ACADÉMIE DE MONTPELLIER

UNIVERSITÉ MONTPELLIER II

- SCIENCES ET TECHNIQUES DU LANGUEDOC –

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ MONTPELLIER II

Discipline : Génie Informatique, Automatique et Traitement du Signal
Formation Doctorale : Systèmes Automatiques et Microélectroniques
École Doctorale : Information, Structures et Systèmes

présentée et soutenue publiquement

par

Alfredo TORIZ PALACIOS

Titre :

Exploration intégrée probabiliste pour robots mobiles évoluant en environnements complexes

JURY :

Mr. Michel DEVY	Directeur de recherche CNRS au LAAS	Rapporteur
Mr. Luc JAULIN	Professeur, ENSIETA Brest	Rapporteur
Mr. René ZAPATA	Professeur, Université Montpellier II	Directeur de Thèse
Mr. Abraham SANCHEZ	Professeur, H. Université Autonome de Puebla	Examinateur
Mr. Philippe FRAISSE	Professeur, Université Montpellier II	Examinateur
Mr. Lionel LAPIERRE	Maître de conférences, Université Montpellier II	Examinateur

Contents

List of figures

Chapter 1. Introduction

One of the fundamental challenges of today's robotics is to obtain robust and efficient mechanism for modeling increasingly complex environments, using mobile robots for their exploration. This is known as the Simultaneous Localization and Mapping (SLAM) problem which consists of using the map that the robot is currently building to determine its own position. This problem can be technical challenge because the robot position and the world features must be estimated simultaneously from noisy sensor data.

Probabilistic solutions are the most popular for the SLAM problem, especially those based on the use of an Extended Kalman Filter to estimate the map. Despite all the research done in this field that has resulted in substantial progress in autonomous map-building, still significant barriers arise in the implementation of these algorithms. The first comes because normally SLAM algorithms implicitly assume a naive control where the robot is driven around the environment by hand while it records the sensor data resulting in a system that is not really autonomous and even the quality of the maps in some cases is poor when the sensor data is collected from a robot being controlled manually by a novice.

Efficient exploration of unknown environments is a fundamental problem in mobile robotics which answer to the question of where to go next in order to build the map efficiently. Even though the exploration strategy can have a real impact on the quality of the resulting map, the area of exploration for SLAM or Integrated exploration is relatively new. This paradigm was first explicitly stated in [**Feder et al. 1999**] and can be seen as the problem where a mobile robot incrementally builds a map of this

environment and simultaneously uses this map to compute its absolute localization, and make local decisions on where to move next in order to minimize the error in the estimation of the mobile pose and the configuration locations. Planning actions for SLAM requires fast algorithms that can adapt to changes in the environment when new features or obstacles are detected.

A second issue is regarding to the representation of objects in the map and to the information that such representation can provide in the SLAM process. B-Spline curves have been used recently in [**Pedraza et al. 2007**] as a form of representation of environments. In this work, these curves have shown great efficiency and versatility to describe complex environments where it is not possible to extract simple features such as lines and points. However, the use of this type of representation is too recent and therefore not fully exploited, especially talking of the kind of information that the B-Splines can provide to the SLAM process.

Finally, limitations on the size of the environments due to calculations that must be performed and the inconsistencies that the linearizations of the problem produce in some methods make it necessary to find alternative ways of addressing the problem of SLAM.

1.1 Motivation

Robotics is the science that pursues the perception and manipulation of the physical world around us through programmable and controllable devices with computers. These devices are known as robots and depending on the objective can be found in a wide variety of forms.

Perhaps the first real example of mobile robot was the turtle developed by Walter in 1948 which was capable of moving exhibiting an apparently intelligent behavior at the moment of reacting against the presence of obstacles. Since then, the attempts of creating mobile robots provided with autonomy have been growing more and more to the point that today there are numerous commercial applications that assist humans in many tasks.

A mobile robot, to be considered truly autonomous should be able to answer the following three questions that define the basic problem of its own navigation [**Leonard et al. 1992**]. Where am I?, Where am I going? How i get that destination?.

Many efforts have been made in the scientific community to answer these questions. The first question concerns the localization of the robot and to get an answer, the robot must use the information obtained through its sensors and the information available over the environment. The localization problem is a key problem in mobile robotics. Occasionally, this has been mentioned as “The most fundamental problem to provide to a mobile robot of autonomous capacities”. The answer to this question will be used as a starting point to obtain the solution of the other two questions because the current position will give a great number of possible target positions while the way to reach them will come in part conditioned by the starting position of the robot.

In order to answer the third question (how to reach that destination?) many path planning algorithms have been proposed adapted to the own characteristics of the vehicles, the perception system and the type of task to be performed. Although all these factors have a commitment to develop a navigation system, the available information over the environment and how the robot is able to perceive and to reason it are without doubt the determining elements

The second question, however, is a problem that remains open or whose solution is imposed by human interaction that defines the destination and the objective of the task to be performed by the robot. The autonomous choice of a destination has been typically left in the hands of algorithms designed for the exploration of environments. Although in recent years these kinds of algorithms have made a great progress, it is a problem that still remains open.

Given this, it is clear that any movement system needs to have some kind of model of its environment to solve the basic problem of navigation; either to determine their own position, to define a target position, or to plan a path to follow. This objective has been pursued in recent decades where the first solutions tried to decouple the problem of map building and the problem of robot localization; however, rapidly was discovered that this would not be possible without simultaneously considering both aspects. This phenomenon is because during the exploration, the robot performs measurements of the environment that are later placed spatially considering its own position, at the same time, previously detected objects are used to determine its location.

In [**Smith *et al.* 1987**] Smith et al provided the basis to solve the two problems simultaneously and that in these days is known as SLAM (Simultaneous Localization and Mapping). The SLAM problem responds to the autonomous capabilities of a

robot, it is possible for an autonomous vehicle to start at an unknown position, in an unknown environment and then start building a map that will be simultaneously used to calculate the absolute position of the vehicle to allow the robot to navigate.

Although many approaches have tried to deal with this problem, the huge dimensionality (temporal, spatial and statistical) of the SLAM problem makes that, mathematically, it does not exist a complete solution. Instead suboptimal solutions have been presented in the current literature being probabilistic techniques based on linear approximations of first order as the extended Kalman filter which has best deal with the problem; however, these solutions suffer from some statistic inconsistency that causes corrupt results or completely incorrect results in the long term. Because of that the solution to this problem is still in study.

The representation of the physical world surrounding the robot in itself, represents also a problem. Here the available methods attempt to describe and model the environment as it is presented and try to get from it the information necessary for the representation. While that some approaches try to extract geometric features and to represent their positions on a map, others try to discretize the space into cells and classified each one as occupied or empty. One last category known as scan-matching has been presented in [**Lu et al. 1997**]. Here, the use of laser sensors to take accurate measurements of the environment is essential. The methods in this last category are able to represent the environment without relying on any assumption about the geometric characteristics. What these methods seek, is the way to align together two consecutive measurements so that the discrepancy obtained between them serves to correct the position of the robot.

Although the representation methods mentioned have been widely developed, these are adapted only to simple primitives such as points or line segments and therefore in more complex environments they lose their validity. For this reason, a new form of representing complex environments has been presented by Pedraza et al. in [**Pedraza et al. 2007**] where B-Splines curves are taken as basis for the representation. With this, the new challenge that now faces the problem of SLAM is to consider new representations to model non-standard environments.

As we have mentioned earlier in this section, navigation gives a real autonomy to the mobile robot. However, the classical methods of SLAM require human intervention to take the decision on where the robot has to go in order to continue the mapping. These navigation methods are known as methods of exploration and correspond to another area of research related to the prediction of the unexplored region. Thus,

exploration is the task of guiding a vehicle in an unknown environment where the mobile robot has to decide the next exploration target that offers the most important benefit of an unexplored region. Considering this, in recent years a new paradigm has emerged which considers the motion planning in the context of SLAM. This new paradigm is known as SPLAM (Simultaneous planning, localization and mapping) or Integrated Exploration and it requires a balanced evaluation of the obtained information, the quality of the localization and the cost of navigation.

Since the problem of integrated exploration is a relatively new area, a lot of work remains to be done.

1.2 OBJECTIVES

Based on these motivations and needs, this section describes the general objective and the particular objectives of this thesis.

The main objective of this thesis work is to develop an effective and robust SPLAM tool destined to the construction of maps of complex environments in an autonomous way. As we have mentioned, the strategies of SPLAM solve simultaneously the planning, localization and mapping problems. It is then necessary to develop several strategies that coexist harmoniously to achieve in a joint way the proposed objective.

With this in mind, we will try to substantially improve some methods from the current boundaries of the theory and technique in the field of exploration, localization and mapping. Thus, from the general objective arise the following specific objectives:

1. To establish the theoretical basis for the understanding of the problem of Integrated Exploration, specifying how a solution can be formulated.

2. To propose an evolution of the SRT (Sensor-based Random Tree) exploration method, transforming the structure it uses into a more versatile one that could be used throughout the entire process of exploration. The proposed evolution transforms the exploration tree into an exploration graph that continues using the probabilistic efficiency that these kinds of methods have shown. Also, with the new structure it is possible to use more complex algorithms to determine the next best position to explore once the region where the robot is currently has been completely covered.

3. To study a new SLAM methodology based on unclamped B-Splines taking the representation of the environment presented by Pedraza et al. in [**Pedraza et al. 2007**],for which will be necessary to extend the area of application of existing techniques as follows:

    - The localization presented in our method is classified into the group of scan matching or topological methods. For this reason the data association in an efficient way is a key step. From the foregoing, an improvement is proposed for the current association data methods which exploit the information gain that represents the parametric description of arbitrary geometries. This way, once available a map of modeling in the form of parametric curves, it is possible to use the information contained in them as curves such as curvatures, length of the curve and curvature zero crossing with the objective of improving and strengthening the existing methodologies of data association.

    - Given the type of curves used in this project, there will be studied also the form in which the environment will be gradually extended with the new information that the robot will get of the unexplored zones

4. To study an alternative for the construction of map of great dimensions so that the computational cost of the algorithm allow its utilization in real time. For this end it is considered the structure used by the proposed exploration method so that the information necessary for the localization process will be the local information that is found on the node on which the robot is currently navigating. Simultaneously, the information with corrected position will be used to extend the global map.

1.3 Structure Of The Thesis

The thesis work presented consists of 6 chapters and a bibliography. In this first chapter, we have tried to transmit to the reader the motivations that have led us to tackle this topic showing the current challenges in the research field from which are originated the objectives pursued in our work.

Chapter 2 gives an overview of the current state of the art for the simultaneous planning, localization and mapping (SPLAM or integrated exploration) problem. Here, it is intended to familiarize the reader by introducing the main algorithms and

methods developed in the involved fields. In this way, we perform a study of the main SLAM strategies used until today showing the advantages and disadvantages of each one of them. Also we propose a study that shows the latest methods in the field of exploration environments and finally we present the work carried out in the relatively new field of integrated exploration.

Chapter 3 introduces an approach to the problem of integrated exploration using some familiar tools in the field of exploration and in the field of SLAM, showing in this last one the version based on the classical extended Kalman filter (based on punctual landmarks) and also a new version presented in [**Pedraza et al. 2007**] which also uses the EKF but with new representation of the environment based on B-Spline curves. The final objective of this chapter is to build a SPLAM strategy using known tools, which will serve as comparison for the approach developed in this project presented in Chapter 4.

Chapter 4 is the central chapter of this thesis. It presents a new approach to the SPLAM problem where the objects are modeled using parametric curve as is proposed in [**Pedraza et al. 2007**]. Because of this representation, we have improved or developed the following methodology:

- An evolution of the SRT exploration method, in which the main structure of exploration is transformed to a graph. Also, the choice of the next position to explore once the robot is in a fully explored zone is performed by using the introduced concept of border control and the graph search method A* in a bi-directional way.

- A topological location method that considers the new representation of the environment using the information about the curvature of the spline for the data association and position correction process.

- A method for incremental construction of maps of complex environments using unclamped splines as modeling tools.

At the same time, the proposed method uses only partial information contained in the nodes of the exploration graph for the process of SLAM. So the method can be used in environments of large dimensions.

Chapter 5 contains the results that show and support the practical application of the methods presented as well as the necessary comparisons to show the validity of our

proposal comparing it with the methodologies presented in Chapter 3. Results include simulations carried out in built environments as well as experiments in real environments.

Finally, Chapter 6 presents the main contributions of the thesis and the conclusions that can be extracted from them. In addition, an analysis of the proposal is performed considering possible improvements for future work and possible extensions of the project.

Chapter 2.State of the Art of Integrated Exploration

In general, the task of acquiring models of unknown environments requires the solution of three subtasks, which are mapping, localization, and motion control. Mapping is the problem of integrating the information gathered with the robot's sensors into a given representation. Localization is the problem of estimating the position of the robot. Finally, the motion control problem involves the question of how to steer a vehicle in order to efficiently guide it to a desired location or along a planned trajectory.

The diagram in Figure 2.1 depicts also the overlapping areas of these three tasks.

- **Simultaneous localization and mapping (SLAM).** It is a fundamental and complex problem in mobile robotics research. In this problem, a mobile robot explores and senses an unknown region; besides it constructs a map and localizes itself in the map.

- **Active localization**. It seeks to guide the robot to locations within the map to improve the pose estimate.

- **Classic exploration.** It does not take localization uncertainty into account and direct the exploration in order to minimize the distance travelled while maximizing the information gained. When the robots travel through unknown environments, the uncertainty over their position increases and the construction of the map becomes difficult. Consequently, the result can be a useless and inaccurate map.

- **Integrated Exploration.** Represented in the center area of the diagram; it address mapping, localization, and motion control simultaneously. The

paradigm of integrated exploration was proposed in [**Feder et al. 1999**], here, the exploration approach calls for a balanced evaluation of alternative motion actions from the point of view of information gain, localization quality and navigation cost.

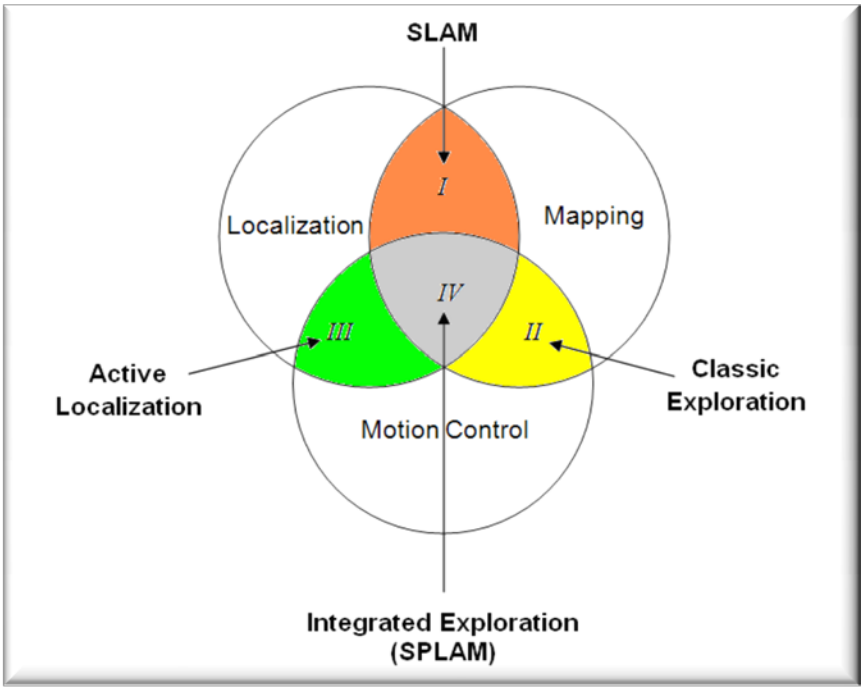

Figure 2.1 The field of robotic exploration with highlighted regions of integration: (I) SLAM, (II) classic exploration, (III) active localization, (IV) integrated exploration

2.1 SLAM

In the world of robotics, we can find a wide range of robots designed for an extremely wide range of applications and for an equally wide range of circumstances; it is unthinkable that a robot can be built without a preconceived purpose. Some of these robots that we can find today have been conceived to be capable to exploring, from environments with regular shapes such as office buildings [**Bosse et al. 2004**], to difficult terrain as in the case of planetary exploration [**Maimone et al. 2004**] and even in underwater environments [**Williams et al. 2004**]. These robots can operate alone [**Leonard et al. 1991**] or in teams even of hundreds of them [**Howard et al. 2006**].

One point that share most of the many implementations is that all the robots need a map. Most of these applications are location based, in one way or another. Tasks like terrain exploration, underwater inspection and many others need that the robots know their localization. Therefore a map is necessary to make it easy regardless of its representation or amount of information to be stored that vary from application to application. By providing the robot with a spatial context, a much more elaborate and intelligent behavior can developed. It means that maps allows to robot go beyond of a purely reactive behavior.

It can be thought that a map can be given in advance to the robot; however, this can be done only in a small circle of applications. Consequently, in most applications the robot has to be equipped with sensors that allow him to observe the environment and make a map by himself. This challenge is addressed by the mappings algorithms; while the robot travels through an environment the information read by the sensors is translated into a map. In this way, and assuming a perfect knowledge of the location of the robot, the challenge of mapping consists in make the most accurate description of the physical reality based on sensor readings even with the noise associated to the measurement system.

Taking this into account, the Simultaneous Localization and Mapping (SLAM) strategy, in a very simple definition, is about having a robot in response to question 'Where am I?'. Nevertheless, this question about the robot's position is almost never raised for its own benefit; on the contrary, the information over the map and the robot pose granted by SLAM is the key toward many intelligent behaviors. As an example, we can mention that this information is used by many navigation and path-planning algorithms as a prerequisite and it may contribute to a more elaborate behavior and motion planning. In general, the map and the robot pose information supply a natural context to relate observations, decisions and actions over time. So, only by the use of SLAM techniques we can successfully do an autonomously complete location-based task where the mobile robot can be placed on an unknown location into an unknown environment. Given that SLAM has not need of the a priori knowledge of the environment, Hugh Durrant-Whyte et al. [**Durrant-Whyte et al 2006**] asserts that the solution of the SLAM problem is the "holy grail" for the mobile robotics community where a robust method would make a robot truly autonomous.

Today, we can find on internet several SLAM algorithms for science research. Despite the great progress that has been made in the past decades, some existing SLAM methods are limited to specialized robot platforms, small environments and certain sensor technologies. For this and other issues, the SLAM problem still

remains unresolved. Given these matters, it is necessary to find robust SLAM solutions that can work for a large variety of robots without altering the model and also to build accurate large maps of environments. It has to be considered that the methods should run in real time and to work with the available memory, even for large maps.

It can be considered that the primary task of SLAM is to choose one representation which can facilitate the subsequent algorithms; this is due to the intelligent behaviors that depend on him. There are a great number of these representations, each with their capabilities and limitations in terms of accuracy, performance and memory allocation. So, the election of one representation is a delicate and most be done considering a broad context that at least includes the algorithms that will actually make use of the map.

2.1.1 Localization

To perform the exploration of a given environment, the robot most travel through it. This means that some motions commands have to be sent to the robot's motions actuators to get some movement from him and this gives rise to its own particular kind of problem often known as odometric errors. The odometric errors are a mismatch between the desired movement specified by the control commands and the movement achieved by the actuators. This difference may have its root in any of the following problems: inaccuracy in the actuator, slippery or uneven surfaces, or some other problems caused by the environment itself.

Given the differences between the desired position and the position reached, many researchers have focused their attention on solving the autonomous localization problem in the past two decades. The result is a great variety of paradigms that seek to determine the position and orientation of a robot with respect to the objects in the environment. Once the robot's pose is obtained by localization, this information is used as a reference frame for mapping to interpret and localize the sensor observations and build a map from these. At the same time, localization estimates the current pose of the robot comparing the current observations with the information contained in the map. Thus, localization and mapping are interdependent.

2.1.1.1 Types of maps

This section considers the four types of maps most commonly used in current localization systems: metric maps, feature maps, topological maps and hybrid Maps.

a) Metric Maps

The level of metric representation of the map contains a model in which the coordinates and properties of objects are represented numerically. So this model can be geometric or discretized (Figure 2.2). In the geometric model, the discrete elements that represent the environment are stored using their geometric parameterization. And in the discretized model the occupation of space is represented by a division of it.

In the geometric model, the maps contain the positions and properties of a group of objects in the environment with certain geometrics characteristics: The robot, the walls, natural or artificial specific markers, etc. Depending in the sensorial capacity and feature extraction of the environment, more complex geometric objects could be distinguish and used.

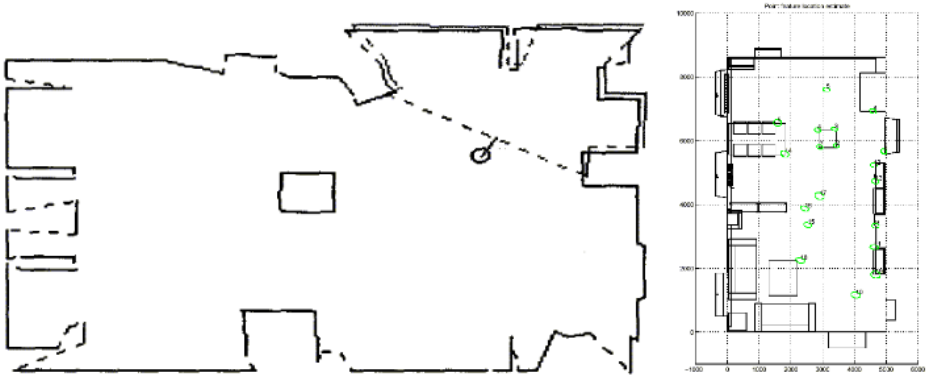

Figure 2.2 Geometric metric Maps. The left image [Zhang et al. 2000] show a map composed by line segments that represents the walls of the environment in the map obtained. On the right image [Zunino et al. 2001], the objects are represented as dots extracted from thin object and corners of the environment.

This representation is often used on structured environments where can be possible an extraction robust enough of the geometric features. Popularity lies in the representation anthropomorphic of the space, which makes it especially useful for displaying maps and to compare them with man-made maps. The compactness of its representation directly influences on the storage size map. However, these algorithms require more accurate sensors like laser scanners or more complete like vision systems.

Between the disadvantages of the geometric model, we find the inability to make a complete model of the environment. This means than only the geometric features will be stored and the rest of information will be discarded and won't be considered in the map. This is, in order to get compactness and robustness, most of the sensorial information is discarded. For this reason, these kinds of maps are often useless for common tasks of navigation of robot mobiles like path planning because the planner would not consider all the possible obstacles that physically exist but that mapping was not able to assimilate as geometric features. In exploration tasks, the situation is not better. This because they don't maintain the notion of which part of the environment has been already explored and which one not.

In the discretized model, individual objects are not extracted from the sensorial system, but the information is treated without a segmentation process to construct a probability density function of the space occupancy. Like this function of density is impossible to maintain analytically, the space is divided into cells and it considers the probability of every single cell is occupied or free. Every cell makes the description of a small rectangular area in the environment, and indicates the probability that the area is occupied by a value in the range (0, 1). The localization here is made by registering observation data with the map using cross-correlation methods.

This model is considered as a continuous representation of the space even if it is discretized, this because none analysis of the ownership of each cell to a single object is realized. Such maps are called occupancy grids (figure 2.3). These kinds of maps are perfect for the tasks of path planning and exploration due to the continuous and complete representation of the environment where none sensorial information has been discarded.

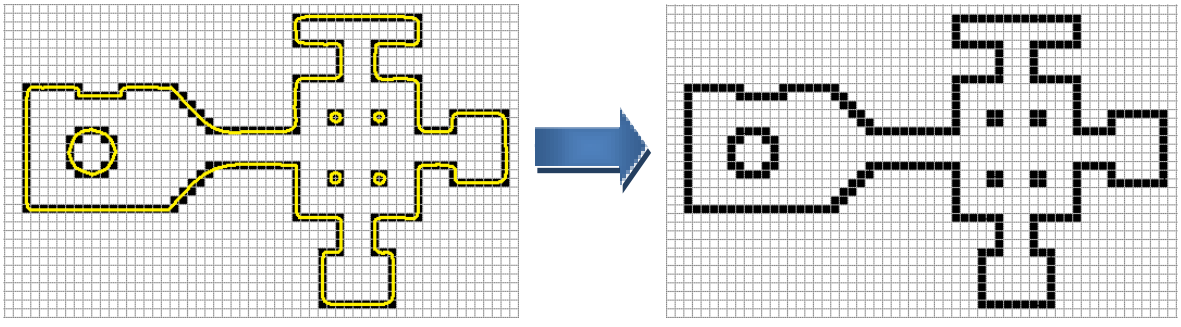

Figure 2.3 Occupancy grid map

The most important benefit of the occupancy grid over other representations is that this model can be used directly by a good number of navigation, obstacle-avoidance

and learning algorithms. But in the other hand, one difficulty concerning occupancy grids SLAM is data association. Within the region of the vehicle pose uncertainty, the cross- correlation search is expensive if the search-space is large. Also, a maximum likelihood correlation search may fail by converging to the wrong mode if the cross correlation result is multi-modal within the search-space region. However this problem can be solved using the Monte Carlo localization procedure as in [**Dellaert et al. 1999**].

For the Occupancy grid SLAM applications [**Yamauchi et al. 1998**], we observe that steps of localization and map update are interlaced by locating the map segment using the global map in the first place and then extending the map by updating the perceived occupancy of the global map grid cells. This method has shown robust results in dynamic indoor environments over a limited period of time. The reason why these maps only work on a limited period of time is because they don't have an appropriate uncertainty model and so will tend to diverge in the long term. The uncertainty represented in occupancy grid is only at a local level (vehicle-centric), which is not enough for SLAM where an integrated representation of sensor and vehicle pose uncertainty and their correlations are essential for map convergence. That is not supported by the occupancy grid framework. By not defining criteria for convergence, the developing map is able to drift with each observation update and this divergence exhibits itself as a slow blurring of the map.

The greatest drawbacks of using occupancy grids are:

- They are not really well-suited for online SLAM, especially in large-scale environments; this because their space and time complexities grow exponentially with the grid resolution.

- Other issue is the map update; even when the maps are easily constructed, the problem lies on the fact that there is no trivial way to undo or alter past grid modifications; this is because of the inherent data aggregation in occupancy grid cells. For example, when the SLAM algorithm re-observes a feature and a large accumulated error is exposed, SLAM would want to correct this error and update the map accordingly, this implies that the current ray-casting should be undone and then redone based on the updated pose estimate. In practice the backwards editing of occupancy grids is considered not desirable during online usage. Therefore, pose estimates have to remain fixed once determined. This has led to the situation where approaches employing occupancy grids are often equipped with means to

accommodate for this inflexibility, like particle filters where multiple possible pose sequences are maintained concurrently [**Schultz et al. 1999**].

Occupancy grids are also used after the main SLAM process as a post-processing step. Using them in this way, mapping in not really in SLAM, but they are used for the visual reporting of the maps or to make easy subsequent algorithms. In the first case, some representation of the environment used by the SLAM algorithm do not easily allow for visual rendering on screen or print. In the second case, we can find that there are SLAM approaches that use representations of the maps that won't be suitable for subsequent use by other methods. Such is the case of learning or obstacle avoidance algorithms.

b) Feature Maps

Feature maps represent specific objects of the environment by the global locations of parametric features (such as points, lines and lately curves) as shown in Figure 2.4. In this type of maps, localization is performed by extracting features from the information obtained by sensors and associating them to features already existing in the map. So, the vehicle pose is calculated using the differences between the predicted feature locations and the measured locations. In this way, localization targets are static and the observer is in motion.

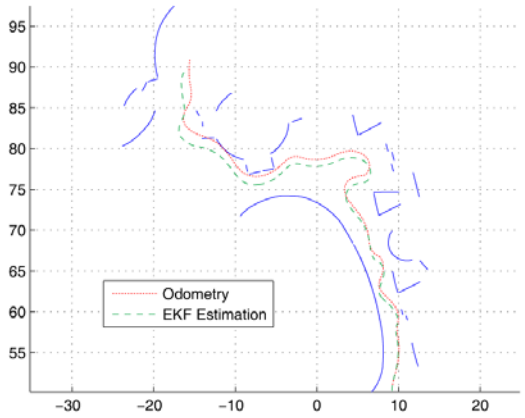

Figure 2.4 Feature map. The environment is defined by parameterized features (point locations in this example). These static landmarks are tracked using target tracking methods to determine the motion of the observer. Image taken from [Pedraza et al. 2007]

Feature map SLAM [**Smith et al. 1987**] comprises the dual task of adding observed features to the map, using the vehicle pose as a reference, while using existing map features to estimate the vehicle pose. In this way, localization using a feature map is

a parameter estimation problem to determine the vehicle pose (x, y, φ) given the map feature information and a set of feature observations. Assuming the measurements are correctly associated to the appropriate map features, the vehicle pose can be tracked using standard estimation techniques (Been the EKF being the most common method applied to this problem). Here, the uncertainty of sensor measurements result is the uncertain estimates of both the vehicle pose and the map feature locations, and these uncertainties are dependent (or correlated). Correlated uncertainty has an important consequence for feature-based SLAM as it inextricably couples the individual features to each other and the vehicle to the map. Attempts to estimate the vehicle pose and map features independently have been shown to produce inconsistent uncertainty estimates [**Leonard et al. 1991**].

Data association is certainly the main weakness of feature map localization. To obtain a good localization we have to find a correct correspondence between a feature observation and its associated feature contained in the map. A wrong association results in an inconsistency where the vehicle location uncertainty decreases but the estimate error actually increases. Significant false associations increase the pose estimate error and consequently degradation in the accuracy of the map. These inconsistencies tend to be self-propagating, causing divergence, i.e., the sensitivity of the SLAM algorithms to incorrect data association.

Several feature map localization implementations are susceptible to data association failure because normally they rely on the association methods developed for target tracking, which treat each measurement in isolation. This approach is sensitive observer pose uncertainty and high feature density when the correlation between fixed marks is done in a wrong way. Robustness can be achieved using batch association: a group of observations is assigned as one and allows us to distinguish the association on the base of his probability of association combined utilizing the geometric character of the local region.

Other problem of data association is the administration of the non-associated observations. These unassociated features can be:

- New map features
- Outlier measurements
- Observations of dynamic objects

Identify the latter two is essential to avoid including items that should not be part of the map.

The most difficult data association complication comes when a cycle is detected. This is difficult because, not only is the vehicle location uncertain, but the new and old portions of the map are also uncertain in relation to each other.

