. Neira, and J.D. Tardós. The SPmap: A Probabilistic Framework for Simultaneous Localization and Map Building. *IEEE Trans. Robotics and Automation,*, 15:948–953, 1999. Available from: http://webdiis.unizar.es/~jdtardos/papers/Castellanos_TRA_1999.pdf. 1.2.3

[CNT04] J. A. Castellanos, J. Neira, and J. D. Tardos. Limits to the consistency of the EKF-based SLAM. In *Intelligent Autonomous Vehicles*, 2004. Available from: http://webdiis.unizar.es/~jdtardos/papers/Castellanos_IAV_2004.pdf. 1.2.2

[CSS04] P.I. Corke, D. Strelow, and S. Singh. Omnidirectional visual odometry for a planetary rover. In *Proceedings of IROS 2004*, 2004. Available from: http://www.ri.cmu.edu/pubs/pub_4913.html. 3.4

[Dav03] A.J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. International Conference on Computer Vision, Nice*, October 2003. Available from: http://www.robots.ox.ac.uk/ActiveVision/Papers/davison_iccv2003/davison_iccv2003.pdf. 2.1, 2.3.2, 4.4.2, 5.3

[Dav05] Andrew J. Davison. Active search for real-time vision. In *ICCV*, 2005. Available from: http://www.doc.ic.ac.uk/~ajd/Publications/davison_iccv2005.pdf. 1.2.3

[DCK04] Andrew J. Davison, Yolanda Gonzalez Cid, and Nobuyuki Kita. Real-time 3D SLAM with Wide-Angle Vision. In *Proc. IFAC Symposium on Intelligent Autonomous Vehicles*, Lisbon, july 2004. Available from: http://www.robots.ox.ac.uk/ActiveVision/Papers/davison_etal_iav2004/davison_etal_iav2004.pdf. 2.1, 5.2

[Dea02] Matthew C. Deans. *Bearing-Only Localization and Mapping*. PhD thesis, Carnegie Mellon University, 2002. 3.4

[DH00] Matthew Deans and Martial Hebert. Experimental comparison of techniques for localization and mapping using a bearings only sensor. In *Proc. of the ISER '00 Seventh International Symposium on Experimental Robotics*, December 2000. Available from: http://www.ri.cmu.edu/pubs/pub_3453_text.html. 2.1, 3.4

[DNDW+01] Dissanayake, Newman, Durrant-Whyte, Clark, and Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotic and Automation*, 17(3):229–241, 2001. Available from: http://oceanai.mit.edu/pnewman/papers/SLAM_TransRandA.pdf. 1.2.1

[DWB06] Hugh Durrant-Whyte and Tim Bailey. Simultaneous Localisation and Mapping (SLAM): Part I - The Essential Algorithms. *Robotics and Automation Magazine*, June 2006. Available from: http://www.acfr.usyd.edu.au/homepages/academic/tbailey/. 1.2.3

[ED06a] Ethan Eade and Tom Drummond. Edge Landmarks in Monocular SLAM. In *BMVC*, 2006. Available from: http://www.macs.hw.ac.uk/bmvc2006/papers/412.pdf. 4.1

[ED06b] Ethan Eade and Tom Drummond. Scalable Monocular SLAM. In *CVPR*, 2006. 2.1, 2.5.3, 2.5.3, 4.1, 5.2

[ENT05] C. Estrada, J. Neira, and J.D. Tardós. Hierarchical SLAM: real-time accurate mapping of large environmentsfox. *IEEE Transactions on Robotics*, 2005. Available from: http://webdiis.unizar.es/~jdtardos/papers/Estrada_TRO_2005.pdf. 1.2.3, 5.2

[ESL05] R. Eustice, H. Singh, and J. Leonard. Exactly sparse delayed-state filters. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2428–2435, Barcelona, Spain, April 2005. 1.2.3

[FAS+02] Bourgault F., Makarenko A.A., Williams S.B., Grocholsky B., , and Durrant-Whyte H.F. Information based adaptive robotic exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002. 5.3

[FB02] M. Fiala and A. Basu. Line segment extraction in panoramic images. In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2002. Available from: http://wscg.zcu.cz/wscg2002/Papers_2002/C51.ps.gz. 4.7

[FJC05] J. Folkesson, P. Jensfelt, and H.I. Christensen. Vision slam in the measurement subspace. In *Intl Conf. on Robotics and Automation*, Barcelona, Spain, April 2005. Available from: http://www.cas.kth.se/~hic/publications.html. 4.1

