



UNIVERSITÀ DEGLI STUDI DI SIENA  
FACOLTÀ DI INGEGNERIA

Corso di Laurea Specialistica in Ingegneria Informatica

# Multi-Robot SLAM using M-Space Feature Representation

**Relatore**

Prof. Andrea Garulli

**Correlatore**

Ing. Antonio Giannitrapani

**Tesi di laurea di**

Daniele Benedettelli

ANNO ACCADEMICO 2007/2008



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Localization and SLAM</b>	<b>4</b>
2.1	Robot Localization . . . . .	5
2.1.1	Local techniques . . . . .	6
2.1.2	Global techniques . . . . .	7
2.2	Single-robot SLAM . . . . .	9
2.2.1	Topological approach . . . . .	9
2.2.2	Grid-based approach . . . . .	10
2.2.3	Feature-based approach . . . . .	10
2.3	Multi-robot SLAM . . . . .	12
<b>3</b>	<b>M-Space EKF SLAM</b>	<b>16</b>
3.1	Generic Feature Parametrization . . . . .	16
3.2	M-Space representation . . . . .	21
3.2.1	Projection Matrix . . . . .	21
3.2.2	Feature Initialization and Growing Dimensions . . . . .	24
3.3	SLAM . . . . .	25
3.3.1	General SLAM algorithm . . . . .	25
3.3.2	Problem Formulation . . . . .	26
3.4	EKF Implementation . . . . .	31
3.5	Feature Extraction . . . . .	34
3.5.1	Algorithm overview . . . . .	35
3.5.2	Line fitting . . . . .	36
3.6	Data Association . . . . .	37
<b>4</b>	<b>Map fusion</b>	<b>42</b>
4.1	Map Alignment . . . . .	43
4.1.1	Data Produced by EKF SLAM algorithms . . . . .	43
4.1.2	Relative Distance and Bearing Measurements . . . . .	44
4.1.3	Transformation from Global Frame $\langle G_2 \rangle$ to $\langle G_1 \rangle$ . . . . .	46
4.2	Error Transformation . . . . .	48

---

4.2.1	Projecting M-Space Errors into Feature Space . . . . .	49
4.2.2	Error on ${}^{G_1}\mathbf{p}_{R_2}$ . . . . .	52
4.2.3	Error on ${}^{G_1}\phi_{R_2}$ . . . . .	53
4.2.4	Error on ${}^{G_1}\mathbf{x}_{f_j}$ . . . . .	53
4.3	Transforming the Map . . . . .	55
4.4	Matching and Eliminating Duplicate Landmarks . . . . .	57
<b>5</b>	<b>Single-robot SLAM Simulations</b>	<b>61</b>
5.1	The SLAM simulator software . . . . .	61
5.2	Simple Environment . . . . .	65
5.3	Complex Environment . . . . .	67
<b>6</b>	<b>Multi-robot SLAM Simulations</b>	<b>75</b>
6.1	Simple Environment . . . . .	77
6.2	Complex Environment . . . . .	82
6.3	Performance Comparison . . . . .	88
<b>7</b>	<b>Conclusions</b>	<b>93</b>
	<b>Bibliography</b>	<b>96</b>



# List of Figures

3.1	Robot pose and feature coordinates in global reference frame. . .	18
3.2	Line parameters. . . . .	20
3.3	Robot pose and waypoint coordinates in global reference frame.	27
3.4	Line parameters. . . . .	29
3.5	Estimated measurement variances. . . . .	37
3.6	Measured and estimated feature overlapping: a) $\tau = 1$ ; b) $\tau < 1$ .	40
4.1	Robot-to-robot measurements. . . . .	45
4.2	Geometric relations in map alignment. . . . .	47
5.1	The lines must be seen multiple times before being considered reliable features. . . . .	63
5.2	The endpoints of the lines are detected in correspondence of two walls intersection. . . . .	64
5.3	The map produced by the simulated single-robot SLAM in the small test environment. . . . .	65
5.4	Simple map - The robot pose error components $\tilde{x}_r$ , $\tilde{y}_r$ and $\tilde{\theta}_r$ compared with the correspondent 99.9% confidence bands. . . .	66
5.5	Simple map - Multiple figure showing the absolute values of the detected line endpoints average error, of the line orientation average error, and of the line normal distance average error. . .	67
5.6	Simple map - The average standard deviation of the line orien- tation errors and of the line distance errors. . . . .	68
5.7	The map produced by the simulated single-robot SLAM in the S. Niccolò building. . . . .	69
5.8	A zoomed view of the S. Niccolò map, showing the round wall of the Electronics Lab. . . . .	70
5.9	Another zoomed view of the S. Niccolò map, showing the area where the robot starts and ends the SLAM experiment. . . . .	70
5.10	The third zoomed view of the S. Niccolò map. . . . .	71
5.11	Big map - The robot pose error components $\tilde{x}_r$ , $\tilde{y}_r$ and $\tilde{\theta}_r$ com- pared with the correspondent 99.9% confidence bands. . . . .	71

5.12	Big map - Multiple figure showing the absolute values of the detected line endpoints average error, of the line orientation average error, and of the line normal distance average error. . . . .	72
5.13	Big map - The covariance submatrix trace square root plotted against time. The peaks are in corrispondence of the initialization of new features. . . . .	73
5.14	Big map - The average standard deviation of the line orientation errors and of the line distance errors. The dashed vertical lines indicate the loop closures. . . . .	73
5.15	Big map - The percentage of inconsistent features. The dashed vertical lines indicate the loop closures. . . . .	74
6.1	The maps produced by the simulated multi-robot SLAM in the small test environment. (a) The map of robot $R_1$ . (b) The map of robot $R_2$ . . . . .	77
6.2	Simple map fusion - The matching duplicate features are used as constraints to improve the map alignment: (a) before the updates, (b) after imposing the constraints. . . . .	78
6.3	Simple map fusion - The robots pose errors components $\tilde{x}_r$ , $\tilde{y}_r$ and $\tilde{\theta}_r$ compared with the correspondent 99.9% confidence bands: (a) robot $R_1$ , (b) robot $R_2$ . The loop closures are marked with L.C. . . . .	79
6.4	Simple map fusion - Multiple figure showing the absolute values of the detected line endpoints average error, of the line orientation average error, and of the line normal distance average error: (a) robot $R_1$ , (b) robot $R_2$ . The star marks indicate the instant of the rendezvous. . . . .	80
6.5	Simple map fusion - The average standard deviation of the line orientation errors and of the line distance errors: (a) robot $R_1$ , (b) robot $R_2$ . The star marks indicate the instant of the rendezvous. . . . .	81
6.6	The maps produced by the simulated multi-robot SLAM in the S. Niccolò building: (a) the map of robot $R_1$ , (b) the map of robot $R_2$ . . . . .	83
6.7	Big map fusion - The matching duplicate features are used as constraints to improve the map alignment: (a) the map of robot $R_1$ before the rendezvous, (b) the map of robot $R_2$ before the rendezvous, (c) the fused map before imposing the constraints (d) the fused map after imposing the constraints. . . . .	84
6.8	Big map fusion - The robots pose errors components $\tilde{x}_r$ , $\tilde{y}_r$ and $\tilde{\theta}_r$ compared with the correspondent 99.9% confidence bands: (a) robot $R_1$ , (b) robot $R_2$ . . . . .	85

---

6.9	Big map fusion - Multiple figure showing the absolute values of the detected line endpoints average error, of the line orientation average error, and of the line normal distance average error: (a) robot $R_1$ , (b) robot $R_2$ . The star marks indicate the instant of rendezvous, while the loop closures are marked with L.C. . . . .	86
6.10	Big map fusion - The average standard deviation of the line orientation errors and of the line distance errors: (a) robot $R_1$ , (b) robot $R_2$ . . . . .	87
6.11	Big map - The percentage of inconsistent features in the maps built by: (a) robot $R_1$ , (b) robot $R_2$ . The star marks are indicate the instant of the rendezvous. . . . .	88
6.12	Test environment used to compare the performance of multi-robot SLAM against single-robot SLAM. . . . .	89

# Chapter 1

## Introduction

A *robot* is usually defined as a virtual or electro-mechanical artificial agent. Today, commercial and industrial robots are in widespread use, performing jobs more cheaply or with greater accuracy and reliability than humans. They are also employed for jobs which are too dirty, dangerous or dull to be suitable for humans. Robots are widely used in manufacturing, assembly and packing, transport, mass production of consumer and industrial goods, surgery, and weaponry. Recently, robots are gaining space in houses, as vacuum cleaners, lawn-mowers and toys. The above mentioned robots often work under strict men control.

Robotics is a vast field of study; an important application subfield is *mobile robotics*, that concerns many topics, such as space exploration, traffic control, factories and warehouses automation, or logistics. A desirable feature for robots that perform earth and space exploration is *autonomy*. One of the key question a real autonomous mobile robot should be able to answer is “*Where am I?*”. In other terms, the robot should be capable of localizing itself in the environment it is exploring. If the spatial model of the physical environment (the map) is not available, the robot should acquire it, while localizing itself with respect to the map. This problem has been well studied over the last two decades, and goes under the name of *Simultaneous Localization and Mapping (SLAM)* problem. Towards the goal of building truly autonomous mobile robots, SLAM is generally regarded as one of the most important problems by

researchers in mobile robotics.

The difficulty of SLAM increases with the size of the environment, and specifically with the size of possible cycles performed by the robot. An issue in SLAM is the ability of the robot to close the loops, i.e. to recognize places that have been already visited, after a journey. Because robot motion is inaccurate, the longer the journey, the bigger the accumulated pose error will be. Teams of cooperating robots could be employed to overcome this problem, but also to perform the exploration and mapping tasks more quickly and robustly, and to increase the quality of the map.

Multi-robot SLAM is a problem that has been tackled intensively only in recent years. Almost all of the work has been aimed at 2D environments. Additionally, nearly all of the research takes an existing algorithm developed for single robot mapping, localization, exploration, and extends it to multiple robots, as opposed to developing a new algorithm that is fundamentally distributed.

This thesis main contribution is a multi-robot SLAM algorithm for a team of mobile robots that have to build a 2D map of the environment with line segments, representing the feature uncertainty in the *measurement subspace* (M-Space) [9]. The M-Space feature representation allows one to specify the line segment feature location and extent (feature space), while expressing its uncertainty in a different space (measurement subspace), corresponding to the partial information provided by the robot sensors. The uncertainty is expressed in a frame attached to the features, so to avoid the *lever-arm effect*, i.e. the feature uncertainty dependency on the location of the feature itself.

The SLAM problem is formulated as a state estimation problem, where the state is composed by the robot pose and the features parameters. The state estimation technique adopted to tackle this problem is the Extended Kalman Filter (EKF). Initially, the robots perform SLAM independently sensing the environment through a laser range finder, estimating their pose and map by

using the EKF. When the robots *meet*, the independent maps are fused together using relative distance and bearing measurements [30], thus making no assumption on the initial location of the robots. The estimation error covariances of the two maps are transformed accordingly. The map fusion procedure is described for the two-robot case, and can be extended to larger robot teams by repeating the procedure for each pair of robots. After the alignment, the enlarged maps are searched for duplicate features. These duplicates are used to impose constraints between the two maps via EKF updates, thus improving the map alignment quality. Each robot get a different version of the same merged map, that is used by the robots to continue performing single-robot SLAM, until they meet and share their information again.

The thesis is organized as follows. Chapter 2 provides an overview in the state of art in robotic localization and mapping problem. In Chapter 3, the M-Space feature representation is introduced. The single-robot EKF SLAM algorithm is described together with the SLAM-related main issues, as the feature extraction from laser raw data and the data association. Chapter 4 describes the map fusion algorithm, that is the base of the multi-robot SLAM technique adopted. Experimental results of the single-robot SLAM and multi-robot SLAM algorithms are shown in Chapter 5 and in Chapter 6 respectively. Concluding remarks and future research directions are outlined in Chapter 7.

# Chapter 2

## Localization and SLAM

The key requirement for a robot in order to achieve a true long-range autonomy is *self-localization*. In fact, almost every task an autonomous robot is assigned requires the knowledge of the vehicle position and orientation (in brief, *pose*) with respect to a global reference frame. Since the 1980s, the problem has been deeply studied, and several solutions have been proposed, allowing the determination of the robot pose, given a map of the environment. However, in real-world applications an autonomous agent is often called to face more challenging situations, where the operating environment is only partially known (uncertain map), or even completely unknown. In all those cases, such as exploration tasks, or missions in hostile environments, a mobile robot must build a map of the environment it is navigating, in while simultaneously localize itself within the map.

For this reason, in recent years, the research in mobile robotics and AI has focused on the study of efficient solutions to the *Simultaneous Localization and Map building* (SLAM) problem. Despite significant progresses in this area, the problem still poses great challenges. At present, robust methods for mapping environments that are static, structured, and of limited size, have been proposed. Mapping unstructured, dynamic, or large-scale environments remains largely an open research problem.

This chapter is organized as follows. An historical overview of the robot localization problem is introduced in Section 2.1. Some of the most popular

techniques proposed in literature to tackle the SLAM problem are presented in Section 2.2. Finally, the recent work on multi-robot SLAM is reviewed in Section 2.3.

## 2.1 Robot Localization

Robot localization has been widely referred to as one of the most fundamental problems in mobile robotics. The aim of the localization problem is to estimate the pose of a robot in the environment, given a map of the surrounding space and sensor data. Most successful mobile robot systems to date utilize localization techniques, since knowledge of the robot pose is essential for a large number of robotic tasks. Although extensively investigated during the last three decades, the localization problem is still an active field of research, because of the lack of a single, generally appropriate method. Surveying the vast literature on mobile robot positioning [3] [27], it turns out that there is no universally valid solution to date.

The heterogeneous localization methods proposed so far in the literature can roughly be categorized into *local* and *global* techniques [10]. Local (tracking) techniques incrementally update the robot position estimate. They require the knowledge (at least approximate) of the initial pose of the autonomous vehicle and they cannot typically recover from a possible failure of the robot position tracking. On the contrary, global techniques are designed to estimate the robot pose even when the initial robot configuration is completely unknown. Clearly, the global techniques are more powerful, at the price of higher computational and memory requirements.

Mobile robots are usually equipped with several sensors, generally distinguished in two classes:

- *proprioceptive* sensors are all those sensors measuring quantities related to the robot itself, such as encoders, gyroscopes, accelerometers, compasses;

- *exteroceptive* sensors are the proximity sensors (infrared, sonars, laser range finders, millimeter wave radars), cameras, stereocams, omnidirectional cameras, and all other sensors performing measurements on the external environment.

Fusion of the two kinds of information is usually obtained via some filtering technique, the EKF being the most popular.

### 2.1.1 Local techniques

The simplest method for tracking the position of an autonomous vehicle is *dead reckoning*. Derived from “deduced reckoning” of sailing days, it denotes a mathematical procedure for determining the current position of a robot by advancing some previous position through known course and velocity information, over a given length of time. In its easiest implementation, such a technique is usually referred to as odometry, since the position of the vehicle along its path is directly provided by some on-board “odometer”. Optical encoders, directly coupled to the wheel axes, are the most common sensors suited to this task. Because of the easiness of the computation involved, and the inexpensiveness of the sensory system required, practically all mobile platforms are supplied with odometric modules [3].

A more elaborate dead reckoning implementation is given by the Inertial Navigation Units (INUs), used in Unmanned Aerial Vehicles (UAVs) [4] or in robots that move on legs. Position estimate is computed by integrating measurements of rate of rotation and linear acceleration, provided by gyroscopes and accelerometers respectively. Both odometry and INUs are prone to systematic errors (kinematic and sensor models inaccuracy) and stochastic errors (noise affecting the measurements). Unfortunately, the integration process, underlying the principle of dead reckoning, leads to error accumulation. Off-line calibration procedures aiming at minimizing the effect of systematic errors on odometric systems have been proposed, but unless additional infor-

mation from some absolute position sensing mechanism is used, the position error grows without bound, as the robot moves on. Notwithstanding these inherent flaws, dead reckoning still represents an effective solution for tracking the pose of a mobile robot, as far as short-range navigation is concerned.

### 2.1.2 Global techniques

Depending on the representation of the robot surroundings, several global localization methods have been developed. One of the most common descriptions of the environment consists in a map given in terms of *landmarks*. A landmark denotes a remarkable feature of the environment which, depending on the sensory system, the robot is able to successfully recognize in successive time instants. They are usually classified in *artificial* and *natural* landmarks. Natural landmarks can be thought of as objects or features that are already present in the environment and need algorithms of feature extraction from raw sensor data. Artificial landmarks are specially designed objects or markers placed in the environment, that is properly modified with the sole purpose of enabling robot navigation [3]. Most techniques exploit angular and/or distance measurements with respect to landmarks having known position, to compute an estimate of the current robot pose.

A different description of the environment is represented by *occupancy grids* [19]. They consist in discretizing the robot space into cells, marked with a score ranging from  $-1$  to  $1$ , representing the belief that the corresponding region is occupied, where  $-1$  denotes a certainly empty cell while  $1$  denotes a certainly occupied one. In this setting, the robot pose can be estimated by matching a local map, built from the current sensor readings, against an a priori known map. A measure of the goodness of the match between two maps, and hence a trial displacement and rotation, is found by computing the sum of products of the corresponding cells [6].

In recent years, several probabilistic techniques for tackling global localiza-

tion problems have been investigated. For example, Markov localization computes a discrete approximation of a probability distribution over the whole set of possible robot poses. The techniques proposed in [5] [23] represent a first application of Particle Filters to mobile robotics problems. The main idea is to approximate the posterior of the robot pose by means of a fixed number of sample poses, or particles, generated according to probabilistic models of the robot dynamics and of the measurement process. This approach enjoys several appealing properties: it is able to deal with arbitrary distributions; the update of the estimates, given the observations, can be carried out in constant time (it depends only on the cardinality of the particle set); its accuracy increases with the availability of computational resources, as more particles are used.

It is worth noticing that EKF-based methods can be regarded as Markov localization algorithms as well, provided that a unimodal Gaussian is used to model the robot pose. Experimental comparisons have shown that EKF-based tracking techniques are usually more accurate, but are less robust, since they lack the ability to globally localize the robot and to recover from localization errors.

To overcome this problem, the *Multiple Hypothesis Tracking* algorithm has been proposed in [16]. Bayesian hypothesis testing is applied to combine the Kalman filter estimates of robot displacements with all the possible ambiguous associations measurement/landmark. As new features are extracted, the number of valid hypotheses decreases and eventually a single location is determined. The robot pose estimates can thus be represented by multi-modal density functions, to achieve global localization.

In [11] [13], a different approach is adopted, based on the assumption that both measurement noises and model uncertainties are unknown but bounded. This hypothesis is motivated by the fact that a bound on the measurement and model errors is often the only available information in several practical situations. This choice naturally leads to a *set-theoretic* formulation of the

problem: all the estimates are derived in terms of feasible uncertainty sets, defined as those regions where the quantities to be estimated are guaranteed to lie. For more details about the set-theoretic approach, the reader is referred to [13] and the works cited therein.

## 2.2 Single-robot SLAM

In Section 2.1, it has been pointed out that several methods for localizing a mobile robot are available, using a priori knowledge on the surrounding environment. Similarly, the task of building an accurate map of the environment, given the exact robot position has been widely addressed [24]. However, in several real-world applications, both issues have to be tackled simultaneously, because a map of the working space is not available. In those cases, the harder Simultaneous Localization And Map building (SLAM) problem has to be faced. The agent must build a map of the environment, while simultaneously localizing itself within the map.

All the proposed SLAM solutions can be roughly categorized into three main approaches: topological, grid-based and feature-based. While the maps based on the former approach describe the connectivity of different places, the last two approaches relies on the geometric properties of the environment. The reader is referred to [24] for a comprehensive overview of the state of the art in robotic mapping.

### 2.2.1 Topological approach

Driven by the observation that human beings are able to successfully navigate, in spite of the poor accuracy of their sensory systems and in presence of incomplete knowledge, topological techniques adopt a qualitative description of the surrounding environment, rather than building a metrically accurate map. Distinctive places and paths are defined by their properties at the control level, and serve as the nodes and arcs of the topological model. During

the exploration, location matching is performed by using global topological constraints as well as local metrical comparison, in order to discriminate new places from already visited ones. In general, these methods are appropriate for navigation in simple environments and for creating models of human and animal cognition. However, for many operations, metrically accurate maps, that are globally referenced, are necessary in order to accomplish the goal of the mission [24].

### 2.2.2 Grid-based approach

Grid-based techniques represent the simplest approach to map building. They adopt the most intuitive environment representation, i.e. an occupancy grid (see Section 2.1.2). The space is discretized into cells, labelled with a value denoting the probability for that portion of space to be occupied. From the sensor data, the robot creates a local grid, related to its current pose. A search through a set of previously acquired certainty grids is performed to find the best match. The configuration of the robot giving the highest correlation is chosen as the robot pose estimate. Once the vehicle is localized, the probability distribution of the local grid is merged into the global one, thus reducing its uncertainty. The main drawback of the grid-based approach relies in its intrinsically high computational burden and storage requirement. Moreover, the effectiveness of these methods strongly depends on the accuracy of the sensory system model, since misinterpreted noisy data may lead to inconsistent maps.

### 2.2.3 Feature-based approach

In the feature-based approach, the environment is mapped in terms of a set of remarkable features (landmarks), whose positions have to be estimated, together with the robot pose. The landmarks are extracted from sensor data, and can be point landmarks, representing tree trunks, poles (outdoor mapping) or the corners formed by two intersecting walls (indoor mapping); line segment

landmarks are used to represent the walls for indoor SLAM.

In the early work [17] an EKF-based SLAM algorithm that builds a map with 2D point landmarks, is proposed. In [12] a line-based EKF mapping algorithm is proposed. In that work, the line features uncertainty is represented with respect to the map global frame, yielding to the *lever-arm effect*, i.e. the feature uncertainty dependency on the location of the feature itself. In order to solve the lever-arm problem, in [22], the *Symmetries and Perturbations model* (SP-model) is introduced; this feature representation allows one to represent the feature uncertainty with respect to a reference frame attached to each feature, thus removing the uncertainty dependency on the feature location. In the work [20] SLAM is performed by using EKF in a dynamic indoor environment, extracting line features from laser data, and representing the features uncertainty using the SP-model. The M-Space feature representation, introduced in [9], extends the SP-model: the M-Space framework allows building maps with point landmarks in 2D or 3D, or line segments.

The estimation errors of landmark locations are necessarily correlated, since they are based on relative measurements taken from the same uncertain positions (the robot pose). As proved by [2] for the linear case, in the limit, the landmark estimates are fully correlated; also, the determinant of any submatrix of the map covariance matrix decreases monotonically as observations are successively made. Therefore, maintaining the full state covariance matrix is of paramount importance, as far as map consistency is concerned. In [2], SLAM is performed outdoor by using EKF to build a map using artificial landmarks.

Unfortunately, when exploring large areas, densely populated of features, the computational burden required by the update of the full covariance matrix can rapidly become too heavy for real-time operations. For instance, if  $n$  landmarks have to be estimated, the storage requirements of the EKF grows as  $n^2$  and the computational complexity of the measurement update is usually  $O(n^3)$ . The cost involved in the management of the full state covariance matrix

limits the application of such techniques to middle-sized environments (map-scaling problem).

A simple strategy for reducing the cost of a SLAM algorithm is to fix the maximum number of features to be mapped, by “forgetting” far landmarks. This choice is clearly efficient, but an important amount of information is discarded. Other solutions for reducing the computational complexity of the optimal SLAM algorithm divide the environment into overlapping subregions, each with its own global stochastic map with a reasonable small number of features.

Generally all those approaches still require quadratic time for maintaining global consistency between different submaps, as well as quadratic storage requirements. A different approach is the one proposed by [25], based on the information form of the Extended Kalman Filter. Driven by the observation that in the SLAM problem the covariance matrix is generally dense, whereas its inverse (the information matrix) is naturally sparse, a constant-time technique is developed.

Given the difficulty of maintaining a huge number of features, the mapping task could be splitted among several robots. In the multi-robot approach, each robot builds a small local map, having to estimate a reasonable small number of features (light computational burden), and then fuses its contribution with the others, producing a larger map. In the next section, some of the recent techniques developed to solve the multi-robot SLAM problem are reviewed.

## 2.3 Multi-robot SLAM

As discussed in Sections 2.1 and 2.2, an extensive amount of research has been carried out in the area of localization, mapping and exploration for single autonomous robots. Only fairly recently has much of this work been applied to multi-robot teams. For an extensive discussion about the progresses of the state of the art in distributed mobile robotics, the reader is referred

to [21] [1] [28].

In order to increase the accuracy and efficiency when mapping large areas, it is often necessary for two or more robots to participate in this task. This process is known as multi-robot SLAM or Cooperative SLAM (C-SLAM). While enhancing efficiency, the complexity of SLAM increases when robots cooperate to build a single, joint map of the area they are exploring. This problem becomes especially challenging when the coordinate transformation (*initial correspondence*) between the initial poses of the robots is unknown.

In order to fuse maps created by different robots, the transformation between their reference frames needs to be determined; this can be done in two ways.

- To search for landmark matches in the two maps. The most probable transformation is the one that produces the maximum number of landmark correspondences. In this case, the underlying assumption is that the two maps are significantly overlapped.
- Robot-to-robot mutual measurements can be used for computing the unknown coordinate transformation: when two robots meet and measure their relative distance and bearing, this information can be used to compute the transformation required for merging the two maps. Due to noise in these measurements, the estimated transformation may be inaccurate, which in effect will reduce the quality of the merged map.

The former approach is pursued in [18]. Here, the robots relative locations are assumed to be unknown. In the marginal representation, the robots share a common map and their locations are independent given this map. The multi-robot Marginal-SLAM algorithm estimates the transformations of coordinates between the robots to produce a common global map. The SEIF introduced in [25] (see Section 2.2.3) is extended to the multi-robot case in [26]. Map fusion is performed by finding correspondences between maps using

SR-tree hill-climbing search; the most probable transformation between two robots frames is the one producing the maximum number of landmark correspondences. A similar method for fusing the maps is reported in [7]: here the Joint Compatibility Branch and Bound algorithm (JCBB) is used to search for matching 2D point landmarks between the two maps to be fused.

The second approach assumes that the robots meet at least once (rendezvous case). In [14], the manifold map concept is introduced. This representation takes maps out of the plane and onto a two-dimensional surface embedded in a higher-dimensional space. Compared with standard planar maps, the key advantage of the manifold representation is self-consistency: this facilitates a number of important autonomous capabilities, including robust retro-traverse, lazy loop closure, active loop closure using robot rendezvous, and, ultimately, autonomous exploration. The merging of the maps is made after the robots encounter. In [15], a Particle Filter-based method is introduced to integrate observations collected prior to the first robot encounter, using the notion of a virtual robot travelling backwards in time. This approach allows one to integrate all data from all robots into a single common map.

An alternative approach to C-SLAM is presented in [29], that examines the prospect of applying the Constrained Local Submap Filter (CLSF) to the multi-vehicle SLAM problem. Rather than incorporating every observation directly into the global map of the environment, the CLSF algorithm relies on creating an independent, local submap of the features in the immediate vicinity of the vehicle. This representation has been shown to reduce the computational complexity of maintaining the global map estimates as well as improving the data association process.

Finally the works [30] [31], propose an algorithm for merging (not necessarily overlapping) maps that are created by different robots independently. In order to perform map fusion, the robots must meet and measure their relative poses. Noise in the robot-to-robot observations, propagated through the

---

map-alignment process, increases the error in the position estimates of the transformed landmarks, and reduces the overall accuracy of the merged map. When there is overlap between the two maps, landmarks that appear twice provide additional information, in the form of constraints, which increase the alignment accuracy.

In this thesis, a multi-robot EKF-based SLAM algorithm is introduced. The features are represented using the M-Space framework [9]. The submaps built by different robots are fused together after the robots meet, using a technique based on [30] [31].

# Chapter 3

## M-Space EKF SLAM

This chapter describes the indoor SLAM technique in detail. The algorithm adopted is the Extended Kalman Filter (EKF), using M-Space uncertainty representation. This chapter is structured as follows: first, a generic feature geometric parametrization will be introduced in Section 3.1; then the M-Space theory will be described in Section 3.2; then the SLAM problem will be formally stated, and its EKF implementation showed in Section 3.3. Finally, various SLAM-related issues as feature extraction, data association and delayed feature initialization will be discussed.

### 3.1 Generic Feature Parametrization

When performing SLAM, one can deal with the following types of coordinates:

- 3D coordinates, used for example for a point feature in 3D space;
- 2D coordinates, used for points constrained on a plane;
- Scalar coordinates, used to model the radius of a pole (e.g. a tree trunk).

Any feature can be parametrized by a combination of these three kind of coordinates, allowing a generic treatment of the coordinate transformations.

A line segment feature is uniquely represented by its endpoints 2D coordinates,

$$\mathbf{x}_f = \begin{bmatrix} x_A \\ y_A \\ x_B \\ y_B \end{bmatrix}^m \quad (3.1)$$

where the  $[\cdot]^m$  superscript denotes the global reference frame; see Fig. 3.1 for reference. The pose of the robot expressed in the global frame is

$$\mathbf{x}_r = \begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix}^m. \quad (3.2)$$

Let  $\mathbf{R}(\psi)$  be the rotation matrix

$$\mathbf{R}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix}, \quad (3.3)$$

its inverse  $\mathbf{R}^{-1}(\psi) = \mathbf{R}^T(\psi) = \mathbf{R}(-\psi)$  and

$$\mathbf{E} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad (3.4)$$

the robot pose extender matrix.