In summary, feature maps are a viable representation for long-term convergent SLAM in fairly small-scale environments where the computational time necessary for the map update and the data association are very efficient. Computation is tractable, and accumulated state uncertainty does not exceed conservative limits.

c) Topological Maps

Unlike other representations, Topological maps [**Kuipers et al. 1991**] show an important conceptual change in the representation of the environment. In Occupancy grids and in feature maps the location is defined as a set of coordinates in Cartesian space where they rely on metric measurements, instead, Topological maps represent the environment in terms of places and connecting paths as shown in Figure 2.5. The two types of map organizations mainly used in topological SLAM are:

- **Graph maps.** This are designed to literally capture the navigability of the environment. Here, all the estimated poses along of the robot's trajectory are turned into nodes that define particular locations in the environment (termed distinctive places) and links define procedural information for traveling between nodes (i.e., links show the path that the robot has traversed between consecutive poses).

  The term topological consistency means that the graph does not indicate connectedness that is not actually in the environment and a consistent graph can be achieved by using a conservative method that only adds links every time that a robot traverses a path between two nodes [**Howard et al. 2006b**].

- **Voronoi diagrams.** In this case, the map is divided into a non-overlapping regions based on obstacle detection. In Voronoi diagrams, as in the graph maps, nodes and links are used to delineate the free space navigable. However they are positioned in a different way. The nodes are positioned at equidistant points, which are those points that are exactly at the same distance away from all near obstacles. For placing them, only obstacles for which there is not another node at a closer distance from the current one are considered. In this way, a link is a straight line that represents a safest path

between two connected nodes that is equally far away from the obstacles on both sides of him. Hence, like graphs, Voronoi diagrams lend themselves excellently for path planning purposes [**Lisien et al. 2005**].

As we can see the most important difference between these two representations is the form as links are treated. In graphs, links are added once the robot has traveled the path between two nodes, and in Voronoi diagrams links are estimated and their position is estimated according to the position of obstacles. Thus, graphs remain true to the actual traversed paths but Voronoi diagrams generalize beyond the actually traversed pathways as they infer the safest pathways based on the obstacle estimates. This characteristic makes this diagram be used especially in planar environments because the use in all three spatial dimensions can result in side-effects on the performance and memory consumption of SLAM.

These two types of map, as all topological representation seek to provide a compact description of the free space areas on the environment and their interconnectedness. Thus, navigation between two non-adjacent locations is determined by a sequence of transitions between intermediate place nodes. The concept works on the assumptions that distinctive places are locally distinguishable from the surrounding area and that the procedural information is sufficient to enable the robot to travel within recognizing distance of a specified place.

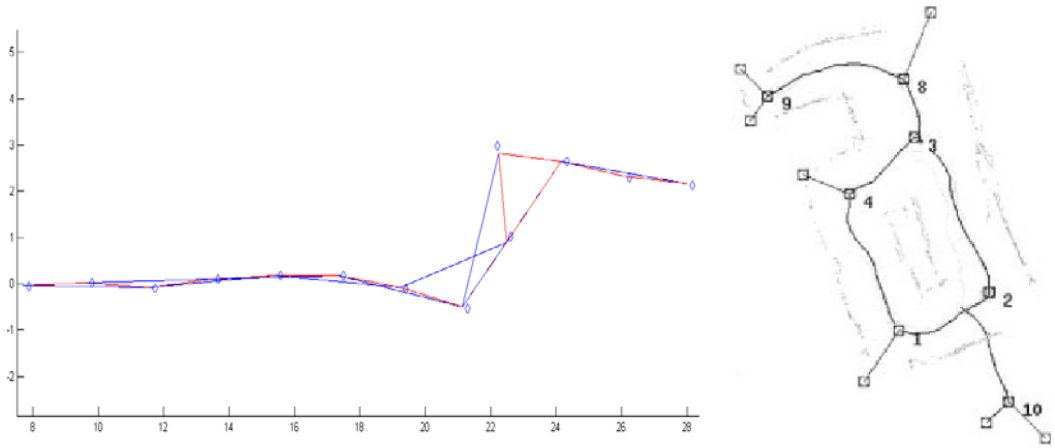

Figure 2.5 Topological Maps. At the Left side we have a Graph map. To the Right Side a Voronoi diagram [Choset et al. 2001]

The principal motivation behind the topological representations is their excellent support to path planning algorithms which is ideal for autonomous robots because they need to go from one place to another to complete their tasks. In the case of

exploration, we assume that at some point the robot will reach a dead-end of the environment and the exploration has to continue into another position. For this, the robot will have to return to a known position and start again from there. Here nodes and links in graphs and Voronoi diagrams tell any path planning algorithm precisely how to get around the explored areas safely. Given the nature of their construction, topological maps make this kind of tasks very easy.

Another motivation on topological maps is that, while feature-based and occupancy grid maps grow exponentially with the size of the environment or the number of detected features, topological maps can represent huge environments in a very compact way because they only grow linearly in size as nodes and links are added to denote newly explored areas.

In the other hand, the weakness over topological maps lies in ensuring reliable navigation between places, and subsequent place recognition, without the aid of some form of metric location measure. Regarding to navigation, for static structured environments, the use of purely qualitative trajectory information is enough to travel between nodes. However, for more complex and dynamic environments rely only on this information may result in failure to guide the robot to the right place. With respect to place recognition problems we can find two scenarios:

- The first is when a place is not recognized (false negative) due to the alteration in the appearance of the place by circumstances such as as viewpoint variation, occlusion, structural change, dynamic objects or changed lighting conditions. Geometric and visual recognition are sensitive to this type of failure.

- In the second, an alternate location is mistaken for a place (false positive) which is a symptom of inadequate place definition but also can be found in highly structured environments (such as rows of office cubicles) where another portion of the environment appears similar to the place definition (the place is not locally unique). Most of geometric recognition methods offer simple descriptions where ambiguous associations may be created even by the presence of transient objects. Vision-based recognition is probably more immune to false positives because of the increased level of information defining the node **[Devy et al. 1995, Sola et al. 2008].**

In all this cases the topological sequence is broken and the robot becomes lost.

Topological approaches to SLAM attempt to overcome the drawbacks of geometric methods. They operates by performing the exploration of the environment guided by a set of path following criteria, and recording place descriptions at appropriate locations. The maps for topological SLAM are constructed adding new places found and connecting them to the previous place according to the path following specifications required to reach it. This means that they are built as a linear sequence of places, which continues until a place is observed that matches a previously stored place description (assuming that the robot is not simply traversing old sections of the map).

For geometric place recognition, these would be locations affecting certain patterns in the sensory data and, for vision-based recognition, they could be either regularly-spaced locations or locations where a given distinctiveness metric is maximized.

As each new place is found, it is connected to the previous place according to the path following specifications required to reach it. In this way, the map is built as a linear sequence of places, which continues until a place is observed that matches a previously stored place description (presuming the robot is not simply traversing old sections of the map).

This matching place description can be found in new places that have a similar appearance to others content already on the map or by re-observing old locations when the robot takes alternate routes.

If the match can be identified unequivocally as the old place location, then a cycle is created, linking the topological sequence back upon itself to form a closed path.

There are certain issues that remain important over topological SLAM like qualitative path following and sequential data association, but the most important concern is cycle detection. Topological SLAM removes the difficulties of uncertainty in the representation and nonlinearities by avoiding metric location measurement, but, instead, data association process receives the great responsibility to work robustly.

For cycle detection, which is the key weakness in the topological map paradigm, data association can be vague when an observed place resemble to a previous place or may be a several places, and is not clear if it is one newly discovered location or one already stored. Using the methods of rehearsal [**Kuipers et al. 1991**] we can

distinguish correct associations. In this method the resulting sequence of places is traced until the number of candidate cycles is reduced to one or none (a new place). This method is appropriate if a place is unique in the world or a place sequence exists in the map. However, the reality is that in many environments such uniqueness cannot be ascertained. So, even if the observation of an expected place sequence serves to increase confidence of cycle detection, it is impossible to confirm this hypothesis.

The solution to this problem is the use of metric information that will limit the cycle search-space. So, that place sequences need only be locally unique and therefore will make possible the estimation of pose uncertainty between places.

d) Hybrid Maps

As illustrated in the previous sections, each elementary approach towards representing a map has several positive and negative sides. The tree levels may and should be treated to find one solution to the problem of modeling the environment. Based on this, it should come as no surprise that many researchers have attempted to acquire better solutions where the strengths of multiple elementary map representations are combined in hybrid data structures. Hybrid approaches are categorized into ones that integrate multiple representations in a single layer and ones that use a data structure with multiple layers where usually a topological layer at the top is used to decompose the lower layer into small-scale feature or grid maps.

Given that, their qualities are complementary. Topological and metric map representations are integrated in order to provide a versatile data structure that can serve the map information in multiple forms. Metric maps, with an appropriate uncertainty representation, constrain data association and permit non-qualitative trajectory planning, while topological maps are great for navigation and path-planning purposes and are capable of mapping large environments breaking the world up into locally connected regions and avoid the problems of maintaining a global reference frame.

Hybrid topological metric maps are basically topological frameworks where the place definitions and/or path definitions contain metric information. Importantly, this means that places are no longer restricted to discrete locations but can describe regions of arbitrary size and shape as local metric maps. ’Hierarchical Atlas’ presented by Lisien et al. [**Lisien et al. 2005**] is one example of hybrid approaches that use map decomposition. In this work, the global description of the environment is made using

a Voronoi diagram while every link in the diagram is stored in a local level in the so-called “edge maps”. The “edge maps” are occupancy grids that stores the geometric properties of the environment as they are found when a particular edge is been traversed. As all information on an edge map is stored with respect to a local coordinate frame, the equidistant points and hence the links can be kept dynamic without compromising the validity of the information stored on the edge-maps. This yields two advantages: first, by using grid-maps on a small scale the limitations due to their static nature are avoided and second, grid-maps are only constructed to cover the extent of the actually explored area instead of having them grow on a global scale whenever the boundaries are explored.

Finally, given all the characteristics of the maps used for the SLAM process, we present a recapitulative table showed in figure 2.6.

	Complex Objects Representation	Compactness	Complete model of the environment	Suited for path planning	Suited for exploration	Suited for SLAM	Main Weakness
Metric Geometric Maps	✓	✓	✗	✗	✗	✓	Inability to make a complete model of the environment
Metric Discretized Maps	✗	✗	✓	✓	✓	✗	Not suitable for online use
Feature Maps	✓	✓	✓	✓	✓	✓	Data association
Topological Maps	✗	✓	✓	✓	✓	✓	Cycle Detection
Hybrid Maps	✓	✓	✓	✓	✓	✓	

Figure 2.6 Recapitulative table of the properties of each type of map used in SLAM

2.1.2 SLAM Methodologies

2.1.2.1 Classic EKF-SLAM

Historically the earliest and perhaps the most influential SLAM algorithm is based on the extended Kalman filter (EKF), which is an extension of the Kalman Filter where nonlinear equations are linearized. One of the first solutions using this approach was proposed by Smith et al. in 1987 [**Smith et al. 1987**] where the map construction was considered as an extension of the localization task. In this work both the robot and

the map construction should be estimated simultaneously using a single state vector representing all the system variables in which we are interested. In addition, as the sensors used during the estimation provide noisy measurements, the evolution of the state vector in time is considered as a stochastic process where the final state estimate and its uncertainty are obtained using an EKF.

Based on this initial work numerous researchers have further developed Kalman SLAM. The result is that more sophisticated algorithms have appeared which deal with techniques for extracting and representing features, algorithms to fuse different types of sensors, methods to improve data association and techniques to reduce the computational cost and improve filter consistency. These improvements have made of the EKF one of the most popular SLAM approaches currently used.

Maps created by EKF-SLAM generally are limited to the current robot pose and landmark position estimates which must be represented as a Gaussian distribution. Therefore, the map is transformed into a state vector X that contains all the relevant variables and that constitute the means of this Gaussian distribution. For the own nature of the EKF SLAM the entities that compose the map must be described using a set of parameters that fit easily into the state vector of the system.

The EKF-SLAM can be considered as a three steps process:

1. **The Prediction Stage**. It deals with vehicle motion based on incremental dead reckoning estimates where the resultant distribution after a movement of the robot is calculated based on the previous robot state X_k and the control inputs u_{k-1}. However, each time the robot moves, the position uncertainty grows and the correlation with map features decreases.

2. **The Update Stage**. When a feature is re-observed an update stage must applied to reduce the uncertainty and improve the overall state estimate. As long as no landmark re-detections occur, the pose estimate will become less and less accurate due to the actuator uncertainty. As depicted in Figure 2.7, the accumulation of this error is reflected in the covariance matrix by increasing values for the robot pose, and through the pose-landmark correlations on all landmark location estimates. This continues until a landmark is re-detected.

   The re-detection of a landmark results in a significant drop in pose uncertainty, which then propagates towards increased certainty on all the landmark

locations. This jagged pattern of slow rises followed by sudden drops in map uncertainty is typical for Kalman SLAM.

3. **Add New Landmarks**. This is done by adding information about the relation between the new landmarks and the old landmarks. The information about the current robot position must be used to place the new information.

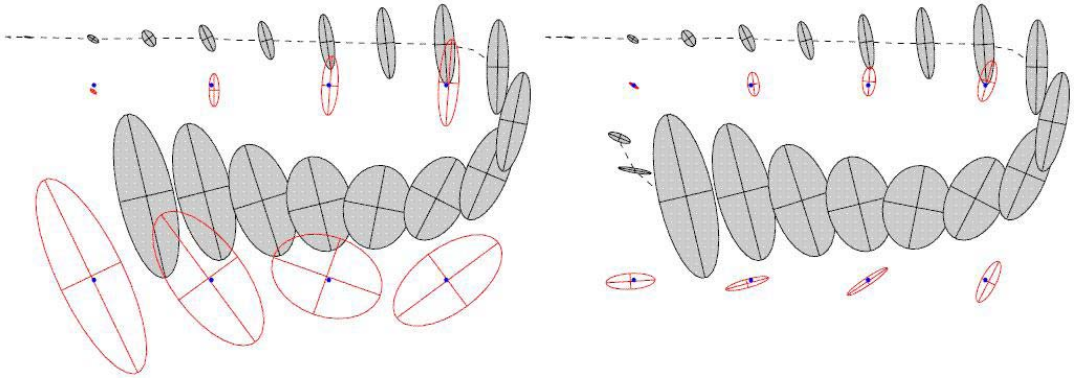

Figure 2.7 Rise and drop of map uncertainty with Kalman SLAM. Robot trajectory is shown in consecutive robotposition dotted line while the estimates are indicated with shaded ellipses and landmark position estimates with unshaded ellipses. The left picture show how the landmark uncertainty increases as robot pose uncertainty and to the right the robot pose drops after the re-observation of a previously stored landmark and through the information propagation this affects also all landmark estimates. Images take from [Thrun et al. 2005]

Initially, the algorithm starts with the robot's position and a covariance matrix that represents the pose uncertainty, i.e., none landmark has been added to the map. When a new landmark is detected the state vector and the covariance matrix are extended through an initialization process called state augmentation. The state-vector typically has a length of (3 + 2N), where the three first entries describe the robot pose in a 2D environment (x,y,θ), and only two entries for every landmark position (x, y), with N denoting the number of landmarks.

The standard Kalman algorithm is summarized in compact form in figure 2.8.

- **Prediction (Lines 1 to 2).** On line 1 the process equations describe the movement of the robot incorporating the control u_k. Line 2 denotes the associated uncertainty P_k^-. Note that this update only affects the robot pose estimate. The equations stated in line 1 and 2 will leave all landmark position

estimates untouched. This is because in most applications the landmarks are either fixed or at least assumed to be static within the robot's operating time.

- **Update (Lines 3 to 5).** On line 3, the Kalman gain K_k is computed, which incorporates the measurement uncertainty (measure that is associated with the current observation z_k). The Kalman gain matrix propagates changes in pose certainty throughout the map estimate, as the information gain is folded back into the robot's belief at lines 4 and 5. Note that the matrix H is just a convenience matrix that describes a mapping from the state-vector $\hat{X}_k$ to an observation z .

- **Add new landmarks (Lines 6 to 9).** On line 6, the state vector X is update with the new landmark. Finally, it is also necessary to add the covariance for the new landmark to the system covariance matrix P (lines 7 to 9).

A more extensive description will be proposed in section 2.

Algorithm of Extended Kalman Filter ($\hat{X}_{k-1}, P_{k-1}$, u_k, z_k)

1. $\hat{X}_k^- \leftarrow f(\hat{X}_{k-1}, u_k, 0)$
2. $P_k^- \leftarrow A_k P_{k-1} A_k^T + Q_k$ //Prediction
3. $K_k \leftarrow P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k \right)^{-1}$
4. $\hat{X}_k \leftarrow \hat{X}_k^- + K_k(z_k - h(\hat{X}_k^-, 0))$ // Correction
5. $P_k = \left(I - K_k H_k \right) P_k^-$
6. $\hat{X} \leftarrow \left[\hat{X} + x_N y_N \right]^T$
7. $P^{N+1N+1} \leftarrow J_x P_k J_x^T + J_z R J_z^T$ //Add new landmark
8. $P^{rN+1} \leftarrow P^{rr} J_x^T$
9. $P^{N+1r} \leftarrow \left(P^{rN+1} \right)^T$

Figure 2.8 EKF algorithm. Here, $\hat{X}$ represents the system states, P is the system covariance matrix, A is the Jacobian of the prediction model, u is the control entry, z are the observations, Q is the associated uncertainty, R is the observation covariance, H is the Jacobian of the measurement model h, K is the Kalman gain, J_x is basically the Jacobian A but without the rotation term and J_z is also the Jacobian of the prediction model but with respect to measurement model.

Unlike other SLAM approaches, the main characteristic that distinguishes EKF is that it estimates the full joint posterior over the map online. This means that the full uncertainty is maintained at all times during the online construction of the map. Moreover Kalman has proved to converge through the full posteriori; however this convergence is only assured as long as landmarks are observed infinitely often. So, when few landmarks are involved in the localization process it becomes harder and less accurate and therefore data association in turn becomes harder since larger pose uncertainties have to be overcome. A degrading data association performance negatively influences map accuracy again, and so this vicious circle of degradation continues. In the other hand, maps are often limited in the number of landmarks they will maintain. This because the update step involves the whole covariance matrix and when too many landmarks need to be maintained, the online SLAM algorithm become slow.

Another characteristic of Kalman filters is that it assume that all noises in the system are governed by Gaussian distributions; which in the context of SLAM is translated to assume Gaussian sensor noise, actuator noise and data association error. Even if sensor noise is often well approximated by a zero-mean Gaussian, the same assumption for odometry and data association uncertainty imposes a severe limitation of Kalman SLAM. This because odometry is typically governed by trigonometric functions and for data association uncertainty when a particular landmark is mistaken for another close, the resultant distribution of potential robot poses is obviously not best approximated by the bell-shaped Gaussian distribution.

Despite its relative success, the classical SLAM solution based on the EKF algorithm suffers from the next limitations:

1. Gaussian distribution assumed for the state of the system may not correspond to reality. In the best of cases linearizations made by the EKF will make estimates of the moments of this distribution degenerate over time producing optimistic values for the map covariance matrix, which may result in inconsistency [**Julier et al. 2001**].

2. It requires updating the full map covariance matrix after each measurement, giving a memory complexity of $O(n^2)$ and a time complexity of $O(n^2)$ per step, where n is the total number of features stored in the map [**Paz et al. 2007**]. So, the computational cost grows to the square with the number of objects contained in the map. This fact limits its application in real time.

Both problems become critical in large scenarios (scaling problem) since linearization errors get worse because the uncertainty in the robot position and surrounding features far from the map origin is larger and the memory and calculation requirements increase because the number of features grows.

2.1.2.2 Particle Methods

Particle filters are mathematical models that represent probability distribution as a set of discrete particles which occupy the state space. These particles can be thought of as a finite set of samples that has been obtained from the posterior distribution they represent. This way, valuable statistics of the original distribution can be easily estimated through the samples. The main objective of particle filtering is to track a variable of interest, typically with non-Gaussian and potentially multi-modal probability distribution functions. Multiple copies (particles) of the variable of interest are used, each one associated with a weight that signifies the quality of that specific particle.

The main drawback of particle filters is that they scale exponentially with the dimension of the underlaying state space. This represents a problem in the context of SLAM since the space of map features and robot paths is usually huge.

Given this, Montemerlo et al. in [**Montemerlo et al. 2002**] introduced the concept of FastSLAM to make particle filters amenable to the SLAM problem. The FastSLAM algorithm is a solution to stochastic SLAM that involves three important concepts: Rao-Blackwellization, conditional independence and resampling.

- **The Rao-Blackwellisation** [**Doucet et al. 2000**] approach consists in partitioning the joint distribution of the state-space into a sampled part whereby the robot pose states are represented by particles and into an analytical part where the landmark states are estimated analytically by Kalman filters. The state partitioning is defined as follows.

$$\begin{aligned} & p\left(X_{v0:k}, m \mid Z_{0:k},\ U_{0:k}\right) \\ &= p\left(X_{v0:k} \mid Z_{0:k},\ U_{0:k}\right)\ p\left(m \mid X_{v0:k}\ ,\ Z_{0:k},\ U_{0:k}\right) \\ &= p\left(X_{v0:k} \mid Z_{0:k}, U_{0:k}\right)\ p\left(m \mid X_{v0:k},\ Z_{0:k}\right) \end{aligned} \qquad (2.1)$$

Where $X_{v0:k}$ is vehicle pose history, m the set of all landmarks, $Z_{0:k}$ is a sequence of measurements and $U_{0:k}$ is a sequence of control inputs. Here the joint posterior is factored into a vehicle pose part and a map part conditioned on the pose.

- **The conditional independence** property basically says that features in SLAM get correlated due to the uncertainty in the robot position. If the trajectory of the robot were perfectly known we could estimate each feature independently (figure 2.9). The practical importance of the conditional independency between features is that, given a particle of the robot trajectory, we can estimate each of the features independently what makes the cost of the algorithm linear in the number of particles.

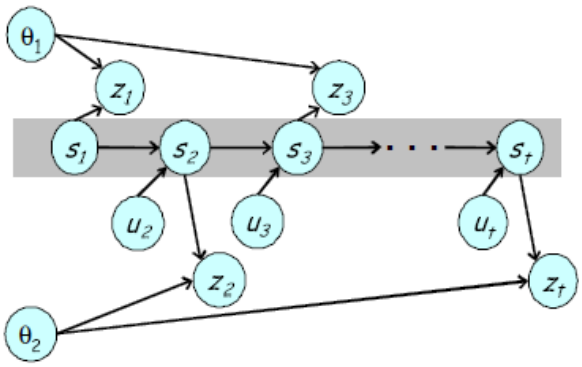

Figure 2.9 Probabilistic dependencies between SLAM variables in a Bayesian Network. Given the trajectory S_k of the robot (obtained through a sequence of controls U_k), map features θ_k are Conditionally Independent. The measurements are denoted as Z_k (range and bearing). Image taken from [Montemerlo et al. 2002].

- **The resampling step** allows the robot to concentrate particles in high probability regions of the distribution. It is useful when working with distributions that evolve in time (dynamic states).

As a Bayesian Filter, the FastSLAM algorithm consists of the steps of prediction and update like the explained in the Classic EKF-SLAM. Each particle in the FastSLAM is of the form:

$$X_k^{[m]} = (x_k^{[m]}, \mu_{1,k}^{[m]}, P_{1,k}^{[m]}, \ldots, \mu_{N,k}^{[m]}, P_{N,k}^{[m]}) \qquad (2.2)$$

Where $[m]$ indicates the index of the particle, $\mathrm{x}_k^{[m]}$ is its path estimate, and $\mu_{N,\mathrm{k}}^{[m]}$ and $\mathrm{P}_{N,\mathrm{k}}^{[m]}$ are the mean and covariance of the Gaussian, representing the Nth landmark location that is attached to the m^{th} particle.

Prediction Step

In this step, the probability distribution of the next position of the robot is obtained from previous distribution at time *k-1* and new poses sampled using the most recent motion command u_k as:

$$x_k^{[m]} = p\left(x_k \mid x_{k-1}^{[m]}, u_k\right) \qquad (2.3)$$

As we can see, this proposal distribution ignores completely the measurements z_k and the pose $x_k^{[m]}$ is predicted only using the motion control u_k. In this case, when the observation is more accurate in relation to the vehicle's motion noise, this becomes problematic. To rise above this problem, an improved version called FastSLAM2.0 has appeared. Here the poses are sampled under consideration of the motion u_k and the measurement z_k. Formally, this is denoted by the following sampling distribution, which now takes the measurement into consideration:

$$x_k^{[m]} = p\left(x_k \mid x_{k-1}^{[m]}, u_k, z_k, n_k\right) \qquad (2.4)$$

Where the variables $n_k = n_1, \ldots, n_k$ are data association variables, in which each n_k specifies the identity of the landmark observed at time *k*.

Update Step

When a new measurement z_k of a feature n is obtained, each particle of the trajectory is weighted according to the agreement between the expected measurement and the actual observation as:

$$w_k^{[m]} = N(z_k \mid x_k^{[m]}, (\mu_{N,k}^{[m]}, P_{N,k}^{[m]})) \qquad (2.5)$$

where factor $w_k^{[m]}$ is called the ***importance weight*** of particle *m*.

$$w_k^{[m]} = \frac{target\ distribution}{proposal\ distribution}$$

$$= \frac{p\left(x_k^{[m]} \mid z_k, u_k, n_k\right)}{p\left(x_{k-1}^{[m]} \mid z_{k-1}, u_{k-1}, n_{k-1}\right) p\left(x_k^{[m]} \mid z_k, u_k, n_k\right)} \quad (2.6)$$

After this operation, the importance weight of all particles is normalized to sum up 1.Once the weights have been obtained, the particles are resampled according to them, what adjust the particle population in response of the new information provided by the measurement. Finally, features of the survival particles are updated following the standard EKF update equations. For a complete derivation of the importance weight in FastSLAM2.0, see [**Kim et al. 2008**] and [**Montemerlo et al. 2003**].

More advanced and complex algorithm based in particle filters can be found in literature. For example in [**Nieto et al. 2003**], Nieto et al. made an extension of the basic FastSLAM to unknown landmark associations, which is important for generating occupancy grid or metric maps. In [**Grisetti et al. 2007**] the feature based map is replaced with an occupancy grid map obtaining very impressive results in large environments.

2.1.2.3 Information Filter Methods (IF)

The Information Filter (IF) is an equivalent definition of the Kalman Filters based on an alternative representation of the Gaussian distribution. While the KF solution for SLAM is based on the moment parametrization of the Gaussian distribution of the state, we can obtain an alternative representation of the distribution which is known as the information form or also called the canonical representation of the Gaussian distribution (because it stems from expanding the quadratic in the exponential of the Gaussian distribution). The result is that rather than parameterizing the normal distribution in terms of its mean and covariance as in $p(\xi_k) = N(X_k, P_k)$ it is instead parametrized in terms of its information vector and information matrix, $N^{-1}(\eta_k, \Lambda_k)$. The process is done by:

$$
\begin{aligned}
p(\xi_k) &= N(X_k, P_k) \\
&\alpha \ \exp\left\{-\frac{1}{2}(\xi_k - X_k)^T P_k^{-1}(\xi_k - X_k)\right\} \\
&= \exp\left\{-\frac{1}{2}(\xi_k{}^T P_k^{-1}\xi_k - 2X_k{}^T P_k^{-1}\xi_k + X_k{}^T P_k^{-1} X_k)\right\} \\
&\alpha \ \exp\left\{-\frac{1}{2}\xi_k{}^T P_k^{-1}\xi_k + X_k{}^T P_k^{-1}\xi_k\right\} \\
&= \exp\left\{-\frac{1}{2}\xi_k{}^T \Lambda_k \xi_k + \eta_k{}^T \xi_k\right\} \\
&\alpha \ N^{-1}(\eta_K, \Lambda_k)
\end{aligned}
$$

Here the information matrix Λ and information vector η are defined as $\Lambda_k = P_k^{-1}$ and $\eta_k = P_k^{-1} X_k$ respectively.

Analogous to the EKF, when the state dynamics and measurements are non-linear, the equations of the IF have to be linearized obtaining an extended version of the IF known as the Extended Information Filter (EIF). The EKF and the EIF are considered as dual filters in the same sense that the canonical and moment parametrization of a Gaussian are reciprocal. But instead of maintaining the covariance matrix P_k and mean state estimate $\hat{X}_k$ like EKF does, the EIF maintains the inverse of the covariance, called the information or precision matrix Λ_k, and an information vector η_k. This can be seen in figure 2.10.

	Covariance Form	Information Form
Distribution *p(x,y)*	$N\left(\begin{bmatrix}\hat{X}_x\\ \hat{X}_y\end{bmatrix}, \begin{bmatrix}P_x & P_{xy}\\ P_{yx} & P_y\end{bmatrix}\right)$	$N^{-1}\left(\begin{bmatrix}\eta_x\\ \eta_y\end{bmatrix}, \begin{bmatrix}\Lambda_x & \Lambda_{xy}\\ \Lambda_{yx} & \Lambda_y\end{bmatrix}\right)$
Marginalization $p(x) = \int p(x,y)dy$	$\hat{X} = \hat{X}_k$ $P = P_k$	$\eta = \eta_x - \Lambda_{xy}\Lambda_y^{-1}\eta_y$ $\Lambda = \Lambda_x - \Lambda_{xy}\Lambda_y^{-1}\Lambda_{yx}$
Conditioning $p(x \mid y) = p(x,y) / p(y)$	$\hat{X}_c = \hat{X}_x + P_{xy}P_y^{-1}(y - \hat{X}_y)$ $P_c = P_x - P_{xy}P_y^{-1}P_{xy}$	$\eta_c = \eta_x - \Lambda_{xy} y$ $\Lambda_c = \Lambda_x$
Perfect Information	$\hat{X} = [x^T, y^T]^T, \ P = 0$	$\eta_c = NaN \ , \ \Lambda \to \infty$
Null Information	$\hat{X} = NaN \ , \ P \to \infty$	$\eta = 0 \ , \ \Lambda = 0$

Figure 2.10 Duality between the Covariance and Information form of a Gaussian distribution

Like in the EKF, the EIF has also the steps of prediction and update to track the posterior of the state. These steps are implemented as follow. Consider the information vector and the information matrix describing the state ξ in canonical form at time k as:

$$\eta_{k\text{-}1} = \begin{bmatrix} \eta_{x_{k\text{-}1}} \\ \eta_{m_{k\text{-}1}} \end{bmatrix} \;;\; \Lambda_{k-1} = \begin{bmatrix} \Lambda_{x_{k\text{-}1}} & \Lambda_{x_{k\text{-}1}m_{k\text{-}1}} \\ \Lambda_{m_{k\text{-}1}x_{k\text{-}1}} & \Lambda_{m_{k\text{-}1}} \end{bmatrix} \tag{2.7}$$

In the prediction stage the control input u_k is applied to the last robot configuration $\hat{X}_{k-1}$ to augment the state with the new robot position obtaining $p\left(\hat{X}_k,\, \hat{X}_{k\text{-}1},\, m \mid z_k,\, u_k\right)$.

$$\begin{bmatrix} \eta_{x_k} \\ \eta_{x_{k-1}} \\ \eta_{m_{k-1}} \end{bmatrix} = \begin{bmatrix} Q^{-1}(\hat{X}_k - F\hat{X}_{k-1}) \\ \eta_{x_{k-1}} - F^T Q^{-1}(\hat{X}_k - F\hat{X}_{k-1}) \\ \eta_{m_{k-1}} \end{bmatrix} \tag{2.8}$$

$$\Lambda_{k-1}^{aug} = \begin{bmatrix} Q^{-1} & -Q^{-1}F & 0 \\ -F^T Q^{-1} & \Lambda_{x_{k-1}} + F^T Q^{-1} F & \Lambda_{x_{k-1}m_{x-1}} \\ 0 & \Lambda_{m_{k-1}x_{k-1}} & \Lambda_{m_{k-1}} \end{bmatrix} \tag{2.9}$$

Where, $\hat{X}_k = f(\hat{X}_{k\text{-}1}, u_k)$ Q and F are the noise of the process and the jacobian for the process model respectively (these variables are also used in the EKF process). Notice that in the Information form, the new robot pose shares information with the previous pose but not with the map. After augmenting the state, when a new element is added, the information matrix starts to becoming sparse; the opposite occurs in the form of covariance, where the covariance matrix is full.