[GBSD97] J. De Geeter, H. Van Brussel, J. De Schutter, and M. Decreton. A smoothly constrained kalman filter. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 19:1171–1177, October 1997. Available from: http://people.mech.kuleuven.be/~jdgeeter. 4.4.2, 4.5.1

[GL02] J. Gonzalez and S. Lacroix. Rover localization in natural environments by indexing panoramic images. In *ICRA02*, 2002. Available from: http://www.laas.fr/~simon/publis/GONZALEZ-ICRA-2002.pdf. 3.4, 3.4.3, 3.11

[GL03a] J. Gancet and S. Lacroix. Pg2p: A perception-guided path planning approach for long range autonomous navigation in unknown natural environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas (USA)*, 2003. 5.3

[GL03b] J. Gancet and S. Lacroix. Pg2p: A perception-guided path planning approach for long range autonomous navigation in unkown natural environments. In *International Conference on Intelligent Robotics and Systems*, 2003. Available from: http://spiderman-2.laas.fr/~simon/publis/GANCET-IROS-03.pdf. 5.3

[Har94] R. Haralick. Propagating covariances in computer vision. In *International Conference on Pattern Recognition*, 1994. Available from: http://www.vision.auc.dk/~hic/perf-ws/haralick.ps.gz. B.3

[HJL+89] R. Haralick, H. Joo, C. Lee, X. Zhuang, V Vaidya, and M. Kim. Pose estimation from corresponding point data. *IEEE Trans on Systems, Man and Cybernetics*, 19:1426–1445, 1989. 3.4.2, B.2

[HKD+05] Shoudong Huang, N. M. Kwok, G. Dissanayake, Q.P. Ha, and Gu Fang. Multi-step look-ahead trajectory planning in SLAM: Possibility and necessity. In *IEEE International Conference on Robotics and Automation*, 2005. Available from: http://services.eng.uts.edu.au/~sdhuang/ICRA05_SLAM_final.pdf. 5.3

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 4.2.1

[JA04] Ebi Jose and Martin Adams. Millimetre wave radar spectra simulation and interpretation for outdoor slam. In *IEEE International Conference on Robotics & Automation*, 2004. Available from: http://www.ntu.edu.sg/home/EAdams/publications2.htm. 1.3

[JFKC06] Patric Jensfelt, John Folkesson, Danica Kragic, and Henrik I. Christensen. Exploiting Distinguishable Image Features in Robotic Mapping and Localization. In *1st European Robotics Symposium (EUROS-06)*, Palermo, Italy, march 2006. Available from: http://www.nada.kth.se/~patric/publications/2006_EUROS_jfkc.pdf. 2.1

[JJL00] Montiel J.M.M., Tardós J.D., and Montano L. Structure and motion from straight line segments. *Pattern Recognition*, 33-8:1295–1307, 2000. Available from: http://webdiis.unizar.es/~josemari/Research.html. 4.1

[JL01] I-K. Jung and S. Lacroix. A robust interest point matching algorithm. In *International Conference on Computer Vision*, Vancouver (Canada), jul 2001. Available from: http://www.laas.fr/~simon/publis/JUNG-ICCV-2001.pdf. 3.2.2

[JL03] I-K. Jung and S. Lacroix. High resolution terrain mapping using low altitude aerial stereo imagery. In *International Conference on Computer Vision*, Nice (France), oct 2003. Available from: http://www.laas.fr/~simon/publis/JUNG-ICCV-03.pdf. 1.3

[JNN03] Jose Guivant Juan Nieto and Eduardo Nebot. Real time data association for fastslam. In *International Conference on Robotics and Automation (ICRA)*, 2003. Available from: http://www.acfr.usyd.edu.au/homepages/academic/enebot/publications/FS_icra_2003.pdf. 1.2.3

[JU97] SJ Julier and JK Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *Proc. American Control Conference 6/1997*, 1997. Available from: http://www.cs.ucl.ac.uk/staff/S.Julier/publications.html. 1.2.3

[Kal60] R.E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, J. Basic Eng.*, pages 35–45, 1960. 1.2.1

[KD04] N. M. Kwok and G. Dissanayake. An efficient multiple hypothesis filter for bearing-only SLAM. In *IROS*, 2004. 2.1, 2.2.4

[KDH05] N. M. Kwok, G. Dissanayake, and Q. P. Ha. Bearing-only SLAM using a SPRT based gaussian sum filter. In *ICRA 2005*, 2005. 2.1, 5.2