The feature coordinates in the robot frame are given by the transformation

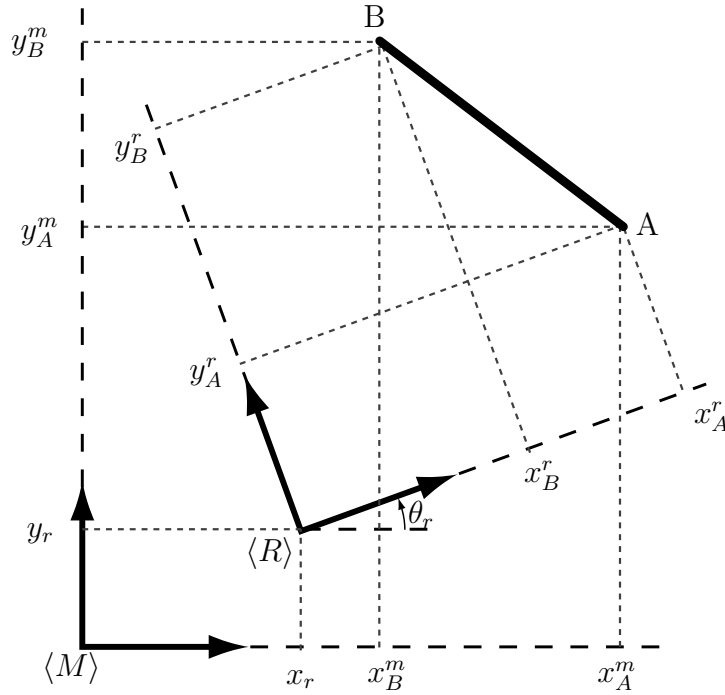
$$\begin{aligned} \mathbf{x}_o &= \mathbf{x}_f \ominus \mathbf{x}_r \\ \mathbf{x}_o &= \begin{bmatrix} x_A \\ y_A \\ x_B \\ y_B \end{bmatrix}^r = \begin{bmatrix} \mathbf{R}(-\theta_r) & \mathbf{0}_{2 \times 2} \\ \mathbf{0}_{2 \times 2} & \mathbf{R}(-\theta_r) \end{bmatrix} (\mathbf{x}_f - \mathbf{E} \mathbf{x}_r) = \\ &= \begin{bmatrix} \cos(\theta_r) & \sin(\theta_r) & 0 & 0 \\ -\sin(\theta_r) & \cos(\theta_r) & 0 & 0 \\ 0 & 0 & \cos(\theta_r) & \sin(\theta_r) \\ 0 & 0 & -\sin(\theta_r) & \cos(\theta_r) \end{bmatrix} \begin{bmatrix} x_A - x_r \\ y_A - y_r \\ x_B - x_r \\ y_B - y_r \end{bmatrix}^m \end{aligned} \quad (3.5)$$

where the  $[\cdot]^m$  superscript denotes that the robot position and feature endpoints coordinates are expressed in the global frame of reference.

The inverse transformation between feature coordinates in robot frame  $\mathbf{x}_o$  and feature coordinates in global frame  $\mathbf{x}_f$  is given by

$$\begin{aligned}
 \mathbf{x}_f &= \mathbf{x}_o \oplus \mathbf{x}_r \\
 \mathbf{x}_f &= \begin{bmatrix} x_A \\ y_A \\ x_B \\ y_B \end{bmatrix}^m = \begin{bmatrix} \mathbf{R}(\theta_r) & \mathbf{0}_{2 \times 2} \\ \mathbf{0}_{2 \times 2} & \mathbf{R}(\theta_r) \end{bmatrix} \mathbf{x}_o + \mathbf{E} \mathbf{x}_r = \\
 &= \begin{bmatrix} \cos(\theta_r) & -\sin(\theta_r) & 0 & 0 \\ \sin(\theta_r) & \cos(\theta_r) & 0 & 0 \\ 0 & 0 & \cos(\theta_r) & -\sin(\theta_r) \\ 0 & 0 & \sin(\theta_r) & \cos(\theta_r) \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ x_B \\ y_B \end{bmatrix}^r + \begin{bmatrix} x_r \\ y_r \\ x_r \\ y_r \end{bmatrix}^m. \tag{3.6}
 \end{aligned}$$

where the  $[\cdot]^r$  superscript denotes that the feature endpoints coordinates are expressed in the frame of reference of the robot.



**Figure 3.1:** Robot pose and feature coordinates in global reference frame.

The matrices  $J_{or}$  and  $J_{of}$  are the Jacobians of the feature coordinates in the robot frame  $\mathbf{x}_o$  with respect to (w.r.t.) the robot pose  $\mathbf{x}_r$  and to the feature coordinates.

$$\begin{aligned}
J_{or} &= \frac{\partial \mathbf{x}_o}{\partial \mathbf{x}_r} = \begin{bmatrix} \frac{\partial x_A^r}{\partial x_r} & \frac{\partial x_A^r}{\partial y_r} & \frac{\partial x_A^r}{\partial \theta_r} \\ \frac{\partial y_A^r}{\partial x_r} & \frac{\partial y_A^r}{\partial y_r} & \frac{\partial y_A^r}{\partial \theta_r} \\ \frac{\partial x_B^r}{\partial x_r} & \frac{\partial x_B^r}{\partial y_r} & \frac{\partial x_B^r}{\partial \theta_r} \\ \frac{\partial y_B^r}{\partial x_r} & \frac{\partial y_B^r}{\partial y_r} & \frac{\partial y_B^r}{\partial \theta_r} \end{bmatrix} = \\
&= \begin{bmatrix} -\cos(\theta_r) & -\sin(\theta_r) & (x_r - x_A^m) \sin(\theta_r) - (y_r - y_A^m) \cos(\theta_r) \\ \sin(\theta_r) & -\cos(\theta_r) & (x_r - x_A^m) \cos(\theta_r) + (y_r - y_A^m) \sin(\theta_r) \\ -\cos(\theta_r) & -\sin(\theta_r) & (x_r - x_B^m) \sin(\theta_r) - (y_r - y_B^m) \cos(\theta_r) \\ \sin(\theta_r) & -\cos(\theta_r) & (x_r - x_B^m) \cos(\theta_r) + (y_r - y_B^m) \sin(\theta_r) \end{bmatrix} \quad (3.7)
\end{aligned}$$

$$\begin{aligned}
J_{of} &= \frac{\partial \mathbf{x}_o}{\partial \mathbf{x}_f} = \begin{bmatrix} \frac{\partial x_A^r}{\partial x_A^m} & \frac{\partial x_A^r}{\partial y_A^m} & \frac{\partial x_A^r}{\partial x_B^m} & \frac{\partial x_A^r}{\partial y_B^m} \\ \frac{\partial y_A^r}{\partial x_A^m} & \frac{\partial y_A^r}{\partial y_A^m} & \frac{\partial y_A^r}{\partial x_B^m} & \frac{\partial y_A^r}{\partial y_B^m} \\ \frac{\partial x_B^r}{\partial x_A^m} & \frac{\partial x_B^r}{\partial y_A^m} & \frac{\partial x_B^r}{\partial x_B^m} & \frac{\partial x_B^r}{\partial y_B^m} \\ \frac{\partial y_B^r}{\partial x_A^m} & \frac{\partial y_B^r}{\partial y_A^m} & \frac{\partial y_B^r}{\partial x_B^m} & \frac{\partial y_B^r}{\partial y_B^m} \end{bmatrix} = \begin{bmatrix} \cos(\theta_r) & \sin(\theta_r) & 0 & 0 \\ -\sin(\theta_r) & \cos(\theta_r) & 0 & 0 \\ 0 & 0 & \cos(\theta_r) & \sin(\theta_r) \\ 0 & 0 & -\sin(\theta_r) & \cos(\theta_r) \end{bmatrix} = \\
&= \begin{bmatrix} \mathbf{R}(-\theta_r) & \mathbf{0}_{2 \times 2} \\ \mathbf{0}_{2 \times 2} & \mathbf{R}(-\theta_r) \end{bmatrix} \quad (3.8)
\end{aligned}$$

The line parametrization that will be described next needs to change the  $\arctan(y/x)$  function to produce values in the range  $(-\pi, \pi]$  and to distinguish between diametrically opposite directions of the vector  $[x \ y]^T$  used as argument. Let us define the function  $\text{atan2}(y, x) : \mathbb{R}^2 \rightarrow (-\pi, \pi]$  as

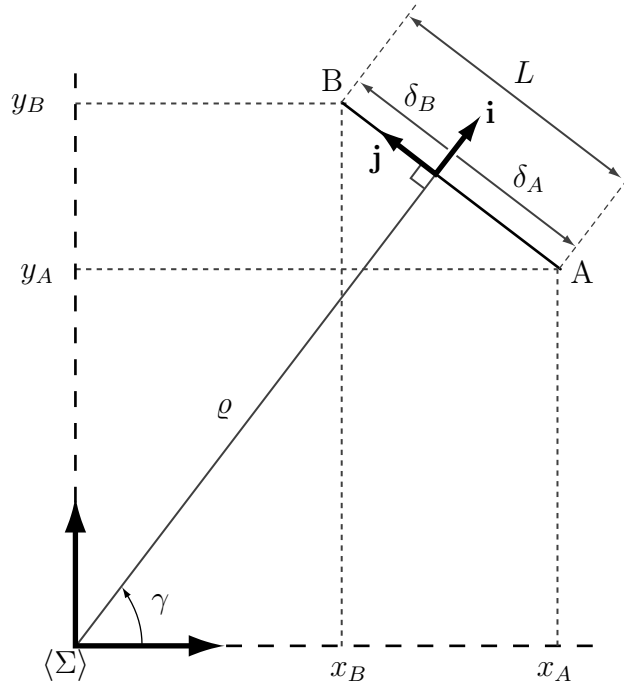
$$\text{atan2}(y, x) = \begin{cases} \arctan(|y/x|) \text{sign}(y) & x > 0 \\ \frac{\pi}{2} \text{sign}(y) & x = 0 \\ (\pi - \arctan(|y/x|)) \text{sign}(y) & x < 0 \\ 0 & y = 0, x > 0 \\ \text{undefined} & y = 0, x = 0 \\ \pi & y = 0, x < 0 \end{cases} \quad (3.9)$$

that will be used also later in this work.

In general, given a reference frame  $\langle \Sigma \rangle$ , a line segment feature can be parametrized in another equivalent form, as a function  $h$  of its endpoints coordinates

$$\ell = \begin{bmatrix} \gamma \\ \varrho \\ \delta_A \\ \delta_B \end{bmatrix} = h \left( \begin{bmatrix} x_A^\Sigma \\ y_A^\Sigma \\ x_B^\Sigma \\ y_B^\Sigma \end{bmatrix} \right) = \begin{bmatrix} \text{atan2}(x_B^\Sigma - x_A^\Sigma, y_A^\Sigma - y_B^\Sigma) \\ x_A^\Sigma \cos(\gamma) + y_A^\Sigma \sin(\gamma) \\ -x_A^\Sigma \sin(\gamma) + y_A^\Sigma \cos(\gamma) \\ -x_B^\Sigma \sin(\gamma) + y_B^\Sigma \cos(\gamma) \end{bmatrix} \quad (3.10)$$

where  $\gamma \in (-\pi, \pi]$  is the angle between the  $x$ -axis and the normal to the line passing through the origin of the chosen frame of reference  $\langle \Sigma \rangle$ ,  $\varrho \geq 0$  is the distance from the origin to the line,  $\delta_A$  and  $\delta_B$  are the line endpoints A and B signed distances from the point of incidence of the normal to the line, as depicted in Fig. 3.2.



**Figure 3.2:** Line parameters.

The line endpoints coordinates  $\mathbf{x}^\Sigma$  are related to the line parameters  $\ell$  (Eq. (3.10)) w.r.t. the chosen frame of reference  $\langle \Sigma \rangle$  as

$$\mathbf{x}^\Sigma = \mathbf{D}\ell$$

$$\begin{bmatrix} x_A \\ y_A \\ x_B \\ y_B \end{bmatrix}^\Sigma = \begin{bmatrix} 0 & \cos(\gamma) & -\sin(\gamma) & 0 \\ 0 & \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & \cos(\gamma) & 0 & -\sin(\gamma) \\ 0 & \sin(\gamma) & 0 & \cos(\gamma) \end{bmatrix} \begin{bmatrix} \gamma \\ \rho \\ \delta_A \\ \delta_B \end{bmatrix} \quad (3.11)$$

## 3.2 M-Space representation

The *measurement subspace*, or M-Space [9], is a feature representation similar to the SP-model [22] that attaches a local frame to each feature element, allowing a generic treatment of many types of features. M-Space is an abstraction of the measured subspace of the feature space, that reflects symmetries and constraints. The main idea is that the features are parametrized to fully specify their location and extent (feature space) but can be initialized in a subspace corresponding to the partial information provided by the robot sensors (measurement subspace).

For example, this allows to initialize a line segment feature even if only its position and direction are measured initially. The information about the line length will be saved in the feature space (as endpoints coordinates), but not initialized in the M-Space: so, the only information available is about the changes perpendicular to the line and about the orientation, while there will be no information along the line, at least until the line endpoints will be later observed and initialized in the M-Space. Since the features are represented as 3D, 2D or scalar coordinates, all the geometric informations have a generic parametrization with well defined transformation rules.

### 3.2.1 Projection Matrix

Let  $\delta\mathbf{x}_p$  denote the change in the M-Space corresponding to a small change in the feature coordinates  $\delta\mathbf{x}_f$ . The actual values of the M-Space coordinates

$\mathbf{x}_p$  are never known or needed, just their changes  $\delta\mathbf{x}_p$  enter into the estimates. These changes are used to adjust the feature coordinates  $\mathbf{x}_f$ , so that the mean of  $\delta\mathbf{x}_p$  is zero. The feature uncertainty estimate is an estimate of the distribution of  $\delta\mathbf{x}_p$  values around its zero mean. The uncertainty is expressed in the frame attached to each feature and can be projected into the global frame using the current global coordinates of the feature.

Small variations in the feature coordinates  $\delta\mathbf{x}_f$  in the feature space are related to small variations of the corresponding M-space coordinates  $\delta\mathbf{x}_p$  by a projection matrix  $\tilde{B}_f(\mathbf{x}_f)$ . In this section, when writing line feature *global* coordinates  $\mathbf{x}_f = [x_A \ y_A \ x_B \ y_B]^T$  the  $[\cdot]^m$  notation is omitted. The fundamental relations between  $\delta\mathbf{x}_p$  and  $\delta\mathbf{x}_f$  are

$$\delta\mathbf{x}_f = \tilde{B}(\mathbf{x}_f)\delta\mathbf{x}_p \quad (3.12)$$

$$\delta\mathbf{x}_p = B(\mathbf{x}_f)\delta\mathbf{x}_f \quad (3.13)$$

$$I_{pp} = B(\mathbf{x}_f)\tilde{B}(\mathbf{x}_f) \quad (3.14)$$

where  $\tilde{B}(\mathbf{x}_f)$  is the right Moore-Penrose pseudo-inverse of  $B(\mathbf{x}_f)$ ,  $\tilde{B} = B^T(B B^T)^{-1}$ : the pseudo-inversion is required since the  $B$  matrix could have less than 4 rows, as it will be shown later in section 3.2.2. Both  $B$  and  $\tilde{B}$  depend on the actual feature coordinates  $\mathbf{x}_f$ .

In the following, the form of the projection matrices will be shown for the line feature case, using Fig. 3.2 as reference. For a line feature, the projection matrix is

$$B_f = \begin{bmatrix} \frac{\cos \gamma}{L\sqrt{2}} & \frac{\sin \gamma}{L\sqrt{2}} & -\frac{\cos \gamma}{L\sqrt{2}} & -\frac{\sin \gamma}{L\sqrt{2}} \\ \frac{\cos \gamma}{\sqrt{2}} & \frac{\sin \gamma}{\sqrt{2}} & \frac{\cos \gamma}{\sqrt{2}} & \frac{\sin \gamma}{\sqrt{2}} \\ -\sin \gamma & \cos \gamma & 0 & 0 \\ 0 & 0 & -\sin \gamma & \cos \gamma \end{bmatrix} \quad (3.15)$$

where  $\gamma$  is the angle of the normal to the line with respect to  $x$ -axis and  $L$  is

the length of the line segment:

$$\gamma = \text{atan2}(x_A - x_B, y_B - y_A) \quad (3.16)$$

$$L = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} \quad (3.17)$$

These entities  $L$  and  $\gamma$  parametrize the infinite line which the feature line segment belongs to, in the same frame in which  $\mathbf{x}_f$  is expressed.

The  $\tilde{B}_f$  matrix that fulfills (3.14) is

$$\tilde{B}_f = \begin{bmatrix} \frac{L \cos \gamma}{\sqrt{2}} & \frac{\cos \gamma}{\sqrt{2}} & -\sin \gamma & 0 \\ \frac{L \sin \gamma}{\sqrt{2}} & \frac{\sin \gamma}{\sqrt{2}} & \cos \gamma & 0 \\ \frac{-L \cos \gamma}{\sqrt{2}} & \frac{\cos \gamma}{\sqrt{2}} & 0 & -\sin \gamma \\ \frac{-L \sin \gamma}{\sqrt{2}} & \frac{\sin \gamma}{\sqrt{2}} & 0 & \cos \gamma \end{bmatrix} \quad (3.18)$$

The form of  $B_f$  can be explained using geometric relations, that are in some case just approximations and thus valid only for small variations of parameters. The first row of the projection matrix  $B_f$  (3.15) corresponds to a rotation around the center of the line, i.e. to changes to  $\gamma$ . The second row is the projection of small movements of the center point of the line in the normal direction  $\mathbf{i}$ . The third and fourth row of  $B_f$  are the signed distances of the endpoints from the normal point of incidence, i.e. the projection of the endpoints coordinates along the line direction  $\mathbf{j}$ .

The projection matrix  $B_f$  projects small changes of the feature endpoints global coordinates  $\mathbf{x}_f$  to small changes of parameters  $\mathbf{x}_p$ . From the analytic point of view, we have

$$\delta \mathbf{x}_p = \frac{\partial \mathbf{x}_p}{\partial \mathbf{x}_f} \delta \mathbf{x}_f \quad (3.19)$$

$\mathbf{x}_p$  are the line parameters equivalent to  $\ell$  of Eq.(3.10), expressed in a chosen frame of reference  $\Sigma$ . In particular, if we choose as  $\Sigma$  the robot frame  $\langle R \rangle$ , the line parameters  $\mathbf{x}_p$  are given by Eq.(3.10),

$$\mathbf{x}_p = h(\mathbf{x}_o)$$

getting

$$\frac{\partial \mathbf{x}_p}{\partial \mathbf{x}_f} = \frac{\partial h(\mathbf{x}_o)}{\partial \mathbf{x}_f} \quad (3.20)$$

where  $\mathbf{x}_o$  are the line endpoints coordinates expressed w.r.t. the robot frame and depend on the feature endpoints coordinates in global frame  $\mathbf{x}_f$  by Eq.(3.5):  $\mathbf{x}_o = \mathbf{x}_f \ominus \mathbf{x}_r$ .

It can be shown that the choice of the local frame, in which the line endpoints  $\mathbf{x}_o$  and relative line parameters  $\mathbf{x}_p$  are expressed, does not influence the form of  $B_f$ , that will depend only by the line parameters  $\gamma$  and  $L$  in the global frame; the proof is beyond the scope of this work. The  $\sqrt{2}$  factors that appear in the first two rows of the  $B_f$  matrix are used to normalize the rows so that Eq.(3.14) can hold.

### 3.2.2 Feature Initialization and Growing Dimensions

A common problem in feature-based SLAM is that the robot cannot initialize a feature in all its dimensions after the first observation. There can be many reasons to explain this:

- the feature is not completely visible: for example, this could be the case of a line, whose endpoints are not be visible because the line is long or the vision field is occluded;
- measurements are affected by noise, so having multiple readings of the same feature can be useful to improve feature initialization, and to reject false detections;
- in bearing-only feature detection, two successive measurements are needed to initialize a feature.

Since the entire line feature is not detected at once, the M-space coordinates  $\mathbf{x}_p$  dimensions  $p$  can grow with time from a minimum of 2, when the robot first measures the line distance and normal direction, up to 4 dimensions, when it has observed both line endpoints. The  $B_f$  matrix will have  $p$  rows, and  $\tilde{B}$  will

have  $p$  columns correspondingly. When an endpoint is measured and initialized in the M-Space, the relative row (column) of the projection matrix  $B_f$  ( $\tilde{B}_f$ ) will be used. These projection matrices change with time, since are always re-evaluated around the current estimate of  $\mathbf{x}_f$ .

### 3.3 SLAM

The M-Space representation can be adopted regardless of the SLAM algorithm used. The map estimation and the filter state vector are separated from the details of maintaining and updating the features. The SLAM problem will be described in the following and then analyzed formally in all its aspects: the robot motion model, the measurement model and the state definition. Finally the SLAM problem will be stated.

#### 3.3.1 General SLAM algorithm

Any state estimation technique can be adopted to address the Simultaneous Localization And Mapping problem. The SLAM algorithm purpose is to let a robot build a map of the environment and localize itself in the same map. The robot starts its exploration from a known location and travels according to the speed commands provided to it. The filter predicts a new robot pose and takes into account the uncertainty increased by this noisy motion. Since the features are considered to be static, their estimates and uncertainties are not changed in this phase. While moving the robot takes noisy measurements, for example using a laser range finder; the line features extracted from the laser raw data are matched with the features already in the filter state.

If a feature cannot be matched with the ones in the filter state, this new measurement is added to a tentative list of new features, in order to avoid augmenting the filter state with spurious readings. Data association is performed also for this tentative list, in order to collect more informations about the same tentative feature: if the tentative line is matched with

one already in the list, the point cloud that generated the new measurement is merged with the cloud stored in the list. After a tentative feature has been observed  $N$  times, it is assumed to be reliable, and it is promoted to become a new feature to augment the filter state.

If a measurement is matched with a feature already in the state, it can be used to update the filter state and uncertainty estimation. The SLAM algorithm cycles through these steps continuously.

### 3.3.2 Problem Formulation

Now the SLAM problem will be introduced formally. Let us consider an autonomous navigating robot constrained to move a plane (2D environment) and let  $\mathbf{x}_r(k) = [x_r(k) \ y_r(k) \ \theta_r(k)]^T$  be its pose, where  $(x_r \ y_r)$  is the position and  $\theta_r$  is the orientation with respect to a global frame of reference, as shown in Fig. 3.3. Assuming that the robot is navigating indoor, the surrounding environment can be described using line segments extracted from walls, doors, or furniture.

#### Robot Motion Model

A non-holonomic differential-drive robot pose can be estimated using relative motion odometry (integrating the readings from the wheels encoders) or using the information of the linear and angular velocity commands. The discrete-time unicycle motion model based on velocity commands  $\mathbf{u}(k) = [v(k) \ \omega(k)]^T$  is

$$\mathbf{x}_r(k+1) = f(\mathbf{x}_r(k), \mathbf{u}(k), \varepsilon_{\mathbf{u}}(k)) \quad (3.21)$$

$$\begin{cases} x_r(k+1) = x_r(k) + T_s \cos(\theta_r(k)) (v(k) + \varepsilon_v(k)) \\ y_r(k+1) = y_r(k) + T_s \sin(\theta_r(k)) (v(k) + \varepsilon_v(k)) \\ \theta_r(k+1) = \theta_r(k) + T_s (\omega(k) + \varepsilon_\omega(k)) \end{cases} \quad (3.22)$$

where  $\mathbf{x}_r(k) = [x_r(k) \ y_r(k) \ \theta_r(k)]^T \in \mathbb{R}^2 \times [-\pi, \pi)$  is the robot pose and  $T_s$  is the sampling time of the discrete-time system. The velocity commands are affected by a white Gaussian noise  $\varepsilon_{\mathbf{u}}$

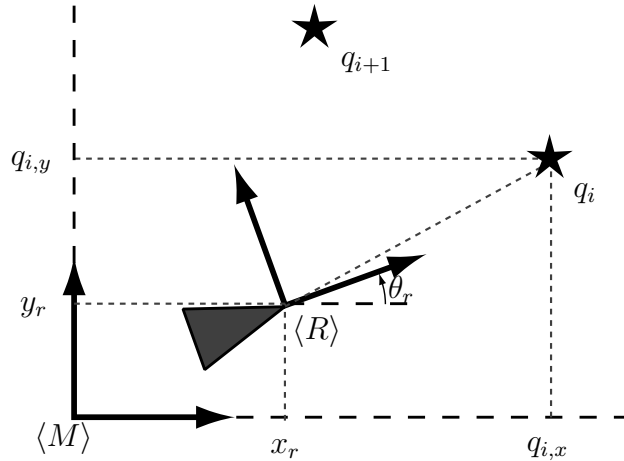
$$\begin{aligned}
\varepsilon_{\mathbf{u}}(k) &= [\varepsilon_v(k) \ \varepsilon_\omega(k)]^T \sim WGN(\mathbf{0}, Q) \\
E[\varepsilon_{\mathbf{u}}(k)] &= \mathbf{0} \\
E[\varepsilon_{\mathbf{u}}(k)\varepsilon_{\mathbf{u}}^T(k)] &= \text{diag}(\sigma_v^2, \sigma_\omega^2) = Q \\
\sigma_v &= \frac{\alpha_v |v(k)|}{3\sqrt{T_s}} \\
\sigma_\omega &= \frac{\alpha_\omega |\omega(k)| + \delta_\omega}{3\sqrt{T_s}}
\end{aligned} \tag{3.23}$$

where the standard deviations  $\sigma_v$  and  $\sigma_\omega$  are proportional to the actual commands by the percentage parameters  $\alpha_v$  and  $\alpha_\omega$ ; the offset  $\delta_\omega$  is introduced to take into account that the robot does not drive perfectly straight even if the command  $\omega(k) = 0$ .

The Jacobians  $J_{rr}$  and  $J_{ru}$  are

$$J_{rr}(k) = \frac{\partial f}{\partial \mathbf{x}_r} = \begin{bmatrix} \frac{\partial f}{\partial x_r} & \frac{\partial f}{\partial y_r} & \frac{\partial f}{\partial \theta_r} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -T_s v(k) \sin(\theta_r(k)) \\ 0 & 1 & T_s v(k) \cos(\theta_r(k)) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.24}$$

$$J_{ru}(k) = \frac{\partial f}{\partial \varepsilon_{\mathbf{u}}} = \begin{bmatrix} \frac{\partial f}{\partial \varepsilon_v} & \frac{\partial f}{\partial \varepsilon_\omega} \end{bmatrix} = \begin{bmatrix} T_s \cos(\theta_r(k)) & 0 \\ T_s \sin(\theta_r(k)) & 0 \\ 0 & T_s \end{bmatrix} \tag{3.25}$$



**Figure 3.3:** Robot pose and waypoint coordinates in global reference frame.

An example of how the velocity commands are generated is as follows: a set of waypoints  $\mathbf{WP} = \{\mathbf{q}_i = [q_{i,x} \ q_{i,y}]^T, i = 1, 2, \dots\}$  is provided to the robot as shown in Fig. 3.3. The linear speed  $v(k)$  is maintained constant, while the

angular speed  $\omega(k)$  is proportional to the error between the robot orientation and the bearing to the current waypoint, thus providing a closed loop control on robot heading.

$$\begin{cases} v(k) &= v(k-1) \geq 0 \quad \forall k \\ \omega(k) &= \mu (\text{atan2}(q_{i,y} - y_r, q_{i,x} - x_r) - \theta_r) \end{cases} \quad (3.26)$$

where  $\mu [s^{-1}]$  is the controller gain. When the distance of the robot from the current  $i$ -th waypoint goes below a certain threshold, the controller is switched to track the next waypoint  $q_{i+1} \in \mathbf{WP}$ . Since the waypoints positions are assumed to be perfectly known to the robot, it will track the waypoint-defined path with null steady-state error.

### Measurements Model

Let  $\mathbf{m}$  denote a measurement of a line segment in the robot frame extracted from the laser raw data, as shown in Fig. 3.4,

$$\mathbf{m}(k) = \begin{bmatrix} \alpha(k) \\ \rho(k) \\ d_A(k) \\ d_B(k) \end{bmatrix} + \varepsilon_{\mathbf{m}}(k) \quad (3.27)$$

affected by white Gaussian noise  $\varepsilon_{\mathbf{m}}(k)$

$$\begin{aligned} \varepsilon_{\mathbf{m}}(k) &= [\varepsilon_{\alpha} \ \varepsilon_{\rho} \ \varepsilon_{d_A} \ \varepsilon_{d_B}]^T \sim WGN(\mathbf{0}, R) \\ E[\varepsilon_{\mathbf{m}}(k)] &= \mathbf{0} \\ E[\varepsilon_{\mathbf{m}}(k) \varepsilon_{\mathbf{m}}^T(k)] &= \text{diag}(\sigma_{\alpha}^2, \sigma_{\rho}^2, \sigma_{d_A}^2, \sigma_{d_B}^2) = R \end{aligned} \quad (3.28)$$

In the following we are dealing with estimated feature coordinates in the robot frame  $\hat{\mathbf{x}}_o = [\hat{x}_A \ \hat{y}_A \ \hat{x}_B \ \hat{y}_B]^T$ , so the  $[\cdot]^r$  notation is omitted. The following quantities are useful:

$$\hat{d}_x = \hat{x}_A - \hat{x}_B \quad (3.29)$$

$$\hat{d}_y = \hat{y}_A - \hat{y}_B \quad (3.30)$$

$$\hat{L} = \sqrt{\hat{d}_x^2 + \hat{d}_y^2} \quad (3.31)$$

The innovation function can be defined in many ways, for example as in [8]. Here, the innovation function  $\eta(\mathbf{m}, \hat{\mathbf{x}}_o)$  is defined as the difference between

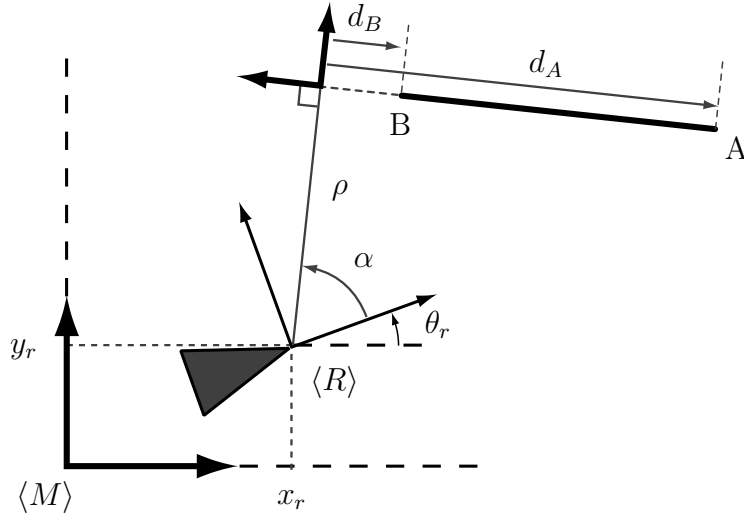


Figure 3.4: Line parameters.

line feature parameters  $\mathbf{m}$  extracted from laser raw data and the expected line feature parameters corresponding to the line endpoints coordinates  $\hat{\mathbf{x}}_o$  in robot frame,

$$\eta = \mathbf{m} - h(\hat{\mathbf{x}}_o) = \mathbf{m} - \begin{bmatrix} \hat{\alpha} \\ \hat{\rho} \\ \hat{d}_A \\ \hat{d}_B \end{bmatrix} \quad (3.32)$$

where

$$h(\hat{\mathbf{x}}_o) = \begin{bmatrix} \hat{\alpha} \\ \hat{\rho} \\ \hat{d}_A \\ \hat{d}_B \end{bmatrix} = \begin{bmatrix} \text{atan2}(-\hat{d}_x, \hat{d}_y) \\ \hat{x}_A \cos(\hat{\alpha}) + \hat{y}_A \sin(\hat{\alpha}) \\ -\hat{x}_A \sin(\hat{\alpha}) + \hat{y}_A \cos(\hat{\alpha}) \\ -\hat{x}_B \sin(\hat{\alpha}) + \hat{y}_B \cos(\hat{\alpha}) \end{bmatrix}, \quad (3.33)$$

which has the same form as Eq. (3.10). The line distance from origin  $\hat{\rho}$  can be equivalently written as  $\hat{\rho} = \hat{x}_B \cos(\hat{\alpha}) + \hat{y}_B \sin(\hat{\alpha})$ , and this is useful later to simplify the expression of the Jacobian  $J_{\eta o}$ .