Using the marginalization operation showed in figure 2.10, we marginalize out the previous robot position from Eq.(2.9) and the information matrix gets full again losing its sparsity:

$$\eta_k^- = \begin{bmatrix} \eta_{x_k^-} \\ \eta_{m_k^-} \end{bmatrix} \;;\; \Lambda_k^- = \begin{bmatrix} \Lambda_{x_k^-} & \Lambda_{x_k^- m_k^-} \\ \Lambda_{m_k^- x_k^-} & \Lambda_{m_k^-} \end{bmatrix} \tag{2.10}$$

Finally with this operation, the prediction stage is finalized.

The prediction step in the standard formulation of the EIF is not easily calculated, since it contains matrix inversions and the number of landmarks grows, so does the difficulty of these inversions. In addition, the cost of the prediction step in Information Form is due to structure of the submatrix Λ_{yx} in the correction term $\Lambda_{xy}\Lambda_{y}^{-1}\Lambda_{yx}$ (known as the Schur complement) during marginalization. Computing this product is quadratic in the number of nonzero elements within this submatrix, that in the current case corresponds to all state elements "linked" to the previous robot pose $\hat{X}_{k-1}$. The cost is then $O(n^2)$.

On the other hand, for information filter, the update step is less complicated and less expensive that in Covariance Form. The corresponding EIF update is given by:

$$\Lambda_k = \Lambda_{k-1}^{-} + H^T R^{-1} H \qquad (2.11)$$

$$\eta_k = \eta_{k-1}^{-} + H^T R^{-1}(z_k - h(\hat{X}_k^{-}, 0) + H\hat{X}_k^{-}) \qquad (2.12)$$

Where H is the jacobian for the measurement equation, R is the noise on the measurements, z is the group of current observations and h are the predicted measurements. These equations show that the information matrix is additively updated by the outer product term $H^T R^{-1} H$. In general, this outer product modifies all elements of the predicted information matrix Λ_k. Also, the outer product $H^T R^{-1} H$ is zero almost everywhere except at the robot pose and observed features.

In [**Eustice et al. 2006**] Eustice et al. give a theoretical insight to understand how the information matrix evolves in time. Here the matrix structure is considered as a Gaussian Markov Random Field [**Bishop 2006**] where its elements correspond to links in the graphical model. Taking the example of figure 2.11, at time k the state vector ξ is composed by the robot position and the features m_1 and m_2 which are completely linked ($\xi^T = [\hat{X}^T, m_1^T, m_2^T]$). At this point, the information matrix is full. After, at the same instant k, the robot observes a new feature m_3 that is only related to the current robot location and consequently is only linked to $\hat{X}_k$; this new feature is added to the state (Fig 2.11 a). In figure 2.11 b, the robot moves at time $k+1$ and the state must be augmented with the new robot position $\hat{X}_{k+1}$ which causes that the information matrix start to becoming sparse. As in the previous case, this new robot position $\hat{X}_{k+1}$ is only linked to $\hat{X}_k$. Finally in image 2.11 c, we marginalize out the previous robot position making all nodes in the graph that were directly linked to $\hat{X}_k$

are fully connected between them. This last step, as we have already said causes that the information matrix become full.

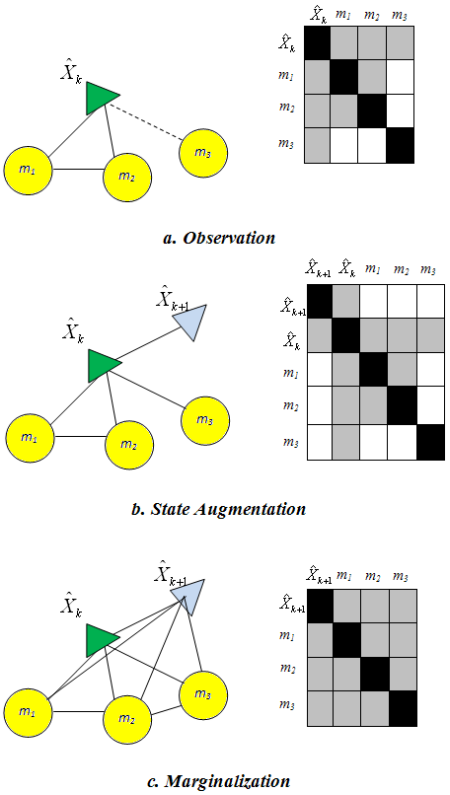

Figure 2.11 Evolution of the Information Filter Method

A drawback of the EIF solution with respect to the EKF is that in order to calculate the jacobians F and H we need to recover the mean vector of the estimate $\hat{X}_k$ from the information vector η_k at each step of the filter. Also, parts of the covariance matrix have to be recovered from the information matrix to perform the data association. The cost, in the best of the cases for partial recoveries, is $O(n^2)$ (even more expensive for full recoveries which have a computational cost of $O(n^3)$).

In order to reduce the computational cost of the EIF a control must be done over the form of fill in the information matrix due to the marginalization operation done in the prediction step and exploit the natural sparsity of the matrix Eq.(2.8, 2.9). Below we summarize the tree most important algorithms based on the EIF.

a) SEIF

Sparse Extended Information Fiter (SEIF) differ from the extended information filter described in the previous section. The number of links to the robot and to each feature in the map is bounded by a constant that is independent of the number of

features in the map. When this bound is over passed the links with smaller value are break until the bound is accomplished again. With this method the information matrix remains sparse avoiding the fill in due to the marginalization operation and reducing the computational cost.

The motivation for maintaining a sparse information matrix is that in SLAM, the normalized information matrix is already *almost* sparse. This suggests that by enforcing sparseness, the induced approximation error is small. However, as it is shown in [**Walter et al. 2007**], an important consequence of the SEIF sparsification algorithm is that the resulting approximation significantly underestimates the uncertainty in the state leading to overconfident state estimates.

b) ESEIF

The Exactly Sparse Extended Information Filter (ESEIF) was presented by Walter *et al.* [**Walter et al. 2007**] as an alternative sparse information filter that achieves the computational benefits of a sparse parametrization while preserving consistency. In contrast to the approximated sparsification enforced in SEIF, the ESEIF imposes exact sparsity in the matrix. Instead of breaking links between the robot and features in the map the algorithm maintains sparsity by controlling the initial formation of the links. More specifically, the ESEIF manages the number of active landmarks by marginalizing out the vehicle pose, essentially “kidnapping” the robot. The algorithm subsequently relocalizes the vehicle within the map based upon new observations to a set of known landmarks. The set of features that were originally active have been made passive and the set of landmarks used for relocalization form the new active map.

The key contribution of the ESEIF is that it avoids the need to approximate conditional independencies and thereby preserves the consistency of the Gaussian distribution. The ESEIF maintains map and pose estimates that are nearly identical to those of the EKF but exploits the sparse SLAM parametrization to track the distribution in near-constant time. The result is a computationally efficient algorithm that is consistent but less precise than the EKF since some of the information is disregarded when the robot is kidnapped. As in the SEIF, additional approximations are made to evaluate the jacobians or perform data association.

c) ESDF

The Exactly Sparse Delayed Filter (ESDF) was proposed by Eustice et al. [**Eustice**

et al. 2006] as an alternative formulation, this technique solves the SLAM problem through a constant-time filtering algorithm. This means that the ESDF computational cost does not grow up with the environment size. This can be achieved because the state vector does not store a map or features but just the historical of robot states corresponding to the vehicle locations where the observations were made. Therefore, ESDF is considered as the solution to the scalability problem for arbitrately large environments. The key idea of this algorithm is that the robot poses are not filtered out and observations are used to constraint relations between pairs of poses; this because the marginalization of robot poses that fills-in the information matrix destroys its sparsity.

The type of representation used by the ESDF algorithm is view-based rather than a feature-based approach. In this kind, the estimation problem consists of tracking the current robot pose in conjunction with a collection of historical poses sampled from the robot's trajectory; consequently, none explicit model of landmarks is necessary. The associated posterior is then defined over a collection of delayed states.

To evaluate jacobians and perform data association, the ESEF as the SEIF have to make some approximations when portions of the mean state vector and covariance matrix are recovered from its information form.

Finally, in the overall of methods based on the EIF we can say that they are able to reduce the computational cost $O(n^2)$ of the classic EKF-SLAM; however, they don't deal with the consistency problem. In fact, the EKF is usually more consistent and precise than the solutions obtained with the IF because of the approximations introduced.

2.1.2.4 Submapping Techniques

Techniques based on building submaps (figure 2.12) have been demonstrated to be well suited for mapping large environments as they reduce the computational cost and confront the complexity and consistency problems of the final estimation. The main idea of the submaps method is that if computational and consistency issues start becoming a problem when the map is large (figure 2.12a), it can be break into smaller sections with local coordinates that can be considered separately from each other (figure 2.12b). The relationships between geometrically adjacent submaps correct each submap's position to acquire the global consistency (figure 2.12c).

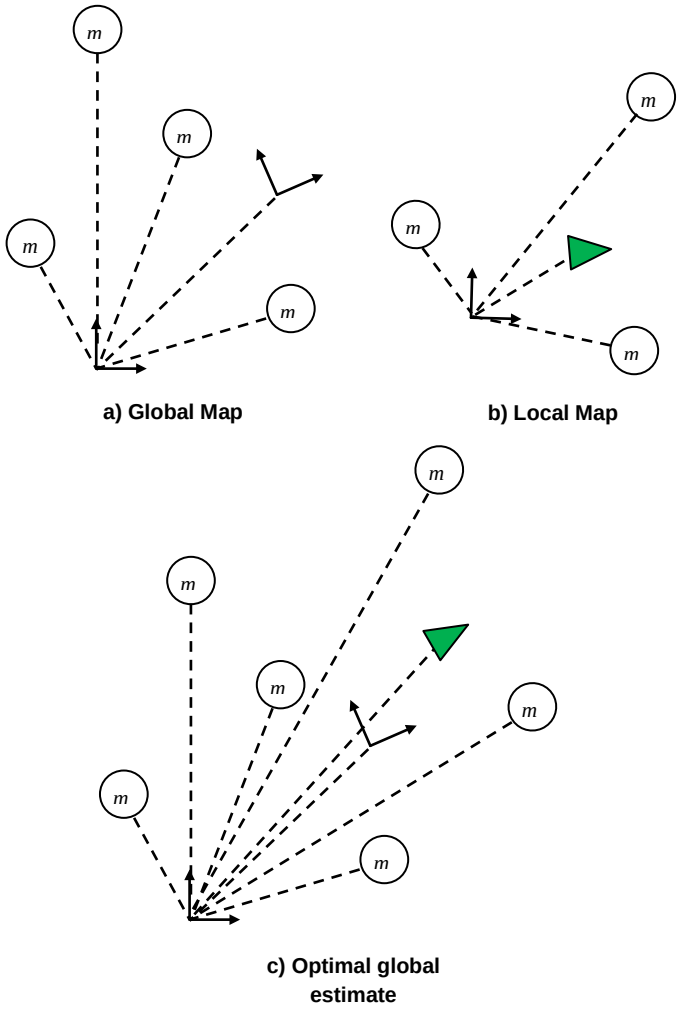

Figure 2.12 Submapping Techniques

Between the advantages in the use of this technique we can mention:

- As each submap uses its own sequence of odometry reading u_k and the information data z_k it is not necessary to compute the correlations between the features in the current submaps and features in other local map. For this, the computational cost of local map building is constant $O(1)$ and independent from the size of the global map.

- The sensor-rate update is independent of the total map size because the landmarks that need to be updated at each time instant is limited only to those who are in the local submap coordinate frame, then the full update, and propagation of local estimates can be done in the background task at a much lower update rate while still permitting sensor-rate global localisation

- Usually, the robot pose and the uncertainties are small during a map construction because each submap is initialized with zero uncertainty. Therefore, the final global map after the after the union of the sub-maps has better consistency properties than the EKF.

- Submap registration can use batch validation gating which improve association robustness.

In literature, we can find several techniques based on independent submaps. The main difference between them is the form in how the submaps are joined to obtain the final global map. The first technique using absolute submaps[1] was Decoupled Stochastic Mapping [**Leonard et al. 2000**]. As this technique uses absolute submaps which are not statistically independent some approximations are needed to get rid of the dependencies, introducing inconsistency in the map. With local submaps[2], these are initialized with zero uncertainty in the robot pose, given that the base reference when the local map is started is usually chosen to be the first robot pose (which is well-defined for the moment parametrization of the EKF). So, local maps are statically independent and thus, uncorrelated [**Tardos et al. 2002**] under the assumption of white noise and only if no information is shared between maps.

Taking this into account, the Map Joining [**Tardos et al. 2002**], Constrained Local Submap Filter (CLSF) [**Williams et al., 2002**] and the Divide & Conquer [**Paz et al. 2008**] methods build a sequence of statistically independent local submaps of limited size where each submap is built using the classical EKF. These three methods will be explained briefly below.

a) Map Joinning and CLSF

The Map Joining [**Tardos et al. 2002**] and the Constrained Local Submap Filter (CLSF) [**Williams et al., 2002**] are two equivalent techniques independently developed to produce efficient global maps by consistently combining local submaps with a cost total of $O(n^2)$. The joining procedure is performed based on a three step procedure.

- In the first step, the robot is at some approximately known location in a local submap from where a new one will be built in a normal SLAM fashion. The

[1] Absolute submap. Submap expressed in global coordinates.
[2] Local submap. Submap expressed with respect to a local coordinate frame.

new submap is then stored when its system state vector contains too many features or the vehicle location is too uncertain. Once we have the two sequential submaps, they are simply accumulated in a single mean vector and covariance matrix $\xi = [\xi_1^{\mathrm{T}}, \xi_2^{\mathrm{T}}]^{\mathrm{T}}$:

$$X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \; ; \; P = \begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix} \qquad (2.13)$$

As both submaps are statistically independents, the covariance matrix P is blockdiagonal.

- In the second step, the process of local map registration results in a global map update where the new submap X_2 is transformed into the reference frame of the submap X_1 using the last robot position contained in it.

$$X_{total} = f(X) = \begin{bmatrix} X_1 \\ X_{R_1}^{W} \oplus X_2 \end{bmatrix} \; ; \; P_{total} = J_{\oplus} P J_{\oplus}^{T} \qquad (2.14)$$

Been $\oplus$ the composition operation defined by Smith et al. in [Ref] and $J_{\oplus}$ is the jacobian of the function f which make the transformation of the elements of submap X_2 into the reference frame of the first submap X_1.

- Finally the last step corresponds to the elimination of duplicate features. Here, features found in different submaps that correspond to the same environment feature are fused using a fusion mechanism to update the global map. In this way, a more precise map is achieved.

The process is repeated joining the next sequential local submap to the recently calculated map until no more local submaps remain to be joined. At the end, the global map will be obtained.

b) Divide & Conquer

Divide and Conquer SLAM is based on the idea of Local Map Sequencing proposed in [**Tardos et al. 2002**] where the idea is to build a sequence of local independent maps of equal constant size while the robot traverses the environment instead of working on a single absolute map. The final absolute map is achieved by joining at

fixed intervals of time during the process. The main difference with the previous approach is that Divide & Conquer SLAM (D&C SLAM) [**Paz et al. 2008**] instead of joining each new local map to a global map sequentially, it carries out map joining in a hierarchical fashion. The basic process to join a pair of submaps is the same as the one described for Map Joining

The algorithm D&C utilize a binary tree of local maps structure to sort the sequence of joining. The lower nodes of the hierarchy represent a sequence of m local maps of minimal size p and the upper level represents the final map of size n. The middles levels represent intermediate joins during the process. The structure is shown in figure 2.13.

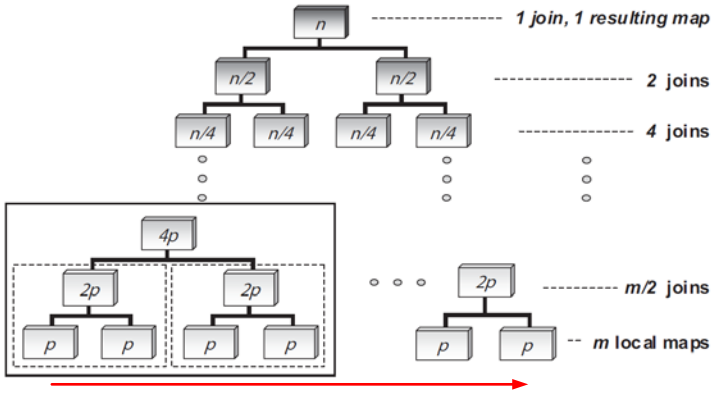

Figure 2.13 Hierarchy of maps that are created and joined in D&C SLAM.
The red arrow represents the sequence in which maps are built and joined [Paz 2008].

In [**Tardos et al. 2002**] is shown that computational cost for local maps of fixed size could be reduced by a large constant factor, but is still $O(n^2)$ in every map joining step.

2.1.2.5 Graph based Methods

The SLAM problem can be addressed in a very intuitive way using the so called graph-based formulation proposed for Lu et al. in [**Lu et al. 1997**]. This approach allows to solve the SLAM problem as a non linear optimization problem where given the sensor measurements is possible to find the robot trajectory and the map with greatest probability.

The graph-based SLAM problem is solved by constructing a graph structure whose nodes represent landmarks or robot locations and in which an edge between two nodes represents a data-dependent spatial constraint. This constraint consists in a

probability distribution over the relative transformations between the two poses. These transformations are either odometry measurements u_k between a consecutive pair of locations X_{k-1}, X_k or are determined by aligning the observations z_k acquired between the two robot locations X_{k-1}, X_k and landmarks m_i, assuming that at time k the robot has sensed landmark i.

For a more easy understanding of graph construction we will use the figure 2.14. At step time $k=1$ the robot measures the feature m_1 and adds an edge between X_1 and m_1 in the graph. This edge can be represented in a matrix format adding a value between the element X_1 and m_1. This is made to correspond to a quadratic equation that defines the resulting constraint (Figure 2.14 a). After, if the robot moves to a new position X_2, an edge between X_1 and X_2 is added representing the odometric readings u_2 and as in the previous step a value representing this movement is added to the matrix between X_1 and X_2 (Figure 2.14 b). The repetition of these two steps leads us to a graph of increasing size (Figure 2.14c).

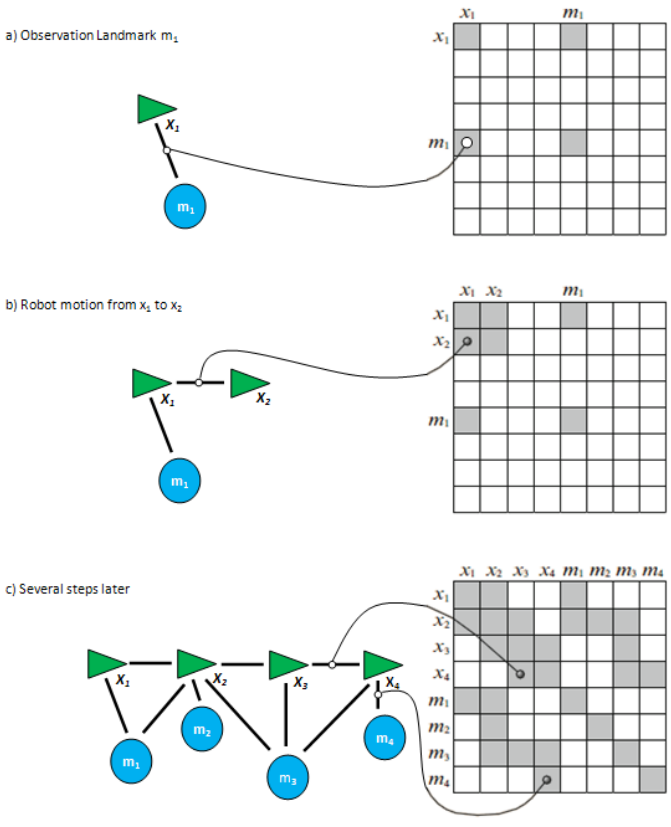

Figure 2.14 Graph constructions. On the left we observe the graph On the right the constraints are showed in a matrix form

Once the graph is already constructed, the configuration of the robot poses that best satisfies the constraints can be found. Given this, we observe that the graph-based

SLAM can be seen as two separate problems. The first is the construction of the graph using the raw measurements, and the second is to determinate the most likely configuration of the poses given the edges of the graph.

A large variety of proposals using this approach have emerged in the robotics community. These algorithms have been subsequently classified according to the particular meaning given to the nodes and edges in the graph as well as the mathematical tools involved during the estimation. Some of these algorithms are presented below.

a) Hierarchy Algorithms

The hierarchy algorithms allow the construction of metric maps of large environments in real time using different levels of construction. In figure 2.15, we can see a map constructed with a two level hierarchy algorithm. The lower (local) level is based on the building of an independent submap (as the ones described in the previous subsection) that captures the local environment and the current robot pose along with the uncertainties of each defined by its mean and covariance matrix. At the upper (global) level, the topology of the environment is represented by a graph of coordinate frames, with each node in the graph representing a local submap with its own local reference frame and each edge representing the transformation between adjacent submaps.

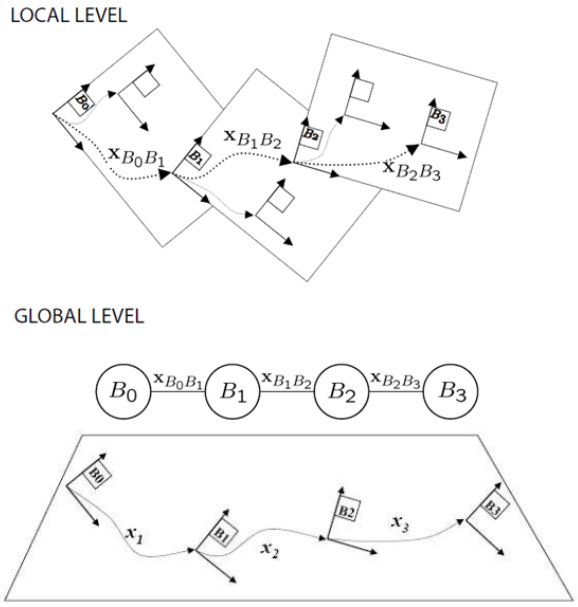

Figure 2.15 Hierarchy Algorithms. Lower level is based on submaps with their own reference frame.Upper level corresponds to a graph whose edges represent the relative transformation between base references of submap pairs [Estrada et al. 2005].

Among the methods that use hierarchy algorithms, we can mention the Constant-Time SLAM (CTS) [**Leonard et al. 2003**], Atlas [**Bosse et al. 2004**], and the Hierarchical SLAM [**Estrada et al., 2005**].

The Constant-Time SLAM framework [**Leonard et al. 2003**] uses multi overlapping local submaps with the frame referenced to one of the features in the submap. This technique maintains a single active map and computes a partial solution independently. The management of each map is limited by a radius r centered in position where the robot was at the time of the creation of a map. This region limits the robot location but not the feature locations. This means that every feature observed from a position in the submap will be added to it.

The information about the robot's position is used to know in which submap the vehicle is in. If the robot has traveled a distance bigger than (r+h), the vehicle is considered out of the submap and it must determine to which one (if the robot has traveled to any of the existing submaps) the vehicle has transitioned to. The parameter h is used to prevent excessive map switching. The basic Flowchart of the CST approach is shown in figure 2.16.

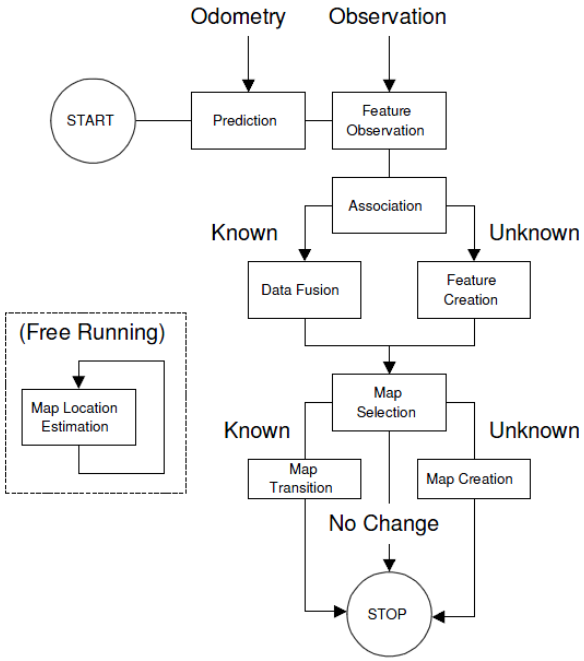

Figure 2.16 Flowchart of the CST algorithm. Image takes from [Newman et al. 2003]

When a vehicle transits from one map to another, the map location estimation is performed in order to improve the global estimation of the features locations in each local submap. The basic idea is to find the best global pose with the lowest global uncertainty of shared features contained in two neighbor submaps and shift the current submap to the correct position by using a minimization function over these

features. So, through the composition of transformation derived from the local maps, it is possible to achieve the consistency of the global Map Location.

The ATLAS framework [**Bosse et al. 2004**] is a relative submap method using both, metric and topological approaches, which is aimed specifically at applying existing small-scale algorithms to the mapping of large-scale cyclic environments.

This approach, as any other graph based technique, maintains an interconnected set of local coordinate frames which are represented by the nodes in the graph instead of a global coordinate frame. In each one of these local frames, the building of a map is carried out by capturing the local environment and the current robot location along with their uncertainties modeled with respect to its own frame. With this, the algorithm restricts the representation of errors to local regions, minimizing linearization problems, but also provides a way of providing global results by combining local maps. In the other hand, the edges of the graph correspond to the transformation between the frames they connect. The uncertainties of the edges are derived from the output of the SLAM algorithm running in a local region and are represented by a Gaussian random variable.

In order to keep efficient, the CTS and Atlas frameworks does not impose loop closing constraints. This is, if more of one path between a pair of nodes exists in the graph, the relative position between the nodes will depend only on the path followed to calculate the composite transformation between them. This means that loop closing are not imposed and it is only used to decide the shortest path.

The Hierarchical SLAM [**Estrada et al., 2005]** is a real-time accurate mapping for large loop environments that combines the use of local maps with a consistent representation of them. The main idea of this method as in the other two presented above is to maintain local submaps independently without share any information in order to maintain convergence and also maintain an adjacency graph for relative positions between these submaps.

In this approach, the last robot position in a submap establishes the new local coordinate frame where the new submap will be built. Every submap generated is bounded by the number of features, uncertainty of the vehicle location, or not matchings found in the data association. Consequently the cost of this algorithm remains linear and bounded.

The main difference between Hierarchical SLAM and other similar approaches as Atlas and CTS is that loop consistency is imposed by constraining cycles in the graph to improve the accuracy in the absolute location of all local maps in the loop. To reduce linearization errors in big loops, a nonlinear optimization algorithm is implemented. This operation improves the accuracy in the absolute location of all local maps in the loop and has an $O(n)$ cost in the number of submaps n involved. If a correction has to be made at a global level as result of a loop closing, this correction is not back propagated to the local level in order to maintain independence between the local maps.

In all the algorithms presented in this subsection, if two features in two submaps correspond to the same object in the environment, the position on the submaps is respected in order to maintain the independency between submaps. This results in weak links between submaps, obtaining approximated solutions.

b) Tree-based algorithms

In this category, we will present two similar approaches that although they have been developed independently they are closely related, these techniques are called **Treemaps** [**Frese 2006**] and **TJTF** [**Paskin et al. 2003**]. Both of them divide the environment into parts the whole map and represent it as a tree data structure (figure 2.17) that dynamically updates and factorizes the joint probability distribution of the system when new observations are obtained. Gaussians densities parameterized in canonical form are used for the probability distributions.