[KDR01] Joss Knight, Andrew Davison, and Ian Reid. 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, October 2001. Available from: http://www.robots.ox.ac.uk/~lav/Papers/knight_etal_iros2001/knight_etal_iros2001.pdf. 1.2.3

[Kon05] Kurt Konolige. SLAM via Variable Reduction from Constraint Maps. In *ICRA 2005*, 2005. 2.1

[KS03] Jong Hyuk Kim and Salah Sukkarieh. Airborne simultaneous localisation and map building. In *Proceedings of IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, September 2003. Available from: http://www.acfr.usyd.edu.au/publications/downloads/2003/Kim194/ICRA2003SLAM. 1.2.1

[LBJL06] Thomas Lemaire, Cyrille Berger, Il-Kyun Jung, and Simon Lacroix. Vision-based SLAM: Stereo and Monocular Approaches. Technical report, LAAS-CNRS, 2006. submitted to IJCV/IJRR special joint issue. Available from: http://www.laas.fr/~tlemaire/publications/lemaireIJCVIJRR2006.pdf. 3.2, 3.2.2

[LDSL06] Simon Lacroix, Michel Devy, Joan Solà, and Thomas Lemaire. Modélisation 3D par vision pour la robotique mobile : approches de cartographie et localisation simultanées. *Journal Français de Photogrammétrie et Télédétection*, 2006. Available from: http://www.laas.fr/~tlemaire/publications/lacroixJFPT.pdf.

[LL06a] Thomas Lemaire and Simon Lacroix. Monocular-Vision based SLAM using line segments. In *Robotic 3D Environment Cognition, Workshop at the International Conference Spatial Cognition*, 2006. Available from: http://www.laas.fr/~tlemaire/publications/lemaireSC2006.pdf.

[LL06b] Thomas Lemaire and Simon Lacroix. SLAM with panoramic vision. Technical report, LAAS-CNRS, 2006. submitted to the Journal for Fields Robotic in the special issue SLAM in the Fields. Available from: http://www.laas.fr/~tlemaire/publications/lemaireJFR2006_SLAM.pdf.

[LL06c] Thomas Lemaire and Simon Lacroix. Vision-based SLAM: achievement of a practical algorithm. In *IROS*, 2006. This video appeared in the video proceedings. Available from: http://www.laas.fr/~tlemaire/publications/lemaireBoSLAMVideo.mp4.

[LL07] Thomas Lemaire and Simon Lacroix. Monocular-vision based SLAM using line segments. Technical report, LAAS-CNRS, 2007. submitted to ICRA 2007. Available from: http://www.laas.fr/~tlemaire/publications/lemaireICRA2007.pdf.

[LLS05] Thomas Lemaire, Simon Lacroix, and Joan Solà. A practical 3D bearing only SLAM algorithm. In *IEEE International Conference on Intelligent Robots and Systems*, august 2005. Available from: http://www.laas.fr/~tlemaire/publications/lemaireIROS2005.pdf. 2.2.4

[LM97] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997. Available from: http://www.cs.dal.ca/~eem/pubs/mapping.pdf. 1.2.3

[Low85] David G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, 1985. 5.3

[LRNB02] J. Leonard, R. Rikoski, P. Newman, and M. Bosse. Mapping partially observable features from multiple uncertain vantage points. *International Journal of Robotics Research*, jan 2002. Available from: http://oe.mit.edu/~jleonard/pubs/leonard_etal_ijrr_w_ransac_result.pdf. 2.1

[LTG+03] P. Lamon, A. Tapus, E. Glauser, N. Tomatis, and R. Siegwart. Environmental modeling with fingerprint sequences for topological global localization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003. 3.4

[MC89] P. Moutarlier and R. Chatila. Stochastic multisensory data fusion for mobile robot location and environment modeling. In *5th Int. Symposium on Robotics Research,*, 1989. 1.1

[MCD06] J. M. M. Montiel, Javier Civera, and Andrew J. Davison. Unified inverse depth parametrization for monocular SLAM. In *RSS 2006*, 2006. Available from: http://www.doc.ic.ac.uk/~ajd/Publications/montiel_etal_rss2006.pdf. 2.1, 2.4.3, 2.5.3, 2.5.3, 2.5.3, 4.1, 5.2

[MD06] J. M. M. Montiel and Andrew J. Davison. A visual compass based on SLAM. In *ICRA 2006*, 2006. Available from: http://www.doc.ic.ac.uk/~ajd/publications.html. 2.4.3