The Jacobian  $J_{\eta o}$  of  $\eta(\mathbf{m}, \mathbf{x}_o)$  is

$$J_{\eta o} = \frac{\partial \eta}{\partial \mathbf{x}_o} = - \left. \frac{\partial h(\mathbf{x}_o)}{\partial \mathbf{x}_o} \right|_{\mathbf{x}_o = \hat{\mathbf{x}}_o} \quad (3.34)$$

where

$$\begin{aligned} \frac{\partial h(\mathbf{x}_o)}{\partial \mathbf{x}_o} &= \left[ \frac{\partial h}{\partial x_A} \quad \frac{\partial h}{\partial y_A} \quad \frac{\partial h}{\partial x_B} \quad \frac{\partial h}{\partial y_B} \right] = \\ &= \frac{1}{L^2} \begin{bmatrix} -dy & dx & dy & -dx \\ -dy d_B & dx d_B & dy d_A & -dx d_A \\ \rho dy - L^2 \sin(\alpha) & -\rho dx + L^2 \cos(\alpha) & -\rho dy & \rho dx \\ \rho dy & -\rho dx & -\rho dy - L^2 \sin(\alpha) & \rho dx + L^2 \cos(\alpha) \end{bmatrix} \end{aligned} \quad (3.35)$$

Small changes of robot pose  $\delta \mathbf{x}_r$  and M-Space coordinates  $\delta \mathbf{x}_p$  cause a small change in expected feature coordinates  $\delta \hat{\mathbf{x}}_o$ ,

$$\begin{aligned} \delta \hat{\mathbf{x}}_o &= \frac{\partial \mathbf{x}_o}{\partial \mathbf{x}_r} \delta \mathbf{x}_r + \frac{\partial \mathbf{x}_o}{\partial \mathbf{x}_p} \delta \mathbf{x}_p = \\ &= \frac{\partial \mathbf{x}_o}{\partial \mathbf{x}_r} \delta \mathbf{x}_r + \frac{\partial \mathbf{x}_o}{\partial \mathbf{x}_f} \frac{\partial \mathbf{x}_f}{\partial \mathbf{x}_p} \delta \mathbf{x}_p = \\ &= [J_{or} \quad J_{of} \tilde{B}_f] \begin{bmatrix} \delta \mathbf{x}_r \\ \delta \mathbf{x}_p \end{bmatrix}, \end{aligned} \quad (3.36)$$

This variation on expected local feature coordinates reflects on the innovation  $\delta \eta$  by

$$\delta \eta = J_{\eta o} \delta \hat{\mathbf{x}}_o = J_{\eta o} [J_{or} \quad J_{of} \tilde{B}_f] \begin{bmatrix} \delta \mathbf{x}_r \\ \delta \mathbf{x}_p \end{bmatrix}. \quad (3.37)$$

For the complete expression of  $J_{or}$ ,  $J_{of}$  and  $\tilde{B}_f$ , see Eqs. (3.7), (3.8) and (3.18).

### State definition

When using M-Space representation, the state estimation filter uses a state vector that includes robot pose and M-Space coordinates, while the geometric feature representation (3D, 2D, scalar coordinates) is kept separately. So the state vector to be estimated is

$$\xi(k) \in \mathbb{R}^{3+4n} : \xi(k) = [ \mathbf{x}_r^T(k), \mathbf{x}_{f_1}^T(k), \dots, \mathbf{x}_{f_n}^T(k) ]^T, \quad (3.38)$$

where  $\mathbf{x}_{f_i}$  is the  $i$ -th feature ( $i = 1 \dots n$ ) described by its endpoints in the global frame (Eq. (3.1)), and  $n$  is the number of features in the state vector. Since static features are considered, the time evolution of the state vector only affects the robot pose  $\mathbf{x}_r$  as by (3.22), leaving the features  $\mathbf{x}_f$  unchanged. Now, the SLAM problem can be stated as follows.

**SLAM problem.** Let  $\hat{\xi}(0)$  be an estimate of the initial robot pose and feature coordinates. Given the dynamic model (3.22) and the measurement (3.27), find an estimate  $\hat{\xi}(k)$  of the robot pose and feature coordinates  $\xi(k)$  for each  $k \in \mathbb{N}_+$ .

### 3.4 EKF Implementation

As already told, any state estimation technique can be used together with M-Space representation. A widely used estimation technique is the Extended Kalman Filter (EKF). The state of the filter is defined as

$$\hat{\mathbf{x}}_{state} \triangleq \begin{bmatrix} \hat{\mathbf{x}}_r \\ \hat{\mathbf{x}}_p \end{bmatrix}, \quad (3.39)$$

$$\hat{\mathbf{x}}_p = [\hat{\mathbf{x}}_{p_1}^T \dots \hat{\mathbf{x}}_{p_n}^T]^T \quad (3.40)$$

where  $\mathbf{x}_{p_i}$   $i = 1 \dots n$  are the M-Space parameters of the  $i$ -th feature. A state vector  $\xi(k)$  (3.38) external to the filter is used to hold the actual feature coordinates  $\mathbf{x}_f$ . The state estimation error covariance matrix  $P_{\mathbf{x}}$  is composed by blocks

$$P_{\mathbf{x}}(k) = \begin{bmatrix} P_{rr}(k) & P_{rp}(k) \\ P_{pr}(k) & P_{pp}(k) \end{bmatrix}, \quad (3.41)$$

where

$$P_{rr}(k) = E[\tilde{\mathbf{x}}_r(k) \tilde{\mathbf{x}}_r^T(k)] \quad (3.42)$$

$$P_{rp}(k) = E[\tilde{\mathbf{x}}_r(k) \tilde{\mathbf{x}}_p^T(k)] \quad (3.43)$$

$$P_{pr}(k) = E[\tilde{\mathbf{x}}_p(k) \tilde{\mathbf{x}}_r^T(k)] = P_{rp}^T(k) \quad (3.44)$$

$$P_{pp}(k) = E[\tilde{\mathbf{x}}_p(k) \tilde{\mathbf{x}}_p^T(k)] \quad (3.45)$$

$$\tilde{\mathbf{x}}_r(k) = \mathbf{x}_r(k) - \hat{\mathbf{x}}_r(k) \quad (3.46)$$

$$\tilde{\mathbf{x}}_p(k) = \mathbf{x}_p(k) - \hat{\mathbf{x}}_p(k) \quad (3.47)$$

and  $E[\cdot]$  is the expected value operator.

The EKF alternates between prediction and update steps:

### ✦ Prediction step:

The new robot pose is predicted according to (3.22) where the commands are given by (3.26)

$$\hat{\mathbf{x}}_r(k+1|k) = f(\hat{\mathbf{x}}_r(k|k), \mathbf{u}(k), 0) \quad (3.48)$$

The feature coordinates expressed in robot frame are computed from the feature coordinates in the external state  $\xi(k)$  and predicted robot pose  $\hat{\mathbf{x}}_r(k+1|k)$  using (3.5)

$$\hat{\mathbf{x}}_o(k+1|k) = \hat{\mathbf{x}}_f(k) \ominus \hat{\mathbf{x}}_r(k+1|k) \quad (3.49)$$

The covariance matrix  $P_{\mathbf{x}}$  is predicted block-wise using the speed noise covariance matrix  $Q$  (3.23), the Jacobians  $J_{rr}$  (3.24) and  $J_{ru}$  (3.25), leaving the  $P_{pp}$  block unchanged (static features)

$$P_{rr}(k+1|k) = J_{rr}(k)P_{rr}(k|k)J_{rr}^T(k) + J_{ru}(k)QJ_{ru}^T(k) \quad (3.50)$$

$$P_{rp}(k+1|k) = J_{rr}(k)P_{rp}(k|k) = P_{pr}^T(k+1|k) \quad (3.51)$$

$$P_{pp}(k+1|k) = P_{pp}(k|k) \quad (3.52)$$

### ✦ Correction step:

In the following, assume that the features in the EKF state have reached the full M-Space dimension of 4 (see section (3.2.2)). The EKF can use batch update, processing all the measurements at the same time, or iterative update, processing the measurements at time  $k$  one after another. Consider the latter approach: if the  $i$ -th feature  $\mathbf{x}_{f_i}$  is being processed, and  $p$  is the M-Space dimension corresponding to the measurement  $\mathbf{m}(k)$  (varying from 2 to 4), then  $\eta(k) \in \mathbb{R}^p$  and

$$\begin{aligned} J_{\eta \mathbf{x}} &= \begin{bmatrix} \frac{\partial \eta}{\partial \mathbf{x}_r} & \frac{\partial \eta}{\partial \mathbf{x}_p} \end{bmatrix} = \\ &= \begin{bmatrix} \frac{\partial \eta}{\partial \mathbf{x}_o} \frac{\partial \mathbf{x}_o}{\partial \mathbf{x}_r} & \frac{\partial \eta}{\partial \mathbf{x}_o} \frac{\partial \mathbf{x}_o}{\partial \mathbf{x}_f} \frac{\partial \mathbf{x}_f}{\partial \mathbf{x}_p} \end{bmatrix} = \\ &= J_{\eta o} [J_{or} \quad J_{of} \tilde{B}_f] \in \mathbb{R}^{p \times (3+p)} \end{aligned} \quad (3.53)$$

is the Jacobian of the innovation  $\eta$  (3.32) w.r.t.  $\mathbf{x}_{state}$ . This Jacobian relates changes in the state to changes in the innovation, as in Eq. (3.37); it is obtained computing (3.34), (3.8) and (3.18) around the estimates  $\hat{\mathbf{x}}_{f_i}(k)$  and  $\hat{\mathbf{x}}_r(k+1|k)$ .

The state Jacobian  $J_{\eta\mathbf{x}}(k)$  is enlarged to reach full state column dimension

$$\bar{J} = [J_{\eta o} J_{or} \mathbf{0}_{p \times (4i-4)} \quad J_{\eta o} J_{of} \tilde{B}_f \quad \mathbf{0}_{p \times (4n-4i)}] \in \mathbb{R}^{p \times (3+4n)} \quad (3.54)$$

where  $(4-p)$  columns are added to the  $\tilde{B}_f \in \mathbb{R}^{4 \times p}$  matrix to reach the full state dimensions  $3+4n$ .

The state vector correction is computed using the innovation  $\eta(k)$  (3.32) as

$$\delta \mathbf{x}_{state}(k) = -K(k) \eta(\hat{\mathbf{x}}_o(k+1|k), \mathbf{m}(k)). \quad (3.55)$$

The matrix  $K(k)$  is the Kalman gain

$$K(k) = P_{\mathbf{x}}(k+1|k) \bar{J}^T S^{-1}(k) \in \mathbb{R}^{(3+4n) \times p} \quad (3.56)$$

$$S(k) = (\bar{J} P_{\mathbf{x}}(k+1|k) \bar{J}^T + R) \quad (3.57)$$

where  $R$  is the measurement noise covariance matrix reduced to  $p \times p$  dimensions (see (3.28)). Notice that eq. (3.55) does not need the actual values of the M-Space parameters included the state  $\hat{\mathbf{x}}_{state}$  (3.39); only the changes  $\delta \mathbf{x}_p$  are used to update the filter.

The external state vector estimate  $\hat{\xi}(k+1)$  is updated using the robot pose update directly

$$\hat{\mathbf{x}}_r(t+1|t+1) = \hat{\mathbf{x}}_r(k+1|k) + \delta \hat{\mathbf{x}}_r(k) \quad (3.58)$$

and by projecting the updated M-Space coordinates  $\delta \mathbf{x}_p(k)$  into the feature space,

$$\hat{\mathbf{x}}_f(k+1) = \hat{\mathbf{x}}_f(k) + \tilde{\mathbf{B}}_f(\hat{\mathbf{x}}_f(k)) \delta \mathbf{x}_p(k) \quad (3.59)$$

where

$$\tilde{\mathbf{B}}_f(\hat{\mathbf{x}}_f(k)) = \begin{bmatrix} \tilde{B}_f(\hat{\mathbf{x}}_{f_1}) & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \tilde{B}_f(\hat{\mathbf{x}}_{f_n}) \end{bmatrix} \quad (3.60)$$

is a block diagonal matrix built using the  $\tilde{B}_f(\hat{\mathbf{x}}_{f_i})$  matrices computed around the current feature estimates  $\hat{\mathbf{x}}_{f_i}$ .

The estimation error covariance matrix correction is

$$\delta P_{\mathbf{x}}(k+1|k+1) = K(k)S(k)K^T(k) \quad (3.61)$$

Finally the estimation error covariance is updated by

$$P_{\mathbf{x}}(k+1|k+1) = P_{\mathbf{x}}(k+1|k) - \delta P_{\mathbf{x}}(k+1|k+1). \quad (3.62)$$

## 3.5 Feature Extraction

The purpose of feature extraction is to obtain the measurements  $\mathbf{m}(k)$  (3.27) and the corresponding covariance matrix  $R(k)$  (3.28) from the raw data provided by the sensor. The feature extraction described in the following is based on the work [12]. The robot is equipped with a laser range finder that produces a data set in the form

$$\{[\phi_j, d_j]^T, i = 0, \dots, N\} \quad (3.63)$$

where  $d_j$  is the distance of the point sensed along the direction  $\phi_j$ . Let

$$C_m = \begin{bmatrix} \sigma_\phi^2 & 0 \\ 0 & \sigma_d^2 \end{bmatrix} \quad (3.64)$$

be the covariance matrix of the white Gaussian noise affecting the sensor readings.

### 3.5.1 Algorithm overview

The sensor readings are processed in order to extract the parameters  $[\alpha, \rho, d_A, d_B]^T$  of the linear features. The pair  $[\phi_j, d_j]^T$  expresses  $j$ -th point in polar coordinates w.r.t. sensor (robot) frame. Let

$$p_j = [x_j, y_j]^T = [d_j \cos(\phi_j), d_j \sin(\phi_j)]^T \quad (3.65)$$

be the Cartesian coordinates of the  $j$ -th point w.r.t. the same frame. The points are processed sequentially, i.e. they are ordered according to ascending values of  $\phi$ . The segmentation phase consists in grouping the sensor readings into subsets  $S_h$  (segments) of points respecting a proximity and collinearity criteria:  $S_h = \{p_1^{(h)}, \dots, p_{n_h}^{(h)}\}$ ,  $h = 1, \dots, q$ , where  $n_h$  is the cardinality of the  $h$ -th segment. Each set is built iteratively: the first two points are used to initialize the segment  $S_1$ . Then a point  $p_j$  is added to the current segment if it respects the following criterion:

- (c1) its normal distance from the current fitting line is below a threshold  $\delta_0$ ,  
and
- (c2) its Euclidean distance from the last point in current segment is below a threshold  $\delta_1$ .

When a new point is added to the current line segment, the parameters of the fitting line are recomputed on the basis of the grown set of points. If the point does not meet the above criterion, the segment  $S_h$  is assumed to be complete and it is stored among the measurements  $\mathbf{m}(k)$ . The new segment  $S_{h+1}$  is instantiated starting from the last point  $p_{n_h}^{(h)}$  of segment  $S_h$  if this point  $p_{n_h}^{(h)}$  did not meet (c2), or starting from next point in laser scan if the point  $p_{n_h}^{(h)}$  did not meet (c1). This condition that determines the end of a segment allows to filter away spurious readings (outlier points) due to the flat surfaces roughnesses, indentations and occlusions. Furthermore, segments  $S_h$  whose length (given by the distance between the first  $p_1^{(h)}$  and last  $p_{n_h}^{(h)}$  points) is

shorter than a minimum length  $L_0$  or segments made up of less than  $N_0$  points are deemed unreliable and discarded.

It may happen that, while processing the  $h$ -th segment  $S_h$ , the next point polar distance  $d_j$  is greater than the laser usable range, heuristically defined as  $d_{max} = d_{range} - 5\sqrt{\sigma_d^2}$ . In this case, the segment  $S_h$  is saved into  $\mathbf{m}(k)$  if it meets the above criteria; similarly, a segment is saved if the next point  $p_j$  is the last point  $p_N$  in the laser scan.

### 3.5.2 Line fitting

The parameters to describe a line segment are in the known form  $m = [\alpha, \rho, d_A, d_B]^T$ .

Let us consider  $N$  points  $p_i = [x_i, y_i]^T, i = 1, \dots, N$  on a plane. The line  $m = [\alpha^*, \rho^*, d_A^*, d_B^*]^T$  minimizing the sum of the squared normal distances from each point  $E(\alpha, \rho) = \sum_{i=1}^N (\rho - x_i \cos(\alpha) - y_i \sin(\alpha))^2$  is given by:

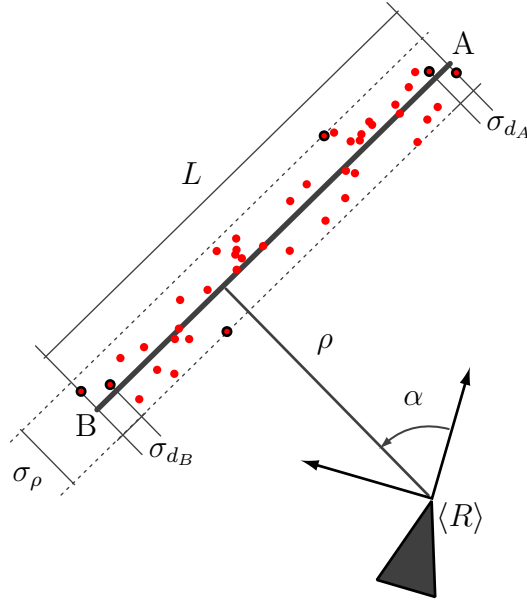
$$\begin{bmatrix} \alpha^* \\ \rho^* \\ d_A^* \\ d_B^* \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \arctan\left(\frac{-2S_{xy}}{S_{y^2} - S_{x^2}}\right) \\ \bar{x} \cos(\alpha^*) + \bar{y} \sin(\alpha^*) \\ -\sin(\alpha^*)x_1 + \cos(\alpha^*)x_1 \\ -\sin(\alpha^*)x_N + \cos(\alpha^*)x_N \end{bmatrix} \triangleq \mathbf{f}_m(x_1, \dots, x_N) \quad (3.66)$$

where

$$\begin{aligned} x &= \frac{1}{N} \sum_{i=1}^N x_i, & y &= \frac{1}{N} \sum_{i=1}^N y_i, \\ S_{x^2} &= \sum_{i=1}^N (x_i - \bar{x})^2, & S_{y^2} &= \sum_{i=1}^N (y_i - \bar{y})^2, \\ S_{xy} &= \sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y}). \end{aligned}$$

In [12] the measurement noise covariance matrix is computed summing the laser covariance matrix (3.64) projected in the line parameter space using the Jacobians of (3.66):  $R = \sum_{i=1}^N J_i C_m J_i^T$ , where  $J_i = \begin{bmatrix} \frac{\partial \mathbf{f}_m}{\partial \phi_i} & \frac{\partial \mathbf{f}_m}{\partial d_i} \end{bmatrix}$ .

Here, the measurement covariance matrix  $R = \text{diag}(\sigma_\alpha^2, \sigma_\rho^2, \sigma_{d_A}^2, \sigma_{d_B}^2)$  diagonal elements are estimated as follows. As shown in Fig. 3.5, the variance of the line normal distance from the origin of the sensor frame  $\sigma_\rho^2$  is estimated as the squared spread of the points used to fit the line segment along the line



**Figure 3.5:** Estimated measurement variances.

normal direction. The variance of the line orientation  $\sigma_\alpha^2$  is estimated dividing the computed  $\sigma_\rho^2$  by the squared line length. The variance of the endpoint distance error  $\sigma_{d_A}^2$  could be estimated as the squared distance along the line direction of the first two points used for fitting; similarly could be done for  $\sigma_{d_B}^2$ . This heuristic for endpoints yields to overestimate the measurement precision, so it is preferable to estimate both  $\sigma_{d_A}^2$  and  $\sigma_{d_B}^2$  as the squared ratio  $(3L/N)^2$  between line length and the number of points used for fitting the line. These estimates are quite conservative, so the EKF will tend to underestimate the accuracy of the measurements.

## 3.6 Data Association

At time  $k$ , a set of measurements are extracted from laser raw data,  $\mathbf{m} = \{m_i, i = 1, \dots, n_{\mathbf{m}}\}$ . These measurements do not provide informations on the identity of the features, that are indistinguishable. In order to perform the EKF correction step, each measurement  $m_i(k)$  must be associated to the corresponding line feature  $\mathbf{x}_f$  in the state vector  $\xi(k)$ . This matching problem

consist in determining the feature  $\mathbf{x}_{f_j}$  (if it exists) in the state vector, that produces the  $i$ -th measurement.

In this work, a Nearest Neighbour (NN) approach based on Mahalanobis distance is used. Intuitively, it reduces to compare the  $i$ -th measurement with each feature estimate and associate the measurement to the “closest” one, in terms of covariance-weighted distance. Having  $n_{\mathbf{m}}$  measurements to compare against  $n$  features already in the state, the NN algorithm complexity is  $O(n \cdot n_{\mathbf{m}})$ .

Under the hypothesis that  $j$ -th feature has originated the  $i$ -th measurement, the error is

$$e = m_i - h(\hat{\mathbf{x}}_{o,j}). \quad (3.67)$$

The form of this error is similar to the innovation of Eq. (3.32). Since the real endpoints of the line could be unknown (M-Space dimension  $p = 2$ ), the distinct Mahalanobis distances  $D_{\alpha,ij}$  and  $D_{\rho,ij}$  and cumulative distance  $D_{C,ij}$

are computed as

$$e_\alpha = \alpha_m - \hat{\alpha} \quad (3.68)$$

$$D_{\alpha,ij} = e_\alpha (\Gamma_\alpha S \Gamma_\alpha^T)^{-1} e_\alpha \quad (3.69)$$

$$e_\rho = \rho_m - \hat{\rho} \quad (3.70)$$

$$D_{\rho,ij} = e_\rho (\Gamma_\rho S \Gamma_\rho^T)^{-1} e_\rho \quad (3.71)$$

$$e_C = [\alpha_m - \hat{\alpha}, \rho_m - \hat{\rho}] \quad (3.72)$$

$$D_{C,ij} = e_C (\Gamma_C S \Gamma_C^T)^{-1} e_C^T \quad (3.73)$$

$$\Gamma_\alpha = [1 \ 0 \ 0 \ 0] \quad (3.74)$$

$$\Gamma_\rho = [0 \ 1 \ 0 \ 0] \quad (3.75)$$

$$\Gamma_C = [\Gamma_\alpha^T \ \Gamma_\rho^T]^T \quad (3.76)$$

$$S = R + H P_x H^T \quad (3.77)$$

$$H = -[J_{\eta_o} J_{or} \ \mathbf{0}_{4 \times (4j-4)} \ J_{\eta_o} J_{of} \tilde{B}_f \ \mathbf{0}_{4 \times (4n-4j)}] \quad (3.78)$$

where  $P_x$  is the estimation error covariance matrix (3.41),  $H$  is the Jacobian of  $e$  w.r.t. the EKF filter state (see (3.54)), and  $R$  is the measurement covariance (see (3.28)).

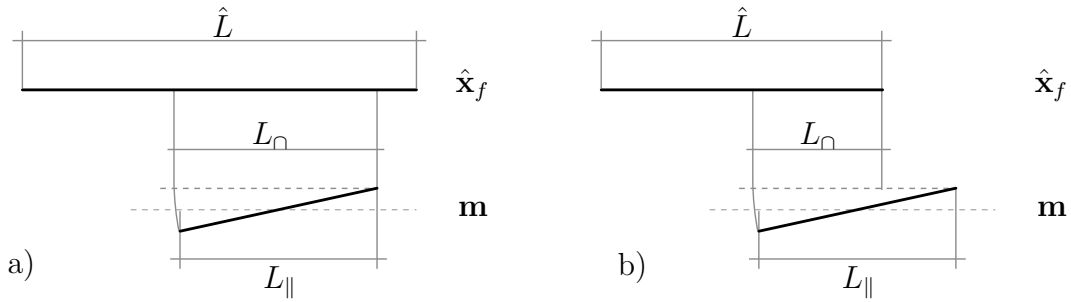
Since different features in the environment may lie on the same line (as two walls divided by a door), another parameter to check is the overlapping rate

$\tau_{ij}$  between the measured line and the estimated one, as shown in Fig. 3.6,

$$\tau_{ij} = \begin{cases} \frac{L_{\cap}}{\min(\hat{L}, L_{\parallel})} & \text{if } (\hat{\epsilon}_{max} \geq \epsilon_{min} \wedge \epsilon_{max} \geq \hat{\epsilon}_{min}) \\ 0 & \text{otherwise} \end{cases}$$

where

$$\begin{aligned} \epsilon_{max} &= \max(d_A, d_B) \\ \epsilon_{min} &= \min(d_A, d_B) \\ \hat{\epsilon}_{max} &= \max(\hat{d}_A, \hat{d}_B) \\ \hat{\epsilon}_{min} &= \min(\hat{d}_A, \hat{d}_B) \\ L_{\cap} &= \min(\hat{\epsilon}_{max}, \epsilon_{max}) - \max(\hat{\epsilon}_{min}, \epsilon_{min}) \\ \hat{L} &= |\hat{d}_A - \hat{d}_B| \\ L_{\parallel} &= |d_A - d_B| \cos(\alpha - \hat{\alpha}) \end{aligned} \quad (3.79)$$



**Figure 3.6:** Measured and estimated feature overlapping: a)  $\tau = 1$ ; b)  $\tau < 1$ .

Each  $i$ -th measurement is compared to all the  $j$ -th feature parameters estimate and three tests are performed:

- weighted error on  $\rho$  must be less than a certain threshold,  $D_{\rho,ij} < T_{\rho}$
- weighted error on  $\alpha$  must be less than a certain threshold,  $D_{\alpha,ij} < T_{\alpha}$
- overlapping rate must be greater than a certain threshold,  $1 \geq \tau_{ij} > T_{\tau}$

Then, among all the correspondent features, the one minimizing  $\min_j(D_{C,ij})$  is selected. If the measurement  $m_i(k)$  is associated with the  $j$ -th feature, its endpoints are suitably updated (actually a union is performed, so the line feature will always grow in length) only if M-Space dimension  $p < 3$ ; the line endpoints are updated via EKF otherwise.

### State Augmentation

If the measurement does not match any of the lines currently present in the state vector, it has to be considered as a new feature, and the state vector should be properly augmented. The new feature  $\mathbf{x}_f^*$  (in form of endpoints coordinates) is added to the state vector  $\hat{\xi}(k|k)$ .

The covariance matrix  $P_{\mathbf{x}}$  is augmented with a large diagonal matrix  $P_{pp}^* \in \mathbb{R}^{4 \times 4}$ ; the new feature is correlated with the other features and robot pose by performing an EKF correction step like (3.61), (3.62), after computing the EKF state Jacobian (3.54) around  $\mathbf{x}_f^*$ .

### Delayed Feature Initialization

To avoid augmenting the state vector with spurious features, an unmatched measurement is first inserted into a list of tentative features; when it is detected a certain number of times over a time interval, it is considered reliable and used to augment the state vector. The tentative features are inserted into the list as a cloud of points measured by the range finder. The clouds are sets of 2D coordinates expressed in the robot frame of reference; since the robot is moving and the clouds are referred to static features, they are updated every time the robot is moving. In order to associate any unmatched measurement to a feature already in the tentative list a matching procedure like the one described above is needed. The new measurements are tested against the parameters got clouds fitting, without considering the covariances: if a match is found, the new cloud of points is added to the cloud already in the list, otherwise the new cloud is inserted as a new item into the list.

# Chapter 4

## Map fusion

In chapter 3, a single-robot SLAM algorithm based on EKF and M-Space feature representation has been presented. This chapter describes an algorithm to fuse the maps built by different robots, assuming that the coordinate transformation (*initial correspondence*) between the initial poses of the robots is unknown. The EKF SLAM algorithm and the map fusion algorithm are used together to perform multi-robot SLAM. In order to fuse maps created by different robots, the transformation between their reference frames needs to be determined. This can be done in two ways: searching for landmark matches in the two overlapping maps, and using as the most probable transformation the one that produces the maximum number of landmark correspondences; or using robot-to-robot mutual measurements for computing the unknown coordinate transformation.

This work is based on the approach proposed in [30] [31]: the method is illustrated for a two-robot case in a 2D environment, but can be easily extended to the case of larger robot teams. First, the two maps are aligned by employing the transformation (rotation and translation) determined by the robot-to-robot measurements (section 4.1). The estimation errors covariance of the maps is then updated according to the transformation employed to express all the maps with regard to the same reference frame (sections 4.2 and 4.3). Then, possible overlap between the two maps is examined by searching for landmark matches. If landmark duplicates are identified, this information

is used to impose constraints that improve the accuracy of the resulting map (section 4.4).

## 4.1 Map Alignment

In this section, we address the problem of map merging, that is aligning the robots' independently created maps to build a single global map of the environment: this method requires that the robots meet at least once. This can be a random event or can be arranged by the two robots. The map alignment problem is solved by processing mutual *relative distance and bearing measurements*, when the two robots are within sensing range of each other. The transformation between the two maps is computed using this pair of robot-to-robot measurements. If the maps created by the two robots overlap, duplicate landmarks can be identified in the merged map. By imposing constraints on the positions of matched landmarks, the quality of the resulting map is improved. Once the two maps are merged, the two robots can continue to cooperatively explore their environment while expressing all new measurements with respect to the common frame of reference (centralized approach), or continue performing single-robot SLAM using the new merged map until they eventually meet again.

### 4.1.1 Data Produced by EKF SLAM algorithms

Suppose there are two robots  $R_1$  and  $R_2$  that have explored and mapped independently the area they have travelled through with respect to their initial global frame,  $\langle G_1 \rangle$  and  $\langle G_2 \rangle$  respectively. The origins of the global frames are usually set to coincide with the robots' starting locations.