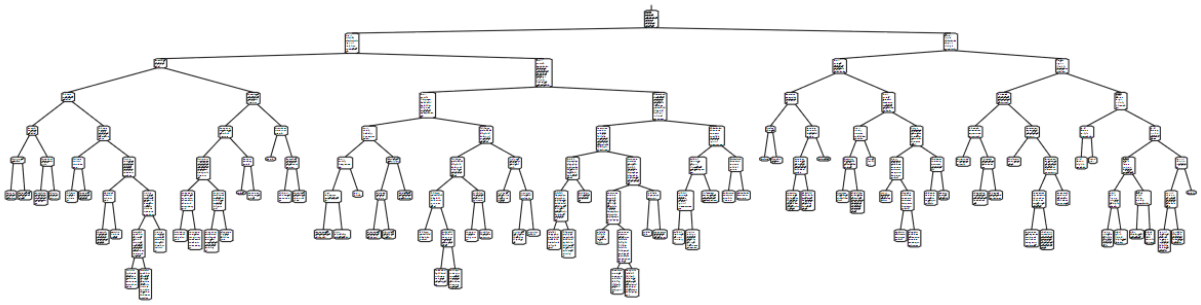

Figure 2.17 Tree representation of the map. The size of the nodes is proportional to the number of features represented. Image taken from [Frese 2006]

Treemap is a sophisticated but also complicated SLAM algorithm that creates a balanced binary tree to perform integration and marginalization, leading to an impressive algorithm that has the ability to deal with very large maps. Each node in the structure represents a specific region of the environment and stores marginal

distributions about the landmarks in it, usually seen simultaneously. The regions corresponding to nodes are defined as a set of landmarks being close to each other instead of be definite geometrically. At every moment the leaf that corresponds to the region of the current robot location is considered. This leaf is known as the actual leaf. While nodes in the tree represent marginal distributions of the joint probability, arcs are used to send “messages” between marginals with elements in common. Finally to integrate a measurement, all nodes from a single leaf up to the root need to be updated by passing Schur-complements along the arcs.

The treemap algorithm can be geometrically sketched in a simple and intuitive way. Let assume that the robot is in a construction that is virtually divided in two parts A and B. Given these parts, now the question that has to be answered is: if the robot is actually in part A, what is the information required about B? In most cases only few features of B are involved in observations while the robot is in A and only these are considered. Probabilistically speaking, the information needed about B is only the marginal distribution of features observed from both A and B conditioned on observations in B.

The construction can be divided into a binary tree of regions by applying recursively the mentioned idea and passing probability distributions along the tree. Figure 2.18 shows the Bayesian justification for this approach. The inputs to treemap during updates (black arrows),are observations z_i of feature positions and robot poses assigned to leaves of the tree modeled as distributions $p_n^I = p(X_{n\downarrow} \left| z_{n\downarrow} \right.)$ of the state vector X (**n** range over all leaves). After, the marginalization ($\otimes$) of features uninvolved is done; the result $p_n^M = p(X_{n\uparrow\downarrow} \left| z_{n\downarrow} \right.)$ is passed to the parent node which integrates ($\odot$) the distributions $p_{n\swarrow}^M$ and $p_{n\searrow}^M$ resulting in $p_{n\swarrow}^M p_{n\searrow}^M = p(X_{n\uparrow\downarrow\vee\swarrow\searrow} \left| z_{n\downarrow} \right.)$. Then the features involved are marginalized out ($\otimes$) by factorizing $p_{n\swarrow}^M p_{n\searrow}^M$ into the marginal p_n^M and the corresponding conditional p_n^C stored at *n*:

$$p_{n\swarrow}^M(y) \cdot p_{n\searrow}^M(y) = p_n^M(v) \cdot p_n^C(y), \quad y = \begin{pmatrix} u \\ v \end{pmatrix}$$

$$p_n^M = p(X_{n\uparrow\downarrow} \left| Z_{n\downarrow} \right.)$$

$$p_n^C = p(X_{n\swarrow\searrow} \left| x_{n\uparrow\downarrow}, z \right.)$$

Where u represents the features only involved below the node *n* and v the features involved above *n*.

The estimate (gray arrows) is computed recursively down the tree, each node n receives a distribution $p_{n\uparrow}$ from its parent, integrates ($\odot$) it with the conditional p_n^C, and passes the result p_n down. In the end, estimates X_n are available at the leaves. As we can see at the figure, the conditional p_n^C is stored at n because it is not need above. On the contrary, the marginal has to pass to the parent $n\uparrow$ to be processed there. Finally, it has to be noted that a feature comes from p_n^I up and passed in p_n^M from the leaves to the least common ancestor of all these leaves. There, it is marginalized out and finally stored in p_n^C. By sending messages (basically multiplying marginals and conditionals) through arcs between nodes, it can be demonstrated that we can consistently update every leaf marginal with observations taken in other leaves.

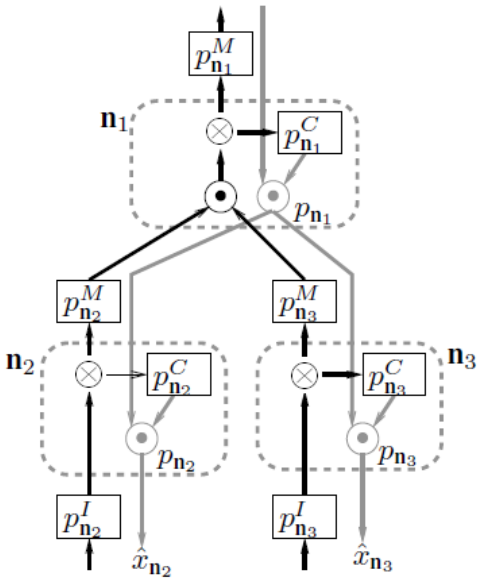

Figure 2.18 Data flow of the probabilistic computations performed by treemap. Figure taken from [Frese 2006b]

Given that the treemap uses a balanced tree structure, the computational cost to recover a part of the state (a leaf) is *O(log n)* and to recover the entire map the cost is *O(n)*, been n the number of leaves.

As the Treemap algorithm, we can find a very similar approach in the **Thin Junction Tree filter**. It creates also a tree structure but instead of forming a balanced binary tree, the properties of the tree created and the type of messages sent are based on the Junction Tree algorithm [**Bishop 2006**].

In the tree structure of the Junction Tree, the nodes C represent clusters, the edges S of the tree represent separators between two connected nodes C_i, C_j and analogously to the nodes, they have a set of variables Vs corresponding to the intersection of Ci's and Cj's variables and also a separator potential ϕ_s (Figure 2.19). In the SLAM case, the variables correspond to landmark and robot states and the potentials are a generalization of probability distributions that are used to factorize the joint probability distribution. In contrast to the basic static Junction Tree, the TJTF develops a group of methods for dynamically updating the junction tree to reflect filtering updates. Given that the junction tree grows under measurement and motion updates, the structure has to be periodically "thinned" to remain tractable via efficient maximum likelihood projections.

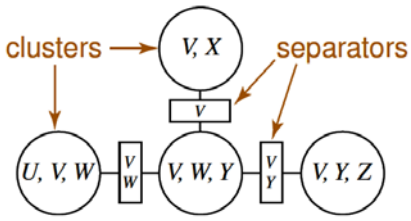

Figure 2.19 Clusters and separators in a junction tree

Initially the filter starts with a single cluster containing x. Once the robot moves, its position most be updated which consists of the prediction and odometry updates and then the state variable of the previous time slice is marginalize (figure 2.20). The prediction update consists of adding a new node X_{k+1} and connecting it to X_k. When the state X is marginalize, all the clusters in which it resides must be merged (figure 2.21). In the worst cases, when X resides in all of the junction tree's clusters, the belief state would collapse to one large cluster. To prevent this, the TJTF first contracts X until it resides in only one cluster and then performs the prediction and odometry updates and then marginalize. Variable contraction is used to reduce the diameter of the junction tree when the cluster X_k caused by marginalization is too large.

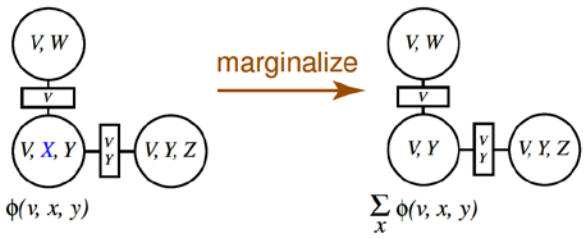

Figure 2.20 Marginalization in a junction tree

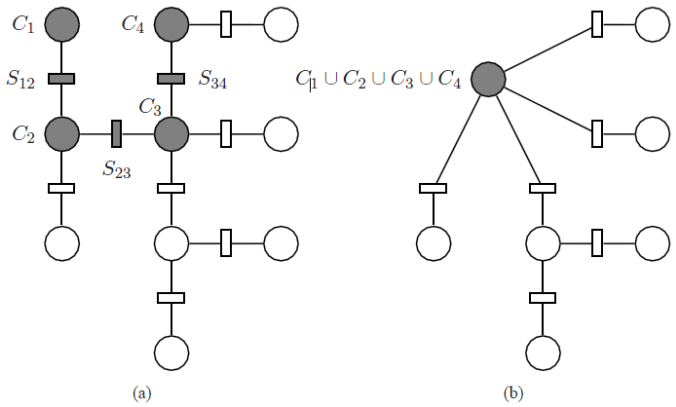

Figure 2.21 Cluster merging. a) The shaded clusters and separators contains the state X b) to marginalize the state X of the junction tree, the clusters C1, C2, C3 and C4 are merged and the separators between them are eliminated [Paskin et al. 2003]

When updating with a measurement of landmark l, the measurement potential $\psi(x_1,l)$ has to be multiplied into the only cluster. This is easily made if the cluster has not achieved its limit size by simply multiplying the observation potential $\psi(x, l)$ into φ_C and distributing evidence in the cluster C. On the other hand we can find two cases:

- If the landmark l has been previously seen, then its potential has to be multiplied into the cluster C that contains it and that is the closest to another that contains X. In the worst case, the robot will reobserve a landmark whose state variable is very far from the robot's state variable X in the junction tree, and in consequence X will have to be added to every cluster to preserve the running intersection property. Finally, if the insertion of X in the clusters increased the diameter beyond some threshold, then we perform variable contractions until the junction tree is thin enough (figure 2.22).

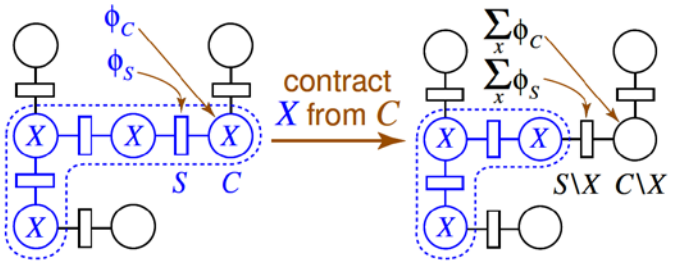

Figure 2.22 Variable contractions in Thin Junction Tree filter

- If a landmark l has not been seen, a new node containing l and X must be attached to the cluster state from where it has been seen.

The Treemap and the TJTF algorithms achieve an extremely efficient operation with some weak approximations even if they are not very intuitive and complex to implement. One of the drawbacks of these algorithms is that as they are based on the information form, thus, well known data association algorithms based on covariances cannot be directly applied. Moreover, based in the tree structure that they use, is not trivial to obtain the corresponding covariance from. Finally is necessary to says that the consistency proprieties of these algorithms are really close to the EKF because they work in an absolute reference system and cannot be improved as when employing local map references.

In this subsection we have made a very simple presentation of the treemaps and TJTF; because of this, we strongly recommend the interested reader to directly study the papers referenced in order to understand the exact process.

c) Batch Techniques

The methods presented in previous subsections are based on filtering algorithms. However, we can found in literature strategies that use batch techniques. The basic idea behind these algorithms is to find the maximum likelihood estimate (MLE) based on the entire history of robot motion and measurement data.

GraphSLAM [**Thrun et al. 2005**] and the Smoothing and Mapping (SAM) [**Dellaert et al. 2006**] are batch strategies that use a graph structure whose nodes corresponds to robot poses and map features and the arcs store information about the motion and measurements constraints (Figure 2.23a). Both methods solve the MLE by optimizing a nonlinear log-likelihood function over a series of iterations, which provides robustness to linearization errors. After each linearization, the information matrix built by these methods becomes completely sparse since the robot trajectory is not marginalized out (Figure 2.23b).

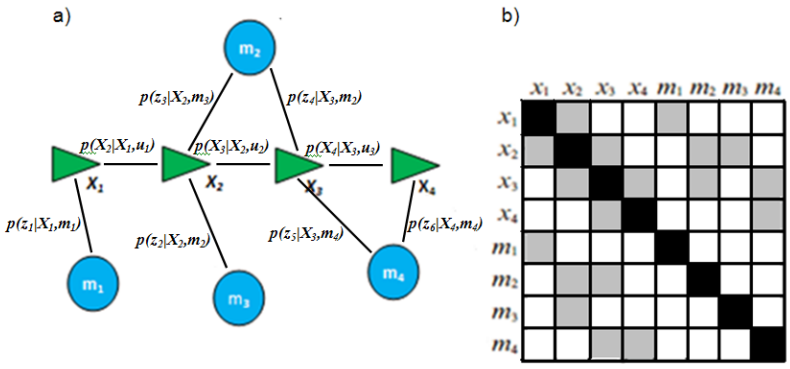

Figure 2.23 Graph structure used by the GraphSLAM and SAM methods with its Information matrix. It can be seen that the entire history of robot motion is estimated

The linearized system for each iteration k solved by these methods is:

$$\Lambda_k \mu_k = \eta_k \qquad (2.15)$$

with Λ_k as the sparse information matrix after the linearization, μ_k the mean vector that we want to calculate and η_k the corresponding information vector. The whole system of nonlinear motion and measurement constrains is linearized around μ_k when it is obtained giving a new system of equations Λ_{k+1} and η_{k+1} that has to be solved again. This cycle is repeated until the system converges.

In **GraphSLAM** algorithm, an elimination algorithm is used in order to solve the linear system in Eq. (2.15) at each iteration. This marginalizes over the map the features and reduces the graph to one with only the pose history. Subsequently, the path posterior map is calculated over the pose history using standard inference techniques. GraphSLAM also computes a map and certain marginal posteriors over the map.

Smoothing and Mapping (SAM) refers to the framework wherein the SLAM problem is solved using smoothing approaches. The SAM algorithm relies on a QR and Cholesky factorization of information matrix in Eq.(2.15) paying attention to variable ordering. Then, the system is jointly solved for robot poses and map features via back-substitution.

Although both algorithms take advantage of the sparsity of the linearized information matrix to speed up the calculations two drawbacks are encountered when using these methods. The first is that the system has to be solved every time that a new observation is introduced. The second drawback occurs when a region in the environment is revisited; this because the computational cost increases although the environment does not change due to the state vector still grows linearly since most of the state elements turn out to be robot poses.

2.1.2.6 Set-membership methods

The strategies in this category rely in the recently developed set membership estimation theory which uses a deterministic unknown-but-bounded description of noise and parametric uncertainty (interval models). These methods verify at any moment the consistency between observed and predicted behavior by using simple

sets to approximate the set of possible behaviors. When an inconsistency is detected a fault can be indicated, otherwise nothing can be stated.

Di Marco et al. present in **[Di Marco et al. 2001]** a Set-membership SLAM approach under the hypothesis that the errors affecting all sensor measurements are unknown but bounded. Estimates of robot position and selected landmarks are derived in terms of feasible uncertainty sets which are defined as those regions where the robot and the landmarks are guaranteed to lie according to the available information. Based on recursive approximations of the uncertainty regions through simply shaped sets, this work exploit the specific structure of the nonlinear SLAM problem to get efficient solutions.

In **[Jaulin et al 2009]**, Jaulin et al. propose a off-line SLAM solution for a underwater vehicle using a set membership method based on interval analysis. Here, seamarks are detected by a human operator after the mission of the robot. The SLAM problem in this work is cast into a constraint satisfaction problem for which interval propagation algorithms are particularly powerful

In **[Le Bars te al. 2010]**, Jaulin et al. presents an experiment using a set-membership approach for SLAM based on interval arithmetic initially developed for the Redermor submarine **[Jaulin et al 2009]** through an improved version of the Guaranteed Estimation of Sea Mines with Intervals. As in its previous work, seamarks are also detected by a human operator.

2.2 Planning Exploration Strategies

Classically, the exploration problem can be understood as follow: Given what you know about the world, where should you move in order to obtain as much new information as possible? Generally exploration techniques work using an occupation probability map and the frontier concept introduced by Yamauchi **[Yamauchi 1997]**. However, there are other approaches that use other forms for identifying the regions of interest for the exploration.

Whatever the chosen strategy, a good exploration algorithm must have two properties:

- ***Completeness*** that requires that the robot covers most of the environment.

- ***Effectiveness*** that requires that the robot achieves the completeness by minimal efforts, such as exploration time, exploration distance, etc.

Focusing on the exploration planning, the techniques developed are classified into two types: deliberative and reactive explorations.

2.2.1 Deliberative Exploration

The methods classified into this group utilize the map information of the environment to take decisions about the frontiers that will be explored and use path planning techniques to lead the robot toward the chosen frontiers. The map can be a complete global map that is known before the start of exploration, or a partial map that is built on-line while the robot explores. In the last case, when the robot moves towards frontiers the known area will increase. The basic idea is to identify the covered area and then to chose appropriate frontier for the robot to move towards.

Two main issues can be found in this category of exploration algorithms for a single robot system:

- When at some point the robot must choose between several frontiers. What frontier should the robot choose such that the information gain is maximized?

- How can the robot travel toward that frontier in a safe and efficient way?

For the first problem the robot needs to estimate and compare the potential information gain of approaching each frontier and choose the best one. A basic case can be found in the work of Yamauchi [**Yamauchi 1997**] where the robot chooses always the nearest frontier. This because the estimation of the potential information gain is made by calculating the distance from the robot to the frontiers and the nearest frontier has the highest potential information gain.

Another way to choose the best frontier in single-robot exploration is using the cost-utility model. One example of the use of this model is the work of Gonzalez-Baños and Latombe [**Gonzalez et al. 2002**] called NBV (Next-Best-View Algorithm). In this algorithm the map of the environment is built iteratively. Initially, the robot builds a local safe region with the information gathered by the sensors at the robot's initial position q_0. At each iteration, the algorithm update the global map by joining the safe region built so far with the local safe region generated at the new position q_k. This

new safe region is then used to choose the next sensing position q_{k+1} based on the expected gain of information that will be sensed at this position, the need of alignment between the global safe region built so far and the new local safe region and finally by the cost of movement to reach the new position.

In the cost-utility paradigm, the cost is the length of the path between the current robot's position and the frontier, whereas utility can be understood in different ways:

In [**Simons et al. 2000**] Simmons et al. consider the utility as the expected visible area behind the frontier. In this work every individual robot construct "bids," which describe their estimates of the expected information gain and costs of traveling to various locations. A central executive receives the bids and assigns tasks in an attempt to maximize overall utility, while trying to minimize overlap in coverage by the robots.

Stachniss et al. [**Stachniss et al. 2006**] use semantic information to increase the utility of the candidate destinations situated in corridors. This is done by introducing a utility function $U(t)$ given by

$$U(t_n \mid t_1, \ldots, t_{n-1}) = Ut_n - \Sigma P_{vis}(t_n, t_i) \qquad (2.16)$$

where $P_{vis}(t_n, t_i)$ describes the probability that the frontier t_n can be observed by a robot moving to t_i.

In his work, Burgard et al. [**Burgard et al. 2005**] present an approach for the coordination of multiple robots, which simultaneously takes into account the cost of reaching a target point and its utility. Whenever a target point is assigned to a specific robot the utility of the unexplored area visible from this target position is reduced. In this way, different target locations are assigned to the individual robots and the exploration speeds up since the robots choose different frontiers that are far from each other.

Others authors have developed some methods based on different representations of the environment. For example, Franchi et al. [**Franchi et al. 2007**] have based their work on the randomized incremental generation of a collection of data structures called Sensor-based Random Trees (SRT), each representing a roadmap of an explored area with an associated safe region. The SRT is incrementally built by using a randomized local planner which privileges the frontier of the Local Safe Region (LSR). In particular, each node of an SRT contains a configuration assumed by the

robot and the LSR perceived from that location, while an arc between two nodes represents a collision-free path between the two configurations. This method will be deepened in later sections.

Others approaches like the method multi-robot developed by Wurm et al. [**Wurm et al. 2008**] take advantage of information about the structure of the environment instead of only considering frontiers between unknown and explored areas. This information is used for assigning optimally a different unexplored room to each robot using the Hungarian method [**Kuhn 1995**].

Once that robot has chosen a target frontier, this position has to be achieved in a safe way and using minimal moves at the same time. This can incur heavy computations since it is a Non-Polynomial (NP) hard problem. Probability-based algorithms can be applied to effectively plan the path between the robot and the frontier such as Probabilistic Road Map [**Choset et al. 2004**] and Rapid-exploring Random Trees [**LaValle 1998**].

2.2.2 Reactive Exploration

In contrast to deliberative exploration, reactive exploration algorithms are behavioral approaches and do not need a map information. Although this kind of exploration is in general challenging to strategically reason about long or short-term objectives, it is well suited to effectively respond to dynamic changes in real-time environments. The basic idea is to find the optimal move based on the current status of robots. For example, if a door is detected, the robot can move towards the door because it is a hint of an unknown area.

The most representative approach for reactive exploration is the Artificial Potential Field [**Khatib 1986**]. The central idea of these methods is to create an artificial potential field that will attract the robot to a target. At the same time another behavior is defined in which each obstacle generate a repulsive field around it. If the robot approaches the obstacle, this repulsive force will act pushing it away from the obstacle. At the end, the two behaviors, seeking and avoiding, can be combined by combining the two potential fields, the robot, then, can follow the force induced by the new filed to reach the goal while avoiding the obstacle. The force of attraction is usually random and is given to trigger the movement of the robot i.e., the random exploration with obstacle avoidance.

In comparison to deliberative exploration, reactive exploration is simple because it does not need the complex process of map building. It can perform well in complex environments and with a large number of robots. However, the main drawback is the occurrence of local minima in the potential field, which may trap the robot and block the exploration process. In this sense, Julia et al. [**Julia et al. 2008**] proposed a technique that enables to detect and escape from these situations by analyzing the potential field generated by the combination of behavioral forces at the robot's neighborhood. Once the local minimum is detected, a technique to force the robot to escape from this point is necessary. A solution can be to plan a path to a frontier cell [**Lau 2003**].

A more reactive but less efficient solution is using a wall-following strategy [**Xiaoping et al. 1997**]; this algorithm switches to a wall-following control mode when the robot falls into a local minimum and switches back to the potential field guided control mode when a certain condition is met. To make a control free of local minima Harmonic functions can be used. However, given that they need to evaluate a global potential field, the technique is computationally expensive. Garrido et al. [**Garrido et al. 2008**] use a similar technique based on the Voronoi Fast Marching method where the robot is directed to the most unexplored areas and where a collision avoidance algorithms is not necessary.

The main disadvantage of the reactive exploration is that complete coverage of the environment cannot be guaranteed. This is because the robot cannot remember the covered area without using a map.

2.3 Integrated Exploration

As we have said in section 2.2, the generic SLAM problem consists of an autonomous system trying to build a map of an unknown environment while simultaneously localizing itself with respect to the map that is been built. This idea however, lack of movement control that guides the robot toward promising areas of the environment that have not yet been integrated into the map.

Having this in mind, it is possible to add to the basic idea of SLAM a motion planner. This adds further to the complexity of the problem giving rise to the Integrated Exploration problem [**Makarenko et al. 2002**] often referred to as SPLAM (Simultaneous Planning Localization and Mapping). In fact, the planning problem on its own is computationally quite intractable under uncertainty. This has been a topic

of interest to multiple communities such as Artificial Intelligence, Control Theory, and Operations Research.

In the integrated exploration approach, the movements of the robot are incrementally planned in order to maximize the information gain and also to increase the possibility of localization (which in long term is obviously related to the information gain).

In general, SLAM algorithms are independent on the exploration algorithm or the motion policy. However, a fully autonomous robot requires to consider SLAM results in its navigation policy.

In the context of the Integrated Exploration, early contributions can be traced back to the work of Feder et al. [**Feder et al. 1999**]. In their work an adaptive motion control technique in SLAM is reported, the robot creates a map and localizes itself simultaneously while making local decisions on where to move next in order to maximize the information obtained and to minimize the error in estimates of the vehicle pose and the landmark locations where the inverse of the estimation error covariance is used as an optimization objective. Bennett et al. use this principle in [**Bennet et al. 2000**] applied to the problem of underwater exploration. Here, the motion command is incorporated into a general behavior based architecture to minimize the vehicle pose and map error.

Makarenko et al. introduce in [**Bourgault et al. 2002**] an integrated exploration method based on the EKF which models the map building and exploration task using an occupancy grid. The adaptive sensing strategy adopted in this work seeks to maximize the expected Shannon-Information gain on the occupancy grid map while simultaneously minimizing the uncertainty of vehicle location in the SLAM process. This approach made the assumption that the robot would observe all the landmarks any time for simplicity, which limited the approach to be applied in large environments. In [**Makarenko et al. 2002**] the same authors have weighted the costs of map exploration, robot position localization and navigation according to their corresponding utility functions under the trade-off of pose uncertainty and mapping. Here the integrated exploration technique is based on a frontier approach using distinguishable special landmarks in the environment. So their algorithm is not appropriate for irregular, unknown environments that cannot be landmarked.

In [**Stachniss et al. 2003**], Stachniss *et al.* develop an integrated exploration strategy conducted by a frontier-based exploration and information gain through entropy minimization strategies. Their algorithm used a grid-based version of the FastSLAM

method where robots dynamically update the probability of coverage status of a region using a Bayesian model. Here, the uncertainty about the pose of the robot can be more accurately reasoned by using this representation and it is taken into account during the whole exploration process. This method proposes one of the few global exploration algorithms where the information available at all locations in the environment is explicitly computed.

Newman *et al.* [**Newmant et al. 2003**] proposed an exploration approach in the context of Bosse's ATLAS [**Bosse et al. 2004**]. Here the robot builds a graph-structure to represent visited areas, and planned motion is motivated by the geometric, spatial, and stochastic characteristics of the current map. Each feature within the map is responsible for determining nearby unexplored areas. They assumed that the location of the features is uncertain and represented by a set of probability functions, which are used in conjunction with the robot path history to determine a robot trajectory suited for exploration.

Sim shows in [**Sim 2005**] a path planning for SLAM with bearings-only sensors. In this work it is proposed to encourage coverage by adding a predefined number of uniformly distributed unvisited dummy features as vague priors in unexplored areas. The path planning policy is based on Voronoi graph with assumptions of perfect data association and unlimited sensor field of view to enhance the stability. The issues related to the initialization of landmarks, which is a key issue in bearing-only SLAM, is not considered in the proposed path planning technique. Yet, this strategy is not effective for systems with short planning horizons and limited sensing as the dummy features will not influence the robot's decision if they are not visible within the planning horizon.

In [**Sim et al. 2005**] Sim carried out the integrated exploration by discretizing the environment into a grid and assuming that the approximate locations of all the features are available at the beginning, thus replanning is not that critical. In this work, the author approximately expressed the state of the EKF as the estimated position of the robot and the trace of the covariance matrix, which reduces the computational cost significantly and makes global planning possible. However it is assumed that the robot is quasi-holonomic with unlimited sensing and robot motion constraints are not considered in the planning process.

In [**Sim 2005b**], Sim presents an approach to information-driven exploration for SLAM which focuses in to overcome the stability issues. In this approach the robot is driven to a globally optimal position for maximizing information gain of the features.

The filter instability is solve by using a virtual minimum range sensor which blocks features that are too close to the robot and that could cause this instability.

Davison et al. [**Davison et al. 2002**] proposed a SLAM framework solution using active vision for real-time. Assuming that the robot trajectory was given, they controlled the active head's movement and sensing on a short term tactical basis, making a choice between a selection of currently visible features. The stereo head is controlled considering uncertainty-based measurement selection, automatic map-maintenance, and goal-directed steering. Persistent features re-detected after lengthy neglect could be re-matched, even if the area was passed through along a different trajectory or in a different direction.

Vidal et al. [**Vidal-Calleja et al. 2006**] considered a single hand-held camera performing SLAM at video rate with generic 6 DOF. They optimized both the localization of the sensor and building of the feature map by computing the most appropriate control actions or movements. The actions belong to a discrete set and are chosen to maximize the mutual information gain between posterior states and measurements.

Leung et al. [**Leung et al. 2008**] considered the trajectory planning problem for line-feature based SLAM in structured indoor environments. In this work, the incremental smoothing and mapping iSAM [**Kaess et al. 2007**] is used to estimate the robot poses and line features whose results are used to efficiently map structured environments through a Model predictive Control with an attractor to optimize the information gain, aid exploration and to incorporate long term planning.

Recently, Juliá et al. in [**Julia et al. 2010**] have presented a hybrid reactive/deliberative approach to the multi-robot integrated exploration problem. In this work, an auxiliary low resolution grid map to represent the free, occupied or unknown state of the space is used. The process consist of a centralized SLAM, which builds the maps and obtains the localization and two processes of exploration (one deliberative and one reactive) per each robot running concurrently.

The reactive exploration is the combination of several basic behaviors that include common behaviours as "go to frontier avoid obstacles" or "go to gateway". This layer operates only with cells within the expected safe zone which is a set of free or unknown cells that can be joined by a straight line without intersect any obstacles. Simultaneously, the deliberative exploration makes the decision between exploring the current expected safe zone, travelling to past poses using the active localization

state or travelling to a gateway cell which is a free cell within the expected safe zone next to a free cell not belonging to this zone.

Furthermore, the model takes into account the degree of localization of the robots to return to previously explored areas when it is necessary to recover the certainty in the position of the robots.

In [**Julia et al. 2011**] the same authors present an integrated exploration solution based on behavioral exploration to model a potential field and on a visual SLAM technique to build the map and to localize the robot. A strategy of detection and escape from local minima is used to avoid the problem of local minima in the potential field. As in their other work this method considers returning to previously explored areas when the localization uncertainty is high.

Chapter 3. EKF-SPLAM Algorithm

In this chapter we are going to develop a SPLAM strategy using some well-known tools for each sub-element that forms it.

- At first, for the exploration part we will use the algorithm of deliberative exploration SRT mentioned in the chapter 2. This tool will help the system to find the next position to explore and will allow us to have a real automatic system.

- Next, for the SLAM algorithm, two EKF methods will be implemented. In first place a classic EKF that will consider only landmarks such as point fixes on the walls, spikes and corners. After an EKF more complex is considered called B-Spline EKF presented by Pedraza et al. [**Pedraza et al. 2007**]. Here the landmarks have the form of B-splines.

3.1 Planning exploration

One of the main tasks for the problem of SPLAM is to travel across the environment in order to build a map in a really autonomous way. For this, a strategy of motion control for exploration most be considered.

Following this thought, in this sub-section we have used one powerful tool known as Sensor-base Random Tree (SRT) presented by Oriolo et al in [**Oriolo et al 2004**].

3.1.1 The SRT method

The SRT method is based on the random generation of robot configurations within a local safe area detected by the sensors. These configurations are stored in a data tree structure that represents the roadmap of the explored area and the associated security region (SR). Each node of the tree (T) consists of a robot's position and its associated local security region (LSR) that is constructed through the perception of the robot system. This LSR is an estimate of the free space surrounding the robot at a given configuration. Two strategies of SRT have been presented by the authors and their difference lie in the form of the LSR and in the way of handle them.