[MDR04] N. D. Molton, A. J. Davison, and I. D. Reid. Locally planar patch features for real-time structure from motion. In *Proc. British Machine Vision Conference*. BMVC, September 2004. (To appear). Available from: http://www.robots.ox.ac.uk/ActiveVision/Papers/molton_etal_bmvc2004/molton_etal_bmvc2004.html. 5.3

[Mei] C. Mei. Omnidirectional calibration toolbox - http://www-sop.inria.fr/icare/personnel/Christopher.Mei/Toolbox.html. Available from: http://www-sop.inria.fr/icare/personnel/Christopher.Mei/Toolbox.html. 3.4.4

[MLD+06] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. 3D reconstruction of complex structures with bundle adjustment: an incremental approach. In *ICRA 2006*, 2006. Available from: http://www.lasmea.univ-bpclermont.fr/Personnel/Maxime.Lhuillier/Papers/Icra06_2.pdf. 2.1

[MLG00] A. Mallet, S. Lacroix, and L. Gallo. Position estimation in outdoor environments using pixel tracking and stereovision. In *IEEE International Conference on Robotics and Automation*, 2000. Available from: http://www.laas.fr/~simon/publis/MALLET-ICRA-2000.pdf. 3.4.2

[MOM04] J. Minguez, J. Osuna, and L. Montano. A divide and conquer strategy based on situations to achieve reactive collision avoidance in troublesome scenarios. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004. Available from: http://webdiis.unizar.es/~jminguez/810.pdf. 5.3

[MR06] C. Mei and P. Rives. Calibrage non biaise d'un capteur central catadioptrique. In *RFIA*, January 2006. Available from: http://www-sop.inria.fr/icare/personnel/Christopher.Mei/Publications.html. 3.4.4

[MTW03] M. Montemerlo, S. Thrun, and B. Wegbreit. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *International Conference on Artificial Intelligence (AAAI)*, 2003. Available from: http://www-2.cs.cmu.edu/~mmde/mmdeijcai2003.pdf. 1.2.3

[NCH06] P. Newman, D. Cole, and K. Ho. Outdoor SLAM using visual appearance and laser ranging. In *ICRA 2006*, 2006. Available from: http://www.robots.ox.ac.uk/~pnewman/papers/LoopClosingWith3DSLAM_ICRA06.pdf. 1.2.3, 3.4.3

[NDW01] Eric W. Nettleton and Hugh F. Durrant-Whyte. Delayed and asequent data in decentralised sensing networks. In *Sensor Fusion and Decentralized Control in Robotic Systems*, 2001. Available from: http://www.acfr.usyd.edu.au/publications/downloads/2001/Nettleton157/SPIE4571_01.pdf. 5.3

[New99] Paul Newman. *On the Structure and Solution of the Simultaneous Localisation and Map Building Problem*. PhD thesis, Australian Centre for Field Robotics - The University of Sydney, March 1999. Available from: http://oceanai.mit.edu/pnewman/papers/pmnthesis.pdf. 1.2.3, 4.4.2

[NGN04] Juan I. Nieto, Jose E. Guivant, and Eduardo M. Nebot. The HYbrid Metric Maps (HYMMs): A Novel Map Representation for DenseSLAM. In *ICRA*, 2004. Available from: http://www.acfr.usyd.edu.au/publications/downloads/2004/Nieto216/Nieto_ICRA04.pdf. 5.3

[Nis03] David Nister. Preemptive RANSAC for live structure and motion estimation. In *Ninth IEEE International Conference on Computer Vision (ICCV'03)*, volume 1, page 199, 2003. Available from: http://www.vis.uky.edu/~dnister/Publications/2005/preemptive/nister_preemptive.pdf. 2.1

[NL03] Paul M. Newman and John J. Leonard. Consistent convergent constant time slam. In *International Joint Conference on Artificial Intelligence*, Acapulco Mexico, August 2003. Available from: http://www.robots.ox.ac.uk/~pnewman/papers/IJCAI2003.pdf. 1.2.3

[NLNT02] P. M. Newman, J. J. Leonard, J. Neira, and J. Tardo's. Explore and return: Experimental validation of real time concurrent mapping and localization. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pages 1802–1809, May 2002. Available from: http://www.robots.ox.ac.uk/~pnewman/papers/ReturnToADime.pdf. 1.2.3, 1.3

[NTDW03] E. Nettleton, S. Thrun, and H. Durrant-Whyte. Decentralised slam with low-bandwidth communication for teams of airborne vehicles. In *Proceedings of the International Conference on Field and Service Robotics*, Lake Yamanaka, Japan, 2003. Available from: http://robots.stanford.edu/papers/Nettleton03a.pdf. 5.3