Initially, each of the two robots performs SLAM independently, i.e. the

SLAM algorithms of the robots  $R_1$  and  $R_2$  estimate the vectors

$$\begin{aligned} G_1 \mathbf{X}_1 &= \begin{bmatrix} G_1 \mathbf{x}_{R_1} \\ G_1 \mathbf{x}_{F_{M_1}} \end{bmatrix} \\ G_2 \mathbf{X}_2 &= \begin{bmatrix} G_2 \mathbf{x}_{R_2} \\ G_2 \mathbf{x}_{f_{M_2}} \end{bmatrix} \end{aligned} \quad (4.1)$$

respectively, with

$$G_k \mathbf{x}_{R_k} = \begin{bmatrix} G_k \mathbf{p}_{R_k} \\ \phi_k \end{bmatrix} \quad G_k \mathbf{p}_{R_k} = \begin{bmatrix} x_k \\ y_k \end{bmatrix}, \quad k = 1, 2 \quad (4.2)$$

$$G_1 \mathbf{x}_{F_{M_1}} = \begin{bmatrix} G_1 \mathbf{x}_{F_1} \\ \vdots \\ G_1 \mathbf{x}_{F_i} \\ \vdots \\ G_1 \mathbf{x}_{F_{n_1}} \end{bmatrix} \quad G_1 \mathbf{x}_{F_i} = \begin{bmatrix} x_{A_{F_i}} \\ y_{A_{F_i}} \\ x_{B_{F_i}} \\ y_{B_{F_i}} \end{bmatrix} \in M_1, \quad i = 1 \dots n_1 \quad (4.3)$$

$$G_2 \mathbf{x}_{f_{M_2}} = \begin{bmatrix} G_2 \mathbf{x}_{f_1} \\ \vdots \\ G_2 \mathbf{x}_{f_j} \\ \vdots \\ G_2 \mathbf{x}_{f_{n_2}} \end{bmatrix} \quad G_2 \mathbf{x}_{f_j} = \begin{bmatrix} x_{A_{f_j}} \\ y_{A_{f_j}} \\ x_{B_{f_j}} \\ y_{B_{f_j}} \end{bmatrix} \in M_2, \quad j = 1 \dots n_2 \quad (4.4)$$

where  $n_1$  ( $n_2$ ) is the total number of landmarks in the map  $M_1$  ( $M_2$ ) created by robot  $R_1$  ( $R_2$ ). For robot  $R_1$ , the state vector produced by the single-robot SLAM algorithm is  $G_1 \mathbf{X}_1 \in \mathbb{R}^{m_1}$ ,  $m_1 = 3 + 4n_1$ , and for robot  $R_2$  the state vector is  $G_2 \mathbf{X}_2 \in \mathbb{R}^{m_2}$ ,  $m_2 = 3 + 4n_2$ .

#### 4.1.2 Relative Distance and Bearing Measurements

We assume that each robot has a range sensor (laser) which can measure the distance  $\rho$  and bearing  $\theta$  towards a target. The measurement of the relative position of robot  $j$  with respect to robot  $i$  is described by:

$${}^i \mathbf{z}_j = \begin{bmatrix} {}^i \rho_m \\ {}^i \theta_{jm} \end{bmatrix} = \begin{bmatrix} \rho \\ {}^i \theta_j \end{bmatrix} + \begin{bmatrix} \varepsilon_{i\rho} \\ \varepsilon_{i\theta_j} \end{bmatrix} \quad i, j = 1, 2$$

where  $\rho$  is the distance between the two robots,  ${}^i \theta_j$  is the direction in which robot  $R_i$  sees robot  $R_j$ , and  $\varepsilon_{i\rho}$ ,  $\varepsilon_{i\theta_j}$  are white zero-mean Gaussian noise processes with variances  $\sigma_{i\rho}^2$  and  $\sigma_{i\theta_j}^2$  respectively. Since the two distance measurements  ${}^1 \rho_m$  and  ${}^2 \rho_m$  are independent, a more accurate estimate of the distance

$\rho$  between the two robots can be computed as the weighted average of the two individual measurements,

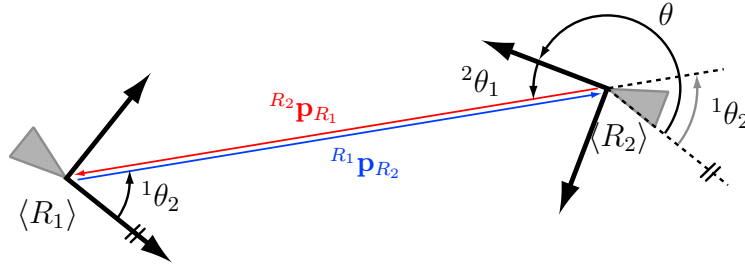
$$\begin{aligned} \rho_m &= \sigma_\rho^2 \left( \frac{{}^1\rho_m}{\sigma_{1\rho}^2} + \frac{{}^2\rho_m}{\sigma_{2\rho}^2} \right), \quad \frac{1}{\sigma_\rho^2} = \frac{1}{\sigma_{1\rho}^2} + \frac{1}{\sigma_{2\rho}^2} \\ \Rightarrow \rho_m &= \frac{\sigma_{2\rho}^2 {}^1\rho_m}{\sigma_{1\rho}^2 + \sigma_{2\rho}^2} + \frac{\sigma_{1\rho}^2 {}^2\rho_m}{\sigma_{1\rho}^2 + \sigma_{2\rho}^2} \end{aligned} \quad (4.5)$$

We form the combined measurement vector as

$$\mathbf{z} = \begin{bmatrix} \rho_m \\ {}^1\theta_{2m} \\ {}^2\theta_{1m} \end{bmatrix} = \begin{bmatrix} \rho \\ {}^1\theta_2 \\ {}^2\theta_1 \end{bmatrix} + \begin{bmatrix} \varepsilon_\rho \\ \varepsilon_{1\theta_2} \\ \varepsilon_{2\theta_1} \end{bmatrix} = \mathbf{z}_t + \varepsilon_z \quad (4.6)$$

$$R_z = E[\varepsilon_z \varepsilon_z^T] = \text{diag}(\sigma_\rho^2, \sigma_{1\theta_2}^2, \sigma_{2\theta_1}^2) \quad (4.7)$$

where  $\mathbf{z}_t$  denotes the vector of the real distance and relative bearings.



**Figure 4.1:** Robot-to-robot measurements.

As depicted in Fig. 4.1, the following relationship holds:

$${}^{R_1}\mathbf{p}_{R_2} = -{}^{R_1}_{R_2}\mathbf{C}(\theta){}^{R_2}\mathbf{p}_{R_1} \quad (4.8)$$

with

$${}^{R_j}\mathbf{p}_{R_i} = \rho \begin{bmatrix} \cos({}^j\theta_i) \\ \sin({}^j\theta_i) \end{bmatrix}, \quad i, j = 1, 2 \quad (4.9)$$

where  ${}^{R_j}\mathbf{p}_{R_i}$  is the position of robot  $R_i$  in robot  $R_j$ 's local frame, and  ${}^{R_1}_{R_2}\mathbf{C}(\theta)$  is the 2D rotation matrix of the form

$$\mathbf{C}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} = \begin{bmatrix} c\psi & -s\psi \\ s\psi & c\psi \end{bmatrix}, \quad (4.10)$$

that describes the angular transformation between robot frames  $\langle R_1 \rangle$  and  $\langle R_2 \rangle$ . The angle between robot frames is

$$\theta = \pi + {}^1\theta_2 - {}^2\theta_1 \quad (4.11)$$

Once the angle  $\theta$  (cf. eq. (4.11)) and the distance between the robots  $R_1$  and  $R_2$  (cf. eq. (4.5)) are computed, the transformation  $({}^{R_1}\mathbf{p}_{R_2}, {}^{R_1}\mathbf{C}(\theta))$  between  $\langle R_1 \rangle$  and  $\langle R_2 \rangle$  is uniquely determined in closed form.

### 4.1.3 Transformation from Global Frame $\langle G_2 \rangle$ to $\langle G_1 \rangle$

In this section, given the distance and bearing measurements recorded by  $R_1$  and  $R_2$ , our goal is to determine the transformation  $({}^{G_1}\mathbf{p}_{G_2}, {}^{G_1}\mathbf{C}(\phi))$  between the global coordinate frames  $\langle G_1 \rangle$  and  $\langle G_2 \rangle$ . This will allow us to express the state estimates  ${}^{G_2}\mathbf{X}_2$  of robot  $R_2$  w.r.t.  $\langle G_1 \rangle$ , i.e., determine the relation

$${}^{G_1}\mathbf{X}_2 = \mathbf{t}({}^{G_1}\mathbf{X}_1, {}^{G_2}\mathbf{X}_2, \mathbf{z}) \quad (4.12)$$

where  $\mathbf{z}$  is the vector described in eq. (4.6).

The rotational transformation matrix  ${}^{G_1}\mathbf{C}(\phi)$  is readily computed from:

$$\begin{aligned} {}^{G_1}\mathbf{C}({}^{G_1}\phi_{G_2}) &= {}^{G_1}\mathbf{C}(\phi_1) {}^{R_1}\mathbf{C}(\theta) {}^{G_2}\mathbf{C}(\phi_2)^T \\ \Rightarrow {}^{G_1}\phi_{G_2} &= \phi_1 + \theta - \phi_2 \end{aligned} \quad (4.13)$$

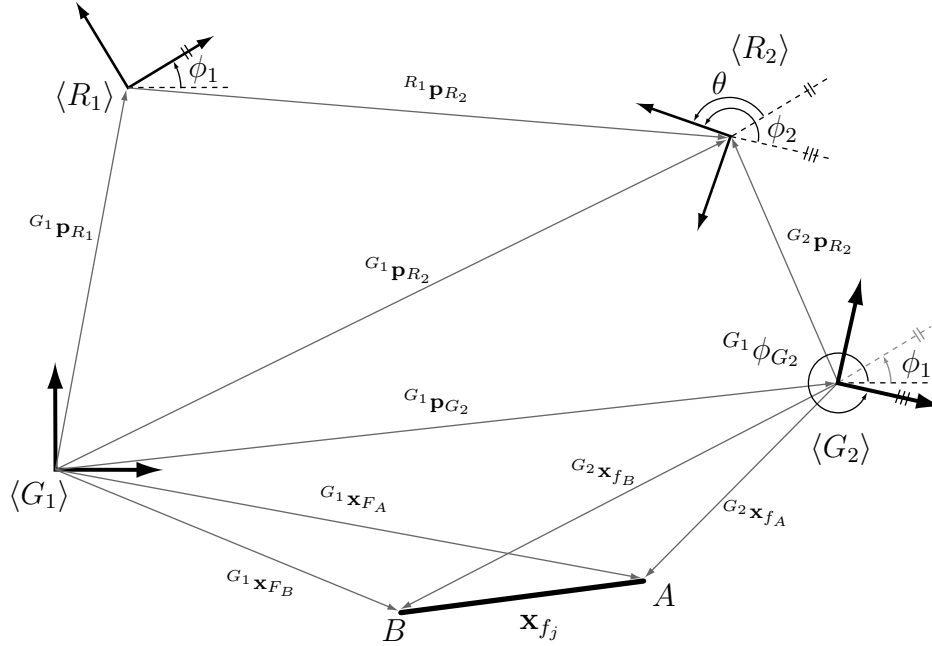
Similarly, the rotational transformation of  $\langle R_2 \rangle$  to  $\langle G_1 \rangle$  is:

$$\begin{aligned} {}^{G_1}\mathbf{C}({}^{G_1}\phi_{R_2}) &= {}^{G_1}\mathbf{C}(\phi_1) {}^{R_1}\mathbf{C}(\theta) \\ \Rightarrow {}^{G_1}\phi_{R_2} &= \phi_1 + \theta \end{aligned} \quad (4.14)$$

Substituting for  $\theta$  from eq. (4.11), we have:

$${}^{G_1}\phi_{R_2} = \phi_1 + \pi + {}^1\theta_2 - {}^2\theta_1 \quad (4.15)$$

where  ${}^{G_1}\phi_{R_2}$  is the orientation of robot  $R_2$  w.r.t. frame  $\langle G_1 \rangle$ ,  $\phi_1$  is the orientation of robot  $R_1$  w.r.t. frame  $\langle G_1 \rangle$ , and  ${}^j\theta_i$  is the relative bearing measurement of  $R_j$  towards  $R_i$ .



**Figure 4.2:** Geometric relations in map alignment.

From Fig. 4.2, the following geometric relations can be obtained:

$${}^{G_1}\mathbf{p}_{G_2} = {}^{G_1}\mathbf{p}_{R_1} + {}^{G_1}\mathbf{C}(\phi_1) {}^{R_1}\mathbf{p}_{R_2} - {}^{G_1}\mathbf{C}({}^{G_1}\phi_{G_2}) {}^{G_2}\mathbf{p}_{R_2} \quad (4.16)$$

$${}^{G_1}\mathbf{p}_{R_2} = {}^{G_1}\mathbf{p}_{R_1} + {}^{G_1}\mathbf{C}(\phi_1) {}^{R_1}\mathbf{p}_{R_2} \quad (4.17)$$

where  ${}^{G_1}\mathbf{p}_{R_1}$  is the position of robot  $R_1$  in frame  $\langle G_1 \rangle$ .

As shown in Fig. 4.2, the position of each of the landmarks  $f_j \in M_2$  expressed w.r.t. the global frame  $\langle G_1 \rangle$  is:

$${}^{G_1}\mathbf{x}_{f_j} = \begin{bmatrix} {}^{G_1}\mathbf{p}_{G_2} \\ {}^{G_1}\mathbf{p}_{G_2} \end{bmatrix} + \begin{bmatrix} {}^{G_1}\mathbf{C}(\phi) & \mathbf{0}_{2 \times 2} \\ \mathbf{0}_{2 \times 2} & {}^{G_2}\mathbf{C}({}^{G_1}\phi_{G_2}) \end{bmatrix} {}^{G_2}\mathbf{x}_{f_j} \quad (4.18)$$

where  ${}^{G_1}\phi_{G_2}$  and  ${}^{G_1}\mathbf{p}_{G_2}$  are given by eqs. (4.13) and (4.16), and  ${}^{G_2}\mathbf{x}_{f_j}$  is the  $j$ -th feature in the map  $M_2$ .

Summarizing, the transformation  $\mathbf{t}$  (eq.(4.12)) that expresses the state vec-

tor  ${}^{G_2}\mathbf{X}_2$  (eq. (4.1)) in frame  $\langle G_1 \rangle$  ( ${}^{G_1}\mathbf{X}_2$ ) is described by

$$\begin{aligned} {}^{R_1}\mathbf{p}_{R_2} &= \rho \begin{bmatrix} \cos({}^1\theta_2) \\ \sin({}^1\theta_2) \end{bmatrix} \\ {}^{G_1}\mathbf{p}_{R_2} &= {}^{G_1}\mathbf{p}_{R_1} + {}_{R_1}^{G_1}\mathbf{C}(\phi_1) {}^{R_1}\mathbf{p}_{R_2} \\ {}^{G_1}\phi_{R_2} &= \phi_1 + \pi + {}^1\theta_2 - {}^2\theta_1 \\ {}^{G_1}\mathbf{p}_{G_2} &= {}^{G_1}\mathbf{p}_{R_1} + {}_{R_1}^{G_1}\mathbf{C}(\phi_1) {}^{R_1}\mathbf{p}_{R_2} - {}_{G_2}^{G_1}\mathbf{C}({}^{G_1}\phi_{G_2}) {}^{G_2}\mathbf{p}_{R_2} \\ {}^{G_1}\mathbf{x}_{f_j} &= \begin{bmatrix} {}^{G_1}\mathbf{p}_{G_2} \\ {}^{G_1}\mathbf{p}_{G_2} \end{bmatrix} + \begin{bmatrix} {}_{G_2}^{G_1}\mathbf{C}(\phi) & \mathbf{0}_{2 \times 2} \\ \mathbf{0}_{2 \times 2} & {}_{G_2}^{G_1}\mathbf{C}({}^{G_1}\phi_{G_2}) \end{bmatrix} {}^{G_2}\mathbf{x}_{f_j}, \quad \mathbf{x}_{f_j} \in M_2 \end{aligned}$$

Since the true quantities are not known, measured and estimated quantities are used in this transformation. The above relations require

- i) an estimate of robot  $R_1$ 's pose  ${}^{G_1}\mathbf{x}_{R_1} = [{}^{G_1}\mathbf{p}_{R_1}^T \ \phi_1]^T$ ;
- ii) an estimate of robot  $R_2$ 's pose  ${}^{G_2}\mathbf{x}_{R_2} = [{}^{G_2}\mathbf{p}_{R_2}^T \ \phi_2]^T$ , and landmarks' positions in the map  $M_2$ ,  ${}^{G_2}\mathbf{X}_{f_{M_2}} = \{{}^{G_2}\mathbf{x}_{f_j}, j = 1 \dots n_2\}$ ;
- iii) the robot-to-robot distance and bearing measurements  $\mathbf{z} = [\rho \ {}^1\theta_2 \ {}^2\theta_1]^T$ .

The accuracy of the transformation between the two coordinate frames will depend on that of the state estimate in (i) and (ii) and the relative measurements in (iii). In the following section, we show the relations that describe how errors in each of these quantities affect the accuracy of the aligned map.

## 4.2 Error Transformation

The augmented state vector, obtained by stacking the two state vectors estimated by the robot independently, is

$$\mathbf{X} = \begin{bmatrix} {}^{G_1}\mathbf{X}_1 \\ {}^{G_2}\mathbf{X}_2 \end{bmatrix} \in \mathbb{R}^{(m_1+m_2)} \quad (4.19)$$

and

$$\tilde{\mathbf{X}} = \begin{bmatrix} {}^{G_1}\tilde{\mathbf{X}}_1 \\ {}^{G_2}\tilde{\mathbf{X}}_2 \end{bmatrix} = \begin{bmatrix} {}^{G_1}\tilde{\mathbf{X}}_{R_1} \\ {}^{G_1}\tilde{\mathbf{X}}_{\mathbf{F}_{M_1}} \\ \dots \\ {}^{G_2}\tilde{\mathbf{X}}_{R_2} \\ {}^{G_2}\tilde{\mathbf{X}}_{\mathbf{f}_{M_2}} \end{bmatrix} \in \mathbb{R}^{(m_1+m_2)} \quad (4.20)$$

is the augmented state vector estimation error. In accordance with eqs. (4.1), (4.3) and (4.4),  $\tilde{\mathbf{X}}$  includes the errors in robot poses and in feature coordinates.

The transformation  $\mathbf{t}$  (4.12) described in Section 4.1 is nonlinear. In order to compute the error in the transformed quantities  ${}^{G_1}\mathbf{x}_{R_2} = [{}^{G_1}\mathbf{p}_{R_2}^T \ {}^{G_1}\phi_{R_2}]^T$  and  ${}^{G_1}\mathbf{x}_{f_j}$ ,  $j = 1 \dots n_2$ , we linearize these equations at the estimated quantities

$${}^{G_1}\tilde{\mathbf{X}}_2 = \frac{\partial \mathbf{t}(\mathbf{X}, \mathbf{z})}{\partial \mathbf{X}} \tilde{\mathbf{X}} + \frac{\partial \mathbf{t}(\mathbf{X}, \mathbf{z})}{\partial \mathbf{z}} \varepsilon_{\mathbf{z}} \quad (4.21)$$

In particular, the Jacobians of  ${}^{G_1}\mathbf{p}_{R_2}$ ,  ${}^{G_1}\phi_{R_2}$  and  ${}^{G_1}\mathbf{x}_{f_i}$  (eqs. (4.17), (4.15), and (4.18)) are computed with respect to the augmented state vector  $\mathbf{X}$  (4.19) and the measurement vector  $\mathbf{z}$  (4.6).

### 4.2.1 Projecting M-Space Errors into Feature Space

The features are saved by the individual SLAM algorithm into the state vectors  ${}^{G_k}\mathbf{X}_k$  (external to the EKF) as the coordinates of their endpoints  ${}^{G_1}\mathbf{X}_{F_{M_1}}$  (4.3) and  ${}^{G_2}\mathbf{X}_{f_{M_2}}$  (4.4)). The estimation error covariance matrices produced by the SLAM algorithm are

$$\mathbf{P}_{11} = \text{Cov} \left( \begin{bmatrix} {}^{G_1}\tilde{\mathbf{X}}_{R_1} \\ {}^{G_1}\tilde{\mathbf{X}}_{\mathbf{P}_{M_1}} \end{bmatrix} \right) \quad (4.22)$$

$$\mathbf{P}_{22} = \text{Cov} \left( \begin{bmatrix} {}^{G_2}\tilde{\mathbf{X}}_{R_2} \\ {}^{G_2}\tilde{\mathbf{X}}_{\mathbf{P}_{M_2}} \end{bmatrix} \right). \quad (4.23)$$

where

$${}^{G_1}\tilde{\mathbf{X}}_{\mathbf{P}_{M_1}} = \begin{bmatrix} {}^{G_1}\tilde{\mathbf{x}}_{P_1} \\ \vdots \\ {}^{G_1}\tilde{\mathbf{x}}_{P_i} \\ \vdots \\ {}^{G_1}\tilde{\mathbf{x}}_{P_{n_1}} \end{bmatrix} \quad (4.24)$$

$${}^{G_2}\tilde{\mathbf{X}}_{\mathbf{P}_{M_2}} = \begin{bmatrix} {}^{G_2}\tilde{\mathbf{x}}_{p_1} \\ \vdots \\ {}^{G_2}\tilde{\mathbf{x}}_{p_j} \\ \vdots \\ {}^{G_2}\tilde{\mathbf{x}}_{p_{n_2}} \end{bmatrix} \quad (4.25)$$

are the error vectors of the M-Space parameters of the two maps. Let

$$\tilde{\mathbf{X}}_{\mathbf{P}} = \begin{bmatrix} {}^{G_1}\tilde{\mathbf{x}}_{R_1} \\ {}^{G_1}\tilde{\mathbf{X}}_{\mathbf{P}_{M_1}} \\ \dots \\ {}^{G_2}\tilde{\mathbf{x}}_{R_2} \\ {}^{G_2}\tilde{\mathbf{X}}_{\mathbf{P}_{M_2}} \end{bmatrix} \in \mathbb{R}^{(m_1+m_2)} \quad (4.26)$$

be the augmented vector including the estimation errors of the robot poses and of the M-Space parameters. Since the two robots estimates are initially independent, the augmented covariance matrix  $\mathbf{P}$  is block diagonal,

$$\mathbf{P} = E \left[ \tilde{\mathbf{X}}_{\mathbf{P}} \tilde{\mathbf{X}}_{\mathbf{P}}^T \right] = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{0}_{m_1 \times m_2} \\ \mathbf{0}_{m_2 \times m_1} & \mathbf{P}_{22} \end{bmatrix}. \quad (4.27)$$

The error  ${}^{G_1}\tilde{\mathbf{p}}_{R_2}$  on transformed robot  $R_2$  position, the error  ${}^{G_1}\tilde{\phi}_{R_2}$  on robot  $R_2$  orientation, and the errors  ${}^{G_1}\tilde{\mathbf{x}}_{f_i}$  of map  $M_2$  features must be expressed in function of the error vector  $\tilde{\mathbf{X}}$ , as shown in eq. (4.21). Unfortunately,  $\tilde{\mathbf{X}}$  includes errors in the feature space, that the SLAM algorithm does not provide. What is provided instead are the M-Space parameters estimation errors  $\tilde{\mathbf{X}}_{\mathbf{P}}$ , whose covariance is described by the covariance matrix  $\mathbf{P}$ . We need a relation between  $\tilde{\mathbf{X}}$  and  $\tilde{\mathbf{X}}_{\mathbf{P}}$ .

The fundamental relation between small variations  $\tilde{\mathbf{x}}_p$  of the M-Space parameters and small variations  $\tilde{\mathbf{x}}_f$  of feature coordinates are

$$\begin{aligned} \tilde{\mathbf{x}}_f &= \tilde{B}_f(\mathbf{x}_f)\tilde{\mathbf{x}}_p \\ \tilde{\mathbf{x}}_p &= B_f(\mathbf{x}_f)\tilde{\mathbf{x}}_f \\ I_{pp} &= B_f(\mathbf{x}_f)\tilde{B}_f(\mathbf{x}_f) \end{aligned}$$

Let  ${}^{G_1}\mathbf{x}_{F_i}$  be a feature in the map  $M_1$  expressed w.r.t. frame  $\langle G_1 \rangle$ , and  ${}^{G_1}\mathbf{x}_{f_j}$  a feature in the map  $M_2$  expressed w.r.t. frame  $\langle G_1 \rangle$ . The M-Space parameters errors can be projected in feature space,

$${}^{G_1}\tilde{\mathbf{x}}_{F_i} = \tilde{B}_f({}^{G_1}\mathbf{x}_{F_i}) {}^{G_1}\tilde{\mathbf{x}}_{P_i} \quad (4.28)$$

$${}^{G_2}\tilde{\mathbf{x}}_{f_j} = \tilde{B}_f({}^{G_2}\mathbf{x}_{f_j}) {}^{G_2}\tilde{\mathbf{x}}_{P_j} \quad (4.29)$$

Stacking the robot poses and all the features together, one obtains the needed relation

$$\begin{aligned} \tilde{\mathbf{X}} &= {}^{G_1 G_2} \tilde{\mathbf{B}} \tilde{\mathbf{X}}_{\mathbf{P}} \\ \begin{bmatrix} {}^{G_1} \tilde{\mathbf{X}}_{R_1} \\ {}^{G_1} \tilde{\mathbf{X}}_{\mathbf{F}_{M_1}} \\ \dots \\ {}^{G_2} \tilde{\mathbf{X}}_{R_2} \\ {}^{G_2} \tilde{\mathbf{X}}_{\mathbf{f}_{M_2}} \end{bmatrix} &= {}^{G_1 G_2} \tilde{\mathbf{B}} \begin{bmatrix} {}^{G_1} \tilde{\mathbf{X}}_{R_1} \\ {}^{G_1} \tilde{\mathbf{X}}_{\mathbf{P}_{M_1}} \\ \dots \\ {}^{G_2} \tilde{\mathbf{X}}_{R_2} \\ {}^{G_2} \tilde{\mathbf{X}}_{\mathbf{P}_{M_2}} \end{bmatrix} \end{aligned} \quad (4.30)$$

where  ${}^{G_1 G_2} \tilde{\mathbf{B}}$  is a block diagonal matrix

$$\begin{aligned} {}^{G_1 G_2} \tilde{\mathbf{B}} &= \begin{bmatrix} {}^{G_1} \tilde{\mathbf{B}}_{\mathbf{F}} & \mathbf{0}_{m_1 \times m_2} \\ \mathbf{0}_{m_2 \times m_1} & {}^{G_2} \tilde{\mathbf{B}}_{\mathbf{f}} \end{bmatrix} \\ {}^{G_1} \tilde{\mathbf{B}}_{\mathbf{F}} = \tilde{\mathbf{B}}_{\mathbf{F}} &= \begin{bmatrix} I_{3 \times 3} & & & \mathbf{0} \\ & \tilde{B}_f({}^{G_1} \mathbf{x}_{F_1}) & & \\ & \mathbf{0} & \dots & \\ & & & \tilde{B}_f({}^{G_1} \mathbf{x}_{F_{n_1}}) \end{bmatrix} \\ {}^{G_2} \tilde{\mathbf{B}}_{\mathbf{f}} &= \begin{bmatrix} I_{3 \times 3} & & & \mathbf{0} \\ & \tilde{B}_f({}^{G_2} \mathbf{x}_{f_1}) & & \\ & \mathbf{0} & \dots & \\ & & & \tilde{B}_f({}^{G_2} \mathbf{x}_{f_{n_2}}) \end{bmatrix} \end{aligned} \quad (4.31)$$

It is important to remark that the  $\tilde{B}_f$  projection matrices in eq. (4.31) are  $4 \times 4$  matrices, regardless of the fact that the endpoints have been initialized or not. This is needed to correlate the line endpoints with the rest of the map.

From now on, the  ${}^{G_1}(\cdot)$  notation will be implied, and the reference frame will be explicitly indicated only if it is different from  $\langle G_1 \rangle$ . So,  $\tilde{\mathbf{B}}_{\mathbf{F}}$  is equivalent to  ${}^{G_1} \tilde{\mathbf{B}}_{\mathbf{F}}$ .