For the first approach, the LSR is a ball and has a conservative attitude appropriated to noise or low resolution sensor. This version is called SRT-Ball (figure 3.1a). The second strategy is called SRT-Star (figure 3.1b). Here the shape of the LSR reminds us a star. The SRT-Star involves a perception strategy that takes all the information reported by the sensors and exploits them in all directions. The star form of the LSR is formed by several “cones” with different radius. Here the $k\text{-}th$ cone radius can be the range minimum between the closest obstacle and the robot or if any object is detected, the radius will be the maximum distance of the sensor. Thus for calculate the radius r of a random direction θ_{rand} we have to identify the corresponding cone of that particular direction. In the other hand, the conservative perception of the SRT-ball ignores the information directional granted by most sensorial systems.

A third strategy for the exploration SRT was proposed by Espinoza et al. in [**Espinoza et al. 2007**] called SRT-radial, in this paper the form of the proposed LSR in absence of obstacles is a circumference. Here, once generated the direction to explore θ_{rand}, the radius is the distance from the robot to the edge of the LSR in that particular direction. Of course, in obstacle presence, the form of the LSR will be deformed and the radius will be different in different directions (Figure 3.1c and d).

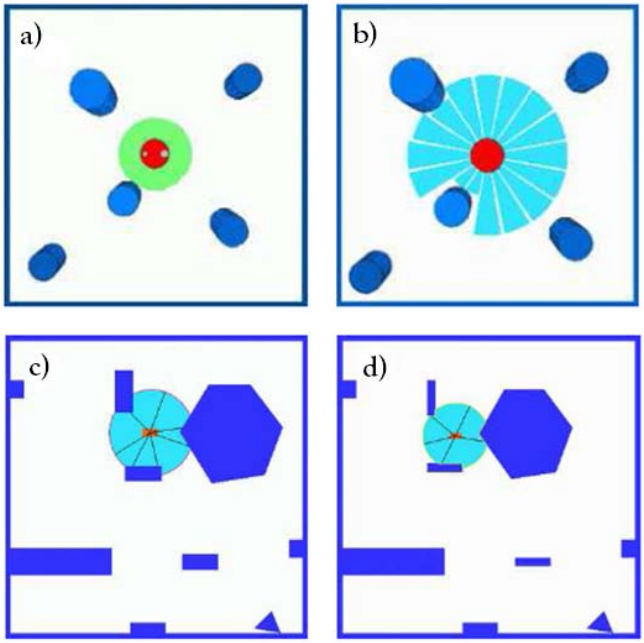

Figure 3.1 a) Safe local region S obtained with the strategy of SRT-Ball. b) Local security region S obtained by the SRT-Star perception system. Notice that the extension of S in some cones is reduced by the obstacle presence. c) and d) Different radius obtained in the safe local region S with the SRT-Radial perception's strategy. [Espinoza et al. 2007].

The figure 3.2 shows how the SRT method works. At the beginning of each iteration, the algorithm gets the LSR associated with the current configuration of the robot, q_{curr}. Once the LSR is obtained, the function **EXTEND_TREE** is responsible for updating the tree, adding the robot position and its corresponding LSR to each node. At the same time, S will store the environment features and will be updated with new features extracted from the LSR that are not parts of the environment yet. This process is performed by the procedure **UPDATE**.

The next step is to process the local boundary F, i.e. to identify obstacles and free areas. Generally, F is a collection of discrete arcs. After obtaining these boundaries and if there are still free zones, the procedure **RANDOM_DIR** will generate random direction in order to choose one that meets the characteristics of a free arc, then, it will generate a q_{curr} configuration taking a step of α length in the direction θ_{rand}. The step size α is chosen as a fixed fraction of the radius of the LSR in that particular direction. Due to the shape of S, q_{cand} will be free of collision. If no border arc is free, then the robot will return to the position of the parent of q_{curr} and the exploration cycle will start again.

Once the q_{cand} configuration is obtained, the **VALID_CONF** procedure will ensure that

this new configuration is valid, i.e. that this new position is outside of the LSRs of the other nodes in the tree. If this new configuration is valid, it will be the new destination q_{dest} that the robot should achieve. On the contrary, if after a maximum number of attempts it is not possible to find a q_{cand} configuration, the parent´s node will be the new configuration q_{dest} (i.e., the robot will return to the father of the current node's configuration).

INTEGRATED_EXPLORATION (q_{init}, k_{max})

```
1.  q_act ← q_init;
2.  for k=1 to K_max
3  .      S ← LSR(q_act);
4.        T ← EXTEND_TREE(q_act,S,T);
5.        Amb_BS ← UPDATE (S);
6.        F ← FORNTIER(q_act,S) ;
7.        if F≠0
8.           i ← 0;
9.           VALID ← FALSE;
10.          While ((i<MAX_ITER) && (!VALID))
11.                θ_rand ← RANDOM_DIR(F);
12.                q_cand ← DISPLACE(q_act, θ_rand);
13.                VALID ← VALID_CONF(q_cand);
14.                i++;
15.          end
16.          if (VALID )
17.                q_dest ← q_cand;
18.          else
19.                q_dest ← q_act.parent;
20.                if q_dest=NULL
21.                      return [T,Amb_BS];
22.                end
23.          end
24.       else
25.          q_dest ← q_act.parent;
26.          if q_dest=NULL
27.                return [T,Amb_BS];
28.          end
29.       end
30.       MOVE_TO(q_dest, q_act, Amb_BS );
31.       q_act ← q_dest;
32. end
33. return [T,Amb_BS];
```

Figure 3.2 SRT-based integrated exploration algorithm

After the configuration q_{dest} is obtained, the function **MOVE_TO** (Figure 3.3) allows the robot to move to this configuration. The process is performed by looking at the list of control inputs (*list_U*), an input $u_{control}$ that allows the robot to approach q_{dest} from

the position q_{curr} (**BEST_U** function). Once choosing the best input $u_{control}$, this is applied to the robot.

At this point, the odometric position, and the increase in X, Y and θ between the previous and the current odometric positions (ΔX, ΔY, $\Delta\theta$) are obtained. The information reported by the robot will be essential to get the estimated position that will be used by the **LOCALIZATION** method to obtain the real position. The algorithm is repeated until the q_{curr} and q_{dest} configurations are the same.

MOVE_TO (q_{act} ,$q_{dest,}$, Amb_BS**)**

1. **while** q_{act}!= q_{dest}
3. $u_{control}$ ← **BEST_U**(List_U,$q_{act,}$ q_{dest});
4. ROBOT ← $u_{control}$;
5. $\hat{q}$ ← **ODOMETRY**;
6. q_{act} ← **LOCALIZATION**($\hat{q}$, $q_{act,}$ Amb_BS);
7. **end**
8. **return** q_{act}

Figure 3.3 MOVE_TO method form the SRT-based integrated exploration algorithm

3.2 EKF-SLAM Classic

The exploration of unknown environments requires an additional functionality because the odometric information reported by the robot, in most cases is not accurate, resulting in inaccurate maps useless for future navigations. The proposed algorithm assumes that the robot's initial position is well located and, consequently, the first observation of the environment has a perfect location. Once the robot has moved from a position q_{last} to a position q_{curr}, the new position of the robot is obtained by adding to the last located position, the increments ΔX, ΔY and $\Delta\theta$ reported by the robot's odometric system. After this position is estimated, the robot will collect the information of the surrounding environment for the localization process.

3.2.1 Review on EKF

As we have said in the small presentation of the EKF in chapter 2, the Kalman filter addresses the general problem of trying to estimate the state $X \in R^n$ of a discrete-time controlled process governed by a linear stochastic difference equation. But what happens if the process to be estimated and (or) the measurement relationship to the

process is non-linear? Some of the most interesting and successful applications of Kalman filtering have been such situations. A Kalman filter that linearizes about the current mean and covariance is referred to as an extended Kalman filter or EKF.

Let us assume that our process has a state vector $X \in R^n$, but that the process is now governed by the non-linear stochastic difference equation:

$$X_k = f\left(X_{k-1}, u_{k-1}, w_{k-1}\right) \qquad (3.1)$$

With a measurement $z \in R^n$ that is

$$z_k = h\left(X_k, v_k\right) \qquad (3.2)$$

Where the random variables w_k and v_k represent the process and measurement noise. In this case the non-linear function f in (3.1) relates the state at the previous step $k\text{-}1$ to the state at the current time step k. It includes as parameters any driving function u_{k-1} and the zero-mean process noise w_k. The non-linear function h in the measurement equation (3.2) relates the state X_k to the measurement z_k.

In practice of course one does not know the individual values of the noise w_k and u_k at each time step. However, one can approximate the state and measurement vector without them as:

$$\tilde{X}_k = f(\hat{X}_{k-1}, u_{k-1}, 0) \qquad (3.3)$$

and

$$\tilde{z}_k = h(\tilde{X}_k, 0) \qquad (3.4)$$

Where $\hat{X}_k^-$ is some a posteriori estimate of the state (from a previous time step $k\text{-}1$).

It is important to note that fundamental flaw of the EKF is that the distributions (or densities in the continuous cases) of the various random variables are no longer normal after undergoing their respective nonlinear transformations. The EKF is simply and ad hoc state estimator that only approximates the optimality of Bayes' rule by linearization.

To estimate a process with non-linear difference and measurement relationships, we

begin by writing new governing equations that linearize and estimate about (3.3) and (3.4).

$$X_k \approx \tilde{X}_k + A(X_{k-1} - \hat{X}_{k-1}) + Ww_{k-1} \quad (3.5)$$

$$z_k \approx \tilde{z}_k + H(X_k - \hat{X}_k^-) + Vv_k \quad (3.6)$$

Where

- X_k and z_k are the actual state and measurement vectors.
- $\tilde{X}_k$ and $\tilde{z}_k$ are the approximate state and measurement vectors from (3.3) and (3.4).
- $\hat{X}_k$ is a posteriori estimate of the state at step k.
- The random variables w_k and v_k represent the process and measurement noise.
- A is the Jacobian matrix of partial derivatives of f with respect to X.
- W is the Jacobian matrix of partial derivatives of f with respect to w.
- H is the Jacobian matrix of partial derivatives of h with respect to X.
- V is the Jacobian matrix of partial derivatives of h with respect to v.

For simplicity, we do not use the time step subscript k with the Jacobians A,W, H and V , even though they are in fact different at each time step.

Now we define a new notation for the prediction error:

$$\tilde{e}_{xk} \equiv X_k - \tilde{X}_k \quad (3.7)$$

and the measurement residual,

$$\tilde{e}_{zk} \equiv z_k - \tilde{z}_k \quad (3.8)$$

In practice, one does not have access to X_k in (3.7), it is the *actual* state vector, i.e., the quantity one is trying to estimate. On the other hand, one does have access to z_k in (3.8), it is the actual measurement that one is using to estimate X_k. Using (3.7) and (3.8) we can write governing equations for an *error process* as:

$$\tilde{e}_{xk} \approx A(X_{k-1} - \hat{X}_{k-1}) + \varepsilon_k \qquad (3.9)$$

$$\tilde{e}_{zk} \approx H(\tilde{e}_{xk} - \eta_k) \qquad (3.10)$$

Where ε_k and η_k represent new independent random variables having zero mean and covariance matrices WQW^T and VRV^T, Q and R are the process noise covariance and the measurement noise covariance respectively.

We can note that the equations (3.9) and (3.10) are linear, this motivates us to use the actual measurement residual $\tilde{e}_{zk}$ in (3.8) and a second (hypothetical) Kalman filter to estimate the prediction error $\tilde{e}_{xk}$ given by (3.9). This estimate, call it $\hat{e}_k$, could then be used along with (3.7) to obtain the a posteriori state candidate estimates for the original non-linear process as

$$\hat{X}_k = \tilde{X}_k + \hat{e}_k \qquad (3.11)$$

Random variables in (3.9) and (3.10) have approximately the following distributions: $p(\tilde{e}_{xk}) \sim N(0, E[\tilde{e}_{xk}\tilde{e}_{xk}^T])$, $p(\varepsilon_k) \sim N\left(0, E\left[0, WQ_kW^T\right]\right)$, $p(\eta_k) \sim N\left(0, E\left[0, VR_kV^T\right]\right)$. Given these approximations and letting the predicted value of $\hat{e}_k$ be zero, the Kalman filter equation used to estimate $\hat{e}_k$ is:

$$\hat{e}_k = K_k\tilde{e}_{zk} \qquad (3.12)$$

By substituting (3.12) back into (3.11) and making use of (3.8) we see that we do not actually need the second (hypothetical) Kalman filter:

$$\hat{X}_k = \tilde{X}_k + K_k\tilde{e}_{zk} = \tilde{X}_k + K_k(z_k - \tilde{z}_k) \qquad (3.13)$$

Equation (3.13) can now be used for the measurement update in the extended Kalman filter, with $\hat{X}_k$ and $\tilde{z}_k$ coming from (3.3) and (3.4), and K_k is the Kalman gain with the appropriate substitution for the measurement error covariance.

The complete set of EKF equations is shown below. Note that we have substituted $\hat{X}_k^-$ for $\tilde{X}_k$ to remain consistent with the earlier “super minus” a priori notation, and the that we now attach the subscript k to the Jacobians A,W,H and V to reinforce the

notion that they are different at (and therefore must be recomputed) at each time step.

$$\hat{X}_k^- = f(\hat{X}_{k-1}, u_{k-1}, 0) \qquad (3.14)$$

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \qquad (3.15)$$

As with the basic discrete Kalman filter, the time update equations (3.5, 3. 6) project the state and covariance estimates from the previous time step *k-1* to the current time step k. Again f in (3.14) comes from (3.3), A_k and W_k are the process Jacobians at step k, and Q_k is the process noise covariance at step k.

$$K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + V_k R_k V_k^T \right)^{-1} \qquad (3.16)$$

$$\hat{X}_k = \hat{X}_k^- + K_k(z_k - h(\hat{X}_k^-, 0)) \qquad (3.17)$$

$$P_k = \left(I - K_k H_k \right) P_k^- \qquad (3.18)$$

As with the basic discrete Kalman filter, the measurement update equations (3.16, 3.17 and 3.18) correct the state and covariance estimates with the measurement z_k. Again h in (3.17) comes from (3.4), H_k and V are the measurement Jacobians at step k, and R_k is the measurement noise covariance at step k. The subscript R changes with each measurement. The next figure offers a complete picture of the operation of the EKF.

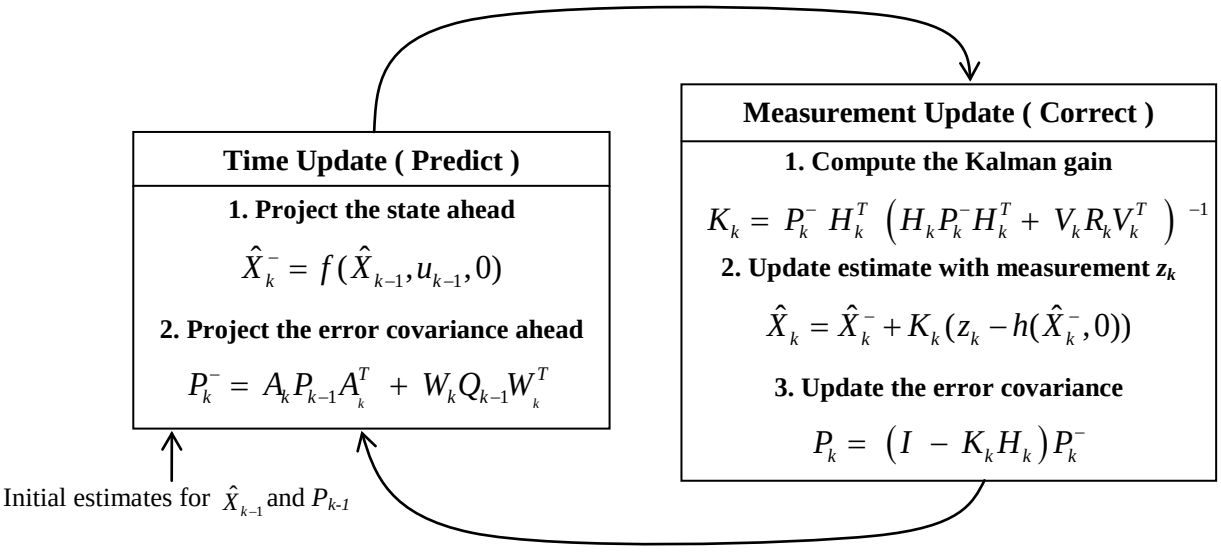

Figure 3.4 A complete picture of the operation of the extended Kalman filter

The algorithm of Extended Kalma Filter is described in the following figure:

Algorithm of Extended Kalman Filter ($\hat{X}_{k-1}, P_{k-1}, u_k, z_k$)

1. $\hat{X}_k^- \leftarrow f(\hat{X}_{k-1}, u_k, 0)$
2. $P_k^- \leftarrow A_k\ P_{k-1}\ A_k^T + W_k\ Q_{k-1} W_k^T$ **//Prediction**
3. $d_k \leftarrow z_k - h(\hat{X}_k^-, 0)$
4. $D_k \leftarrow H_k\ P_k^- H_k^T + V_k R_k V_k^T$
5. $K_k = P_k^- H_k^T D_k^{-1}$
6. $\hat{X}_k = \hat{X}_k^- + K_k d_k$
7. $P_k = \left(I - K_k H_k\right) P_k^-$ **// Correction**
8. **return** ($\hat{X}_k, P_k$)

Figure 3.5. EKF algorithm

The EKF algorithm is described in figure 3.5, It accepts as input the located position $\hat{X}_{k-1}$ and the covariance P_{k-1} , the control u_k and the measurement z_k. Its outputs are the estimations $\hat{X}_k$ and P_k at the instant k. This algorithm is implemented in two steps:

1. **The prediction step** (Lines 1 and 2). Calculates the predicted belief $\hat{X}_k^-$ and P_k^-. This belief is obtained by incorporating the control u_k before to incorporate the measurement z_k.

2. **The correction step** (Lines 3 to 8). The re-observed landmarks are considered. Using the estimate of the current position it is possible to estimate where the landmark should be. There is usually some difference, this is called the innovation. So the innovation is basically the difference between the estimated robot position and the actual robot position, based on what the robot is able to see.

   The variable d_k at the line 3 is called the measurement innovation (or residual). Like we have said, it corrects the discrepancy between the predicted measurement $h(\hat{X}_k^-, 0)$ and the actual mesurement z_k. Line 4 calculates the

innovation (or residual) covariance D_k. Line 5 computes a variable K_k that is called Kalman gain. It specifies to what extent the innovation should be taken into account in the posterior state estimate. The new mean of the posterior belief is calculated in line 6, by adjusting it in proportion to the Kalman gain K_k and the innovation d_k.

Finally, the new covariance of the posterior belief is determined in line 7. If the algorithm is implemented accurately, the initials values $\hat{X}_0^-$ and P_0 may reflect accurately the distribution of the initial state, so, we have some properties:

$$E\left(X_k - \hat{X}_k\right) = E\left(X_k - \hat{X}_k^-\right) = 0 \qquad (3.19)$$

$$E\left(d_k\right) = 0 \qquad (3.20)$$

Equations (3.19) and (3.20) reflect that expected values (mean errors) of all estimates are zero.

3.2.2 Application of EKF to robot localization

So far we have talked about the EKF in a very general way. From this point, we will talk about the concrete implementation of an algorithm of localization based on the extended Kalman filter for SLAM.

In this work we use a differential robot, so our equation of movement will have the following form:

$$\begin{pmatrix} x_k \\ y_k \\ \theta_k \end{pmatrix} = \begin{pmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{pmatrix} + \begin{pmatrix} \Delta t * \tilde{v}_k * \cos(\theta_{k-1} + \tilde{\omega}_k) \\ \Delta t * \tilde{v}_k * \sin(\theta_{k-1} + \tilde{\omega}_k) \\ \Delta t * (\theta_{k-1} + \tilde{\omega}_k) \end{pmatrix} \qquad (3.21)$$

Where $\tilde{v}_k$ and $\tilde{\omega}_k$ denote the true translational and rotational velocity generated by the motion control $u_k = (v_k, \omega_k)^T$ with added Gaussian noise.

$$\begin{pmatrix} \tilde{v}_k \\ \tilde{\omega}_k \end{pmatrix} = \begin{pmatrix} v_k \\ \omega_k \end{pmatrix} + \begin{pmatrix} \varepsilon_{\sigma_1^2} \\ \varepsilon_{\sigma_2^2} \end{pmatrix} \qquad (3.22)$$

where $\varepsilon_{\sigma_1^2}$ and $\varepsilon_{\sigma_2^2}$ are independent Gaussian error variables with zero-mean and standard deviations σ_1 and σ_2 respectively relative to the control velocities v and ω. Therefore, the motion model can be decomposed into a noise-free model with a random Gaussian noise.

$$\underbrace{\begin{pmatrix} x_k \\ y_k \\ \theta_k \end{pmatrix}}_{X_k} = \underbrace{\begin{pmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{pmatrix} + \begin{pmatrix} \Delta t * v_k * \cos(\theta_{k-1} + \omega_k) \\ \Delta t * v_k * \sin(\theta_{k-1} + \omega_k) \\ \Delta t * (\theta_{k-1} + \omega_k) \end{pmatrix}}_{f\,(X_{k-1},\, u_k,\, 0\,)} + N(0, Q_k) \qquad (3.23)$$

This decomposition can be also applied to the perception model. Let $j=c_{ki}$ be the identity of the i^{th} feature observed at time t corresponds to the j^{th} landmark in the map.

$$\underbrace{\begin{pmatrix} r_{ki} \\ \theta_{ki} \end{pmatrix}}_{Z_{ki}} = \underbrace{\begin{pmatrix} \sqrt{(m_{j,x} - x_k)^2 + (m_{j,y} - y_k)^2} \\ \tan^{-1}((m_{j,y} - y_k) / (m_{j,x} - x_k)) \end{pmatrix}}_{h\,(X_k, 0\,)} + N(0, R_k) \quad (3.24)$$

Where $m_{j,x}$ and $m_{j,y}$ denote the coordinates of the i^{th} landmark detected by the robot (that is identical to j^{th} landmark in the map).

The algorithm of localization EKF used in our scheme SPLAM is showed in the algorithm 3.6.

LOCALIZATION_EKF($\hat{q}$, $\hat{X}_{k-1}$, LandM_Amb, P_{k-1})

1. [$\hat{X}_k^-$, P_k^-] ← **PREDICTION**($\hat{q}$, $\hat{X}_{k-1}$, P_{k-1});
2. D ← **SENSOR_DATA**($\hat{X}_k^-$);
3. Ss ← **LANDMARKS_EXTRACTION**(D);
4. [LM_{asoc}, LM_{new}] ← **DATA_ASSOCIATION(**Ss, LandM_Amb**)**;
5. [$\hat{X}_k$, P_k] ← **UPDATE** (LM_{asoc}, q_{est})
6. **return** $\hat{X}_k$, P_k

Figure 3.6 Localization EKF algorithm

This algorithm is derived from the general EKF algorithm of figure 3.5. The inputs are: the posterior belief of the robot pose at time *k-1* ($\hat{X}_{k-1}$), the increases in x, y, θ ($\hat{q}$),

the landmarks stored in the environment, and the covariance P at time *k-1*. The output is a new estimate of the robot pose at time *k* represented by $\hat{X}_k$ and P_k .

The EKF for SLAM as the EKF used for navigation use the steps of prediction and update, however, the EKF for SLAM adds one extra function for the actualization of the map that is been building.

a) PREDICTION

The function **PREDICTION**, computes the necessary Jacobians for linearizing the motion model.

$$A_k = \frac{\partial f(\hat{X}_{k-1}, u_k)}{\partial \hat{X}_{k-1}} \tag{3.25}$$

$$W_k = \frac{\partial f(\hat{X}_{k-1}, u_k)}{\partial u_k} \tag{3.26}$$

Where A_k is the partial derivative of the function of movement with respect to the pose of the robot and W_k is the partial derivative of the function of movement with respect to the control.

Next, the motion noise covariance must be determined as follow:

$$Q_k = \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix} \tag{3.27}$$

Where σ_1 and σ_2 are two variables relative to the control velocities. Now the estimation of the new position of the robot is calculated using:

$$\hat{X}_k^- = f\left(\hat{X}_{k-1}, u_k, 0\right) \tag{3.28}$$

Where $\hat{X}_{k-1}$ is the localizated position at time *k-1*, *f* is the model of our robot and u_k are the control entries. We can add directly to the previous localized position the increases Δx, Δy and $\Delta\theta$ of the previous state of time *k-1* to the state of time *k* reported by the robot:

$$\begin{pmatrix} x_k \\ y_k \\ \theta_k \end{pmatrix} = \begin{pmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{pmatrix} \tag{3.29}$$

Finally the matrix of covariances is updated using the equation (3.15):

$$P_k^- = A_k \, P_{k-1} \, A_k^T + W_k Q_k \, W_k^T$$

b) SENSOR_DATA

After the function of prediction, the function **SENSOR_DATA** will obtain the information of the surrounding environment to the robot caught by the sensors in this instant k. This information will be placed spatially in the current position estimated by the function of prediction.

c) LANDMARKS_EXTRACTION

Taking the gathered information for the function **SENSOR_DATA**, the function **LANDMARKS_EXTRACTION** will be in charge to look for characteristics of the environment that are easily re-observable.

The EKF used in this section is conceived to work in feature maps formed by lines (Figure 3.7a), but before to try to find this type of features we have to find one other kind called break point (Figure 3.7b). This type indicate discontinuities in the scan process and usually occur due to the existence of objects or surfaces that hinder the detection of other elements more distant. The detection of these points allows classifying the measures in groups called “clusters”

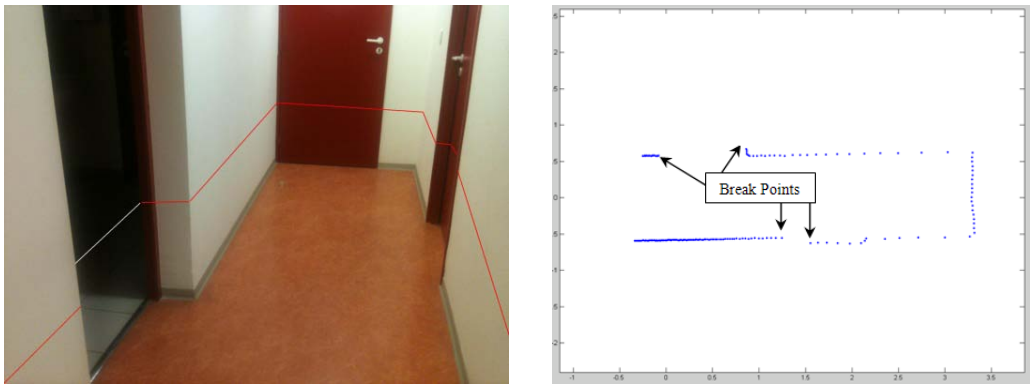

Figure 3.7 a) Environment. b) Break points found in the environment

Several strategies to find these clusters have been proposed in literature. In particular the clustering process used in this section is based on the classic criteria of Dietmayer [**Dietmayer et al. 2001**], whose operation can be explained in the figure 3.8. P_a and P_b represent two consecutive points detected by the laser, while r_a and r_b are the distances of these points to the coordinates origin. Given the triangle OP_aP_b, where r_a and r_b are known and α is the angular resolution of the laser, we can apply the cosines theorem to calculate the distance between P_a and P_b:

$$r_{ab} = \sqrt{r_a^2 + r_b^2 - 2r_a r_b \cos(\alpha)} \qquad (3.30)$$

The criteria used to form the clusters is that, if the distance between P_a and P_b is less than

$$r_{ab} \leq C_0 + C_1 \cdot min\{r_a - r_b\} \qquad (3.31)$$

Where $C_1 = \sqrt{2(1-\cos(\alpha))}$, then P_b belongs to the same cluster than P_a. Otherwise, the points P_a and P_b belong to different clusters. The constant C_0 represents a noise adjustment in the laser measures. The other constant, C_1, takes a value not explained by Dietmayer, but, that can be explained using the figure 3.8, where it can be appreciated that $min\{r_a, r_b\} = ra$, therefore:

$$C_1 \min\{r_a, r_b\} = r_a\sqrt{2(1-\cos(\alpha))} = 2r_a\sqrt{\frac{1-\cos(\alpha)}{2}} \qquad (3.32)$$

On the other side, the variable named z in the figure 3.8 will take the value:

$$z = r_a \sin(\frac{\alpha}{2}) = r_a\sqrt{\frac{1-\cos(\alpha)}{2}} \qquad (3.33)$$

Finally, we get:

$$2z = C_1 \cdot r_a \qquad (3.34)$$

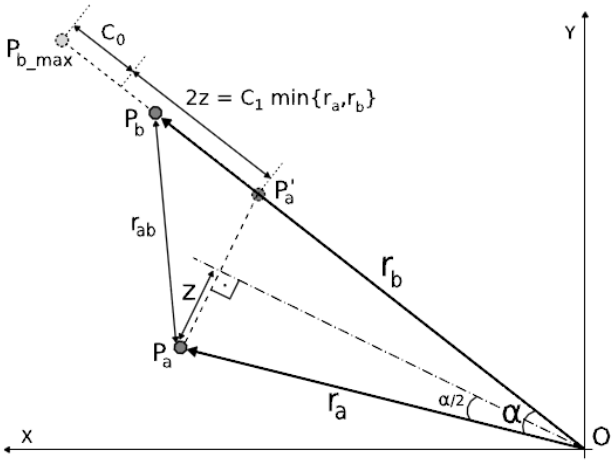

Figure 3.8 Dietmayer's criteria

With the raw data segmented in clusters the next step is to find features in each one of them. As we have already said, this EKF works on feature maps, so given this restriction the system looks essentially for 3 types of landmarks:

1. **SPIKES**. They are identified finding cluster with less of tree laser measures. This type of Landmark is considered by the algorithm once that it has been seen a number I of times, since own mistakes of the measure system might be interpreted like landmarks.

2. **STRAIGHT LINES**. To extract this type of landmarks we have used the work by Pavlidis et al. [**Pavlidis et al. 1974**] named "Split and Merge". The algorithm has two parts. The first phase is recursive, and consists in dividing the available segments into smaller ones, while the second is used to merge segments that are almost colinear (Figure 3.9 and 3.10).