[OMSM00] C. Olson, L. Matthies, M. Schoppers, and M. Maimone. Robust stereo ego-motion for long distance navigation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2000. 3.4.2

[Pea95] N. Peach. Bearing-only tracking using a set of range-parametrised extended kalman filters. In *IEEE Proceedings on Control Theory Applications*, volume 142, pages 73–80, 1995. 2.1, 2.2.3, 2.2.4, 2.3.2, 2.5.2

[RD05] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1508–1511, October 2005. Available from: http://mi.eng.cam.ac.uk/~er258/work/rosten_2005_tracking.pdf. 5.2

[RLSP03] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3D Object Modeling and Recognition Using Local Affine-Invariant Image Descriptors and Multi-View Spatial Constraints. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003. 5.3

[SC87] R.C. Smith and P. Cheesman. On the representation of spatial uncertainty. In *Int.J.Robotics Research*, 1987. 1.1

[SDML05] Joan Solà, Michel Devy, André Monin, and Thomas Lemaire. Undelayed initialization in bearing only SLAM. In *IEEE International Conference on Intelligent Robots and Systems*, august 2005. Available from: http://www.laas.fr/~tlemaire/publications/solaIROS2005.pdf. 2.1, 5.2

[SGH02] Williams SB, Dissanayake G, and Durrant-Whyte HFbo. An efficient approach to the simultaneous localisation and mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 406–411, Washington, DC, May 2002. International Conference on Robotics and Automation. Available from: http://www.acfr.usyd.edu.au/publications/downloads/2002/Williams165/ICRA02.pdf. 1.2.3

[SH95] A. Stentz and M. Hebert. A complete navigation system for goal acquisition in unknown environments. *Autonomous Robots*, 2, 1995. 5.3

[Sim05] Robert Sim. Stabilizing information-driven exploration for bearings-only slam using range gating. In *Proceedings of Intelligent Robots and Systems (IROS)*, 2005. Available from: http://www.cs.ubc.ca/~simra/publications/sim_iros05.pdf. 2.1

[Sol07] Joan Sola. *Towards Visual Localization, Mapping and Moving object tracking by a Mobile robot: a Geometric and Probabilistic Approach.* PhD thesis, Institut National Polytechnique de Toulouse, Ecole Doctorale Systèmes, 2007. 4

[SRD06] Paul Smith, Ian Reid, and Andrew Davison. Real-time monocular SLAM with straight lines. In *BMVC*, 2006. Available from: http://www.macs.hw.ac.uk/bmvc2006/papers/162.pdf. 4.1

[SS03] Dennis Strelow and Sanjiv Singh. Online motion estimation from image and inertial measurements. In *Workshop on Integration of Vision and Inertial Sensors (INERVIS)*, 2003. Available from: http://www.dennis-strelow.com/publications/documents/inervis03.pdf. 2.1

[SSSa] 2002 SLAM Summer School in stockholm - http://www.cas.kth.se/SLAM. Available from: http://www.robots.ox.ac.uk/~SSS06/Website/index.html. 1.2.3

[SSSb] 2004 SLAM Summer School in toulouse - http://www.laas.fr/SLAM. Available from: http://www.robots.ox.ac.uk/~SSS06/Website/index.html. 1.2.3

[SSSc] 2006 SLAM Summer School in oxford- http://www.robots.ox.ac.uk/~SSS06/Website/index.html. Available from: http://www.robots.ox.ac.uk/~SSS06/Website/index.html. 1.2.3

[TFBD00] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2000. Available from: http://robots.stanford.edu/papers/thrun.robust-mcl.pdf. 1.1

[TL03] S. Thrun and Y. Liu. Multi-robot slam with sparse extended information filers. In *Proceedings of the 10th International Symposium of Robotics Research (ISRR'03)*, Sienna, Italy, oct 2003. Available from: http://robots.stanford.edu/papers/liu.multiseif03.pdf. 5.3

[TLK+04] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous Localization and Mapping With Sparse Extended Information Filters. *International Journal of Robotics Research*, April 2004. Submitted for journal publication. Available from: http://robots.stanford.edu/papers/thrun.seif.pdf. 1.2.3

[Ume91] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13:376–380, 1991. B.2

[WHD05] Zhan Wang, Shoudong Huang, and Gamini Dissanayake. Dslam: Decoupled localization and mapping for autonomous robot. In *ISRR*, 2005. Available from: http://robots.stanford.edu/program.html. 1.2.3