### 4.2.2 Error on ${}^{G_1}\mathbf{p}_{R_2}$

The equation (4.17) can be developed into:

$$\begin{aligned}
{}^{G_1}\mathbf{p}_{R_2} &= {}^{G_1}\mathbf{p}_{R_1} + {}_{R_1}^{G_1}\mathbf{C}(\phi_1)^{R_1}\mathbf{p}_{R_2} \\
&= \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} c\phi_1 & -s\phi_1 \\ s\phi_1 & c\phi_1 \end{bmatrix} \begin{bmatrix} \rho c^1\theta_2 \\ \rho s^1\theta_2 \end{bmatrix} \\
&= \begin{bmatrix} x_1 + \rho c(\phi_1 + {}^1\theta_2) \\ y_1 + \rho s(\phi_1 + {}^1\theta_2) \end{bmatrix} \tag{4.32}
\end{aligned}$$

The Jacobians of  ${}^{G_1}\mathbf{p}_{R_2}$  w.r.t.  ${}^{G_1}\mathbf{x}_{R_1} = [x_1 \ y_1 \ \phi_1]^T$ ,  ${}^{G_1}\mathbf{X}_{M_1} = [\mathbf{x}_{F_1}^T \dots \mathbf{x}_{F_{n_1}}^T]^T$ ,  ${}^{G_2}\mathbf{x}_{R_2} = [x_2 \ y_2 \ \phi_2]^T$ ,  ${}^{G_2}\mathbf{X}_{M_2} = [\mathbf{x}_{f_1}^T \dots \mathbf{x}_{f_{n_2}}^T]^T$  and  $\mathbf{z} = [\rho \ {}^1\theta_2 \ {}^2\theta_1]^T$  are:

$$\begin{aligned}
\nabla_{{}^{G_1}\mathbf{x}_{R_1}}({}^{G_1}\mathbf{p}_{R_2}) &= \begin{bmatrix} 1 & 0 & -\rho s(\phi_1 + {}^1\theta_2) \\ 0 & 1 & \rho c(\phi_1 + {}^1\theta_2) \end{bmatrix} \\
\nabla_{{}^{G_1}\mathbf{X}_{M_1}}({}^{G_1}\mathbf{p}_{R_2}) &= \mathbf{0}_{2 \times 4n_1} \\
{}^{\mathbf{p}_{R_2}}\mathbf{T}_1 &= \begin{bmatrix} \nabla_{{}^{G_1}\mathbf{x}_{R_1}}({}^{G_1}\mathbf{p}_{R_2}) & \nabla_{{}^{G_1}\mathbf{X}_{M_1}}({}^{G_1}\mathbf{p}_{R_2}) \end{bmatrix} \tag{4.33}
\end{aligned}$$

$$\begin{aligned}
\nabla_{{}^{G_2}\mathbf{x}_{R_2}}({}^{G_1}\mathbf{p}_{R_2}) &= \mathbf{0}_{2 \times 3} \\
\nabla_{{}^{G_2}\mathbf{X}_{M_2}}({}^{G_1}\mathbf{p}_{R_2}) &= \mathbf{0}_{2 \times 4n_2} \\
{}^{\mathbf{p}_{R_2}}\mathbf{T}_2 &= \begin{bmatrix} \nabla_{{}^{G_2}\mathbf{x}_{R_2}}({}^{G_1}\mathbf{p}_{R_2}) & \nabla_{{}^{G_2}\mathbf{X}_{M_2}}({}^{G_1}\mathbf{p}_{R_2}) \end{bmatrix} \tag{4.34}
\end{aligned}$$

$${}^{\mathbf{p}_{R_2}}\mathbf{\Gamma} = \nabla_{\mathbf{z}}({}^{G_1}\mathbf{p}_{R_2}) = \begin{bmatrix} c(\phi_1 + {}^1\theta_2) & -\rho s(\phi_1 + {}^1\theta_2) & 0 \\ s(\phi_1 + {}^1\theta_2) & \rho c(\phi_1 + {}^1\theta_2) & 0 \end{bmatrix} \tag{4.35}$$

Using these Jacobians, the error in the position of robot  $R_2$  expressed w.r.t. global frame  $\langle G_1 \rangle$  is:

$${}^{G_1}\tilde{\mathbf{p}}_{R_2} = \begin{bmatrix} {}^{\mathbf{p}_{R_2}}\mathbf{T}_1 & \vdots & \mathbf{0}_{2 \times (3+4n_2)} \end{bmatrix} \tilde{\mathbf{X}} + {}^{\mathbf{p}_{R_2}}\mathbf{\Gamma} \varepsilon_{\mathbf{z}} \tag{4.36}$$

### 4.2.3 Error on ${}^{G_1}\phi_{R_2}$

As before, in order to compute the error on  ${}^{G_1}\phi_{R_2}$  (eq. (4.15)), the Jacobians are needed:

$$\nabla_{G_1\mathbf{x}_{R_1}}({}^{G_1}\phi_{R_2}) = [0 \quad 0 \quad 1]$$

$$\begin{aligned} \nabla_{G_1\mathbf{x}_{M_1}}({}^{G_1}\phi_{R_2}) &= \mathbf{0}_{1 \times 4\mathbf{n}_1} \\ \phi_{R_2}\mathbf{T}_1 &= \left[ \nabla_{G_1\mathbf{x}_{R_1}}({}^{G_1}\phi_{R_2}) \quad \nabla_{G_1\mathbf{x}_{M_1}}({}^{G_1}\phi_{R_2}) \right] \end{aligned} \quad (4.37)$$

$$\nabla_{G_2\mathbf{x}_{R_2}}({}^{G_1}\phi_{R_2}) = \mathbf{0}_{1 \times 3}$$

$$\begin{aligned} \nabla_{G_2\mathbf{x}_{M_2}}({}^{G_1}\phi_{R_2}) &= \mathbf{0}_{1 \times 4\mathbf{n}_2} \\ \phi_{R_2}\mathbf{T}_2 &= \left[ \nabla_{G_2\mathbf{x}_{R_2}}({}^{G_1}\phi_{R_2}) \quad \nabla_{G_2\mathbf{x}_{M_2}}({}^{G_1}\phi_{R_2}) \right] \end{aligned} \quad (4.38)$$

$$\phi_{R_2}\mathbf{\Gamma} = \nabla_{\mathbf{z}}({}^{G_1}\phi_{R_2}) = [0 \quad 1 \quad -1] \quad (4.39)$$

then,

$${}^{G_1}\tilde{\phi}_{R_2} = \left[ \phi_{R_2}\mathbf{T}_1 \vdots \mathbf{0}_{1 \times (3+4\mathbf{n}_2)} \right] \tilde{\mathbf{X}} + \phi_{R_2}\mathbf{\Gamma} \varepsilon_{\mathbf{z}} \quad (4.40)$$

As one can see from eqs. (4.36) and (4.40), only the robot  $R_1$  pose error  ${}^{G_1}\tilde{\mathbf{x}}_{R_1}$  affects the robot  $R_2$  pose error  ${}^{G_1}\tilde{\mathbf{x}}_{R_2}$ , while other errors of the map  $M_2$   ${}^{G_2}\tilde{\mathbf{x}}_{f_j}$  do not contribute.

### 4.2.4 Error on ${}^{G_1}\mathbf{x}_{f_j}$

The eq. (4.16) can be developed into:

$$\begin{aligned} {}^{G_1}\mathbf{p}_{G_2} &= {}^{G_1}\mathbf{p}_{R_1} + {}^{G_1}\mathbf{C}(\phi_1)^{R_1}\mathbf{p}_{R_2} - {}^{G_1}\mathbf{C}(\phi)^{G_2}\mathbf{p}_{R_2} \\ &= \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} c\phi_1 & -s\phi_1 \\ s\phi_1 & c\phi_1 \end{bmatrix} \begin{bmatrix} \rho c^1\theta_2 \\ \rho s^1\theta_2 \end{bmatrix} - \begin{bmatrix} c\phi & -s\phi \\ s\phi & c\phi \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} \\ &= \begin{bmatrix} x_1 + \rho c(\phi_1 + {}^1\theta_2) - (x_2 c\phi - y_2 s\phi) \\ y_1 + \rho s(\phi_1 + {}^1\theta_2) - (x_2 s\phi + y_2 c\phi) \end{bmatrix} \end{aligned} \quad (4.41)$$

The eq. (4.18) can be rewritten by substituting  ${}^{G_1}\mathbf{p}_{G_2}$  from eq. (4.41),

$$\begin{aligned} {}^{G_1}\mathbf{x}_{f_j} &= \begin{bmatrix} {}^{G_1}\mathbf{p}_{G_2} \\ {}^{G_1}\mathbf{p}_{G_2} \end{bmatrix} + \begin{bmatrix} {}^{G_1}\mathbf{C}(\phi) & \mathbf{0}_{2 \times 2} \\ \mathbf{0}_{2 \times 2} & {}^{G_1}\mathbf{C}(\phi) \end{bmatrix} {}^{G_2}\mathbf{x}_{f_j} \\ &= \begin{bmatrix} x_1 + \rho c(\phi_1 + {}^1\theta_2) + (x_{A_{f_i}} - x_2)c(\phi_1 + \pi + {}^1\theta_2 - {}^2\theta_1 - \phi_2) - (y_{A_{f_i}} - y_2)s(\phi_1 + \pi + {}^1\theta_2 - {}^2\theta_1 - \phi_2) \\ y_1 + \rho s(\phi_1 + {}^1\theta_2) + (x_{A_{f_i}} - x_2)s(\phi_1 + \pi + {}^1\theta_2 - {}^2\theta_1 - \phi_2) + (y_{A_{f_i}} - y_2)c(\phi_1 + \pi + {}^1\theta_2 - {}^2\theta_1 - \phi_2) \\ x_1 + \rho c(\phi_1 + {}^1\theta_2) + (x_{B_{f_i}} - x_2)c(\phi_1 + \pi + {}^1\theta_2 - {}^2\theta_1 - \phi_2) - (y_{B_{f_i}} - y_2)s(\phi_1 + \pi + {}^1\theta_2 - {}^2\theta_1 - \phi_2) \\ y_1 + \rho s(\phi_1 + {}^1\theta_2) + (x_{B_{f_i}} - x_2)s(\phi_1 + \pi + {}^1\theta_2 - {}^2\theta_1 - \phi_2) + (y_{B_{f_i}} - y_2)c(\phi_1 + \pi + {}^1\theta_2 - {}^2\theta_1 - \phi_2) \end{bmatrix} \end{aligned} \quad (4.42)$$

The Jacobians needed to compute the error for feature  $f_j$  are:

$$\begin{aligned} \nabla_{G_1\mathbf{x}_{R_1}}({}^{G_1}\mathbf{x}_{f_j}) &= \begin{bmatrix} 1 & 0 & -\rho s(\phi_1 + {}^1\theta_2) - s\phi(x_{A_{f_j}} - x_2) - c\phi(y_{A_{f_j}} - y_2) \\ 0 & 1 & \rho c(\phi_1 + {}^1\theta_2) + c\phi(x_{A_{f_j}} - x_2) - s\phi(y_{A_{f_j}} - y_2) \\ 1 & 0 & -\rho s(\phi_1 + {}^1\theta_2) - s\phi(x_{B_{f_j}} - x_2) - c\phi(y_{B_{f_j}} - y_2) \\ 0 & 1 & \rho c(\phi_1 + {}^1\theta_2) + c\phi(x_{B_{f_j}} - x_2) - s\phi(y_{B_{f_j}} - y_2) \end{bmatrix} \\ \nabla_{G_1\mathbf{X}_{M_1}}({}^{G_1}\mathbf{x}_{f_j}) &= \mathbf{0}_{2 \times 4n_1} \\ f_j \mathbf{T}_1 &= \begin{bmatrix} \nabla_{G_1\mathbf{x}_{R_1}}({}^{G_1}\mathbf{x}_{f_j}) & \nabla_{G_1\mathbf{X}_{M_1}}({}^{G_1}\mathbf{x}_{f_j}) \end{bmatrix} \end{aligned} \quad (4.43)$$

$$\begin{aligned} \nabla_{G_2\mathbf{x}_{R_2}}({}^{G_1}\mathbf{x}_{f_j}) &= \begin{bmatrix} -c\phi & s\phi & s\phi(x_{A_{f_j}} - x_2) + c\phi(y_{A_{f_j}} - y_2) \\ -s\phi & -c\phi & -c\phi(x_{A_{f_j}} - x_2) + s\phi(y_{A_{f_j}} - y_2) \\ -c\phi & s\phi & s\phi(x_{B_{f_j}} - x_2) + c\phi(y_{B_{f_j}} - y_2) \\ -s\phi & -c\phi & -c\phi(x_{B_{f_j}} - x_2) + s\phi(y_{B_{f_j}} - y_2) \end{bmatrix} \\ \nabla_{G_2\mathbf{X}_{M_2}}({}^{G_1}\mathbf{x}_{f_j}) &= \begin{bmatrix} \mathbf{0}_{4 \times 4(j-1)} & \begin{bmatrix} {}^{G_1}\mathbf{C}(\phi) & \mathbf{0}_{2 \times 2} \\ \mathbf{0}_{2 \times 2} & {}^{G_1}\mathbf{C}(\phi) \end{bmatrix} & \mathbf{0}_{4 \times 4(n_2-j)} \end{bmatrix} \\ f_j \mathbf{T}_2 &= \begin{bmatrix} \nabla_{G_2\mathbf{x}_{R_2}}({}^{G_1}\mathbf{x}_{f_j}) & \nabla_{G_2\mathbf{X}_{M_2}}({}^{G_1}\mathbf{x}_{f_j}) \end{bmatrix} \end{aligned} \quad (4.44)$$

$$\begin{aligned} f_j \mathbf{\Gamma} &= \nabla_{\mathbf{z}}({}^{G_1}\mathbf{x}_{f_j}) = \\ &= \begin{bmatrix} c(\phi_1 + {}^1\theta_2) & -\rho s(\phi_1 + {}^1\theta_2) - s\phi(x_{A_{f_j}} - x_2) - c\phi(y_{A_{f_j}} - y_2) & s\phi(x_{A_{f_j}} - x_2) + c\phi(y_{A_{f_j}} - y_2) \\ s(\phi_1 + {}^1\theta_2) & \rho c(\phi_1 + {}^1\theta_2) + c\phi(x_{A_{f_j}} - x_2) - s\phi(y_{A_{f_j}} - y_2) & -c\phi(x_{A_{f_j}} - x_2) + s\phi(y_{A_{f_j}} - y_2) \\ c(\phi_1 + {}^1\theta_2) & -\rho s(\phi_1 + {}^1\theta_2) - s\phi(x_{B_{f_j}} - x_2) - c\phi(y_{B_{f_j}} - y_2) & s\phi(x_{B_{f_j}} - x_2) + c\phi(y_{B_{f_j}} - y_2) \\ s(\phi_1 + {}^1\theta_2) & \rho c(\phi_1 + {}^1\theta_2) + c\phi(x_{B_{f_j}} - x_2) - s\phi(y_{B_{f_j}} - y_2) & -c\phi(x_{B_{f_j}} - x_2) + s\phi(y_{B_{f_j}} - y_2) \end{bmatrix} \end{aligned} \quad (4.45)$$

So, the error on feature  $f_j$  expressed w.r.t. frame  $\langle G_1 \rangle$  is :

$${}^{G_1}\tilde{\mathbf{x}}_{f_j} = \begin{bmatrix} f_j \mathbf{T}_1 \\ \vdots \\ f_j \mathbf{T}_2 \end{bmatrix} \tilde{\mathbf{X}} + f_j \mathbf{\Gamma} \varepsilon_{\mathbf{z}} \quad (4.46)$$

$$\stackrel{(4.30)}{=} \begin{bmatrix} f_j \mathbf{T}_1 \\ \vdots \\ f_j \mathbf{T}_2 \end{bmatrix} {}^{G_1 G_2} \tilde{\mathbf{B}} \tilde{\mathbf{X}}_{\mathbf{P}} + f_j \mathbf{\Gamma} \varepsilon_{\mathbf{z}} \quad (4.47)$$

To project the error  ${}^{G_1}\tilde{\mathbf{x}}_{f_j}$ , expressed in feature space, back into M-Space  ${}^{G_1}\tilde{\mathbf{x}}_{p_j}$ , the fundamental relation must be used again,

$${}^{G_1}\tilde{\mathbf{x}}_{p_j} = B_f({}^{G_1}\mathbf{x}_{f_j}) {}^{G_1}\tilde{\mathbf{x}}_{f_j} \quad (4.48)$$

thus obtaining

$${}^{G_1}\tilde{\mathbf{x}}_{p_j} = B_f({}^{G_1}\mathbf{x}_{f_j}) \begin{bmatrix} f_j \mathbf{T}_1 \\ \vdots \\ f_j \mathbf{T}_2 \end{bmatrix} {}^{G_1 G_2} \tilde{\mathbf{B}} \tilde{\mathbf{X}}_{\mathbf{P}} + f_j \mathbf{\Gamma} \varepsilon_{\mathbf{z}} \quad (4.49)$$

### 4.3 Transforming the Map

Using the results of the previous sections, the pose and the map of robot  $R_2$  can now be described w.r.t. frame  $\langle G_1 \rangle$ . Before the two robots met, two independent Kalman filter estimators were used for estimating the vectors  ${}^{G_1}\mathbf{X}_1$  and  ${}^{G_2}\mathbf{X}_2$  (4.1), with feature estimation errors covariance given by  $\mathbf{P}_{11}$  and  $\mathbf{P}_{22}$  (eqs. (4.22) and (4.23)).

In order to align the two maps, we express  ${}^{G_2}\mathbf{X}_2$  in the same frame as  ${}^{G_1}\mathbf{X}_1$  (eq. (4.12) or equivalently eqs. (4.17), (4.15) and (4.18)). The new augmented state vector is:

$$\mathbf{X}' = \begin{bmatrix} {}^{G_1}\mathbf{X}_1 \\ {}^{G_1}\mathbf{X}_2 \end{bmatrix} = \begin{bmatrix} {}^{G_1}\mathbf{X}_1 \\ \mathbf{t}({}^{G_1}\mathbf{X}_1, {}^{G_2}\mathbf{X}_2, \mathbf{z}) \end{bmatrix} \quad (4.50)$$

while the errors are described as:

$$\tilde{\mathbf{X}}' = \mathbf{B} \left( \begin{bmatrix} \mathbf{I}_{m_1 \times m_1} & \mathbf{0}_{m_1 \times m_2} \\ \mathbf{T}_1 & \mathbf{T}_2 \end{bmatrix} {}^{G_1 G_2} \tilde{\mathbf{B}} \tilde{\mathbf{X}}_{\mathbf{P}} + \begin{bmatrix} \mathbf{0}_{m_1 \times 3} \\ \mathbf{\Gamma}_2 \end{bmatrix} \varepsilon_{\mathbf{z}} \right) \quad (4.51)$$

where

$$\mathbf{T}_1 = \begin{bmatrix} p_{R_2} \mathbf{T}_1 \\ \phi_{R_2} \mathbf{T}_1 \\ f_1 \mathbf{T}_1 \\ \vdots \\ f_{n_2} \mathbf{T}_1 \end{bmatrix}, \quad \mathbf{T}_2 = \begin{bmatrix} p_{R_2} \mathbf{T}_2 \\ \phi_{R_2} \mathbf{T}_2 \\ f_1 \mathbf{T}_2 \\ \vdots \\ f_{n_2} \mathbf{T}_2 \end{bmatrix}, \quad \mathbf{\Gamma}_2 = \begin{bmatrix} p_{R_2} \mathbf{\Gamma} \\ \phi_{R_2} \mathbf{\Gamma} \\ f_1 \mathbf{\Gamma} \\ \vdots \\ f_{n_2} \mathbf{\Gamma} \end{bmatrix} \quad (4.52)$$

are the stacked Jacobians, and

$$\begin{aligned}
\mathbf{B} &= \begin{bmatrix} G_1 \mathbf{B}_{\mathbf{F}} & \mathbf{0}_{m_1 \times m_2} \\ \mathbf{0}_{m_2 \times m_1} & G_1 \mathbf{B}_{\mathbf{f}} \end{bmatrix} \\
G_1 \mathbf{B}_{\mathbf{F}} = \mathbf{B}_{\mathbf{F}} &= \begin{bmatrix} I_{3 \times 3} & & & \mathbf{0} \\ & B_f(G_1 \mathbf{x}_{F_1}) & & \\ & \mathbf{0} & \ddots & \\ & & & B_f(G_1 \mathbf{x}_{F_{n_1}}) \end{bmatrix} \\
G_1 \mathbf{B}_{\mathbf{f}} = \mathbf{B}_{\mathbf{f}} &= \begin{bmatrix} I_{3 \times 3} & & & \mathbf{0} \\ & B_f(G_1 \mathbf{x}_{f_1}) & & \\ & \mathbf{0} & \ddots & \\ & & & B_f(G_1 \mathbf{x}_{f_{n_2}}) \end{bmatrix}
\end{aligned} \tag{4.53}$$

$\mathbf{B}$  is a block diagonal matrix of all the  $B_f$  projection matrices, computed at the features expressed w.r.t. frame  $\langle G_1 \rangle$ . As before,  $\mathbf{B}_{\mathbf{F}}$  is equivalent to  $G_1 \mathbf{B}_{\mathbf{F}}$ , and  $\mathbf{B}_{\mathbf{f}}$  is equivalent to  $G_1 \mathbf{B}_{\mathbf{f}}$ .

If the non-initialized endpoints of the map  $M_2$  are not correlated with the rest of the merged map, the EKF-based update procedure described next would not affect them, reducing the quality of the merged map. For this reason, similarly to eq. (4.31), the  $B_f$  projection matrices in eq. (4.53) are  $4 \times 4$  matrices, regardless of the fact that the endpoints have been initialized or not. This is needed to correlate the line endpoints with the rest of the map.

Finally, the covariance of the new system is computed as:

$$\begin{aligned}
\mathbf{P}' &= \mathbf{B} \left( \begin{bmatrix} \mathbf{I}_{m_1 \times m_1} & \mathbf{0}_{m_1 \times m_2} \\ \mathbf{T}_1 & \mathbf{T}_2 \end{bmatrix} \bar{\mathbf{P}} \begin{bmatrix} \mathbf{I}_{m_1 \times m_1} & \mathbf{T}_1^T \\ \mathbf{0}_{m_2 \times m_1} & \mathbf{T}_2^T \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{m_1 \times 3} \\ \mathbf{\Gamma}_2 \end{bmatrix} R_{\mathbf{z}} \begin{bmatrix} \mathbf{0}_{3 \times m_1} & \mathbf{\Gamma}_2^T \end{bmatrix} \right) \mathbf{B}^T \\
\text{where} & \\
\bar{\mathbf{P}} &= {}^{G_1 G_2} \tilde{\mathbf{B}} \mathbf{P} {}^{G_1 G_2} \tilde{\mathbf{B}}^T
\end{aligned} \tag{4.54}$$

Its blocks are

$$\begin{aligned}
\mathbf{P}' &= \begin{bmatrix} \mathbf{P}'_{11} & \mathbf{P}'_{12} \\ \mathbf{P}'_{21} & \mathbf{P}'_{22} \end{bmatrix} \\
\mathbf{P}'_{11} &= \mathbf{B}_F \tilde{\mathbf{B}}_F \mathbf{P}_{11} \tilde{\mathbf{B}}_F^T \mathbf{B}_F^T = \mathbf{P}_{11} \\
\mathbf{P}'_{12} &= \mathbf{B}_F \tilde{\mathbf{B}}_F \mathbf{P}_{11} \tilde{\mathbf{B}}_F^T \mathbf{T}_1^T \mathbf{B}_f^T = \mathbf{P}_{11} \tilde{\mathbf{B}}_F^T \mathbf{T}_1^T \mathbf{B}_f^T \\
\mathbf{P}'_{21} &= \mathbf{P}'_{12}{}^T = \mathbf{B}_f \mathbf{T}_1 \tilde{\mathbf{B}}_F \mathbf{P}_{11} \\
\mathbf{P}'_{22} &= \mathbf{B}_f (\mathbf{T}_1 \tilde{\mathbf{B}}_F \mathbf{P}_{11} \tilde{\mathbf{B}}_F^T \mathbf{T}_1^T + \mathbf{T}_2^{G_2} \tilde{\mathbf{B}}_f \mathbf{P}_{22}^{G_2} \tilde{\mathbf{B}}_f^T \mathbf{T}_2^T + \mathbf{\Gamma}_2 \mathbf{R}_z \mathbf{\Gamma}_2^T) \mathbf{B}_f^T
\end{aligned}$$

This new merged map can now be used by the two robots to perform SLAM independently, until they meet again.

## 4.4 Matching and Eliminating Duplicate Landmarks

In this section we describe a method for determining duplicate landmarks and use the landmarks common between the two maps as constraints to update the *merged* map. It is very likely that the areas the two robots covered have common regions before rendezvous. This means that a number of landmarks might appear as duplicates in the new state vector that resulted by aligning and merging the state estimates of the two robots. By employing this information, one can improve the accuracy of the final map. Specifically, if all duplicate landmark pairings are known, these can be used to reduce both the alignment errors and the size of the state vector.

Two well known methods for searching for matching landmarks are the Nearest Neighbor (NN) and Joint Compatibility Branch and Bound (JCBB) algorithms. NN simply pairs only the closest two landmarks. If for each landmark in  $M_2$ , we search through all landmarks in  $M_1$  for its nearest neighbor, then the total complexity is  $O(n_1 \cdot n_2)$ . JCBB finds the largest number of jointly compatible pairings by searching among all the possible feature combinations, pruning the search tree with branch and bound heuristics. The algorithm is more robust but has a much higher computational cost.

Considering the trade-off between robustness and complexity, we have selected to use the NN algorithm. NN was deemed sufficient for the purposes of the work presented here, because the positioning uncertainty of the landmarks in the vicinity of the rendezvous location, expressed with respect to frame  $\langle R_1 \rangle$ , is relatively small. The landmarks located close to the area where the two robots meet have a higher probability of being within the intersection of the two maps and are the ones that can be reliably matched within the first few iterations of the algorithm, to improve map alignment. For this purpose, the search for matching landmarks is performed sorting the landmarks of the map  $M_1$  ( $M_2$ ) in order of ascending distance from the robot  $R_1$  ( $R_2$ ).

After map alignment, the merged state vector is

$$\mathbf{X}' = \begin{bmatrix} \mathbf{x}_{R_1} \\ \mathbf{x}_{F_1} \\ \vdots \\ \mathbf{x}_{F_{n_1}} \\ \mathbf{x}_{R_2} \\ \mathbf{x}_{f_1} \\ \vdots \\ \mathbf{x}_{f_{n_2}} \end{bmatrix}$$

The distance between two features  $F_i \in M_1$  and  $f_j \in M_2$  is computed in a form similar to the innovation  $\eta$  (cf. eq. (3.32)),

$$\begin{aligned} Z &= \mathbf{m} - h(\mathbf{x}_{F_i}), & i &= 1 \dots n_1 \\ \mathbf{m} &= h(\mathbf{x}_{f_j}), & j &= 1 \dots n_2 \end{aligned} \quad (4.55)$$

where the function  $h(\cdot)$  has the same form as eq. (3.33).

If features  $\mathbf{x}_{F_i}$  and  $\mathbf{x}_{f_j}$  are the same, then  $Z$  should be zero. The corresponding estimate  $\hat{Z} = h(\hat{\mathbf{x}}_{F_i}) - h(\hat{\mathbf{x}}_{f_j})$  can be used together with the pseudo-measure (4.55) to compute an innovation  $\eta$ . So, Mahalanobis distance hypothesis test

can be formulated as for the SLAM data association problem:

$$Z = h(\mathbf{x}_{f_j}) - h(\mathbf{x}_{F_i}) = 0 \quad (4.56)$$

$$\hat{Z} = h(\hat{\mathbf{x}}_{f_j}) - h(\hat{\mathbf{x}}_{F_i}) \quad (4.57)$$

$$\eta = \Phi(Z - \hat{Z}) = -\Phi\hat{Z} \quad (4.58)$$

$$S = H \mathbf{P}' H^T \quad (4.59)$$

$$D_{ij} = \eta^T S^{-1} \eta \quad (4.60)$$

where

$$\begin{aligned} H &= \frac{\partial \eta}{\partial \mathbf{X}'} = \\ &= \Phi \left[ \mathbf{0}_{4 \times 3} \quad \mathbf{0}_{4 \times (4i-4)} \quad H_{F_i} \quad \mathbf{0}_{4 \times (4n_1-4i)} \quad \vdots \quad \mathbf{0}_{4 \times 3} \quad \mathbf{0}_{4 \times (4j-4)} \quad H_{f_j} \quad \mathbf{0}_{4 \times (4n_2-4j)} \right] \\ H_{F_i} &= -J_{\eta_o} J_{o_f} \tilde{B}_f(\hat{\mathbf{x}}_{F_i}) \\ H_{f_j} &= J_{\eta_o} J_{o_f} \tilde{B}_f(\hat{\mathbf{x}}_{f_j}) \\ \Phi &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{aligned} \quad (4.61)$$

$D_{ij}$  is the Mahalanobis distance between features  $F_i$  e  $f_j$ . Each  $F_i$  in  $M_1$  is compared to every  $f_j$  of  $M_2$  ( $j = 1 \dots n_2$ ) and the minimum  $D_{ij}$  is determined. If the minimum  $D_{ij}$  is less than a certain threshold  $T_D$  and the overlapping  $\tau_{ij}$  is greater than a certain threshold  $T_\tau$ , then it can be stated that landmarks  $F_i$  and  $f_j$  are the same. This constraint is imposed via an EKF update using the innovation  $\eta$  (4.58),

$$K = -\mathbf{P}'_ - H^T S^{-1} \quad (4.62)$$

$$\mathbf{X}'_+ = \mathbf{X}'_ - + \tilde{\mathbf{B}} K \eta \quad (4.63)$$

$$\mathbf{P}'_+ = \mathbf{P}'_ - - K S K^T \quad (4.64)$$

where

$$\begin{aligned}
 \tilde{\mathbf{B}} &= \begin{bmatrix} G_1 \tilde{\mathbf{B}}_{\mathbf{F}} & \mathbf{0}_{m_1 \times m_2} \\ \mathbf{0}_{m_2 \times m_1} & G_1 \tilde{\mathbf{B}}_{\mathbf{f}} \end{bmatrix} \\
 G_1 \tilde{\mathbf{B}}_{\mathbf{f}} &= \begin{bmatrix} I_{3 \times 3} & & & \mathbf{0} \\ & \tilde{B}_f(G_1 \mathbf{x}_{f_1}) & & \\ & \mathbf{0} & \dots & \\ & & & \tilde{B}_f(G_1 \mathbf{x}_{f_{n_2}}) \end{bmatrix}
 \end{aligned} \tag{4.65}$$

After this update, the feature  $\mathbf{x}_{f_j}$  is eliminated from the state vector  $\mathbf{X}'_+$  together with the corresponding rows and columns of the covariance matrix  $\mathbf{P}'_+$ . This process of searching for duplicates is repeated for all landmarks  $F_i$  belonging to map  $M_1$ .

# Chapter 5

## Single-robot SLAM Simulations

Chapter 3 introduced the single-robot SLAM algorithm using EKF and M-Space feature representation. On the purpose of testing the SLAM algorithm, a simulator has been developed using the MATLAB environment. The simulator can run single-robot SLAM simulations, as well as two-robot SLAM simulations with map fusion. In this chapter the simulations results for the single-robot SLAM algorithm are presented and discussed.

### 5.1 The SLAM simulator software

In order to test the algorithm performance in realistic situations, CAD maps of real buildings can be provided to the simulator for the robot to perform SLAM therein. The robot travels across the given environment following a given set of waypoints (see Fig. 3.3).

Robot moves according to eq. 3.22, and takes measurements as shown in Fig. 3.4. The laser range finder provides a set of points, taking a measurement every 1 degree, in a field of view of 180 degrees: its working is simulated with a *raytracing* algorithm that computes the intersection of the laser beam with the obstacles (lines) present in the CAD map. The line segment features are extracted from the laser raw data as described in Section 3.5. The simulator logs relevant data about the running experiment and can also capture a video, to be played-back later in real-time.

The simulations settings for the following experiments are listed below (see Chapter 3 for reference about symbols).

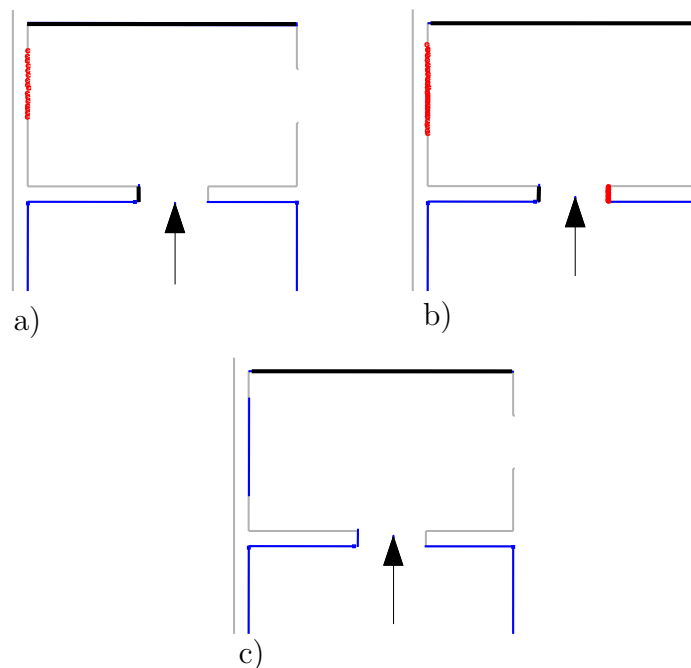
### Robot settings

- The robot linear speed is constant,  $v = 0.2m/s$ .
- The maximum achievable angular speed of the wheels is  $\omega_{MAX} = 9rad/s$ ; this influence the turning radius of the unicycle vehicle.
- The wheels have a radius of  $15cm$  and the distance between the two wheels is  $50cm$ .
- The robot odometer runs at  $10Hz$  ( $T_s = 0.1 s$ ), while the measurements are taken at a  $2Hz$  frequency.
- The robot motion is affected by Gaussian white noises: the standard deviation of linear speed noise is  $\sigma_v = \frac{\alpha_v |v(k)|}{3\sqrt{T_s}}$ , and the standard deviation of angular speed noise is  $\sigma_\omega = \frac{\alpha_\omega |\omega(k)| + \delta_\omega}{3\sqrt{T_s}}$ , where  $\alpha_v = 0.03$ ,  $\alpha_\omega = 0.02$ ,  $\delta_\omega = 0.1 \frac{\pi}{180} rad/s$ . The standard deviations are scaled by the  $\sqrt{T_s}$  term, because a discrete-time motion model (3.22) is adopted.