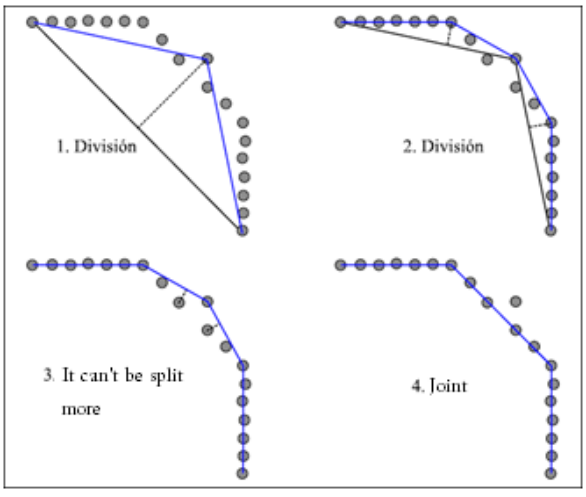

Figure 3.9 Split and Merge Algorithm evolution [Tardos et al. 2002b]

Split and Merge Algorithm

1: **function SPLIT(*P*)**
2: Input: $P \quad \{p_1, \cdots, p_n\}$
3: Put P in the list L.
4: $R \leftarrow$ line that fits the points of P
5: $p_t \leftarrow$ point with the biggest distance T to the line R
6: If $T > T_{max}$ then
7: $P' \leftarrow \{p_1, \cdots, p_t\}$
8: $P'' \leftarrow \{p_t, \cdots, p_n\}$
9: $L \xleftarrow{add}$ **SPLIT**(P')
10: $L \xleftarrow{add}$ **SPLIT**(P'')
11: End If
12: **Return** L

14: **function MERGE(*L*)**
15: Input: $L \leftarrow$ list of sets $\{P_i, \cdots, P_n\}$
16: For $i \leftarrow 1$ to $n - 1$ do
17: If P_i y P_{i+1} are colinear within an error margin then
18: $Q \leftarrow P_i \cup P_{i+1}$
19: $R \leftarrow$ line that fits the points of Q
20: q_t point $\in Q$ with the biggest distance T to the line R
21: If $T \leq T_{max}$ then
22: $L \xleftarrow{eliminate} P_i$ y P_{i+1}
23: $L \xleftarrow{add} Q$
24: End If
25: End If
26: End For
27: **Return** L

Figure 3.10 Split and Merge Algorithm

At the end of this algorithm and due to its nature, every processed cluster will give n sub-clusters of raw data laser measures corresponding to n noncolinear segments. The line contained in these sub-clusters is approximate using the method of least median square (LMS Least Median Square) proposed by Mount et al. in [**Mount et al. 2007**].The decision to use this method is based on a comparative study of different methods proposed in the literature (including RANSAC and its variants (MSAC and NAPSAC), and Least Squares approximation) where the LMS method proved to give more stable results obtaining in all tests conducted a slope almost equal.

3. **CORNERS**. These landmarks will be obtained taking all the straight lines found in the environment and verifying which ones intersect.

d) DATA_ASSOCIATION

A critical aspect of the localization algorithm is the data association. The objective of this function is to match observed features from different scans and to assign measurements from which they originate and reject fake measurements. This process is also known as re-observing landmarks. Given the three types of features used in this method, the **SPIKE** and **CORNER** kind are the easiest to associate. For them, we look for the closers features stored in the system of the same type where the Euclidean distance between them is less than a threshold D_{min}.

In the other hand the Straight lines features most be translated into a fixed point, this is made by taking the robot predicted position and calculating his orthogonal point to the straight lines. This process is done for both lines, the one observed and the line associates stored in the system figure 3.11.

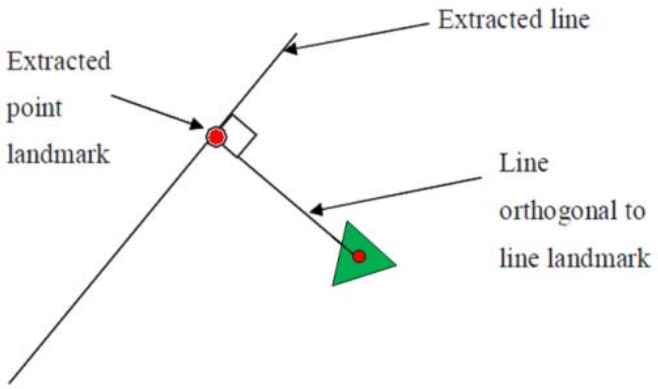

Figure 3.11 Line feature as a point

The measurement-to-feature association is performed using a gating approach in the innovation space incorporating both measurement uncertainty and robot uncertainty. So using the innovation matrix:

$$S_{ki} = H_{ki}\begin{bmatrix} P^{rr} & P^{ri} \\ P^{ir} & P^{ii} \end{bmatrix} H_{ki}^{T} + V_k R_k V_k^{T} \qquad (3.35)$$

Where

$$H_{ki} = \frac{\partial h_i(\hat{X}_k, i)}{\partial \hat{X}_k} \qquad (3.36)$$

And defining the innovation $d_{ki} = z_{ki} - h_i\left(\hat{X}_k^-\right)$, we can establish a validation gate to determine a correct association in the form of:

$$d_{ki}^T S_{ki}^{-1} d_{ki} \leq \lambda \qquad (3.37)$$

Where λ is a constant chosen heuristically.

The validation gate uses the fact that our EKF implementation gives a bound on the uncertainty of an observation of a landmark. Thus we can determine if an observed landmark is a landmark in the database by checking if the landmark lies within the area of uncertainty.

e) UPDATE

If a feature present in the state vector is re-observed the update step of the EKF is used to update the state of the map including the robot pose. The model of observation for the characteristic i has the form:

$$Z_{ki} = \begin{bmatrix} \mathrm{r_{ki}} \\ \theta_{\mathrm{ki}} \end{bmatrix} = \begin{bmatrix} \sqrt{\left(\mathrm{m_{j,x}} - \mathrm{x_k}\right)^2 + \left(\mathrm{m_{j,y}} - \mathrm{y_k}\right)^2} \\ \tan^{-1}\left(\dfrac{\mathrm{m_{j,y}} - \mathrm{y_k}}{\mathrm{m_{j,x}} - \mathrm{x_k}}\right) \end{bmatrix} + N(0, R_k)$$

$$= h\left(\hat{\mathrm{X}}_{\mathrm{ki}}\right) + N\left(0, \mathrm{R_k}\right) \qquad (3.38)$$

$$R_k = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix} \qquad (3.39)$$

The noise process $N(0,R_k)$ is assumed to be white Gaussian with covariance R_k. If the N features are observed the observation model becomes:

$$Z_k = \begin{bmatrix} z_{k1} \\ \vdots \\ z_{kn} \end{bmatrix};\quad h_k = \begin{bmatrix} h_{k1} \\ \vdots \\ h_{kn} \end{bmatrix};\quad R_k = \begin{pmatrix} R_{k1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R_{kn} \end{pmatrix} \qquad (3.40)$$

$$H_k = \frac{\partial h(\hat{X}_k)}{\partial \hat{X}_k} \tag{3.41}$$

With this information, we calculate the covariance of the innovation S_k given for:

$$S_k = H_k \, P_k^- \, H_k^T + V_k R_k V_k^T \tag{3.42}$$

Which depicts the uncertainty corresponding to the predicted measurement Z_k. Again with this information we calculate Kalman's profit K_k of the following way:

$$K_k = P_k^- \, H_k^T \, S_k^{-1} \tag{3.43}$$

Finally, the new pose estimate is obtained $\hat{X}_k^-$ and the matrix of covariances P_k^- are updated as it follows:

$$\hat{X}_k = \hat{X}_k^- + K_k(Z_k - h(\hat{X}_k^-, 0)) \tag{3.44}$$

$$P_k = \left(I - K_k \, H_k \right) P_k^- \tag{3.45}$$

3.2.3 Extension of the Map

This section is completely bound to the **UPDATE** function on the SRT algorithm (figure 3.2 lines 5). Achieve the new position to explore q_{dest}, his new LSR will be used to do update the map on the EKF system state X.

a) Add a new feature

When a new feature $L_{new} = [r \ \theta]$ is observed, the new feature state X_{N+1} is incorporated in the system vector state. For CORNER and SPIKES we add just the range and bearing to the feature as:

$$X_{N+1} = m(\hat{X}_k, L_{new}) = \begin{bmatrix} x_{rk} + r\cos(\theta_{rk} + \theta) \\ y_{rk} + r\sin(\theta_{rk} + \theta) \end{bmatrix} \tag{3.46}$$

$$\hat{X}_k \leftarrow \begin{bmatrix} \hat{X}_k \\ X_{N+1} \end{bmatrix} \tag{3.47}$$

But for the STRAIGHT LINE case the new feature have the range and bearing of the star and final point $L_{new} = [r_s\ \theta_s\ r_f\ \theta_f]$, so in this case the new feature has the following form:

$$\mathrm{X}_{\mathrm{N+1}} = m(\hat{X}_k, L_{new}) = \begin{bmatrix} x_{rk} + r_s \cos(\theta_{rk} + \theta_s) \\ y_{rk} + r_s \sin(\theta_{rk} + \theta_s) \\ x_{rk} + r_f \cos(\theta_{rk} + \theta_f) \\ y_{rk} + r_f \sin(\theta_{rk} + \theta_f) \end{bmatrix} \qquad (3.48)$$

Been the 3 first elements in $\hat{X}_k$ the position x,y and θ localized of the robot.

Once the new landmark is added, it only remains to update the matrix of covariances for this new one landmark. Thus, in the first place we added the covariance for the new landmark.

$$P^{N+1\ N+1} = J_{xr}\ P_k{}^{rr}\ J_{xr}{}^T + J_z\ R_k\ J_z{}^T \qquad (3.49)$$

After the robot – landmark covariance for the new landmark is added.

$$P^{rr\ N+1} = P_k{}^{rr}\ J_{xr}{}^T \qquad (3.50)$$

And finally we add the landmark – landmark covariance.

$$P^{N+1\ i} = J_{xr}\ (P_k{}^{ri})^T \qquad (3.51)$$

J_{xr} and J_z are two jacobians of EKF that are used in SLAM. J_{xr} is the as the jacobian of the prediction of the feature and is basically the same as the jacobian of the prediction model except that there is not the rotation term. In the other hand, J_z is also the jacobian of the prediction model for the feature bus with respect to range and bearing. R is the Gaussian noise proportional to the measurement.

b) Feature Extension

When a new LSR is obtained on a new position q_{dest} achieved, the associations of some lines are made only partially (Figure 3.12)

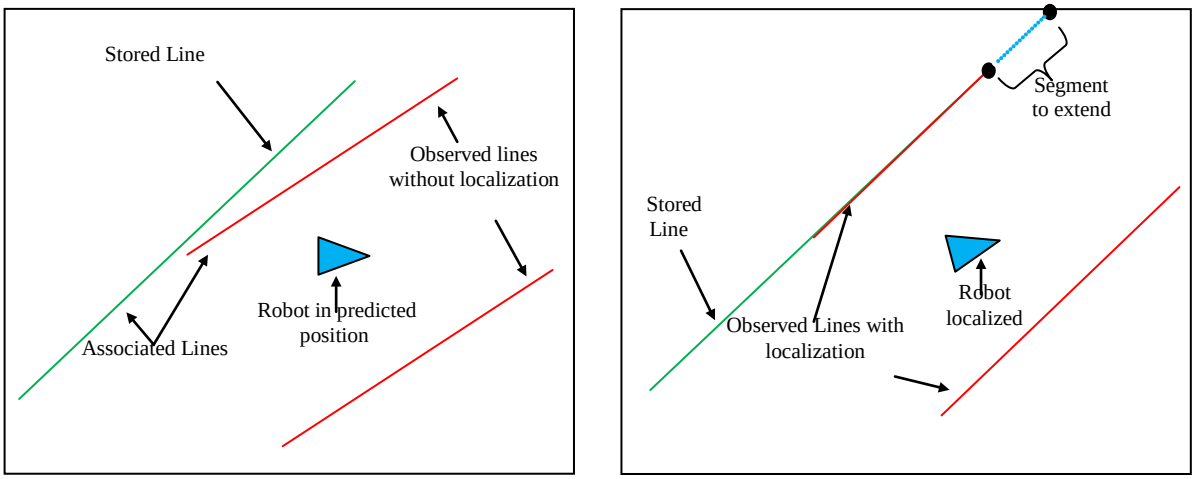

Figure 3.12 Line segment partially associated to be extended

In this case, that line segment has to be extended with the new information. Therefore, to make this extension, in first place we have to extract the k^{th} feature corresponding to the line to be extended as follow:

$$\left[x_i, y_i, x_f, y_f\right] \leftarrow \begin{bmatrix} X_r \\ X_1 \\ \vdots \\ X_k \\ \vdots \\ X_N \end{bmatrix} \tag{3.52}$$

Whit this information, we can easily calculate the elements m and b in the equation of the line.

$$m = \frac{y_f - y_s}{x_f - x_s}; \qquad b = y_s - mx_s \tag{3.53}$$

And to do the extension (in the case of the figure 3.12 toward the extreme right) with the farthest x_{est} and y_{ext} coordinates of this segment.

$$y_{ext} = mx_{ext} + b \tag{3.54}$$

Finally, the new line extended is added to the system vector in the same position from which was extracted.

$$[x_i, y_i, x_{ext}, y_{ext}] \rightarrow \begin{bmatrix} X_r \\ X_1 \\ \vdots \\ X_k \\ \vdots \\ X_N \end{bmatrix} \quad (3.55)$$

3.3 EKF-SLAM with B-Splines

3.3.1 Foundations of B-Splines.

a) B-Splines Definition

Let U be a set of $m + 1$ non-decreasing numbers, $u_0 \leq u_2 \leq u_3 \leq \ldots \leq u_m$. The u_i's are called knots, the set U the knot vector, and the half-open interval $[u_i, u_{i+1})$ the *i-th* knot span. Note that since some u_i's may be equal, some knot spans may not exist. If a knot u_i appears k times (i.e., $u_i = u_{i+1} = \ldots = u_{i+k-1}$, where $k > 1$, ui is a multiple knot of multiplicity k, written as $u_i(k)$. Otherwise, if u_i appears only once, it is a simple knot. If the knot vector does not have any particular structure, the generated curve will not touch the first and last legs of the control polyline as shown in the figure 3.13a and we have an unclamped B-Spline.

We may want to clamp the curve so that it is tangent to the first and the last legs at the first and last control points, respectively. To do so, the first knot and the last knot must be of multiplicity $p+1$, where p is the degree of the curve. This will generate the so-called clamped B-spline curves, figure 3.13b. By repeating some knots and control points, the generated curve can be a closed one. In this case, the start and the end of the generated curve join together forming a closed loop as shown in figure 3.13c.

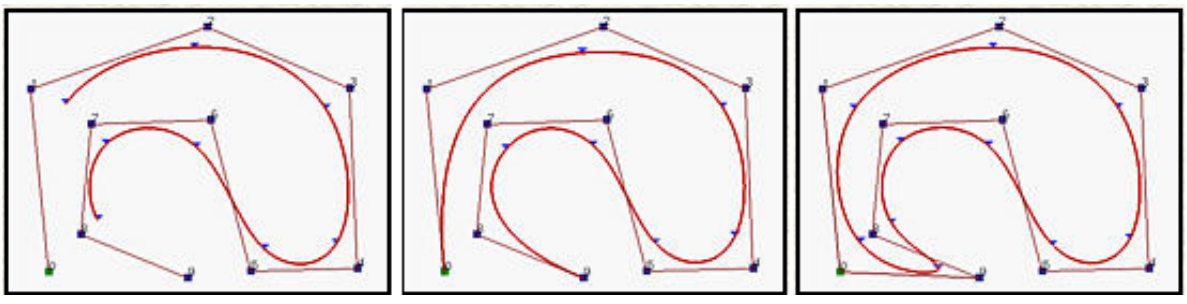

Figure 3.13 a) Unclamped B-Splines. b) Clamped B-Spline. c) Closed B-Spline

If the knots are equally spaced (i.e., $u_{i+1} - u_i$ is a constant for $0 \le i \le m-1$), the knot vector or the knot sequence is said uniform; otherwise, it is non-uniform. The knots can be considered as division points that subdivide the interval $[u_0, u_m]$ into knot spans.

All B-spline basis functions are supposed to have their domain on $[u_0 , u_m]$. To define B-spline basis functions, we need one more parameter, the degree of these basis functions, p. The *i-th* B-spline basis function of degree p, written as $N_{i,p}(u)$, is defined recursively as follows:

$$N_{i,0}(u) = \begin{cases} 1, & if \quad u_i \le u \le u_{i+1} \\ 0, & otherwise \end{cases} \tag{3.56}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{3.57}$$

The above equation is usually referred to as the Cox-de-Boor recursion formula [**de Boor 1978**]. Given $n + 1$ control points $P_0, P_1,\ldots, P_n$ and a knot vector $U = u_0, u_1, \ldots , u_m$, the B-spline curve of degree p defined by these control points and knot vector U is

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u) X_i \tag{3.58}$$

where $N_{i,p}(u)$'s are B-spline functions of degree p. The form of a B-spline curve is very similar to that of a Bézier curve [**Rogers 2001**].

b) Spline Fitting

The simplest method of fitting a set of data points with a B-splines curve is the global interpolation method [**Ishida 1997**]. Suppose we have $n+1$ data points $D_0,D_1, \ldots ,D_n$ and wish to fit them with a B-spline curve of degree p, where $p \le n$ is an input. We select a set of parameter values $t_0, t_1, \ldots , t_n$ (the number of parameters is equal to the number of data points).

Suppose the desired interpolating B-spline curve of degree p is done in the equation (3.58). This B-spline has $n+1$ unknown control points, since parameter t_k corresponds to data point D_k. Plugging t_k into the equation (3.58) yields the following:

$$D_k = C(t_k) = \sum_{i=0}^{n} N_{i,p}(t_k) X_i \quad for \quad 0 \le k \le m \tag{3.59}$$

Because there are $n + 1$ B-spline basis functions in the above equation and $n + 1$ parameters, plugging these t_k's into the $N_{i,p}(u)$'s yields $(n + 1)^2$ values. These values can be organized into a $(n+1)\times(n+1)$ matrix N in which the $k\text{-}th$ row contains the values of $N_{0,p}(u)$,$N_{1,p}(u)$, … , and $N_{i,p}(u)$ evaluated at t_k as shown below:

$$N = \begin{bmatrix} N_{o,p}(t_0) & N_{1,p}(t_0) & N_{2,p}(t_0) & \cdots & N_{n,p}(t_0) \\ N_{o,p}(t_1) & N_{1,p}(t_1) & N_{2,p}(t_1) & \cdots & N_{n,p}(t_1) \\ \vdots & & & \ddots & \vdots \\ N_{o,p}(t_n) & N_{1,p}(t_n) & N_{2,p}(t_n) & \cdots & N_{n,p}(t_m) \end{bmatrix} \tag{3.60}$$

This matrix N, is generally known as ***placement matrix*** and it has for each one of its rows, a maximum of p nonzero elements. In the same way, we can also collect vectors D_k and X_i as follows:

$$D = \begin{bmatrix} D_0 & D_1 & \dots & D_m \end{bmatrix} \tag{3.61}$$

$$X = \begin{bmatrix} X_0 & X_1 & \dots & X_n \end{bmatrix} \tag{3.62}$$

With these representations, we can write the equation (3.59) in a most compact form:

$$D = N\,X \tag{3.63}$$

Given that vector D contains the input data points and matrix N is obtained by evaluating B-spline functions at the given parameters, the only unknown is vector X.

As we can see, the simpler form is a system of linear equations with unknown X, solving for X yields the control points and the desired B-spline interpolation curve becomes available. Therefore, the interpolation problem is solved.

When the problem is over determined, it can be solved in an in a mean sense. This occurs in the most general case when $2 \le p \le n+1 < m+1$. To solve it, we can obtain a least squares solution making use of the pseudoinverse matrix of B:

$$X = \left[B^T B \right]^{-1} B^T D = \Phi D \tag{3.64}$$

3.3.2 EKF with B-Splines

a) Data management

Before the obtained data can be used by the localization algorithm, they need to undergo several processes (Figure 3.14):

- **FIRST SEGMENTATION**. An analysis of the relative position of consecutive data points. The objective is to detect points close enough to belong to the same obstacle. For this segmentation we have used the Dietmayer'criteria presented in section 3.2.2 c.

- **SECOND SEGMENTATION**. The segments obtained in the first segmentation undergo another test to look for straight points whose angle is below a certain threshold. The objective of this segmentation is to detect corners and curves with high curvatures. For this part we used the algorithm "split and merge" presented also in section 3.2.2 to divide every cluster found in the first segmentation into small straight lines segments. After, an analysis of the slopes corresponding to 2 consecutive segments is performed, if the difference between these two slopes is bigger than a certain θ threshold it means that a division most be done. This process is showed in figure 3.15.

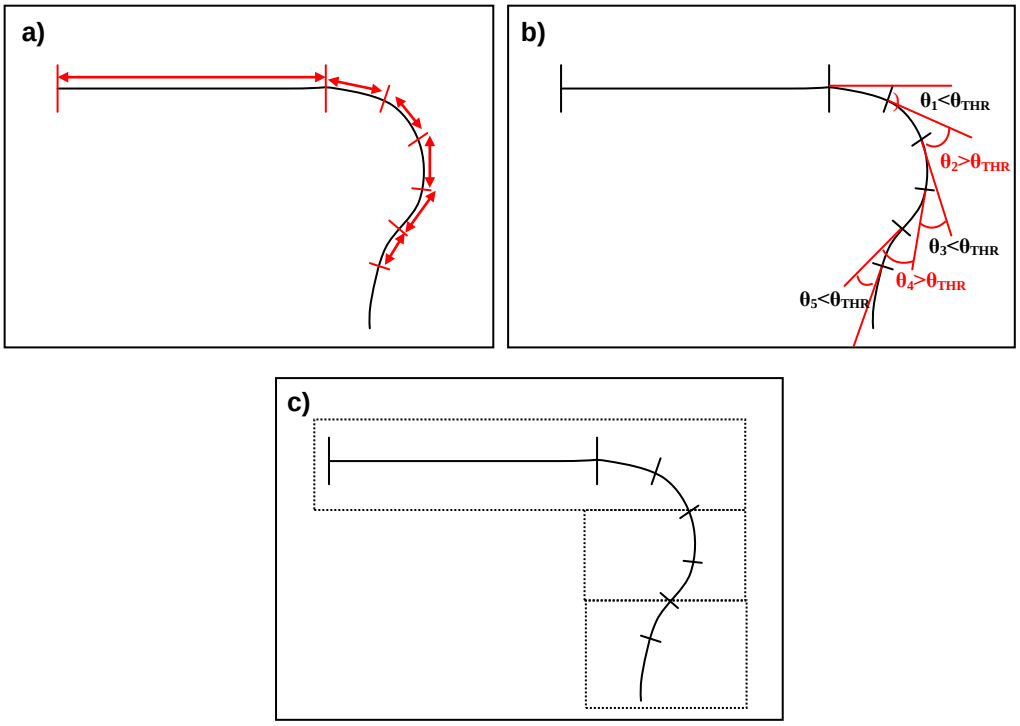

Figure 3.14 a) Line segments found by the "split and merge algorithm". b) Analysis of slopes between adjacent segments. c) Segments obtained after processing.

- **FITTING**. Each of the obstacles of the second segmentation is adjusted to the B-Spline grade 3 that form its control polygons.

The overview of the process performed over the raw data acquired by the sensor is showed in figure 3.15.

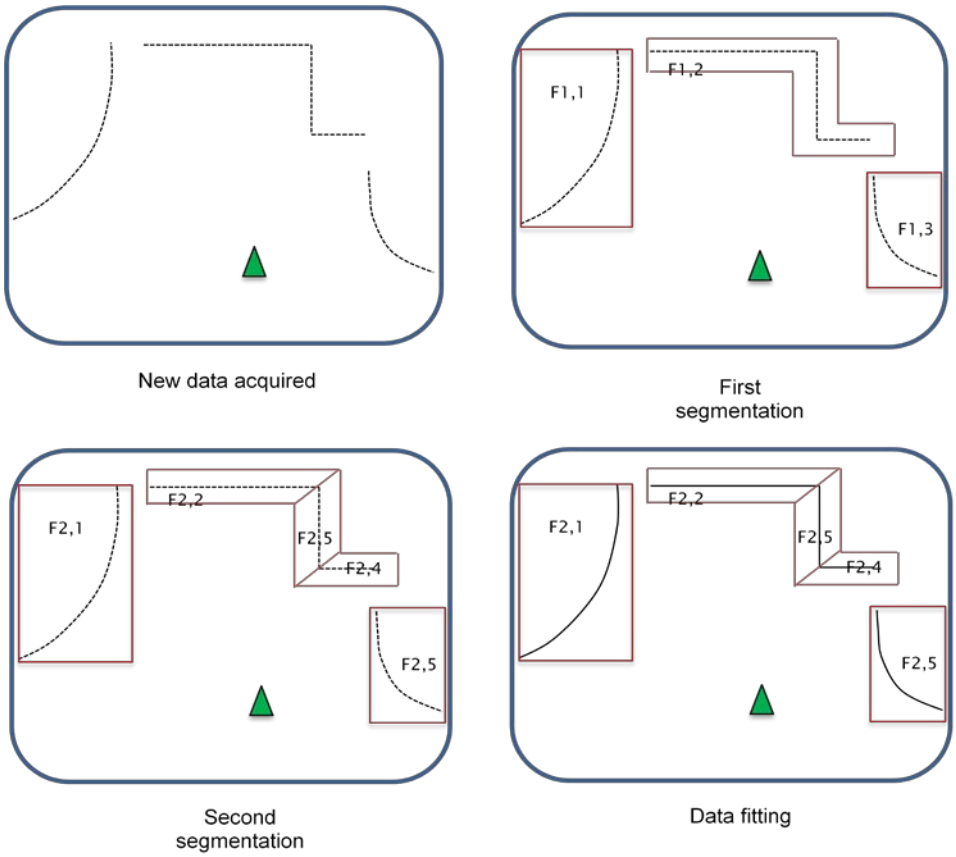

Figure 3.15 Overview of treatment made to the raw data

b) Association of B-Splines

Once the data from the sensor are segmented, a process of data association is performed. The first association is crude, and the control points of each segment obtained in the segmentation process are compared with the control points in the map, using the following criteria:

$$min(dist(X_{m,i}, X_{o,j})) \leq d_{min} \; , \quad \begin{cases} i = 1 \dots n_m \\ j = 1 \dots n_0 \end{cases} \qquad (3.65)$$

Where $X_{m,i}$ and $X_{0,j}$ are the control points of the splines, on the map and on the predicted position, respectively, n_m and n_0 are the number of control points of the splines on the map and on the predicted position, dist($X_{m,i}$, $X_{o,j}$) represents the Euclidean distance between the control points, and finally d_{min} is the parameter that will regulate if the points are or not related.

If no spline in the map is close enough to a detected spline in order to be related, then this new object is added to the map, once the robot's position has been located. By contrast, if a spline is associated with a map's feature, it is necessary to obtain a concordance between its points, as follows:

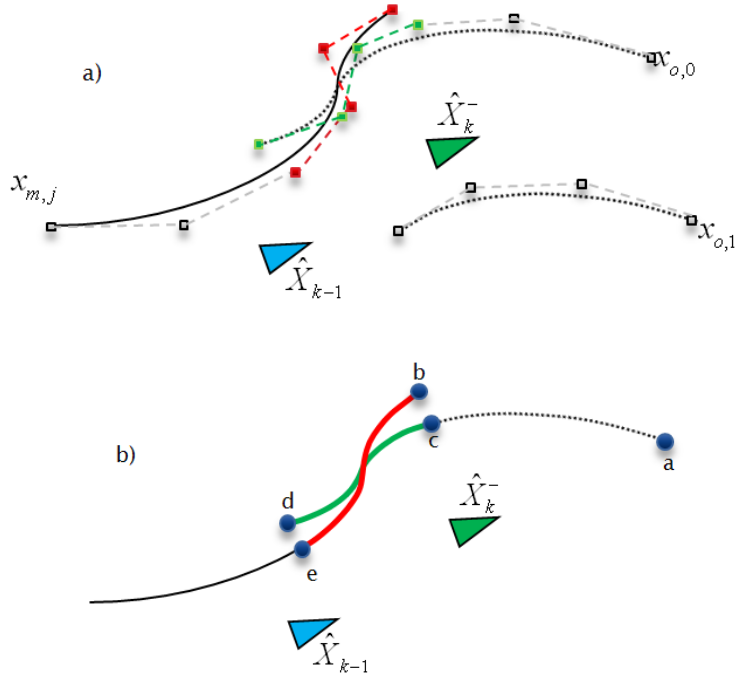

Figure 3.16 Curves Concordance. (a) Rough association. (b) Association fine

- One of the ends of the curve is considered point a.
- The point nearest to point a in the spline on the map is considered point b.
- If b is one of the endpoints of the spline on the map, then, the point nearest to b in the spline is calculated and named point c, if not, point a is associated with point b.
- The process is repeated using the other end of the observed spline as a starting point (point d in the figure 3.16b). This point is associated with the

point e of the spline on the map.

- Thanks to a property of the B-splines about the possibility of knowing the curves' length, eb and dc segments can be adjusted to have the same length. If the difference of the lengths is greater than a threshold l_{max}, the endpoints of the larger curve are eliminated to adjust its size.

c) Model of State

As in the localization EKF showed before, the state of the system in any instant k is composed by the robot's position (considered the only moving object in the environment) and all the map features represented as cubic B-Splines. Now when the splines are expressed as a linear combination of basis functions B-splines, the state of each one of them can be represented by the position of their control points, this is possible if we use a fixed knot vector to generate a basis for all the map B-Spline features. So, referring all positions and orientations to a global reference system $\{u_w, v_w\}$ and considering that the robot is the first element in the map (F_0), the following expressions describe the state of the system composed by the robot pose X_r , and the position of the control points of every feature in the map represented by the vector X_{si} :

$$X_{F0} = X_r = [x_r, y_r, \theta_r]^T \tag{3.66}$$

$$X_{Fi} = X_{si} = [x_{i,0}, \ldots, x_{i,ni}, y_{i,0}, \ldots, y_{i,ni}]^T \quad \text{where} \quad i = 1, \ldots, N \tag{3.67}$$

So, the state of the system can be written as follow:

$$X = \left[X_r^T, X_{s1}^T, \ldots, X_{sn}^T \right]^T \tag{3.68}$$

Been N the number of map features and n_i is the number of control points for each one of them. As we can see, the number of control points in the map for each spline can be variable because they can be extended progressively when new areas of the environment are explored.

The start point for the probabilistic formulation of the estimation problem, is the assumption of that the real state of the system in the instant k is unknown, but can be model with a Gaussian distribution who has all the information at that moment.

$$\hat{X}_k \sim N(\hat{X}_k^-, P_k) \tag{3.69}$$

where

$$\hat{X}_k^- = [\hat{X}_{kr}^- \quad \hat{X}_{ks1}^- \cdots \hat{X}_{ksN}^-] \tag{3.70}$$

And

$$P_k = \begin{bmatrix} P_k^{rr} & P_k^{rs1} & \cdots & P_k^{rsN} \\ P_k^{s1r} & P_k^{s1s1} & \cdots & P_k^{s1sN} \\ \vdots & \vdots & \ddots & \vdots \\ P_k^{sNs1} & P_k^{sNs2} & \cdots & P_k^{sNsN} \end{bmatrix} \tag{3.71}$$

d) Model of Observations.

Like in the section 3.2.2, the use of the EKF for the SLAM problem requires a mathematical expression that allows us to predict the measurement that we expect to get from the robot's sensor given the robot pose and the current knowledge of the environment at that instant.

The model of observation for the case of B-Splines is reduced to find the intersection of the straight line that forms every laser beam with the splines contained in the map figure 3.17.

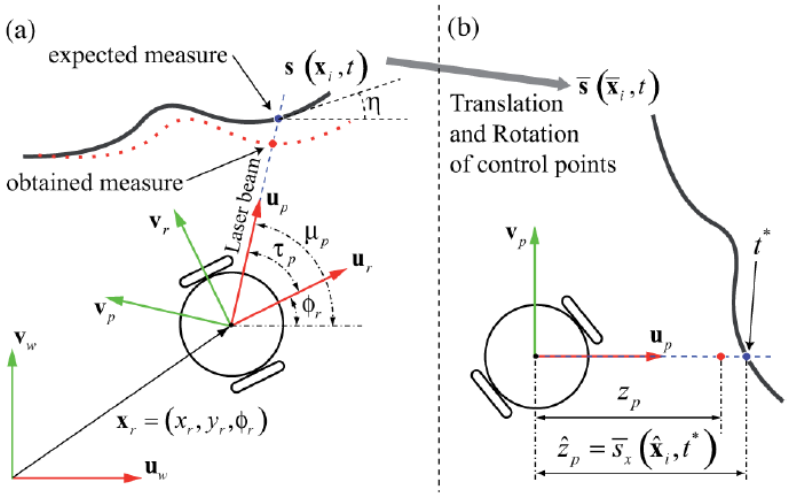

Figure 3.17 Observation model (figure taken from [Pedraza et al. 2007]). The expected intersection of each laser beamacross the angular range of the sensor with the map spline is computed expressing the map spline (a) in the $\{u_p, v_p\}$ reference frame (b).

However, to calculate this intersection with a parametric curve in the form $s(t) = \left[s_x(t), s_y(t) \right]^T$ is not convenient for an explicit mathematical formulation. Given that disadvantage, we use a two steps process that is used iteratively. In first place we apply the property of Affine Invariance which says that when we want to apply an affine transformation to a B-spline curve we can apply the transformation to control points, which is quite easy, and once the transformed control points are obtained the transformed B-spline curve is the one defined by these new points. Therefore, we do not have to transform the curve. Next we use the Newton–Raphson method for calculating the roots of a function.

To begin, we have to define an orthonormal reference frame $\{u_p , v_p\}$ centered in the robot reference frame $\{u_r , v_r\}$, which is defined by the direction and orientation of the sensor beam (Figure 3.17 show one random sensor beam and its intersection with a spline curve). Next, we take $s\ (\overline{x}_i\ (x_i\ , x_r\ )\ , t)$ as the position vector that crosses one curve referred in such system. In this way, the relation between the control points $x_i\ = [x_i\ , y_i]^T$ and $\overline{x}_i\ = [\overline{x}_i\ , \overline{y}_i\ ]^T$ is defined as:

$$\begin{bmatrix} \overline{x}_i \\ \overline{y}_i \end{bmatrix} = \begin{bmatrix} \cos\mu_p & \sin\mu_p \\ -\sin\mu_p & \cos\mu_p \end{bmatrix} \begin{bmatrix} x_i - x_r \\ y_i - y_r \end{bmatrix} \qquad (3.72)$$

Where μ_p is the angle of the considered laser beam in the global reference system. It means that, given the orientation of the laser beam in the frame of the robot, τ, we have:

$$\mu_p\ =\ \theta_r + \ \tau \qquad (3.73)$$

In this context, the measurement prediction $\hat{z}_p = h\ (x_i\ ,\ x_r)$ is given by the value of $\overline{s}_x\ (\overline{x}_i\ (x_i\ ,\ x_r\ )\ ,\ t^*)$, with t^* as the value of the parameter t that makes $\overline{s}_y\ (\overline{y}_i\ (x_i\ ,\ x_r\ )\ ,\ t^*) = 0$. Considering that just a small group of k control points affect the form of the curve for every value of the parameter t, only these points need to be rotated and transferred to the new reference system.

As we can see at this point we don't have an explicit observation model, but even with this absence it is possible to compute (in an approximate way) the derivatives with respect to the state of the robot and of the elements in the map. Once calculated the value t* who gives $\overline{s}_y\ (t^*) = 0$ and assuming small perturbations in the state

vector, the expected measurement in the proximity to this parameter location can be approximated by the following analytical expression:

$$h(x_i, x_r) = \overline{s}_x \left(\overline{x}_i(x_i, x_r), t* - \frac{\overline{s}_y\ (\overline{y}_i\ (x_i\ ,\ x_r\ )\ ,\ t*)}{\overline{s}_y'\ (\overline{y}_i\ (x_i\ ,\ x_r\ )\ ,\ t*)} \right) \qquad (3.74)$$

To obtain this result it is assumed that with minor variations in the state of the system, the behavior of the spline near to the work point $t*$ can be approximated linearly for the tangent to the curve in that point. Now the following clarifications should be made:

- It has been pointed out explicitly the dependency functional between the measure expected and the state of the system in a given instant k. So, the expected measure comes given by the next relation

$$z = h(x_j, x_R) \qquad (3.75)$$

- The measure expected in front of smalls variations of the state comes given by the value of the coordinate x of the curve in the frame $\{u_p, v_p\}$ (figure 3.17) evaluated in the point where the coordinate y is annulled. Calling $t^{\oplus}$ to this new parameter that verifies $s_y(t^{\oplus}) = 0$ we can write:

$$h(x_i, x_r) = \overline{s}_x(\overline{x}_i, t^{\oplus}) \qquad (3.76)$$

- The position of the control points that defines the curve expressed in the frame bound to the position defined by the laser beam of the sensor, depends, on the positions of the control points in the global frame and also on the position and orientation of the robot (3.72).

- Finally, the parameter that annuls the coordinate y in the curve on the frame $\{u_p, v_p\}$can be calculated doing only one iteration of Newton-Rapson near to the solution value for the work point $t*$:

$$t^{\oplus} = t* - \frac{\overline{s}_y(\overline{y}_i(x_i, x_r), t*)}{\overline{s}_y'(\overline{y}_i(x_i, x_r), t*)} \qquad (3.77)$$

All these results allows us to obtain the expression (3.74).

Deriving (4.17) with respect to the position points on the global frame that define the curve, we have the following:

$$\frac{\partial h}{\partial \mathrm{x}_i} = \frac{\partial \overline{s}_x}{\partial \overline{x}_i}\frac{\partial \overline{x}_i}{\partial \mathrm{x}_i} + \overline{s}_x'(t^*)\frac{\frac{\partial \overline{s}_y}{\partial \overline{y}_i}'\frac{\partial \overline{y}_i}{\partial \mathrm{x}_i}\overline{s}_y(t^*) - \frac{\partial \overline{s}_y}{\partial \overline{y}_i}\frac{\partial \overline{y}_i}{\partial \mathrm{x}_i}\overline{s}_y'(t^*)}{\left[\overline{s}_y'(t^*)\right]^2} \quad (3.78)$$

Considering that the coordinate *y* is zero $\overline{s}_y(t^*) = 0$ on the frame $\{u_L, v_L\}$:

$$\frac{\partial h}{\partial \mathrm{x}_i} = \frac{\partial \overline{s}_x}{\partial \overline{x}_i}\frac{\partial \overline{x}_i}{\partial \mathrm{x}_i} - \frac{\overline{s}_x'(t^*)}{\overline{s}_y'(t^*)}\frac{\partial \overline{s}_y}{\partial \overline{y}_i}\frac{\partial \overline{y}_i}{\partial \mathrm{x}_i} \quad (3.79)$$

In figure 3.17 we can see that the slope of the spline in this same frame is been left expressed in function of the angles η and μ

$$\frac{\overline{s}_X'(t^*)}{\overline{s}_y'(t^*)} = \frac{1}{\tan(\eta - \mu)} \quad (3.80)$$

So, we have:

$$\frac{\partial h}{\partial \mathrm{x}_i} = \frac{\partial \overline{s}_x}{\partial \overline{x}_i}\frac{\partial \overline{x}_i}{\partial \mathrm{x}_i} - \frac{1}{\tan(\eta - \mu)}\frac{\partial \overline{s}_y}{\partial \overline{y}_i}\frac{\partial \overline{y}_i}{\partial \mathrm{x}_i} \quad (3.81)$$

Now, with the equation (3.58) we can see:

$$\frac{\partial \overline{s}_x}{\partial \overline{x}_i} = \frac{\partial \overline{s}_y}{\partial \overline{y}_i} = N_{i,p}(t) \quad (3.82)$$

And from (3.72) we have:

$$\frac{\partial \overline{x}_i}{\partial \mathrm{x}_i} = \cos(\mu) \quad (3.83)$$

$$\frac{\partial \overline{x}_i}{\partial \mathrm{y}_i} = \sin(\mu) \quad (3.84)$$

$$\frac{\partial \overline{y}_i}{\partial \mathrm{x}_i} = -\sin(\mu) \tag{3.85}$$

$$\frac{\partial \overline{y}_i}{\partial \mathrm{y}_i} = \cos(\mu) \tag{3.86}$$

So, we can write:

$$\frac{\partial h}{\partial x_i} = N_{i,p}(t^*)\left[\cos\mu + \frac{\sin\mu}{\tan(\eta-\mu)}\right] \tag{3.87}$$

$$\frac{\partial h}{\partial y_i} = N_{i,p}(t^*)\left[\sin\mu + \frac{\cos\mu}{\tan(\eta-\mu)}\right] \tag{3.88}$$

Those, as we have already said are the derivatives of the observation model with respect to the control point's positions that define the spline in the map.

In a similar way and using the property of splines who says $\sum_{i=0}^{n} N_{i,p}(t) = 1$ and the ecuations (3.58) and (3.72), we can calculate the partial derivatives of the measure with respect to the robot's position.

$$\begin{aligned}\frac{\partial h}{\partial x_r} &= \sum \frac{\partial \overline{s}_x}{\partial \overline{x}_i}\frac{\partial \overline{x}_i}{\partial \mathrm{x}_r} - \frac{\overline{s}_x'(t^*)}{\overline{s}_y'(t^*)}\sum \frac{\partial \overline{s}_y}{\partial \overline{y}_i}\frac{\partial \overline{y}_i}{\partial \mathrm{x}_r} \\ &= \left[-\cos\mu - \frac{\overline{s}_x'(t^*)}{\overline{s}_y'(t^*)}\sin\mu\right]\sum N_{i,p}(t^*) \\ &= -\cos\mu - \frac{\sin\mu}{\tan(\eta-\mu)}\end{aligned} \tag{3.89}$$

$$\begin{aligned}\frac{\partial h}{\partial \mathrm{y}_r} &= \sum \frac{\partial \overline{s}_x}{\partial \overline{x}_i}\frac{\partial \overline{x}_i}{\partial \mathrm{y}_r} - \frac{\overline{s}_x'(t^*)}{\overline{s}_y'(t^*)}\sum \frac{\partial \overline{s}_y}{\partial \overline{y}_i}\frac{\partial \overline{y}_i}{\partial \mathrm{y}_r} \\ &= \left[-\sin\mu - \frac{\overline{s}_x'(t^*)}{\overline{s}_y'(t^*)}\cos\mu\right]\sum N_{i,p}(t^*) \\ &= -\sin\mu + \frac{\cos\mu}{\tan(\eta-\mu)}\end{aligned} \tag{3.90}$$

$$\frac{\partial h}{\partial \theta_r} = \sum \frac{\partial \overline{s}_x}{\partial \overline{x}_i}\frac{\partial \overline{x}_i}{\partial \theta_r} - \frac{\overline{s}_x'(t^*)}{\overline{s}_y'(t^*)}\sum \frac{\partial \overline{s}_y}{\partial \overline{y}_i}\frac{\partial \overline{y}_i}{\partial \theta_r}$$
$$= \sum N_{i,p}(t^*)\overline{y}_i + \frac{1}{\tan(\eta-\mu)}\sum N_{i,p}(t^*)\overline{x}_i \qquad (3.91)$$
$$= \frac{\hat{z}}{\tan(\eta-\mu)}$$

These formulas will be used to make the efficient calculation of the relevant Jacobians in the following sections.

3.3.3 Application of EKF with B-Splines to robot localization

In this section, all the methods and equations obtained in section 3.3.2 will be combined in the frame of the EKF algorithm presented in figure 3.6 which will allow us to build incrementally the map of the environment where the features are described as cubic B-Splines.

a) Prediction

As in section 3.2.2 the prediction step gives the relative movement of the robot between the times *k-1* and *k* (Figure 3.18a). So, knowing that we are working in a non dynamic environment (i.e. the robot is the only moving object), the a priori estimation at the time *k* of the state is given by (3.28)

$$\hat{X}_k^- = f\left(\hat{X}_{k-1},\, u_k,\, 0\right)$$

where *f* is the motion model of our differential robot. Also its covariance is calculated using (3.15).

$$P_k^- = A_k\, P_{k-1}\, A_k^T + W_k Q_k\, W_k^T$$

The Jacobian matrices are calculated exactly as in (3.25) and (3.26)

$$A_k = \begin{bmatrix} \frac{\partial f(\hat{X}_{k-1}, u_k)}{\partial \hat{X}_{k-1}} & 0 & \cdots & 0 \\ 0 & I_{n1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I_{nN} \end{bmatrix} \qquad (3.92)$$

$$W_k = \begin{bmatrix} \frac{\partial f(\hat{X}_{k-1}, u_k)}{\partial u_k} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad (3.91)$$

b) Update

Once obtained the measure awaited for every one of the laser positions along of his angular range, the covariance matrix of the innovation of the measure comes given by (3.42):

$$S_k = H_k P_k^- H_k^T + V_k R_k V_k^T$$

where R_k is the sensor covariance matrix and the Jacobian H_k has the following expresion:

$$H_k = \begin{bmatrix} \frac{\partial h}{\partial X_r} & 0 \cdots 0 & \frac{\partial h}{\partial X_{si}} & 0 \cdots 0 \end{bmatrix} \qquad (3.92)$$

where the term $\frac{\partial h}{\partial X_r}$ can be calculated making use of (3.89),(3.90) and (3.91), and the term $\frac{\partial h}{\partial X_r}$ from (3.87) and (3.88). The gain matrix then is calculated as in (3.43):

$$K_k = P_k^- H_k^T S_k^{-1}$$

Finally, the state estimation and its covariance are updated (Figure 3.18b), equations (3.44) and (3.45).

$$\hat{X}_k = \hat{X}_k^- + K_k(Z_k - h(\hat{X}_k^-, 0))$$

$$P_k = \left(I - K_k H_k \right) P_k^-$$

Where the term $z_k - h(\hat{X}_k^-, 0)$ represents the innovation:

$$v_k = z_k - h(\hat{X}_k^-, 0) \qquad (3.93)$$

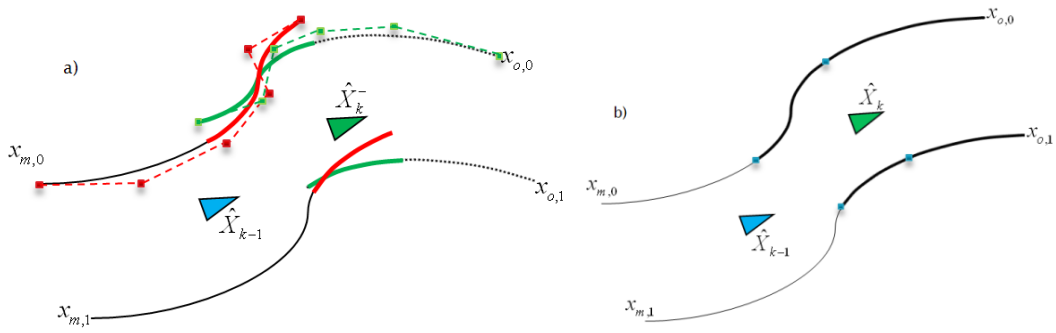

Figure 3.18 EKF. a) Configuration after the prediction of the EKF. b) Configuration after the update of the EKF.

3.3.4 Extension of the map

This is done also by the function UPDATE in the SRT algorithm (figure 3.2 lines 5). Here the stochastic map is incrementally built according to two mechanisms good differentiated:

- ***Add new objects.*** With this mechanism, the objects detected by the robot's sensors, that has not been associated with any of the objects in the map, are initialized inside the vector of state of the system, but besides of this extension also is necessary to extend the covariance matrix to include the new stochastic information.

- ***Extend the objects in the map.*** When one object detected by the robot is associated partially with one or more objects in the map, is possible to extend this last ones with the new information acquired.

***a)* Adding new objects**

When a new observation is not associated to any of the curves in the map, these can be considered as new objects, and the splines that define their geometry must be added to the model. It means that the control point that forms the spline will be added to the vector of state of the system. Then, given the set of measures $z = \{z_i,\ i = p,\ldots,\ p + q\}$ obtained for the laser angular positions in the robot reference frame corresponding to the new feature F_{N+1} and the N static features already stored in the map, the state vector, the state vector is augmented as follow:

$$X^{a} = g(X, Z) \Leftrightarrow \begin{cases} X_r^a = X_r \\ X_{si}^a = X_{si}, \quad i = 1,...,N \\ X_{sN+1}^a = g_{sN+1}(X_r, z) \end{cases} \qquad (3.94)$$

The state of the robot as the N features in the map won't be modified by the fact of to add new elements to him. In the other hand the state of the new element is given by the position of the control point of the curve that represents it, calculated as a function of the state of robot X_r and of the vector of measures acquires z. So, in the equation (3.94) the function $g_{sN+1}(X_r, z)$ is the fitting function of the $q+1$ new data points obtained by the sensor, presented in section 3.3.1. In this way, we can obtain the control points of the new feature as a linear function of the data points and the robot pose:

$$\begin{bmatrix} x_{N+1,0} \\ \vdots \\ x_{N+1,nN} \end{bmatrix} = \Phi \begin{bmatrix} x_r + z_p \cos(\theta_r + \tau_p) \\ \vdots \\ x_r + z_p \cos(\theta_r + \tau_{p+q}) \end{bmatrix} \qquad (3.95)$$

$$\begin{bmatrix} y_{N+1,0} \\ \vdots \\ y_{N+1,nN} \end{bmatrix} = \Phi \begin{bmatrix} y_r + z_p \sin(\theta_r + \tau_p) \\ \vdots \\ y_r + z_p \sin(\theta_r + \tau_{p+q}) \end{bmatrix} \qquad (3.96)$$

Where Φ is the pseudoinverse of the placement matrix.

Like for the classic EKF in equation (3.49), the new covariance matrix for the augmented state vector is

$$P^a = J_x^a\ P_k\ J_x^{aT} + J_z^a\ R_k\ J_z^{aT} \qquad (3.97)$$

Where R is as we have said, the covariance matrix of the laser measures, and with the Jacobians $J_x^a = \frac{\partial g}{\partial X_r}$ and $J_z^a = \frac{\partial g}{\partial Z}$ as:

$$J_x^a = \left[\begin{array}{cccc} I_r & 0 & \cdots & 0 \\ 0 & I_{n1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I_{nN} \\ \hline \frac{\partial g_{sN+1}}{\partial X_r} & 0 & \cdots & 0 \end{array}\right], \quad J_z^a = \left[\begin{array}{c} Ir \\ 0 \\ \vdots \\ 0 \\ \hline \frac{\partial g_{sN+1}}{\partial Z} \end{array}\right] \tag{3.98}$$

With

$$\frac{\partial g_{sN+1}}{\partial X_r} = \left[\begin{array}{c} \Phi\begin{bmatrix} 1 & 0 & -z_p \sin\mu_p \\ \vdots & \vdots & \vdots \\ 1 & 0 & -z_{p+q} \sin\mu_{p+q} \end{bmatrix} \\ \hline \Phi\begin{bmatrix} 0 & 1 & z_p \cos\mu_p \\ \vdots & \vdots & \vdots \\ 0 & 1 & z_{p+q} \cos\mu_{p+q} \end{bmatrix} \end{array}\right] \tag{3.99}$$

$$\frac{\partial g_{sN+1}}{\partial Z} = \left[\begin{array}{c} \Phi\begin{bmatrix} \cos\mu_p & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \cos\mu_{p+q} \end{bmatrix} \\ \hline \Phi\begin{bmatrix} \sin\mu_p & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sin\mu_{p+q} \end{bmatrix} \end{array}\right] \tag{3.100}$$

This way, it is possible to obtain the state vector augmented of the system after the inclusion of the new object as function of the state of the system not augmented (but corrected after the update of the EKF) and of the measures obtained by the laser for this new object.

b) Extend the objects in the map

In the most frequent case, the observations obtained will be associated only partially with some feature in the map, we can see this phenomenon in the figure 3.19a where

the j^{th} feature in the map $x_{m,j}$ is partially associated with the observation $x_{o,0}$ and $x_{m,j+1}$ with $x_{o,1}$

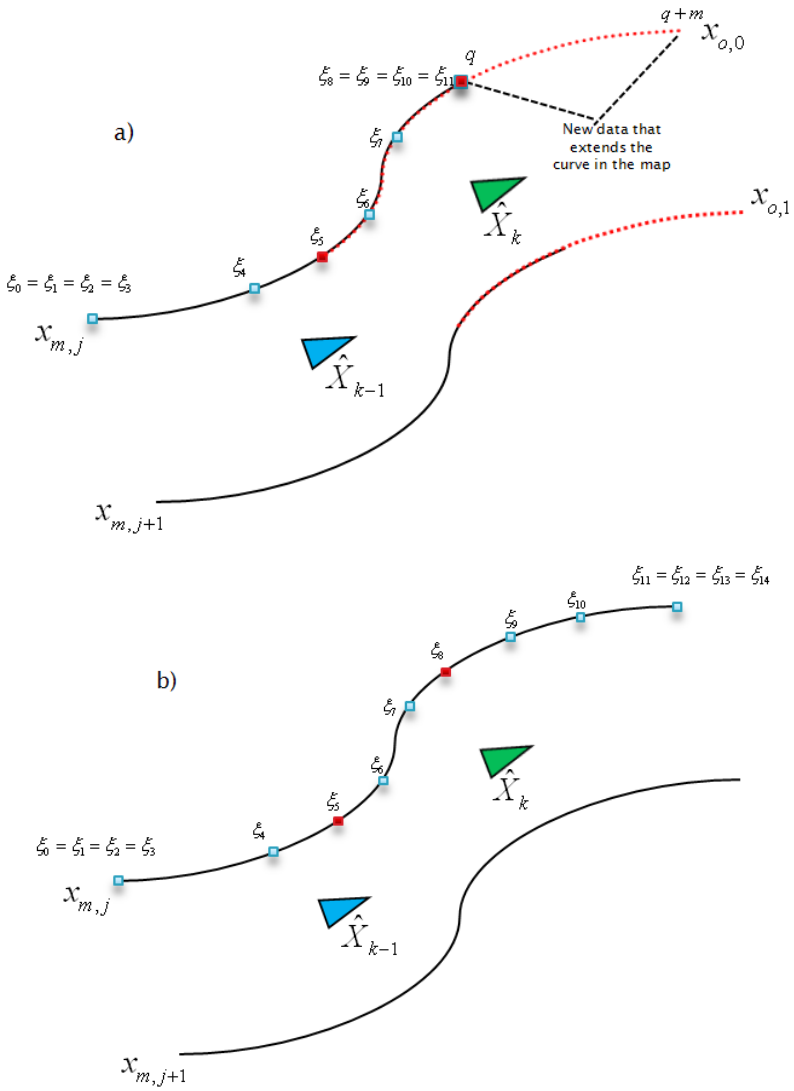

Figure 3.19 Extension of a spline with new data. a) Curve in the map with the m+1 measures corresponding to the new zone detected. The red squares represent the beginning and the end of the association between the map and the observed curve. The blue squares indicate the positions of the knots over the curve. b) Curve after his extension, we can see the need of extend the knot vector for the new information .

This situation indicates that a new area unexplored of one object in the map is been detected by the sensor of the robot and accordingly this new set of $m+1$ data points unassociated must be integrated into the map. Figure 3.19a.

$$d_i = \begin{bmatrix} d_i^x \\ d_i^y \end{bmatrix} = \begin{bmatrix} x_r + z_i \cos \mu_i \\ y_r + z_i \sin \mu_i \end{bmatrix} \quad , \quad i = q, ..., q+m \qquad (3.101)$$

The state vector resulting of the extension of the j^{th} object will be obtained as follow:

$$X^{e} = g_{e}(X, Z) \Leftrightarrow \begin{cases} X_r^e = X_r \\ X_{si}^e = X_{si} \quad , \forall i \neq j \\ X_{si}^e = g_{sj}^e(X_r, X_j, Z_{q:q+m}) \end{cases} \qquad (3.102)$$

This means that only the curve in the map to be extended is modified while the rest remains unchanged.

In function $g_{sj}^{e}(X_r, X_j, Z_{q:q+m})$ in (3.102), X_r is the position and orientation of the robot, X_j are the control points of the j^{th} feature in the map, and $Z_{q:q+m}$ are the sensor measures corresponding to the new surface detected. This function is calculated following a similar scheme to the one used in the data fitting section but considering the following:

- Already exists a start decryption represented by the segment known.
- That will be necessaries to modify the original knot vector extending the range of the independent parameter so that the new information can be added in the resultant curve.
- That the new information will be added establishing a new parameterization for the new points congruent with the parameterization of the existing curve.

At this point, with the new set of measured data points, and using the extension algorithm for B-Splines presented by Hu et al. [**Hu et al. 2002**] and the scheme showed before we can make the extension.

So, given the clamped knot vector that defines a spline curve of order k with X_i control points:

$$\Xi : \underbrace{\xi_0 = \cdots = \xi_{k-1}}_{k} \leq \xi_k \leq \cdots \leq \xi_n \leq \underbrace{\xi_{n+1} = \cdots = \xi_{n+k}}_{k} \qquad (3.103)$$

We will use the extension algorithm for B-Splines [**Hu et al. 2002**] to find another curve geometrically equivalent but defined by a knot vector unclamped (3.3.1). For simplicity we will say “left unclamped” when the multiplicity of the first elements of the knot vector is reduced:

$$\Xi_l : \overline{\xi}_0 \le \cdots \le \overline{\xi}_{k-1} \le \xi_k \le \cdots \le \xi_n \le \underbrace{\xi_{n+1} = \cdots = \xi_{n+k}}_{k} \qquad (3.104)$$

and “right unclamped” when this is done in the k final elements of the knot vector:

$$\Xi_r : \underbrace{\xi_0 = \cdots = \xi_{k-1}}_{k} \le \xi_k \le \cdots \le \xi_n \le \overline{\xi}_{n+1} \le \cdots \le \overline{\xi}_{n+k} \qquad (3.105)$$

With L_i and R_i, i=0,…,n as the new control points obtained from (3.104) and (3.105) respectively.

So, the “right-unclamped” algorithm used in [**Pedraza et al. 2009**] for the particular case of cubic B-Splines that converts (3.103) into (3.105) and obtains the new R_i control points is:

$$\left.\begin{aligned} R_i &= X_i, \quad i = 0,...,n-2 \\ R_{n-1} &= -\Gamma_{n-1}^2 X_{n-2} + \frac{1}{\gamma_{n-1}^2} X_{n-1} \\ R_n &= \Gamma_n^2 \Gamma_{n-1}^2 X_{n-2} - \left(\frac{\Gamma_n^2}{\gamma_{n-1}^2} + \frac{\Gamma_n^1}{\gamma_n^2} \right) X_{n-1} + \frac{1}{\gamma_n^1 \gamma_n^2} X_n \end{aligned}\right\} \qquad (3.106)$$

Where

$$\gamma_i^j = \frac{\overline{\xi}_{n+1} - \overline{\xi}_i}{\overline{\xi}_{i+j+1} - \overline{\xi}_i} \text{ and } \Gamma_i^j = \frac{1-\gamma_i^j}{\gamma_i^j} \qquad (3.107)$$

In a similar way, for the “left-unclamped” we have:

$$\left.\begin{aligned} L_i &= X_i, \quad i = 2,...,n \\ L_1 &= -\psi_1^2 X_2 + \frac{1}{\delta_1^2} X_1 \\ L_0 &= \psi_0^2 \psi_1^2 X_2 - \left(\frac{\psi_0^2}{\delta_1^2} + \frac{\psi_0^1}{\delta_0^2} \right) X_1 + \frac{1}{\delta_0^1 \delta_0^2} X_0 \end{aligned}\right\} \qquad (3.108)$$

Where

$$\delta_i^j = \frac{\bar{\xi}_{k-1} - \bar{\xi}_{i+k}}{\bar{\xi}_{i+k-j-1} - \bar{\xi}_{i+k}} \text{ and } \psi_i^j = \frac{1-\delta_i^j}{\delta_i^j} \qquad (3.109)$$

With these equations and the methodology presented in section 3.3.1, now is easy to extend any feature in the map when new measures of unknown areas are discovered. However, some considerations most be done:

- One parameterization most be established for achieve a consistency between the new data point and the spline over which the integration will be done. This information is obtained from the previous stage of data association.

- The knot vector most be unclamped but also may be he has to be extended with extra knots for the new data points that will be added to the existing feature. Finally, the knot vector for the curve extended has to be a clamped knot vector.