### Measurements settings

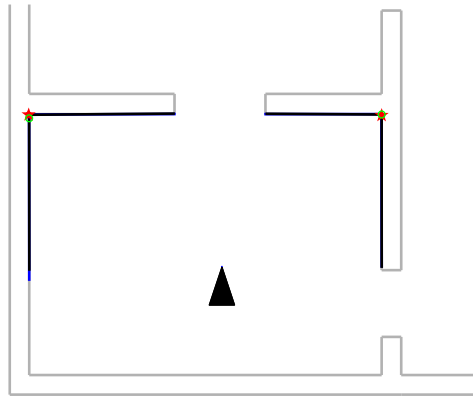
- The maximum distance measurable by the laser is  $\rho_{MAX} = 8 m$ .
- The field of view of the laser range finder is 180 degrees, i.e. the robot can see in a sector of  $\pm 90$  degrees with respect to its heading. The laser takes a range measurement every 1 degree.
- The laser range measurements is affected by a Gaussian noise whose standard deviation is  $\sigma_{RANGE} = 0.003 m$ . The bearing of the measurements is affected by a Gaussian white noise whose standard deviation is  $\sigma_{BEARING} = 0.16 \frac{\pi}{180} rad$ . These noises affect the raw measurements, while the measurement noise covariance matrix  $R$  is estimated as described in Section 3.5.2.

- The line features extracted from laser raw data are at least 20 *cm* long, and are produced by fitting at least 15 consecutive points.
- The thresholds  $T_\rho$ ,  $T_\alpha$ , and  $T_\tau$  are the parameters for data association (see Section 3.6), that determine the tolerance with which the measurements are compared with the features that are already in the state.
- A feature must be seen at least 3 times, before being promoted from the tentative list, and used to augment the EKF external state vector. This parameter should be increased, when dealing with real environments crowded with people. Figure 5.1 shows an example of initialization. In subfigure (a) The point cloud is stored in the tentative list; in (b) the line is seen again, and more points are added to the same cloud. Another short line is detected at the right of the robot; finally, in (c) the long line is considered a reliable feature, while the short line is removed from the tentative list, since it has not been seen enough times.



**Figure 5.1:** The lines must be seen multiple times before being considered reliable features.

- When detected, the lines endpoints are used for augmenting the M-Space dimensions of a feature (see Section 3.2.2) or to compute the innovations for the EKF update. The only reliable endpoints are the ones in correspondence of the intersection of two consecutive walls in the laser scanning (see Figure 5.2).



**Figure 5.2:** The endpoints of the lines are detected in correspondence of two walls intersection.

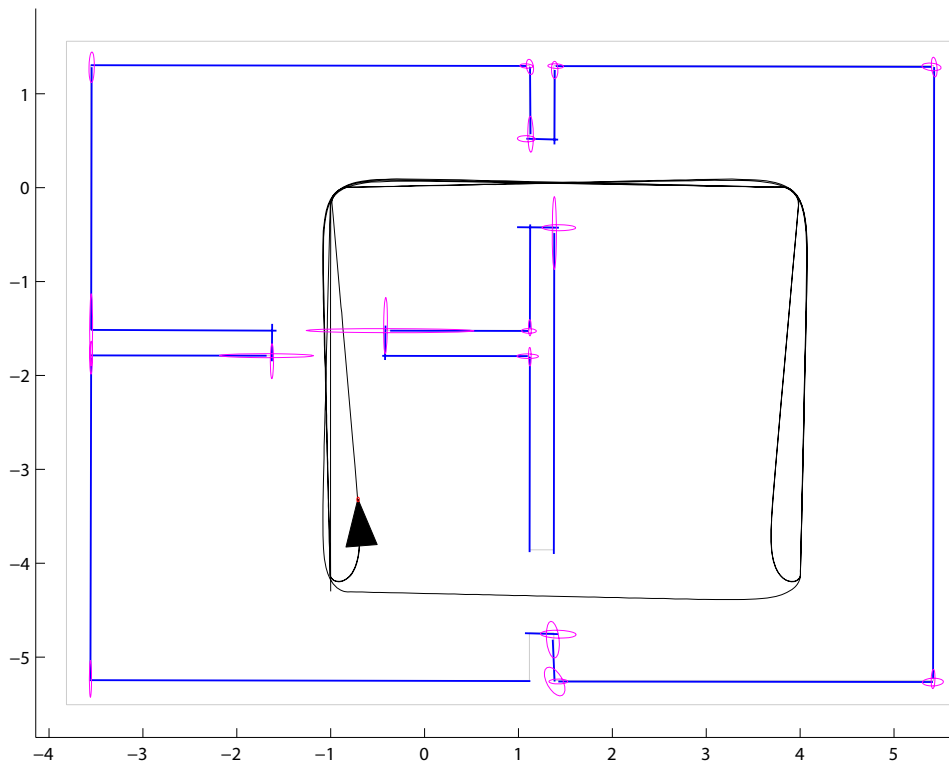
In each experiment, the robot starts from a known pose with  $P_{rr} = \mathbf{0}_{3 \times 3}$ . The SLAM process is relative to the initial robot pose. Assuming initial known location in the simulations is not a strong hypothesis, but is needed just because the simulation graphics (robot path, measured walls, raw measurements clouds) are superimposed on the CAD ground truth.

For each experiment are reported the figure showing the SLAM result superimposed to the ground truth, with the final 99.9% confidence ellipses on the detected lines endpoints and robot position. Other plots show the course of the robot pose error with respect to the confidence bands; the average line parameters errors absolute values against time (orientation, normal distance, and endpoints error, if detected); the standard deviations of the errors of the line orientation and normal distance; finally, the percentage of inconsistent feature estimates during the simulation.

## 5.2 Simple Environment

The first simulation is performed in a  $60 \text{ m}^2$  environment composed by three rooms. This simple map has been created in AUTOCAD. The real-time experiment would take about 6 minutes. Figure 5.3 shows the resulting map. The ground truth lines are light-colored, the lines mapped by the robot are darker. For each detected line endpoint, the relative confidence ellipse is shown. The confidence ellipse for the  $i$ -th feature endpoint is obtained from the feature covariance submatrix  $P_{pp_i}$ , after projecting it to the feature space,

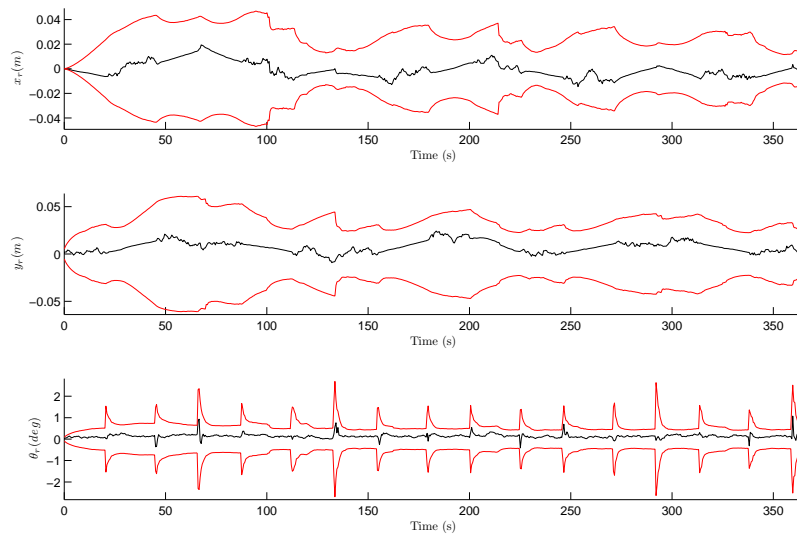
$$P_{ff_i} = \tilde{B}_f(\mathbf{x}_{f_i}) P_{pp_i} \tilde{B}_f^T(\mathbf{x}_{f_i}).$$



**Figure 5.3:** The map produced by the simulated single-robot SLAM in the small test environment.

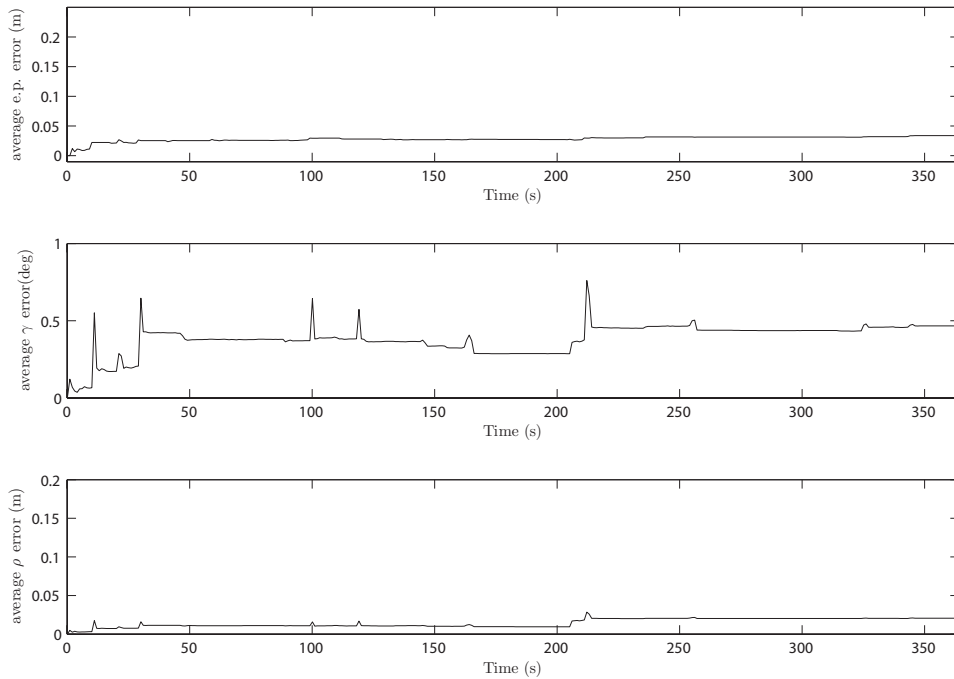
The robot pose error  $\tilde{\mathbf{x}}_r = [\tilde{x}_r \quad \tilde{y}_r \quad \tilde{\theta}_r]^T$  is shown in its components in Figure 5.4, compared with the 99.9% confidence bands. The robot pose estimate

remains consistent during the whole experiment. The peaks in the confidence bands of the robot orientation error correspond to the moments when the robot is turning. The robot remains well localized during the whole experiment, since the environment is small, and most of the features are initialized when the robot has low uncertainty. At the end of the experiment, the robot pose error is nearly zero.



**Figure 5.4:** Simple map - The robot pose error components  $\tilde{x}_r$ ,  $\tilde{y}_r$  and  $\tilde{\theta}_r$  compared with the correspondent 99.9% confidence bands.

The absolute value of the estimated map average errors are shown in Figure 5.5. In particular, the first subplot shows the average error of the detected line endpoints; the second subplot shows the average error of the lines orientations; the third subplot shows the average error on the lines normal distance (position). At the end of the experiment, the average endpoints error is about 3 *cm*, the average lines orientation error is less than 0.5 degrees, while the lines distance error is less than 2 *cm*. In Figure 5.6, the average standard deviation of the lines orientation error and of the line distance error are shown separately. The high peaks correspond to new features initialization. In correspondence

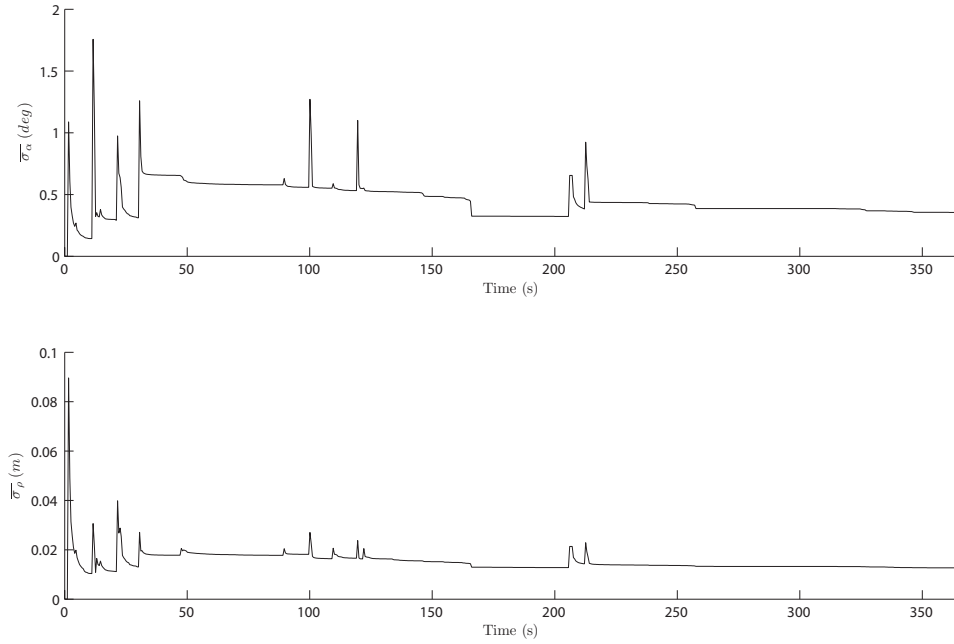


**Figure 5.5:** Simple map - Multiple figure showing the absolute values of the detected line endpoints average error, of the line orientation average error, and of the line normal distance average error.

of a loop closure at time  $t = 166$  s, the average standard deviations decrease.

### 5.3 Complex Environment

The SLAM algorithm has been tested in a  $3000\text{ m}^2$  scenario, using a simplified CAD map of the second floor of the University of Siena Engineering Faculty building, former the Insane Asylum “*S. Niccolò*”. This SLAM experiment is challenging both for the length of the loops that the robot must perform before coming back to already visited places, and for the high dimensionality of the state that comes out to be estimated (more than 200 features). The two loops in the simulation are closed successfully: the first loop is performed around the left courtyard, and the second around the right courtyard. The real-time experiment would take about 23 minutes. The Figure 5.7 shows the result of the single-robot SLAM experiment. Some zoomed views of the resulting map

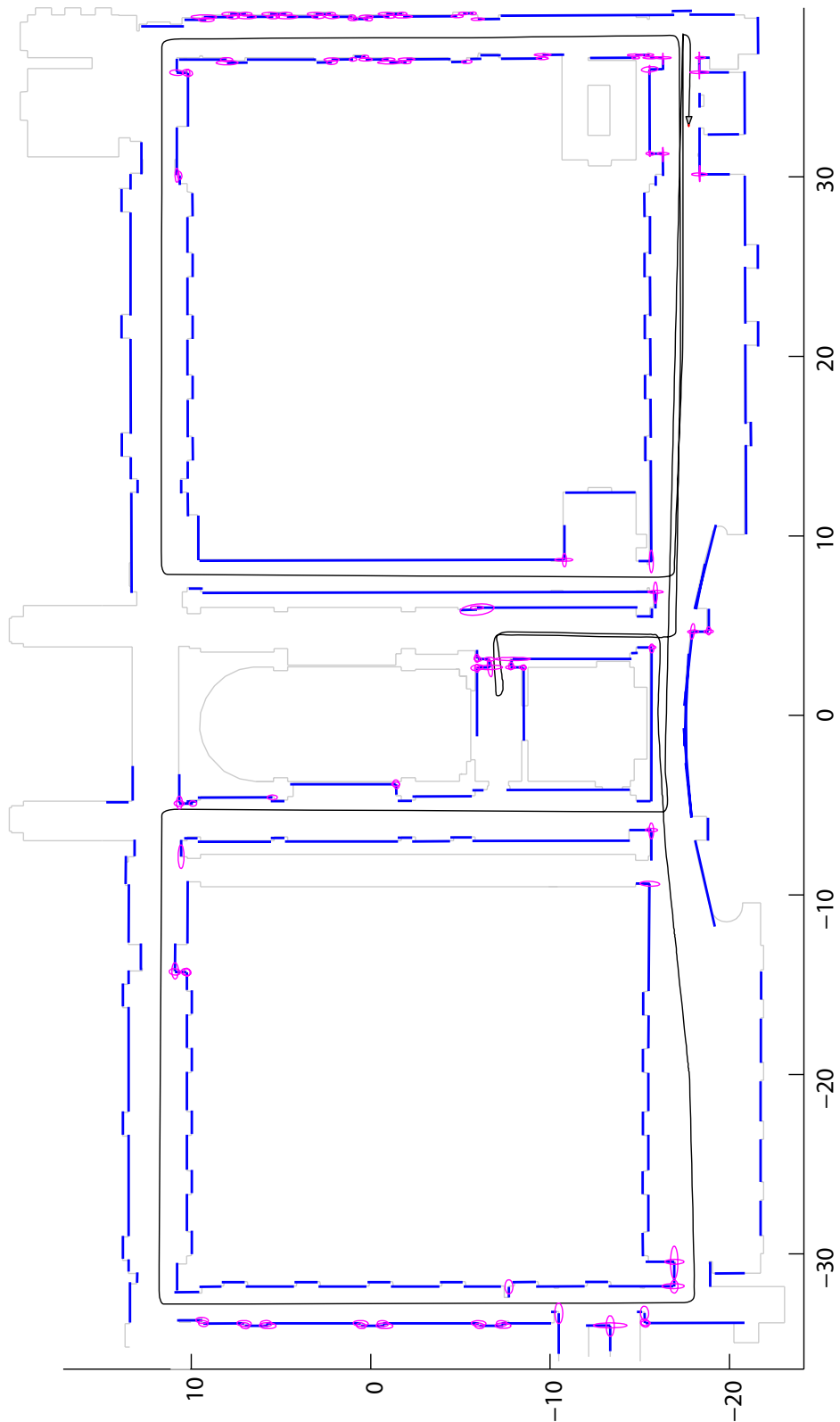


**Figure 5.6:** Simple map - The average standard deviation of the line orientation errors and of the line distance errors.

are shown in Figures 5.8, 5.9, and 5.10.

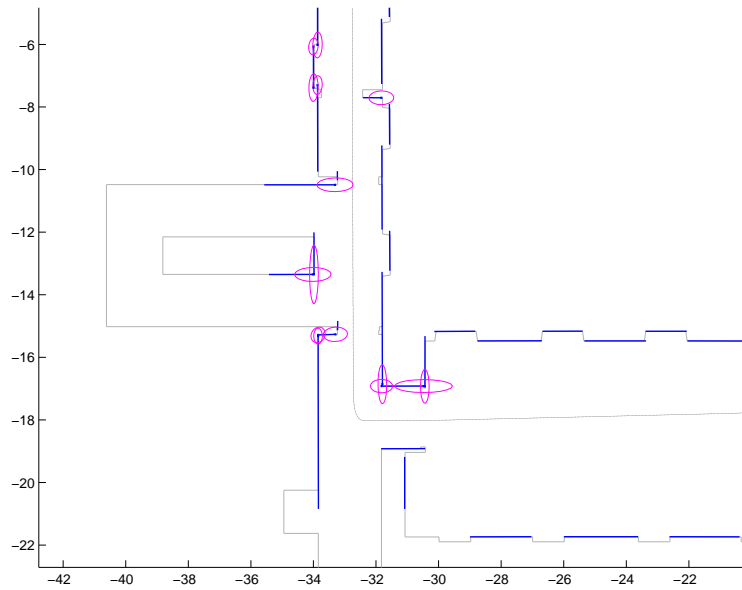
The robot pose error  $\tilde{\mathbf{x}}_r = [\tilde{x}_r \quad \tilde{y}_r \quad \tilde{\theta}_r]^T$  is shown in its components in Figure 5.11, compared with the 99.9% confidence bands. This plot shows that the robot pose estimate remains consistent during the whole experiment. The peaks in the confidence bands of the robot orientation error correspond to the moments when the robot is turning. At time  $t = 470$  s of the simulation, the confidence bands of the robot position collapse, in correspondence to the first loop closure. At time  $t = 1180$  s another loop closure occurs. The last loop closure is at time  $t = 1360$  s, when the robot comes back to its starting position: the robot position error is less than 1 cm and the orientation error is less than 0.2 degrees.

The absolute values of the estimated map average errors are shown in Figure 5.12. In particular, the first subplot shows the average error of the detected line endpoints; the second subplot shows the average error of the lines orientations; the third subplot shows the average error on the lines normal

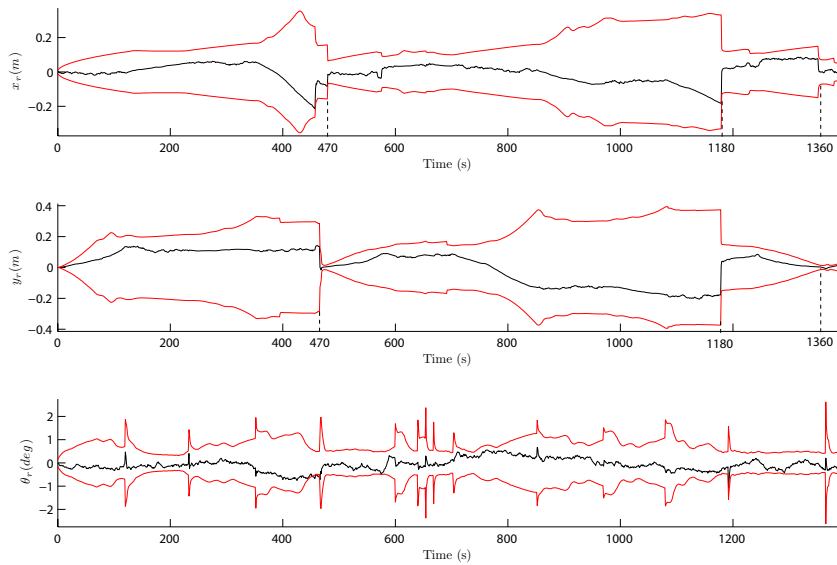


**Figure 5.7:** The map produced by the simulated single-robot SLAM in the S. Niccolò building.



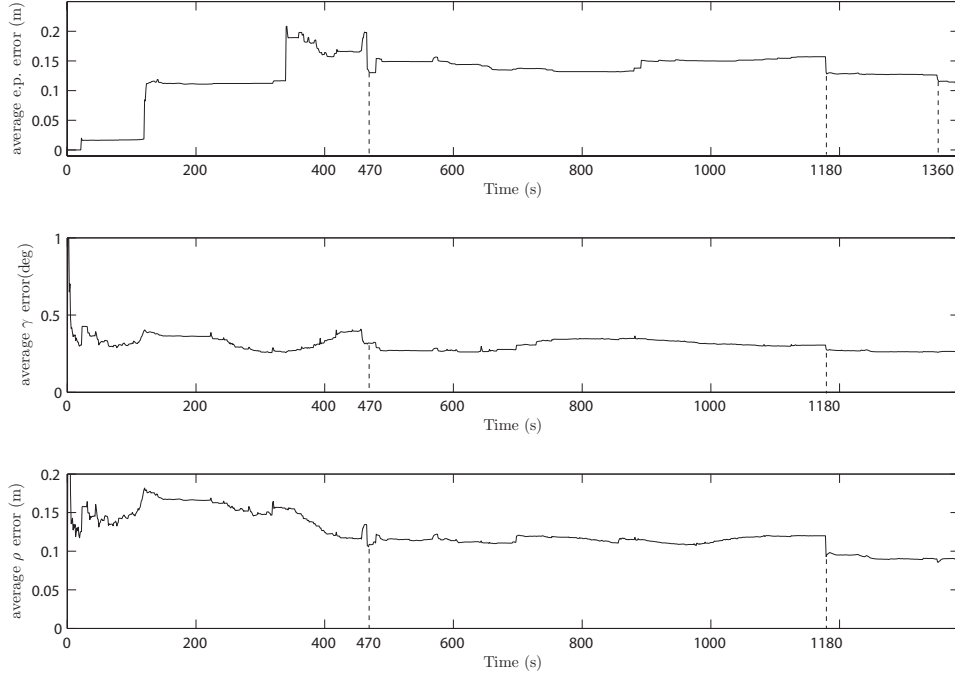


**Figure 5.10:** The third zoomed view of the S. Niccolò map.



**Figure 5.11:** Big map - The robot pose error components  $\tilde{x}_r$ ,  $\tilde{y}_r$  and  $\tilde{\theta}_r$  compared with the correspondent 99.9% confidence bands.

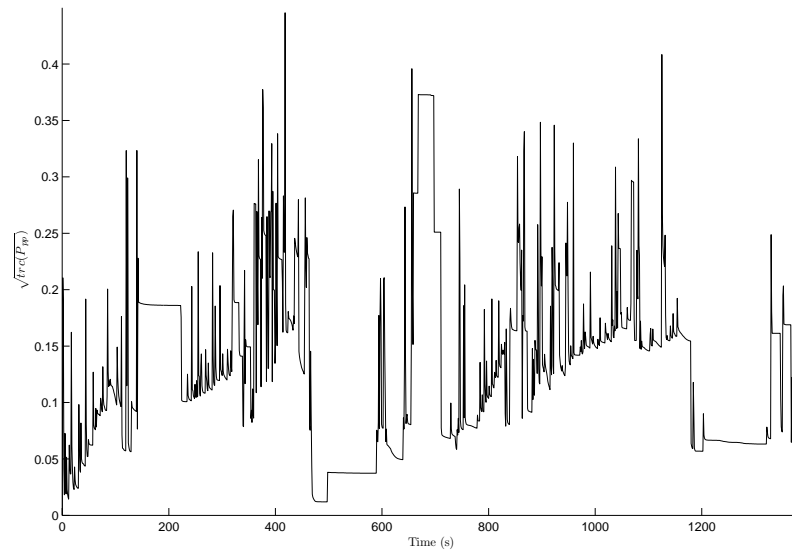
distance (position). At the end of the experiment, the average endpoints error is about 10 *cm*, the average lines orientation error is less than 0.5 degrees, while the lines distance error is less than 10 *cm*.



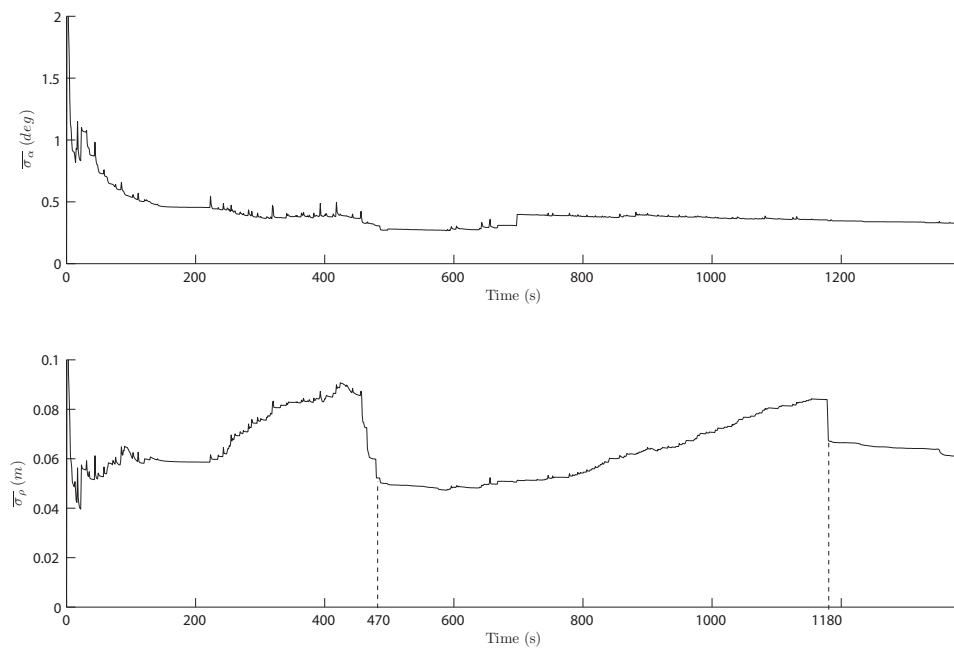
**Figure 5.12:** Big map - Multiple figure showing the absolute values of the detected line endpoints average error, of the line orientation average error, and of the line normal distance average error.

The qualitative plot of Figure 5.13 shows the course of the map covariance submatrix trace square root  $\sqrt{\text{trc}(P_{pp})}$ . The high peaks correspond to new features initialization. The trace remarkably decreases in correspondence of the loop closures mentioned above, at  $t = 470$  s and  $t = 1180$  s. In Figure 5.14, the average standard deviation of the lines orientation error and of the line distance error are shown separately.

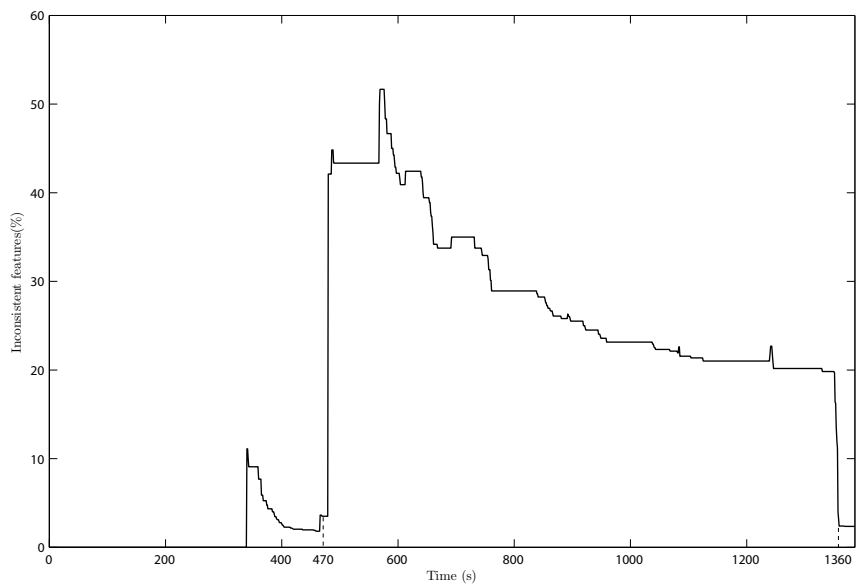
Finally, Figure 5.15 shows the percentage of inconsistent features plotted against time. In correspondence to the first loop closure, the number of inconsistent feature estimates jumps to 42%. This is due more to the shrinkage of the map error confidence bands, than to an effective error increase. At the end of the experiment, the inconsistent features are less than the 3% of the total.



**Figure 5.13:** Big map - The covariance submatrix trace square root plotted against time. The peaks are in correspondence of the initialization of new features.



**Figure 5.14:** Big map - The average standard deviation of the line orientation errors and of the line distance errors. The dashed vertical lines indicate the loop closures.



**Figure 5.15:** Big map - The percentage of inconsistent features. The dashed vertical lines indicate the loop closures.

# Chapter 6

## Multi-robot SLAM Simulations

Chapter 3 introduced the single-robot SLAM algorithm using EKF and M-Space feature representation. In Chapter 4, the map fusion algorithm, at the base of the multi-robot SLAM algorithm adopted in this thesis, is described. In Chapter 5, the experimental results of the single robot SLAM algorithm are shown. In this chapter, the multi-robot SLAM algorithm simulations results are presented and discussed. The experiments are executed using the original MATLAB-based simulator described in Section 5.1, using the same settings for the two robots.

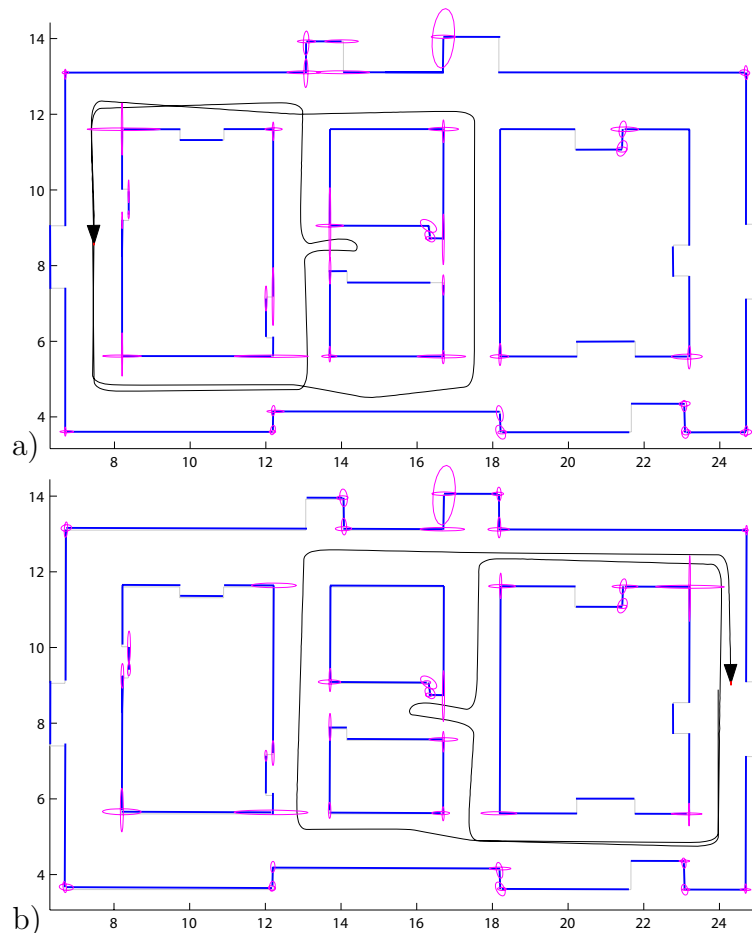
Some additional settings for the two-robot SLAM simulations are described in the following. Each robot is given a different set of waypoints to follow. Their paths are planned so that they can meet once, and fuse their maps. After the rendezvous, the robots continue the exploration in the area that was explored by the other robot. This area is not new, because after the map fusion, each robot gets a copy of the other one's map.

The robots can see each other within a range of  $2 m$ , inside the same field of view of the laser range finder, a 180 degrees sector in the front. The mutual measurements (see Section 4.1.2) are affected by Gaussian white noises, whose standard deviations are  $\sigma_{\rho_m} = 0.02 m$  and  $\sigma_{\theta_m} = 2\frac{\pi}{180} rad$  for the distance and bearing respectively. The data association thresholds  $T_D$  and  $T_\tau$  for the map matching (see Section 4.4) determine the tolerance with which the duplicate features are deemed to be the same.

In each experiment, the robots start from known poses with  $P_{rr_i} = \mathbf{0}_{3 \times 3}$ ,  $i = 1, 2$ . For each experiment are reported the two figures showing the SLAM results for the two robots, superimposed to the ground truth, with the final 99.9% confidence ellipses on the detected lines endpoints and robot position. Other plots show the course of the robot pose error with respect to the confidence bands; the average line parameters errors absolute values against time (orientation, normal distance, and endpoints error, if detected); the standard deviations of the errors of the line orientation and normal distance; finally, the percentage of inconsistent features during the simulation.

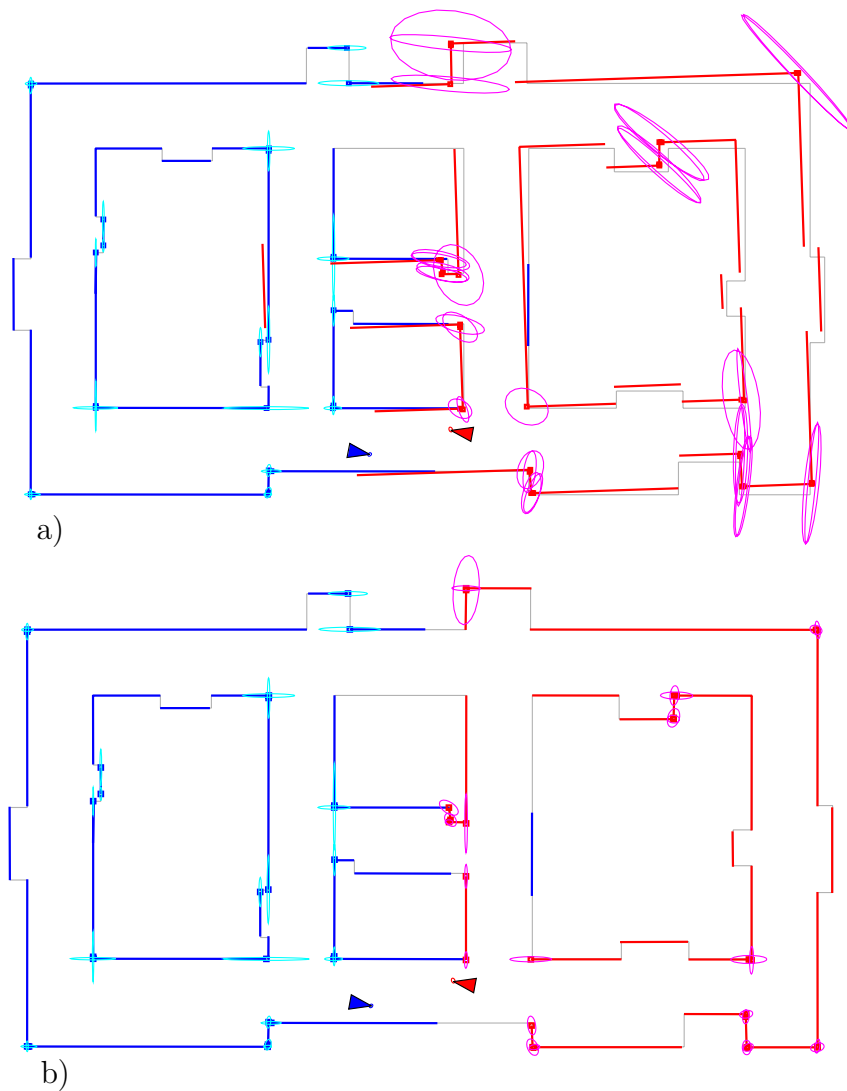
## 6.1 Simple Environment

The first simulation is performed in a  $150\text{ m}^2$  environment, resembling the second floor of the “S. Niccolò” building. This simple map has been created in AUTOCAD. The real-time experiment would take about 6 minutes. The robots close the loop about at  $t = 145\text{ s}$ , and meet at  $t = 209\text{ s}$ . The resulting maps made by the robots  $R_1$  and  $R_2$  are shown in Figure 6.1 (a) and (b) respectively. The ground truth lines are light-colored, the lines mapped by the robot are darker. For each detected line endpoint, the relative confidence ellipse is shown.



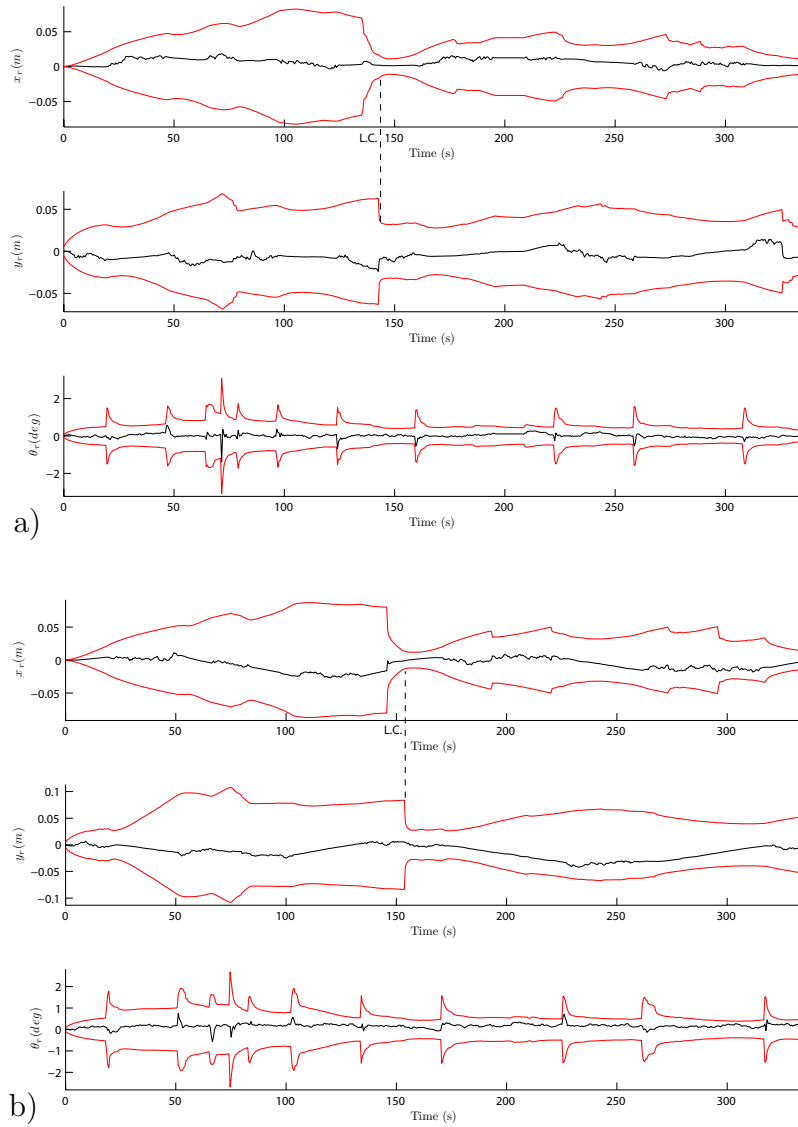
**Figure 6.1:** The maps produced by the simulated multi-robot SLAM in the small test environment. (a) The map of robot  $R_1$ . (b) The map of robot  $R_2$ .

As described in Section 4.4, the duplicate features that are found in the merged map after the alignment are used as constraints to improve the map fusion. The effect of these updates is shown in Figure 6.2, before (a) and after having imposed the constraints and having removed the matching duplicate features (b).



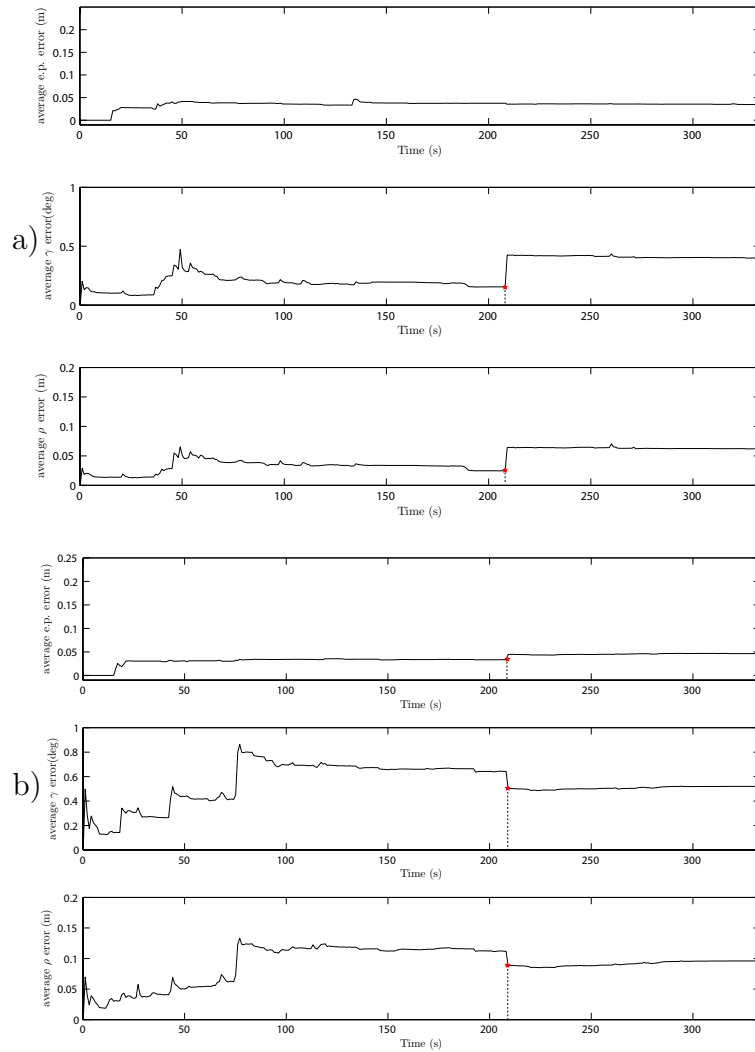
**Figure 6.2:** Simple map fusion - The matching duplicate features are used as constraints to improve the map alignment: (a) before the updates, (b) after imposing the constraints.

The robots pose errors  $\tilde{\mathbf{x}}_r = [\tilde{x}_r \quad \tilde{y}_r \quad \tilde{\theta}_r]^T$  are shown in their components in Figure 6.3, compared with the 99.9% confidence bands. The robots pose estimates are consistent, and the robots remain well localized during the whole experiment. At the end, the robots pose errors are nearly zero.



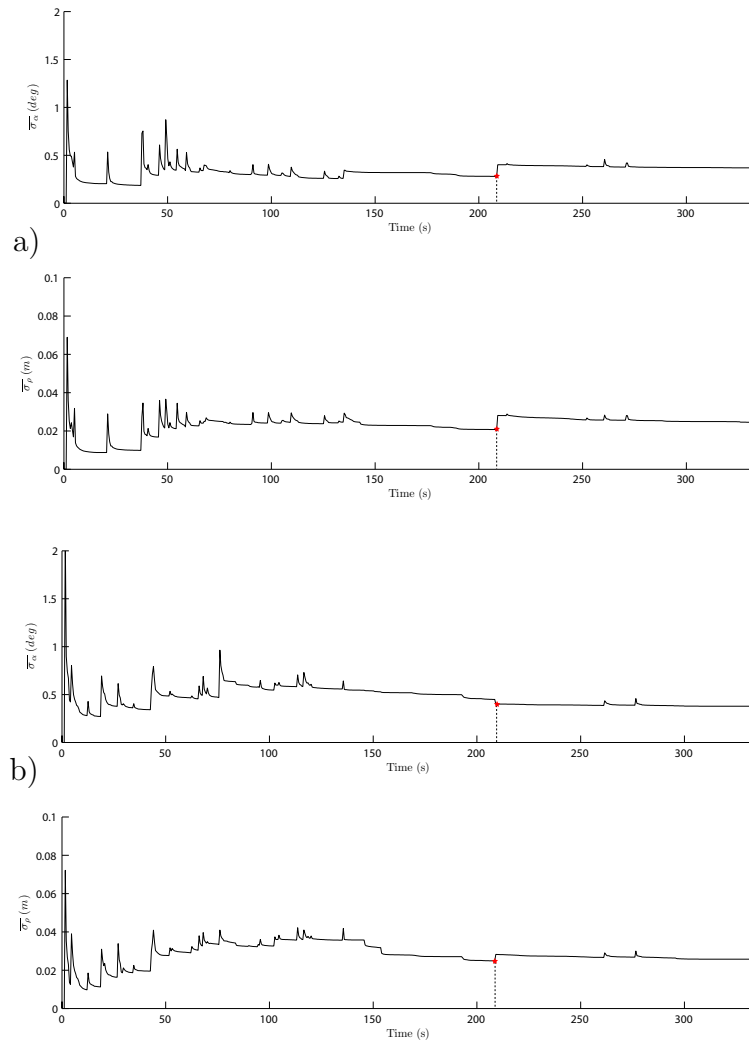
**Figure 6.3:** Simple map fusion - The robots pose errors components  $\tilde{x}_r$ ,  $\tilde{y}_r$  and  $\tilde{\theta}_r$  compared with the correspondent 99.9% confidence bands: (a) robot  $R_1$ , (b) robot  $R_2$ . The loop closures are marked with L.C.

The average error absolute values of the maps estimated by the robots are shown in Figure 6.4. At the end of the experiment, for robot  $R_1$ , the average endpoints error is less than 4 *cm*, the average lines orientation error is less than 0.4 degrees, while the lines distance error is less than 7 *cm*; for robot  $R_2$ , the average errors are less than 5 *cm*, less than 0.6 degrees, and less than 10 *cm* respectively.



**Figure 6.4:** Simple map fusion - Multiple figure showing the absolute values of the detected line endpoints average error, of the line orientation average error, and of the line normal distance average error: (a) robot  $R_1$ , (b) robot  $R_2$ . The star marks indicate the instant of the rendezvous.

In Figure 6.5, the average standard deviation of the lines orientation error and of the line distance error are shown separately, for both robots. The high peaks correspond to new features initialization.



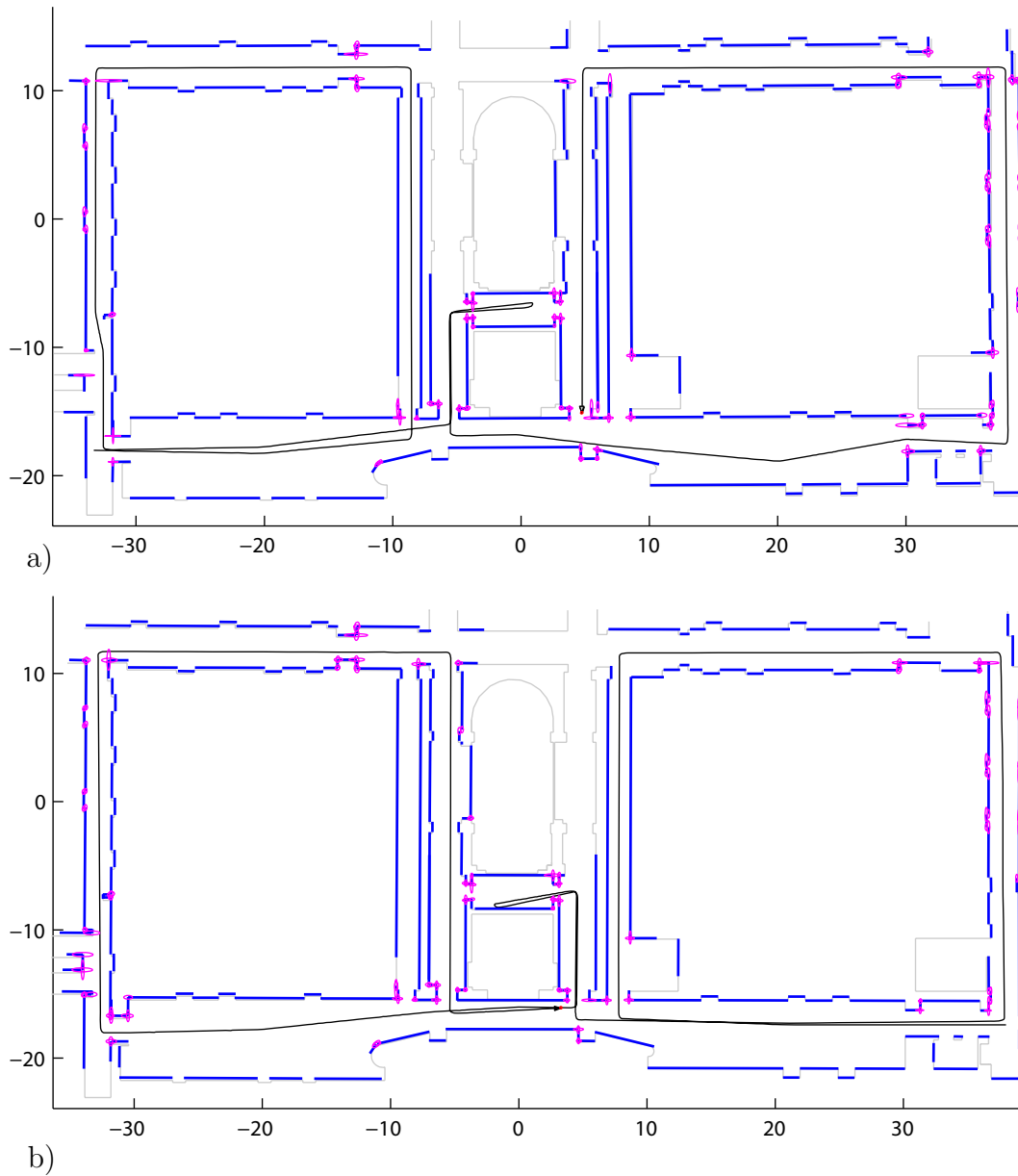
**Figure 6.5:** Simple map fusion - The average standard deviation of the line orientation errors and of the line distance errors: (a) robot  $R_1$ , (b) robot  $R_2$ . The star marks indicate the instant of the rendezvous.

The maps of the two robots are completely consistent before the rendezvous. After the map fusion, the inconsistent features in both maps are less than the 2% of the total.

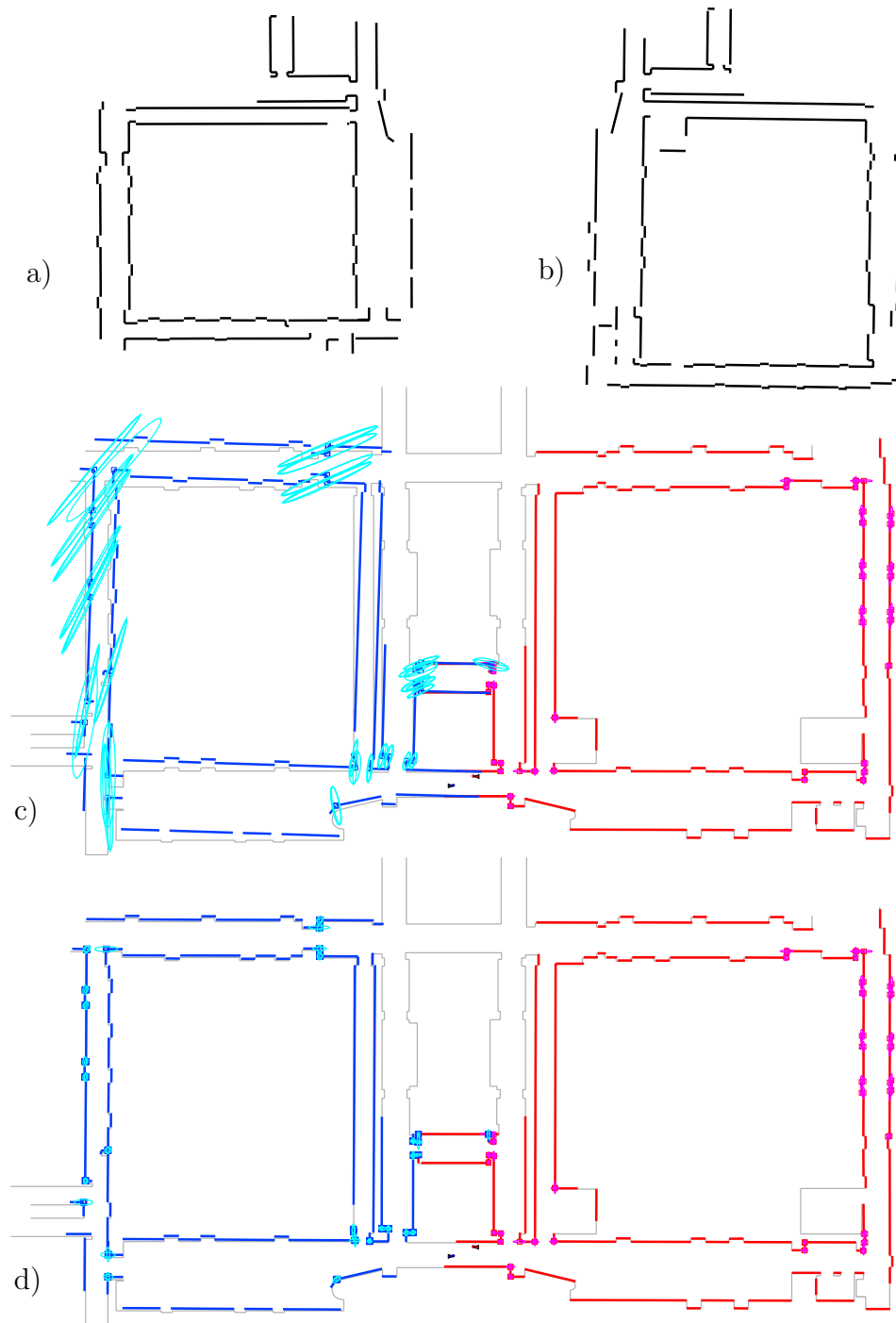
## 6.2 Complex Environment

The multi-robot SLAM algorithm has been tested in the “S. Niccolò” building, the same 3000  $m^2$  scenario of the experiment shown in Section 5.3. The robots first explore the area around a courtyard each, and then meet to fuse the two maps. The robot  $R_1$  closes the loops around the left courtyard around  $t = 530$  s, and robot  $R_2$  closes its loop around the right courtyard at  $t = 590$  s. They meet at  $t = 925$  s: if the experiment ended at this time, each robot would have had the map of almost the entire floor already. But the experiment goes on, and each of the robots goes to explore the area covered by the other robot, remaining localized quite well, exploiting the information got after the rendezvous. The real-time experiment would take about 26 minutes. The Figure 6.6 shows the maps produced by the SLAM algorithm for robot  $R_1$  (subfigure (a)), and robot  $R_2$  (subfigure (b)).

As described in Section 4.4, the duplicate features that are found in the merged map after the alignment are used as constraints to improve the map fusion. The effect of these updates is shown in Figure 6.7, before (c) and after having imposed the constraints and having removed the matching duplicate features (d).

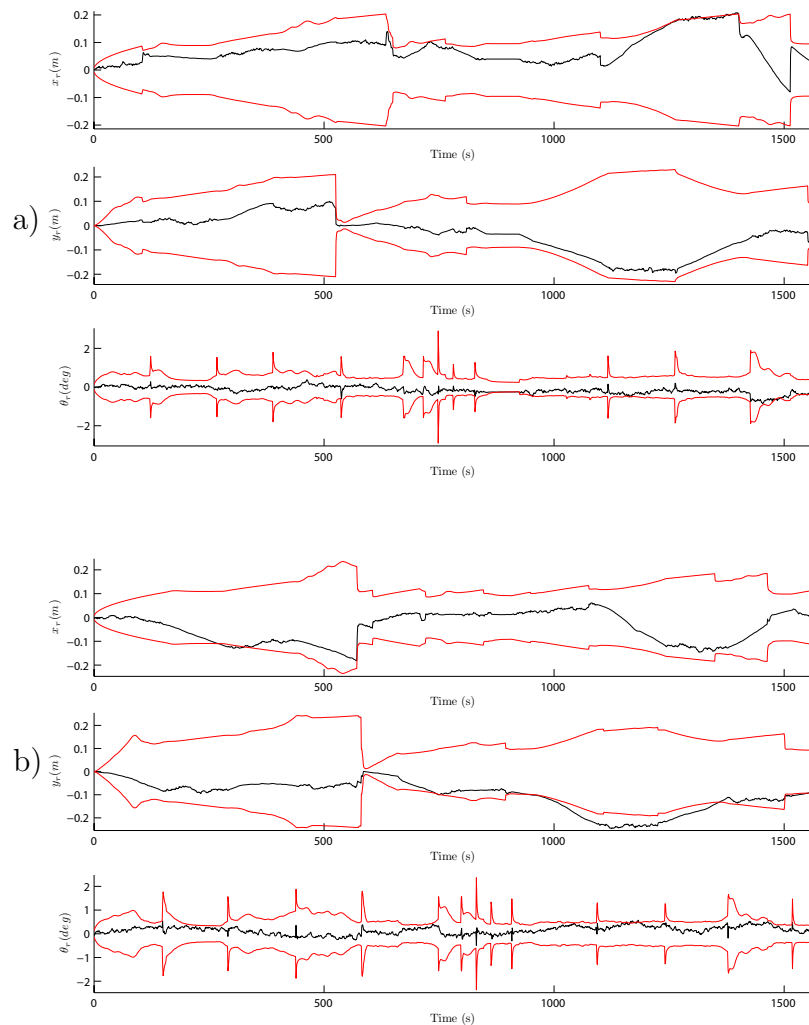


**Figure 6.6:** The maps produced by the simulated multi-robot SLAM in the S. Niccolò building: (a) the map of robot  $R_1$ , (b) the map of robot  $R_2$ .



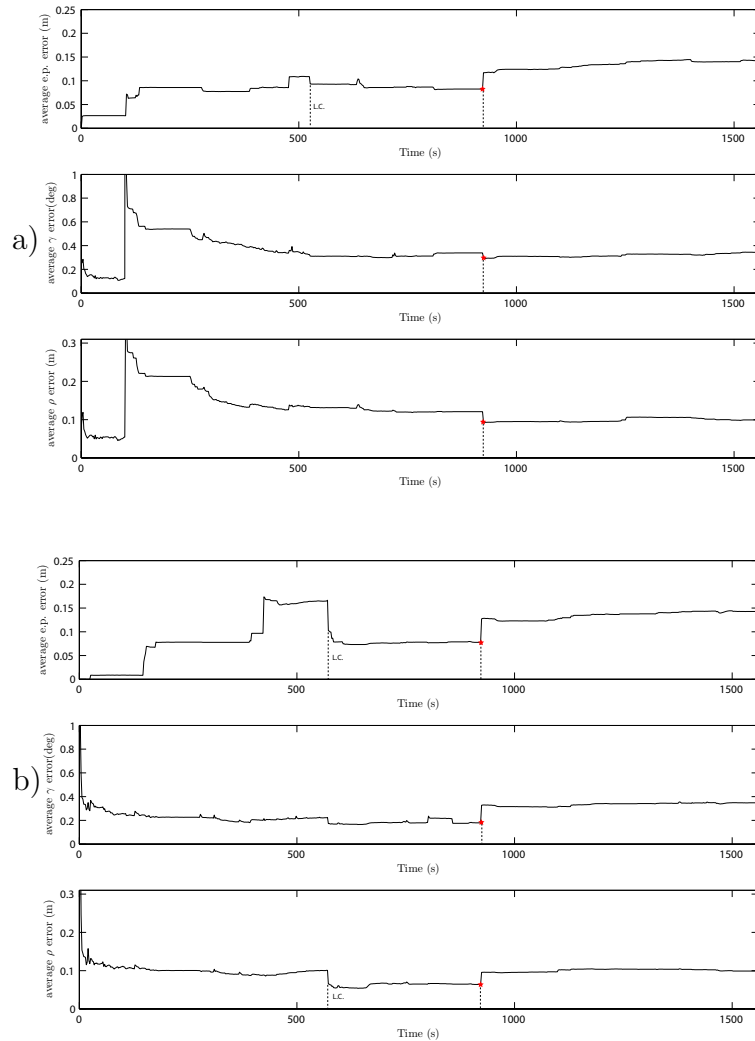
**Figure 6.7:** Big map fusion - The matching duplicate features are used as constraints to improve the map alignment: (a) the map of robot  $R_1$  before the rendezvous, (b) the map of robot  $R_2$  before the rendezvous, (c) the fused map before imposing the constraints (d) the fused map after imposing the constraints.

The robots pose errors are shown in their components in Figure 6.8, compared with the 99.9% confidence bands. The robots pose estimates lose consistency after the map fusion, because the robots try to localize with respect to some inherited features that are not consistent. At the end, the robot  $R_1$  position error is less than 7 cm and orientation error is less than 0.5 degrees; the robot  $R_2$  position error is less than 10 cm and orientation error is less than 0.2 degrees.



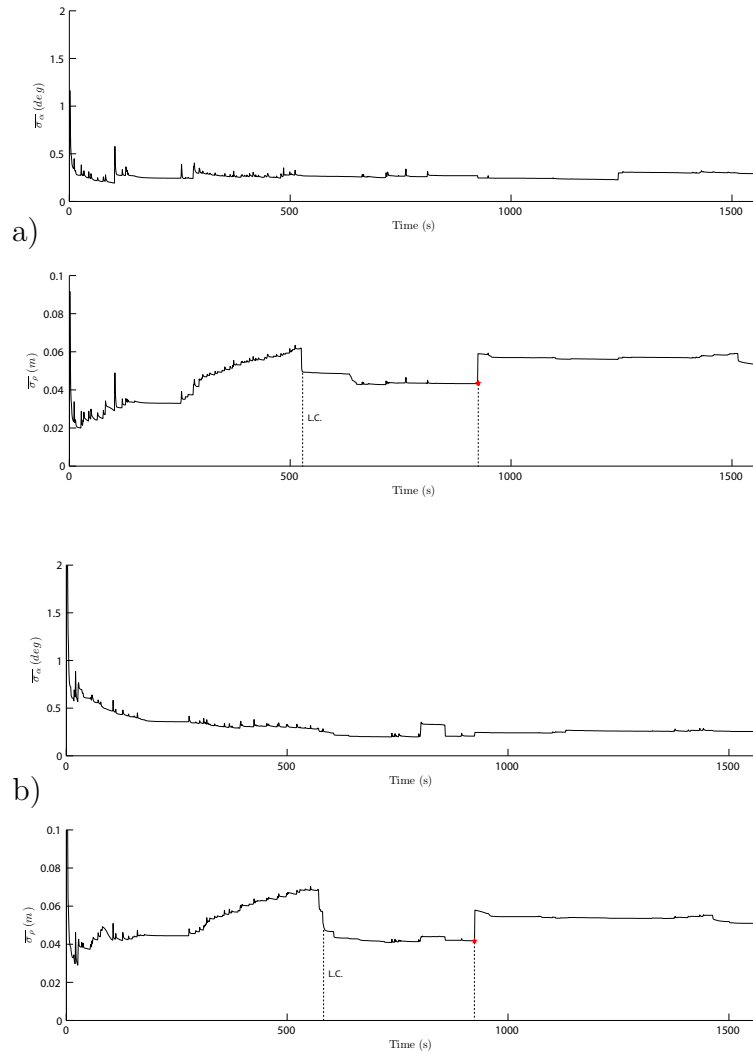
**Figure 6.8:** Big map fusion - The robots pose errors components  $\tilde{x}_r$ ,  $\tilde{y}_r$  and  $\tilde{\theta}_r$  compared with the correspondent 99.9% confidence bands: (a) robot  $R_1$ , (b) robot  $R_2$ .

The average error absolute values of the maps, estimated by the robots, are shown in Figure 6.9. At the end of the experiment, for robot  $R_1$ , the average endpoints error is less than 20 cm, the average lines orientation error is less than 0.4 degrees, while the lines distance error is less than 12 cm; for robot  $R_2$ , the average errors are less than 15 cm, less than 0.35 degrees, and less than 10 cm respectively.



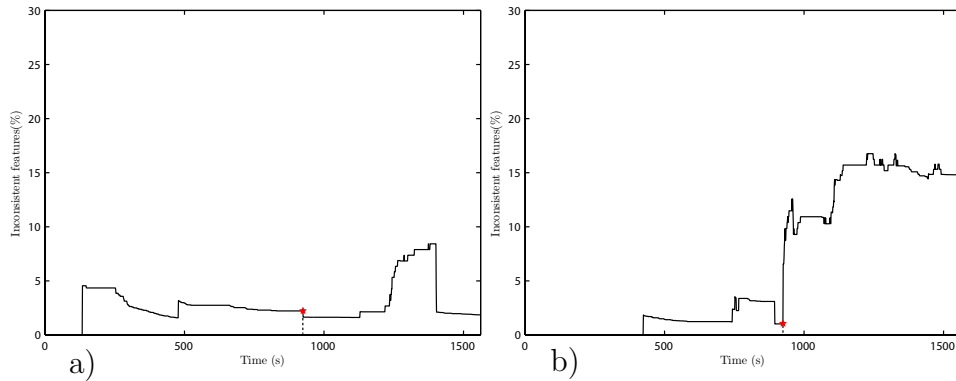
**Figure 6.9:** Big map fusion - Multiple figure showing the absolute values of the detected line endpoints average error, of the line orientation average error, and of the line normal distance average error: (a) robot  $R_1$ , (b) robot  $R_2$ . The star marks indicate the instant of rendezvous, while the loop closures are marked with L.C.

In Figure 6.5, the average standard deviation of the lines orientation error and of the line distance error are shown separately, for both robots. The high peaks correspond to new features initialization.



**Figure 6.10:** Big map fusion - The average standard deviation of the line orientation errors and of the line distance errors: (a) robot  $R_1$ , (b) robot  $R_2$ .

Figure 6.11 shows the percentage of inconsistent features for the two robots plotted against time. The inconsistent features in map  $M_1$  are less than the 3% at the end of the experiment, while the inconsistent features in map  $M_2$  arrive to the 15% after the rendezvous.



**Figure 6.11:** Big map - The percentage of inconsistent features in the maps built by: (a) robot  $R_1$ , (b) robot  $R_2$ . The star marks are indicate the instant of the rendezvous.

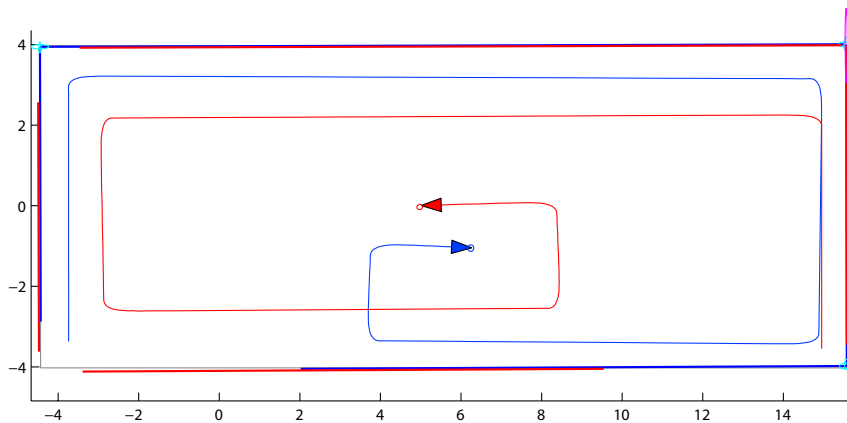
### 6.3 Performance Comparison

In this section, the advantages of multi-robot SLAM are discussed from a qualitative point of view. The map fusion is shown to solve the loop closure problem, using the quantitative results of several experiments carried out in a simple environment. As introduced in Section 2.3, the advantages of multi-robot SLAM compared against single-robot SLAM are:

- the ability to build the map of the same environment in less time, achieving the same quality in terms of errors and uncertainty of the map;
- the improved quality of the map, given the same exploration time;
- the fact that each robot has to maintain a smaller number of features, at least until the rendezvous.

The multi-robot SLAM simulations presented in this chapter showed how a team of two robots can build a map in less time than a single robot would do. In the simple environment case, a single robot would take 7'15" to build the map, while two robots take only 3'29"; after the rendezvous, their maps are mutually completed by the map fusion. In the complex environment case, the time required by two robots is about 15 minutes, instead of 23 minutes needed by a single robot.

In order to test the map quality improvement, given the same exploration time and the same path travelled, some experiments have been performed in particular conditions. The robot-to-robot measurement noises are now set to an order of magnitude less than the laser range finder measurements, i.e. the rendezvous is assumed to be *almost ideal*. The test environment is shown in Figure 6.12. The environment has only 4 features.



**Figure 6.12:** Test environment used to compare the performance of multi-robot SLAM against single-robot SLAM.

During their travel, both robots see all the features, but with increasing uncertainty: in fact, the more the time passed from the experiment beginning, the more the motion uncertainty affects the initialized features. The robots *do not close* the loop on their own, so they can only rely on the information provided by the other robot, after the map fusion, to close the loop and to improve both the map and their pose estimate.

First, some experiments are run without allowing the robots to meet and fuse the maps, i.e. they run in the same environment without interacting. In this case, the pose and map uncertainty cannot be reduced, since the robots do not close the loop and do not meet. The results of these single robot experiments are reported in Tables 6.1 and 6.2. Each table reports, for each row (experiment), the line orientation and position errors  $|e_\gamma|$  and  $|e_\rho|$ , the corresponding standard deviations  $\bar{\sigma}_\gamma$  and  $\bar{\sigma}_\rho$ , the robot pose error  $|e_{\mathbf{p}_R}|$ ,  $|e_{\theta_R}|$ , and the standard deviation of the robot pose components error  $\sigma_x$ ,  $\sigma_y$ ,  $\sigma_\theta$ .

Then, other experiments are run allowing the robot rendezvous and the map fusion. In this case, the uncertainty and the errors of the robots poses and of the maps are reduced, by imposing the constraints between the duplicate features in the maps. The results are reported in Tables 6.3 and 6.4.

**Table 6.1:** Results of single-robot SLAM experiments (robot  $R_1$ ): the last row reports the average values of the columns.

$ e_\gamma $	$ e_\rho $	$\bar{\sigma}_\gamma$	$\bar{\sigma}_\rho$	$ e_{\mathbf{p}_R} $	$ e_{\theta_R} $	$\sigma_x$	$\sigma_y$	$\sigma_\theta$
0.3085	0.0084	0.0815	0.0416	0.0561	0.0239	0.0661	0.0591	0.9315
0.6492	0.0608	0.1105	0.0406	0.0585	0.0235	0.0688	0.0709	0.9825
0.0801	0.0253	0.0803	0.0416	0.1043	0.0119	0.0661	0.0587	0.9166
0.2278	0.0485	0.1125	0.0407	0.1610	0.0020	0.0691	0.0707	0.9733
0.0611	0.0074	0.0804	0.0417	0.0500	0.0016	0.0650	0.0570	0.8688
0.2543	0.0279	0.1110	0.0406	0.0487	0.0091	0.0695	0.0707	0.9815
0.2857	0.0153	0.0787	0.0415	0.0498	0.0081	0.0656	0.0584	0.9235
0.1969	0.0624	0.1129	0.0407	0.1790	0.0038	0.0689	0.0701	0.9673
0.0542	0.0091	0.0794	0.0415	0.0239	0.0034	0.0652	0.0578	0.8908
0.5863	0.0276	0.1141	0.0405	0.1764	0.0230	0.0692	0.0694	0.9627
0.0768	0.0274	0.0806	0.0418	0.0536	0.0052	0.0658	0.0585	0.9088
0.1774	0.0212	0.1127	0.0407	0.0812	0.0080	0.0692	0.0702	0.9661
0.1759	0.0080	0.0803	0.0417	0.0761	0.0034	0.0657	0.0585	0.9019
0.1253	0.0106	0.1122	0.0406	0.1220	0.0057	0.0691	0.0699	0.9650
0.0646	0.0070	0.0778	0.0413	0.0246	0.0017	0.0662	0.0589	0.9441
0.5182	0.0294	0.1116	0.0406	0.0739	0.0361	0.0684	0.0713	0.9766
<b>0.2401</b>	<b>0.0248</b>	<b>0.0960</b>	<b>0.0411</b>	<b>0.0837</b>	<b>0.0107</b>	<b>0.0674</b>	<b>0.0644</b>	<b>0.9413</b>

The results of the single and multi-robot cases are summarized in Table 6.5, averaging the results of the two robots for each case. From these results is clear that in the multi-robot case, the map fusion improve the map quality in term of error and uncertainty reduction. Also the robot poses estimates are improved. Since the uncertainty introduced by the mutual robot measurements is very

**Table 6.2:** Results of single-robot SLAM experiments (robot  $R_2$ ): the last row reports the average values of the columns.

$ e_\gamma $	$ e_\rho $	$\bar{\sigma}_\gamma$	$\bar{\sigma}_\rho$	$ e_{\mathbf{p}_R} $	$ e_{\theta_R} $	$\sigma_x$	$\sigma_y$	$\sigma_\theta$
0.0880	0.0363	0.0798	0.0414	0.0721	0.0028	0.0660	0.0586	0.9205
0.2032	0.0494	0.1130	0.0408	0.0295	0.0197	0.0697	0.0705	0.9797
0.2373	0.0283	0.0809	0.0417	0.0278	0.0236	0.0661	0.0582	0.9068
0.2907	0.0275	0.1111	0.0407	0.0638	0.0302	0.0698	0.0706	0.9794
0.1546	0.0224	0.0809	0.0417	0.0212	0.0034	0.0654	0.0578	0.8874
0.0807	0.0476	0.1124	0.0407	0.1503	0.0068	0.0693	0.0710	0.9827
0.0899	0.0094	0.0810	0.0417	0.0602	0.0156	0.0660	0.0583	0.9026
0.2649	0.0458	0.1129	0.0407	0.1622	0.0044	0.0693	0.0705	0.9815
0.1117	0.0384	0.0813	0.0417	0.1259	0.0014	0.0655	0.0578	0.8867
0.1636	0.0070	0.1108	0.0406	0.0442	0.0031	0.0689	0.0704	0.9694
0.1044	0.0214	0.0792	0.0414	0.1080	0.0215	0.0661	0.0589	0.9420
0.3539	0.0201	0.1132	0.0407	0.1590	0.0152	0.0696	0.0706	0.9795
0.1576	0.0330	0.0819	0.0418	0.0732	0.0071	0.0658	0.0587	0.9063
0.2765	0.0607	0.1138	0.0406	0.1298	0.0190	0.0698	0.0707	0.9802
0.0960	0.0291	0.0800	0.0417	0.0997	0.0043	0.0660	0.0586	0.9158
0.2077	0.0578	0.1139	0.0406	0.1428	0.0230	0.0693	0.0712	0.9851
<b>0.1800</b>	<b>0.0334</b>	<b>0.0966</b>	<b>0.0412</b>	<b>0.0918</b>	<b>0.0126</b>	<b>0.0677</b>	<b>0.0645</b>	<b>0.9441</b>

small, the duplicate features (inherited from the other map), used as pseudo-measurements, have about the same uncertainty as the corresponding features already in the map. This corresponds to measuring the same features twice with comparable precision, thus reducing the uncertainty of the estimates.

**Table 6.3:** Results of multi-robot SLAM experiments (robot  $R_1$ ): the last row reports the average values of the columns.

$ e_\gamma $	$ e_\rho $	$\bar{\sigma}_\gamma$	$\bar{\sigma}_\rho$	$ e_{\mathbf{p}_R} $	$ e_{\theta_R} $	$\sigma_x$	$\sigma_y$	$\sigma_\theta$
0.1806	0.0148	0.0636	0.0209	0.0244	0.0026	0.0253	0.0272	0.4749
0.1960	0.0328	0.0706	0.0206	0.0501	0.0050	0.0231	0.0213	0.2255
0.0785	0.0286	0.0661	0.0326	0.0634	0.0019	0.0376	0.0400	0.5693
0.1335	0.0287	0.0736	0.0316	0.0370	0.0005	0.0336	0.0338	0.2985
0.0879	0.0069	0.0657	0.0326	0.0632	0.0166	0.0374	0.0419	0.6400
0.1216	0.0269	0.0735	0.0316	0.0548	0.0006	0.0335	0.0338	0.2957
0.1855	0.0131	0.0669	0.0327	0.0580	0.0131	0.0375	0.0409	0.5998
0.1197	0.0153	0.0669	0.0324	0.0161	0.0039	0.0372	0.0417	0.6362
0.0813	0.0125	0.0669	0.0327	0.0245	0.0081	0.0376	0.0409	0.6027
0.2336	0.0364	0.0741	0.0316	0.0532	0.0114	0.0337	0.0338	0.2989
0.2627	0.0187	0.0803	0.0428	0.0225	0.0087	0.0377	0.0465	0.5438
0.2397	0.1148	0.0879	0.0419	0.2014	0.0109	0.0338	0.0420	0.3045
0.2514	0.0123	0.0665	0.0326	0.0219	0.0046	0.0375	0.0377	0.4868
0.2459	0.0599	0.0744	0.0317	0.0882	0.0001	0.0336	0.0338	0.2998
0.0864	0.0219	0.0664	0.0327	0.0441	0.0013	0.0375	0.0407	0.5991
0.0677	0.0250	0.0745	0.0317	0.0416	0.0034	0.0336	0.0338	0.2961
<b>0.1608</b>	<b>0.0293</b>	<b>0.0711</b>	<b>0.0320</b>	<b>0.0540</b>	<b>0.0058</b>	<b>0.0344</b>	<b>0.0369</b>	<b>0.4482</b>

**Table 6.4:** Results of multi-robot SLAM experiments (robot  $R_2$ ): the last row reports the average values of the columns.

$ e_\gamma $	$ e_\rho $	$\bar{\sigma}_\gamma$	$\bar{\sigma}_\rho$	$ e_{\mathbf{p}_R} $	$ e_{\theta_R} $	$\sigma_x$	$\sigma_y$	$\sigma_\theta$
0.0674	0.0140	0.0658	0.0323	0.0211	0.0071	0.0376	0.0388	0.5409
0.0205	0.0089	0.0650	0.0324	0.0406	0.0074	0.0375	0.0388	0.5379
0.1228	0.0139	0.0730	0.0317	0.0136	0.0063	0.0339	0.0338	0.3000
0.1221	0.0124	0.0664	0.0324	0.0097	0.0024	0.0374	0.0375	0.4885
0.2073	0.0234	0.0659	0.0325	0.0343	0.0031	0.0376	0.0382	0.5120
0.0760	0.0301	0.0736	0.0317	0.0531	0.0040	0.0337	0.0338	0.2995
0.0665	0.0139	0.0742	0.0317	0.0383	0.0013	0.0335	0.0338	0.2966
0.2859	0.0217	0.0656	0.0324	0.0430	0.0093	0.0373	0.0447	0.7325
0.1939	0.0610	0.0732	0.0315	0.0920	0.0046	0.0331	0.0336	0.2903
0.1628	0.0285	0.0653	0.0325	0.0405	0.0037	0.0374	0.0384	0.5096
0.1114	0.0218	0.0656	0.0326	0.0639	0.0010	0.0375	0.0400	0.5705
0.1185	0.0110	0.0655	0.0325	0.0234	0.0003	0.0374	0.0418	0.6367
0.2232	0.0857	0.0741	0.0317	0.1259	0.0079	0.0337	0.0339	0.3004
0.2083	0.0175	0.0657	0.0325	0.0356	0.0074	0.0377	0.0373	0.4733
0.1439	0.0241	0.0733	0.0316	0.0613	0.0025	0.0338	0.0339	0.3022
0.5528	0.0506	0.0896	0.0466	0.0572	0.0150	0.0586	0.0338	0.2970
<b>0.1677</b>	<b>0.0274</b>	<b>0.0701</b>	<b>0.0331</b>	<b>0.0471</b>	<b>0.0052</b>	<b>0.0374</b>	<b>0.0370</b>	<b>0.4430</b>

**Table 6.5:** Average results of the single and multi-robot SLAM experiments.

	$ e_\gamma $	$ e_\rho $	$\bar{\sigma}_\gamma$	$\bar{\sigma}_\rho$	$ e_{\mathbf{p}_R} $	$ e_{\theta_R} $	$\sigma_x$	$\sigma_y$	$\sigma_\theta$
<b>single</b>	0.2101	0.0291	0.0963	0.0411	0.0878	0.0116	0.0675	0.0644	0.9427
<b>multi</b>	0.1642	0.0283	0.0706	0.0325	0.0506	0.0055	0.0359	0.0369	0.4456

# Chapter 7

## Conclusions

In this thesis, a multi-robot SLAM algorithm has been presented. The map of the environment is built using line segments and the features uncertainty is represented using the M-Space representation.

The M-Space feature representation allows one to specify the line segment feature location and extent, while expressing its uncertainty in a different space, corresponding to the partial information provided by the robot sensors. The uncertainty is expressed in a frame attached to the features, so to avoid the lever-arm effect.

In particular, the single-robot SLAM problem has been posed as a state estimation problem, and solved with the Extended Kalman Filter. The multi-robot SLAM algorithm relies on the fact that the robots meet and measure their relative distance and bearing to each other. These measurements are used to compute the transformation between their maps reference frames. If the two maps significantly overlap, the duplicate features found in the aligned map are used to impose constraints between the two maps, improving the quality of the alignment. This map fusion approach does not assume that the robots initial positions are known. The multi-robot SLAM algorithm has been described for the two-robot case, but can be extended to a larger team of robots, repeating the map fusion procedure for each pair of robots.

The SLAM algorithm has been tested with a custom simulator, developed in the MATLAB environment. The simulated robots are unicycles and are

equipped with a laser range finder, which working is implemented by a raytracing algorithm. The simulator can load CAD maps of real indoor environments, allowing realistic performance tests. First the single-robot SLAM algorithm has been tested in a small map and subsequently in a 3000  $m^2$  scenario, using a simplified CAD map of the second floor of the University of Siena Engineering Faculty building.

The multi-robot SLAM technique has been tested in a small environment, and then on the large map of the Engineering Faculty building, showing that a team of robots can build a map in a shorter time than a single robot. The performance of the multi-robot SLAM and of the single-robot SLAM has been compared through a serie of simulations in a test environment, showing that more robots can collaborate to solve the loop closure problem, exploring smaller areas of the same environment. Also, the computational burden is reduced, since every robot must maintain a smaller submap.

An aspect that remains to be experimentally tested is the capability of a team of mobile robots to perform SLAM of the same an environment with greater precision than a single robot, given the same exploration time and path travelled. In the future, more experiments, aimed to get a quantitative evidence of this improvement, will be carried out.

Besides the encouraging results obtained, there are still some practical issues to deal with, in order to develop successful real world applications. The data association algorithm used in the simulator has a computational complexity of  $O(n \cdot m)$ , where  $n$  is the number of the measurements and  $m$  is the number of the already known features. This computational complexity could be reduced by using different multi-dimensional search techniques. The search for duplicate matching features between the two maps would benefit from the same complexity improvement.

Furthermore, another aspect of the M-Space feature representation could be implemented. When two walls are recognized to share a common corner,

the two corresponding line features in the map share an endpoint. The shared endpoint coordinates cause a change in the M-space, in order to handle the update of these common parameters.

Finally, towards the goal of realizing a team of mobile robots that perform multi-robot SLAM in vast indoor environments, the proposed algorithm could be implemented on real robots.

# Bibliography

- [1] T. Arai, E. Pagello, L. E. Parker (2002). Editorial: advances in multi-robot systems. In *IEEE Transactions on Robotics and Automation*, **18**(5), 655-661.
- [2] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-whyte and M. Csorba (2001). A solution to the simultaneous localization and map building (SLAM) problem. In *IEEE Transactions Robotics and Automation*, **17**(3), 229-241.
- [3] J. Borenstein, B. Everett and L. Feng (1996). *Navigating mobile robots: systems and techniques*. Wellesley: AK Peters.
- [4] D. Caltabiano (2006). *Innovative methods in autonomous field robotics*, Ph.D. Thesis.
- [5] F. Dellaert, D. Fox, W. Burgard, S. Thrun (1999). Monte Carlo localization for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '99)*.
- [6] A. Elfes (1987). Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation* **3**(3), 249-265.
- [7] J. W. Fenwick, P. M. Newman, J. J. Leonard (2002). Cooperative concurrent mapping and localization. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA '02)*, 1810-1817.

- 
- [8] J. Folkesson, P. Jensfelt, H. I. Christensen (2005). Vision SLAM in the measurement subspace. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA '05)*, 30-35.
- [9] J. Folkesson, P. Jensfelt, H. I. Christensen (2007). The M-Space feature representation for SLAM. *IEEE Transactions on Robotics*, **23**(5), 1024-1035.
- [10] D. Fox (1999). Markov localization for mobile robots in dynamic environments. In *Journal of Artificial Intelligence Research*, **11**, 391-427.
- [11] A. Garulli and A. Vicino. Set membership localization of mobile robots via angle measurements (2001). *IEEE Transactions on Robotics and Automation* **17**(4), 450-463.
- [12] A. Garulli, A. Giannitrapani, A. Rossi, A. Vicino (2005). Mobile robot SLAM for line-based environment representation. In *Proc. of the 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05)*, 2041-2046.
- [13] A. Giannitrapani, *A set-theoretic framework for simultaneous localization and map building*. Ph. D. Thesis, 2003.
- [14] A. Howard (2004). Multi-robot mapping using manifold representations. In *IEEE International Conference on Robotics and Automation*, to appear.
- [15] A. Howard (2005). Multi-robot simultaneous localization and mapping using particle filters. In *Proceedings of Robotics: Science and Systems (RSS'05)*. In *The International Journal of Robotics Research*, (2006) **25**(12), 1243-1256.

- 
- [16] P. Jensfelt, S. Kristensen (1999). Active global localization for a mobile robot using multiple hypothesis tracking. In *Proc. of the IJCAI-99 Workshop on Reasoning with Uncertainty in Robot Navigation*.
- [17] J. J. Leonard and H. F. Durrant-Whyte (1991). Simultaneous map building and localization for an autonomous mobile robot. In *Proc. IEEE Int. Workshop on Intelligent Robots and Systems*, 1442-1447.
- [18] R. Martinez-Cantin, N. de Freitas and J. A. Castellanos (2007). Multi-robot marginal-SLAM. *International Joint Conference on Artificial Intelligence (IJCAI), Workshop on Multirobotic Systems for Societal Applications*.
- [19] H. P. Moravec and A. Elfes (1985). High resolution maps from wide angle sonar. In *IEEE Int. Conf. Robotics and Automation*.
- [20] P. Newman, J. Leonard, J. D. Tardós, J. Neira (2002). Explore and return: experimental validation of real-time concurrent mapping and localization. In *Proc. of the IEEE International Conference on Robotics and Automation*, 1802-1809
- [21] L. E. Parker. *Current state of the art in distributed autonomous mobile robotics*. Distributed Autonomous Robotic Systems, Springer 2000.
- [22] J. D. Tardós (1992). Representing partial and uncertain sensorial information using the theory of symmetries. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA '92)*, 1799-1804.
- [23] S. Thrun, D. Fox, W. Burgard and F. Dellaert (2000). Robust Monte Carlo localization for mobile robots. In *Artificial Intelligence*, **128**(1-2), 99-141.
- [24] S. Thrun. *Robotic mapping: a survey*. Exploring Artificial Intelligence in the New Millenium, Morgan Kaufmann, 2002.

- [25] S. Thrun, D. Koller, Z. Ghahmarani, H. Durrant-Whyte, and A.Y. Ng (2002). *Simultaneous mapping and localization with sparse extended information filters*. In *Proc. of the 5th International Workshop on Algorithmic Foundations of Robotics*. In *The International Journal of Robotics Research*, (2004) **23**(7-8) 693-716.
- [26] S. Thrun, Y. Liu (2003). *Multi-robot SLAM with sparse extended information filters*. In *Proc. of 11th International Symposium of Robotics Research (ISRR'03)*. In *Robotics Research Journal* (2005), 254-266.
- [27] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.
- [28] M. M. Veloso, D. Nardi (2006). Special issue on multirobot systems. In *Proceedings of the IEEE*, **94**(7), 1253 - 1253.
- [29] S. B. Williams, H. Durrant-Whyte (2002). Towards multi-vehicle simultaneous localisation and mapping, In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 27-430.
- [30] X. S. Zhou and S. I. Roumeliotis (2006). Multi-robot SLAM with unknown initial correspondence: the robot rendezvous case. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, **9**(15), 1785-1792.
- [31] X. S. Zhou and S. I. Roumeliotis (2006). Multi-robot SLAM map alignment with rendezvous. In *International Journal of Robotics Research*, (2006) **25**(12), 1243 - 1256.

journal = Int. J. Rob. Res., volume = 25, number = 12,  
year = 2006, issn = 0278-3649, pages = 1243-1256, doi =  
<http://dx.doi.org/10.1177/0278364906072250>, publisher = Sage Publica-  
tions, Inc., address = Thousand Oaks, CA, USA,