In this way, the system of equations (3.59) is written for the new data, extended using the equations (3.106), (3.107), (3.108) and (3.109) when necessary, and its least squares solution give a lineal relation in matrix form between the original control points X_j and the new data d_i as:

$$\begin{bmatrix} x_{j,0}^e \\ \vdots \\ x_{j,n_j+P}^e \end{bmatrix} = \Phi^e \begin{bmatrix} d_q^x \\ \vdots \\ d_{q+m}^x \\ x_{j,0} \\ \vdots \\ x_{j,n_j} \end{bmatrix} \quad \begin{bmatrix} y_{j,0}^e \\ \vdots \\ y_{j,n_j+P}^e \end{bmatrix} = \Phi^e \begin{bmatrix} d_q^y \\ \vdots \\ d_{q+m}^y \\ y_{j,0} \\ \vdots \\ y_{j,n_j} \end{bmatrix} \qquad (3.110)$$

With X_j^e as the new control points of the extended spline.

Notice that once chosen the knot vector unclamped for the spline, the Φ^e matrix of extension can be considered constant. So, the new covariance matrix after extend the j^{th} spline can be obtained using (3.97) as follow:

$$P^e = J_x^e \, P_k \, J_x^{eT} + J_z^e \, R_k \, J_z^{eT}$$

Where the jacobians involved $J_x^e = \frac{\partial g^e}{\partial X_r}$ and $J_z^e = \frac{\partial g^e}{\partial z}$ have the following form:

$$J_x^a = \begin{bmatrix} I_r & 0 & \cdots & 0 & \cdots & 0 \\ 0 & I_{n1} & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial g_{sj}^e}{\partial X_r} & \vdots & \ddots & \frac{\partial g_{sj}^e}{\partial X_{sj}} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \vdots & \cdots & 0 & \cdots & I_{nN} \end{bmatrix}, \quad J_z^a = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \frac{\partial g_{sj}^e}{\partial Z} \\ \vdots \\ 0 \end{bmatrix} \qquad (3.111)$$

Been

$$\frac{\partial g_{sj}^e}{\partial X_r} = \begin{bmatrix} \Phi^e \left[\begin{array}{ccc} 1 & 0 & -z_q \sin \mu_q \\ \vdots & \vdots & \vdots \\ 1 & 0 & -z_{q+m} \sin \mu_{q+m} \\ \hline 0 & 0 & 0 \end{array}\right] \\ \hline \Phi^e \left[\begin{array}{ccc} 0 & 1 & z_q \cos \mu_q \\ \vdots & \vdots & \vdots \\ 0 & 1 & z_{q+m} \cos \mu_{q+m} \\ \hline 0 & 0 & 0 \end{array}\right] \end{bmatrix} \qquad \frac{\partial g_{sj}^e}{\partial X_{sj}} = \begin{bmatrix} \Phi^e \left[\begin{array}{c} 0 \\ \hline I \end{array}\right] \\ \hline \Phi^e \left[\begin{array}{c} 0 \\ \hline I \end{array}\right] \end{bmatrix} \qquad (3.112)$$

And

$$\frac{\partial g_{sj}^e}{\partial Z} = \begin{bmatrix} \Phi^e \left[\begin{array}{ccc} \cos \mu_q & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \cos \mu_{q+m} \\ \hline 0 & 0 & 0 \end{array}\right] \\ \hline \Phi^e \left[\begin{array}{ccc} \sin \mu_q & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sin \mu_{q+m} \\ \hline 0 & 0 & 0 \end{array}\right] \end{bmatrix} \qquad (3.113)$$

3.4 Conclusion

In this chapter we have showed the development of some existing tools in order to combine them and to obtain an SPLAM algorithm. Although these techniques are appropriated and allow us to perform a full exploration over environments without a specified geometry (for the case of representation of environments using B-Splines), there are some problems found in them can affect the approach. These problems are:

- For the method of deliberative exploration, we find that the stop criterion used in which once the robot has reached a position where it cannot explore more, it has to go back to parent nodes in the tree structure to search for new unexplored regions, ending when the root of the tree is revisited and no unexplored zone is found in this position. This means that no frontier control is perform and therefore once the method chooses a new frontier to explore in a position q_k , the robot will travel to it without knowing if more areas in this position can be explored until the process of automatic backward movement of the method will bring it back to this position to verify. With the previous we infer that the robot will travel in unnecessary way looking for positions where it could continue exploring in zones that it has already visited.

- Another problem with the method SRT is the tree structure that uses. This does not allows to perform an optimized navigation; i.e., if during the creation of a branch the robot close a loop with the root of the tree and the robot has to return to it, the method does not allow just travel from the last position to the root, instead it has to go all over the branch looking for new zones to explore over it.

- With respect to the SLAM method based on the extended Kalman filter, the main problem is the computational cost that grows quadratically with the number of objects contained in the map. For this reason its application is limited to maps formed by only a few hundred of objects.

- On the other hand, SLAM is a nonlinear problem so applying the EKF has the limitation of reducing the accuracy and consistency due to the effect that the linearizations have on estimates of the robot and hence over the map

Chapter 4. Topologic-SPLAM Algorithm

In chapter 3 we have show an approach for the SPLAM problem using some well known tools both in the field of SLAM as in the field of exploration environments. Although these algorithms have shown good performance, some problems found in them can provide negative effect to the proposed solution. This reason has encouraged the development of new strategies and process that can improve the performance of our SPLAM strategy. Between them we can mention:

- **A Random Exploration Graph (REG).** It is a modified version of the algorithm of exploration SRT. Here, we include a frontier control to carry a registry over the nodes that have not been fully explored. So, every time that the robot goes to new frontiers to explore, the frontiers in the neighbors nodes are updated in the registry and in the case that some node is reported as fully explored, it is removed from the registry.

  Another modification of the SRT method is presented when two nodes without relationship (they are not father and son) are found with their regions of local security intersected, the structure of the method is then transformed to a graph adding a connection between these two nodes.

  The graph structure, next to the frontier control, will allow a more efficient exploration given that the robot knows exactly where to go to continue the exploration. It will be also able to navigate optimally between two nodes of the structure without having to pass through the root node when a change of branch is necessary.

- **A topologic SLAM Approach.** We present also a new method of topologic SLAM based on B-Splines curves that exploits the structure used by the exploration method but also all the information contained in this kind of representation.

4.1 The Random exploration graph approach

Franchi et al present in [**Franchi et al. 2009**] a modified version of their SRT algorithm in which the tree structure is transformed to an exploration graph when a path to travel between two nodes in a safe way is found. Although our approach performs a restructuration of the tree in a similar way, the process of exploration is completely different.

Remembering the process performed by the method SRT, our approach is based also on the random choice of one of the free frontiers in the current node to continue the exploration. On the contrary, in [**Franchi et al. 2009**] the choice of the next position to explore is chosen using a probability proportional to the arc length of the frontier. The choice of random frontiers in our work was taken considering the property of ***completeness*** for the exploration methods, because no matter what frontier is chosen the method will have to return to explore the remaining free frontiers of the node as long as they exist.

Another important aspect of the method is the way in which the nodes with free frontiers will be revisited once that the node where the robot is currently found doesn't have any more free frontiers (the region has been fully explored). For this case, the method of Franchi et al. builds minimum spanning trees with all the nodes in the graph for each node adjacent to the current node without regard this last one. To the end, the tree of the node adjacent with the greatest weighted forward frontier length is chosen. This process may require too much computation time if the number of adjacent nodes and the number of nodes that form the environment built until that moment are too high. Finally, the tree structure generates a discontinuous path that forces the robot to go through the parent nodes if a change of branch is necessary ignoring again the new structure.

Contrary to the method described above, the approach presented in this section fully exploits the use of graph structure to plan a path toward nodes that have not been fully explored once the robot is in a position where it can no longer continue with the exploration.

For this planning can take place is necessary prior knowledge of these nodes. To this end a border control (Figure 4.1) will be performed as follows: if once chosen the random frontier of the current node q_{curr} on which the exploration will continue (as indicates the method SRT that is used as base) there are more unexplored frontiers

in it (Figure 4.1a), the q_{curr} node along with information on the number of remaining frontiers to explore and their arc length will be added to a list that will be used to plan a path to these nodes to continue the exploration (Figure 4.1b).

With this list obtained, the planning will be done using the method of path planning A* in a bidirectional way planning the path from the current node towards the nodes in the list and in the other side from the nodes in the list toward the current node ending when a path between the current node and any of the nodes in the list is found. In this way and unlike to the method of Franchi et al. the planning won't be perform using all the nodes in the graph and what is looked is the path towards the node with possibility of exploration with the shortest distance.

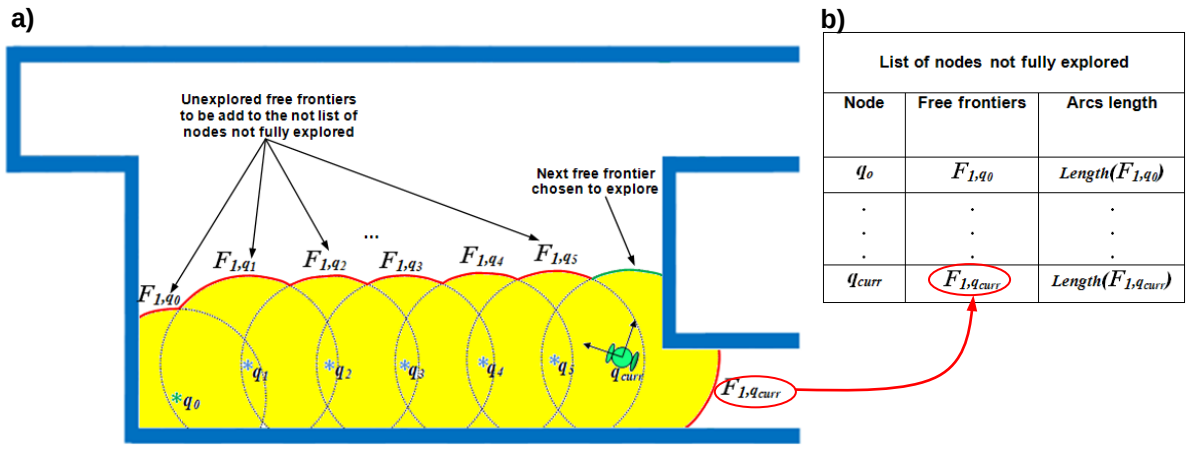

Figure 4.1 Frontier control. a) Environment semi explored where the red arcs represent free frontiers and the green arc is the next frontier chosen at the q_{curr} position to be explored. Yellow region represent the global security region (GSR) at that time. b) List of nodes not fully explored

4.1.1 Random exploration graph algorithm

As we have already said, the method of exploration proposed is an extension of the SRT exploration method in its radial version [**Espinoza et al. 2007**] presented in section 3.1 where the main structure is transformed into an exploration graph if in any position q_K the local safe region (LSR) of the current node intersects the LSR of some previous node that is not its parent node and provided that it is possible to find a collision-free path between the positions q_k and q_i of these nodes (figure 4.3).

As in the SRT, in our method the graph represents the road map of the explored area and is gradually built extending the structure towards borders selected randomly in

such way that the new configuration (and the path that leads to it) is contained in the local safe region. Each node of the graph consists of a free-collision configuration q that the robot has reached together along with the description of the local safe region S surrounding q perceived by the sensors. Moreover, an arc between two nodes represents a free-collision path.

In figure 4.2, we show the algorithm that implements the exploration method proposed. The approach is based on a frontier control that will indicate what nodes can continue being explored once the robot arrive to a position where it is not possible to continue extending the structure. For this, we start a counter that will be used as identifier to indicate the sequential position in which each node has been built. Next, the list that will contain these identifiers is initialized as an empty list.

INTEGRATED_REG_EXPLORATION (q_{init}, k_{max})

```
1.  Node_Ag = 0
2.  q_act = q_init
3.  L_Nodes_Ex = Null
4.  S ← PERCEPTION(q_act)
5.  for k=1 to k_max
6       S ← INTERSECTION_ACT(G,S,q_act,L_Nodes_Ex)
7       F ← FRONTIERS(S)
8       if  F ≠ Null
9           Node_Ag= Node_Ag+1
10          (F_rand,θ_rand) ← FRONT_RAND(F)
11          F ← REMOVE(F , F_rand)
12          q_dest ← DISPLACE(q_act , θ_rand ,α , r)
13          MOVE_TO(q_act , q_dest)
14          S_dest ← PERCEPTION(q_dest )
15          F_rand ← VERIFICATION( F_rand , S_dest)
16          F ← F ∪ F_rand
17          if  F ≠ Null
18              L_Nodes_Ex = L_Nodes_Ex ∪ Node_Ag
19          end
20          G ← ADD(G, Node_Ag , q_act , S, F)
21          q_act = q_dest
22          S = S_dest
23      else
24          (P, Ind_Node) ← FIND_PATH(qact, L_Nodes_Ex)
25          for i = 1 to length(P)
26              MOVE_TO( q_act , P(i) )
27              q_act ← P(i)
28          end
29          L_Nodes_Ex ← REMOVE(L_Nodes_Ex , Ind_Node)
30      end
31. end
32. Return ( G )
```

Figure 4.2 REG algorithm

The initial node referred like the departure and return point is built adding the initial position q_{init} and performing a first reading of the environment around this position. Although it has been created, the node is still not added to the graph because this should not be taken into account when looking for possible connections with neighboring nodes. At this point and with the elements mentioned built, the loop that controls all the process starts.

In every iteration k of the algorithm a process of verification is made over all the neighboring nodes of the q_{curr} position where two objectives are pursued.

- **The first objective** is to verifier what sections of the free-frontier of these nodes are within the current LSR. If it is possible to find these free frontiers intersecting the current LSR the node to which it corresponds is updated removing the frontier or segment of border of it and of the current node; if with this action the node is pointed as fully explored, the list of nodes to be explored is also updated by removing the identifier of the node from it.

- **The second objective** is to identify if between the intersected frontiers of the neighboring nodes with the current node is possible to find a safe path to travel between them. The criterion to ensure a collision-free path is that the straight line joining the two nodes must cross free frontiers in both nodes with a range of security to the left and right of the point over the frontiers where the straight line pass (Figure 4.3). If the looked path is found, an arc between these two nodes will be added to the structure.

The Function **INTERSECTION_ACT** performs the mentioned verifications. This function takes as parameters the graph constructed so far, the q_{curr} position, the LSR of the current node and the node list with possibility of exploration. The outputs of the function are the graph, the current LSR and the update list of nodes with possibility of exploration.

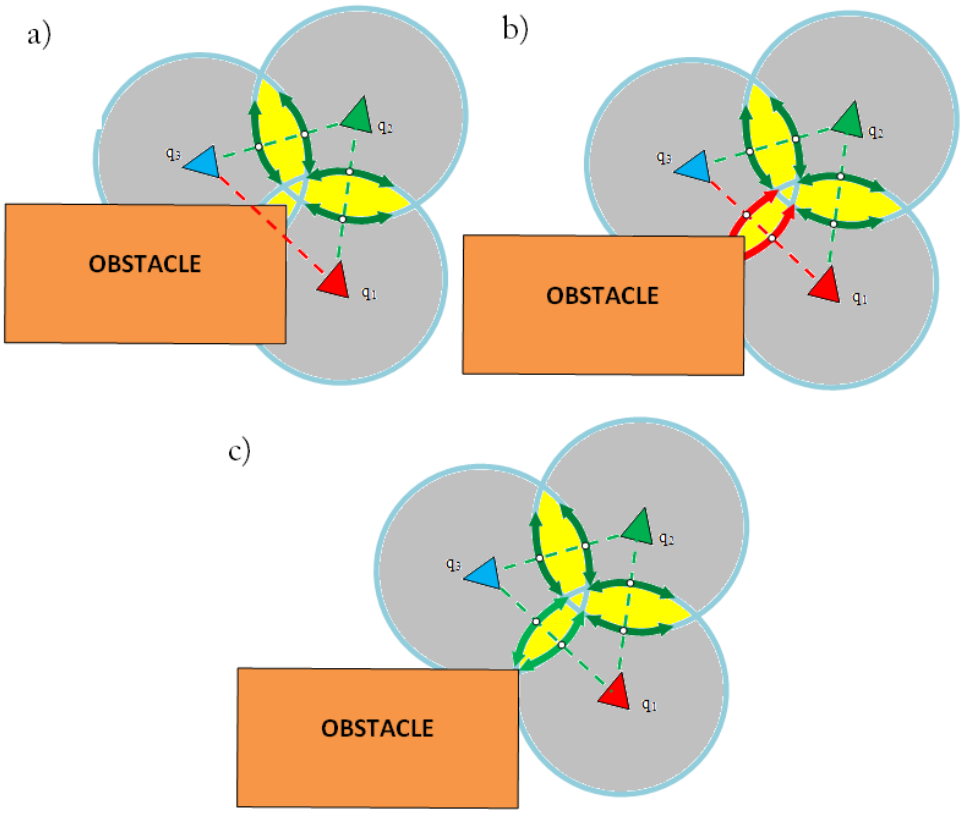

Figure 4.3 Connection between nodes, the red, green and blue triangles represent positions of robot, in gray are shown the RSL of each node, the frontiers are shown in blue color and the intersections between the LSR are shown in yellow color. a) Connections (*q1,q2*) and (*q2,q3*) possible since there is a direct path between the positions (dotted green line) and a range of security (green double-headed arrow) both the left and the right side of the point where the path passes over the frontier (white dots). b) Connections (*q1,q2*) and (*q2,q3*) possible but not for (*q1,q3*) because although there is a direct path between the positions (dotted red line) the requirement for a safety range for the left and right (red double-headed arrow) of the point where the path passes over the frontier (white dots) is not achieved. c) Connections (*q1,q2*) , (*q2,q3*) y (*q1,q3*) possible.

Once performed the verification and update (if necessary) of the structures, the function **FRONTIERS** will obtain the remaining free frontiers F of the LSR. If at least one free frontier is found, the function **FRONT_RAND** will randomly choose one of them and the middle point of the arc length of the chosen frontier will be the new random direction θ_{rand} to visit as long as the arc length of the chosen frontier does not exceed a certain threshold chosen proportional to the distance that the new LSR can cover. Otherwise it will be chosen the frontier segment proportional to the arc length that can be covered starting on the initial extreme of the frontier from which the middle point of this segment will be taken as the direction to explore θ_{rand}. This is done because we do not want to leave two frontiers to explore in opposite sides of the LSR if the current frontier segment to explore is too big.

Unlike the SRT method, our approach does not require a validation function to verify that the new chosen direction is not within the LSR of some other node because the frontier control performed removes frontiers contained in other nodes (function **INTERSECTION_ACT**) and also because each node stores separately frontiers corresponding to obstacles and free frontiers. The random frontier chosen is then eliminated of the group of free frontiers found with the function **FRONTIERS** since by now is not longer considered free.

Once that the random direction has been chosen, the function **DISPLACEMENT** will get the new position q_{dest} to be visited giving one step of length $\alpha \cdot r$ in the direction θ_{rand} where the constant r represents the radio of the LSR. The constant $\alpha < 1$ in turn ensures that is inside of the LSR and that can be achieved through a path contained in it; values close to 1 maximize the exploration but also increase the probability of a collision with objects that have not been detected yet, on the other hand values close to zero augment the margin of safety but it can result in a very slow exploration process. In our case we have chosen a 75% of the length of the radius to find the new position to explore.

With the new target position q_{dest} calculated, the function **MOVE-TO** will carry the robot from the current position to the new target position. The process is performed in the same way that in section 3.1 taking a list of control inputs ($list\_U$) and choosing the one that best approximates the robot from the current position to the target position. The process will be repeated until the current position and the target position have a distance not greater than a threshold Ψ. In this function as in its counterpart of section 3.1 we will use the reported information about the increases in x,y and θ between the previous localized position and the current odometric position as well as the information of the LSR to estimate the real position of the robot using the proposed topological localization method that will be presented in section 4.2.

Once the robot has reached the target position, a new process of perception is perform to estimate the surrounding space S_{dest} of this new position. With this information, the function **VERIFIES** estimate what portion of the previous frontier F_{rand} chosen by **FRONT_RAND** has been covered. In the case where the frontier has not been 100% covered, the function will return the remaining portion of frontier to explore and this will be joined to the frontiers group F of the previous node that must be explored.

The reason to obtain the new LSR as well as the frontier verification is performed at this point, is because the position q_{dest} is never fully achieved but approximated to a certain threshold distance, then if calculations to verify if the frontier has been fully explored were performed immediately after having obtained the position q_{dest} these would have a position error and therefore would be inaccurate (Figure 4.4).

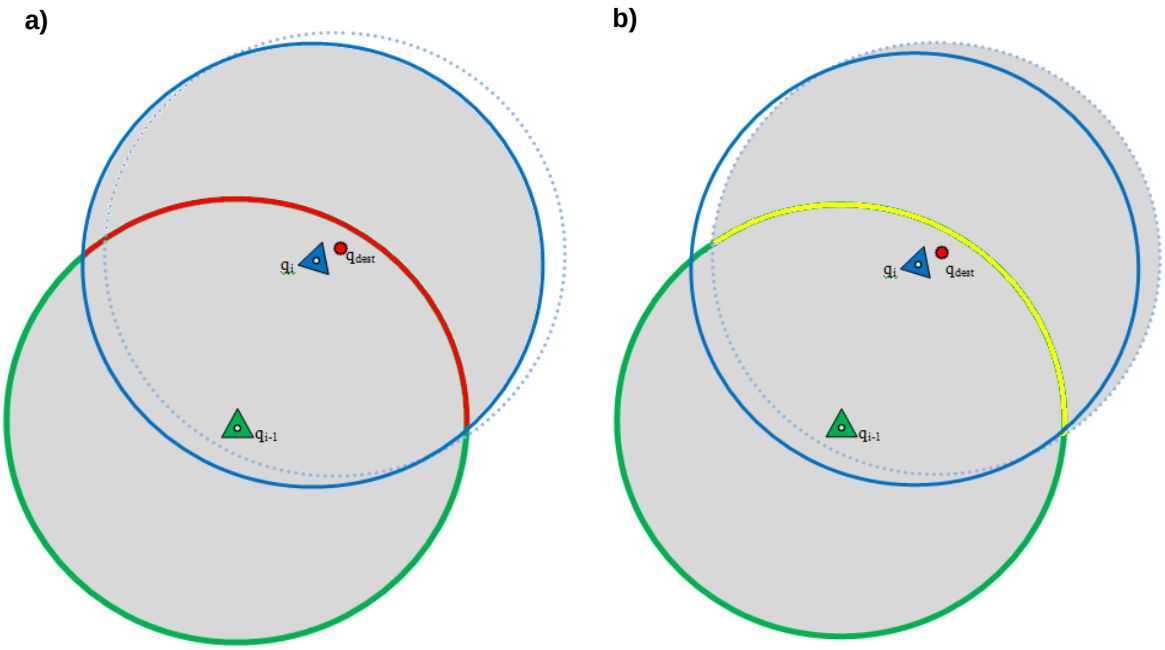

Figure 4.4 Explored frontier calculation based on the position of the robot q_i and in the estimated target position q_{dest}. The red dot represents the target position q_{dest}, the green and blue triangle represents the position q_{i-1} and the position q_i approximated to the position q_{dest} respectively. The green circumference represents the frontier of the LSR on the position q_{i-1}, the blue circumference in continuous line represents the frontier of the LSR on the position q_i, the blue circumference in dotted line represents the estimated frontier of the LSR in the estimated position q_{dest}. The gray shading zone represents the LSRs of the q_{i-1}, q_i and q_{dest} positions. a) The frontier segment in red represents the portion of real frontier that has been explored since the robot only approaches the position q_{dest} but cannot reach it exactly. b) The yellow line shows the estimated frontier segment explored from the target position q_{dest}.

Obtained the information about the remaining free frontiers of the previous node the method classifies the node as a node with possibility of exploring whether there are still free frontiers, in this case the head of the node is added to the list of nodes to be explored.

Finally the node is added to the structure of the graph and the curves in the map will be extended with the new information (as will be shown in section 4.) using the

function **ADD** and the loop begins again now using the information q_{dest} and S_{dest} as q_{act} and S.

Typically when the space has been fully explored the algorithm will fail to find a frontier to explore. In this case, the exploration must continue in any of the nodes stored in the node list with possibility of exploration in case they exist, otherwise the method ends and the robot will return to the initial node. The search for the next node to explore is performed using the graph-search algorithm A* in a bidirectional way.

The method A* evaluates the nodes by combining the cost to reach a node and the estimated cost of going to the goal node:

$$f^*(n) = g(n) + h(n) \qquad (4.1)$$

Since $g(n)$ gives the cost of the path from start node to node n, and $h(n)$ is the estimated cost of cheapest path from n to the objective, $f^*(n)$ represents the cheapest estimated cost of the solution through n.

The bidirectional use of this algorithm implies to extend the path both from the initial position and from the wished position leading the search always towards the final position reached by the opposite side and ending when both paths are in the same node (Figure 4.5).

This strategy is not used individually, it is used simultaneously with all the nodes contained in the list and stopped when a path is found; this procedure is performed by the function **FIND-PATH**. The reason to seek individual paths from the current position to all the nodes and not use simply an Euclidean distance to go to the nearest node that is the aim, is that due to dead rooms and other structures the distance to a node may seem small, however the path to reach it could be too extensive.

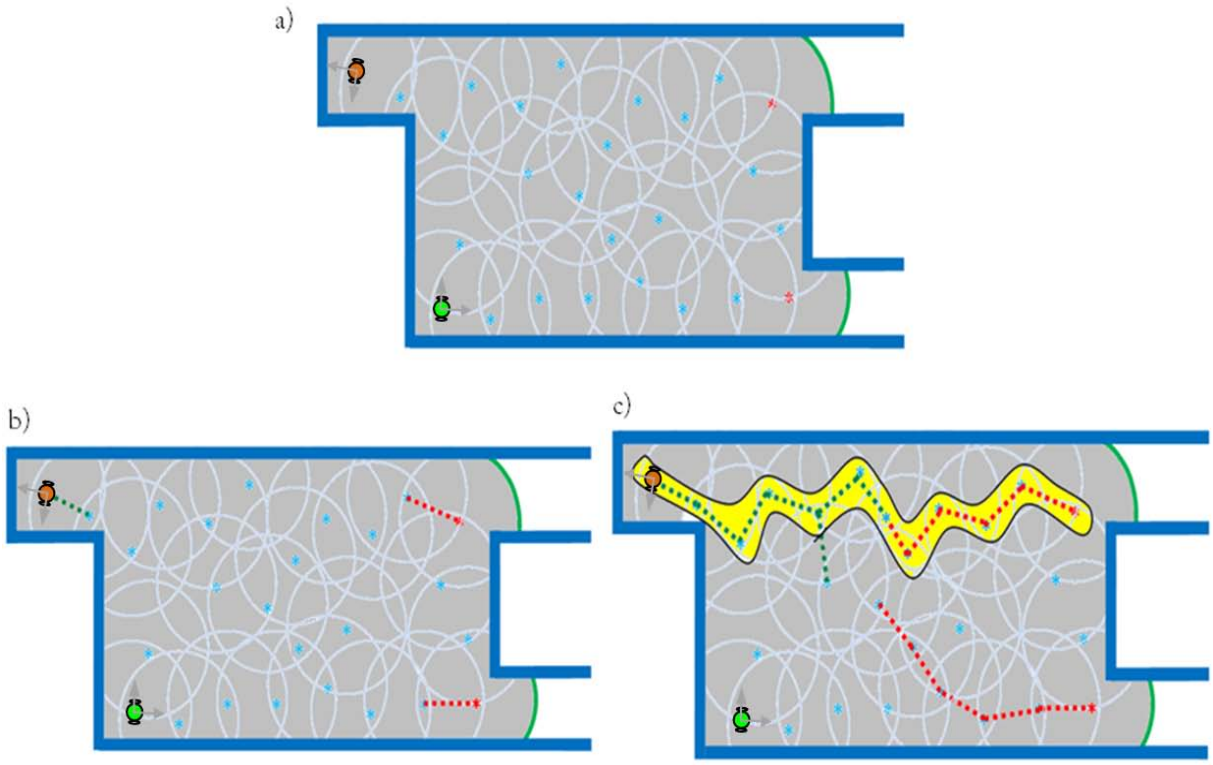

Figure 4.5 Evolution of the bidirectional A * algorithm. a) Environment explored where the green robot represent the initial position, the red asterisks represent nodes with possibility of exploration with its free frontiers in green, the orange robot represents the current position q_{curr} that has been fully explored, the shaded part represents the free space explored and the blue circumferences are the LSR frontiers of each node. b) Evolution of the A * algorithm in t=1; the paths are extended simultaneously from q_{curr} toward the last node in the path partially built that comes from the nodes that still have free frontiers and vice versa. b) The algorithm finishes when a path between q_{curr} and some node with free frontier is found (path contained in the yellow region).

Obtained the trajectory P, the method **MOVE_TO** will lead the robot from the current node to the node where the exploration will be continued. Finally, the index of the chosen node from which the exploration will continue is eliminated of the list of nodes with possibility of exploration. With the new node to explore the method will continue making the same process described in the algorithm until no more frontiers remains to explore being in this moment when the robot returns to the initial node of the graph ending the exploration.

4.2 Topologic SLAM with B-Splines

Although the SLAM strategies based on the Kalman filter have shown good results, some problems are found such as the inevitable inconsistency and accuracy

reduction in the estimation of the system state due to the effect that linearizations produce on the estimates of the robot and the map.

Another problem found in the implementations of the filter is the computational cost required by the method that by being squared with the number of items in the map its application is limited to maps that contain a limited number of them.

Although many works have emerged to solve these problems, we have decided to develop a new topological strategy partially based on the use of subregions contained in each node of the exploration method (section 4.1) and in B-Spline curves for the modeling of the obstacles. The lack of a covariance matrix and the use of limited regions of the environment allow that the SLAM problem be easily attacked even for very large environments in real time.

In this section we present the reasoning, concepts and algorithms needed that allow the map construction where the entities that form it will be modeled by B-spline curves.

4.2.1 Data management

Given that our method is based on the use of B-Splines to the representation of the environment, in this section we will explain how we perform the interpretation of data coming from the sensor to obtain the parametric splines that represents the physical world surrounding the robot as well as the data association process to establish correspondences between the detected splines and the splines contained in the map.

a) Acquisition of the B-Splines

When the robot gets a new set of measures of its environment through its perception system, the surrounding world is just a set of points $p_i \in \Re^2$ (considering in our case only a two-dimensional scenario) meaningless linked to each other only by the logic of the ordination provided by the scanning sensor (Figure 4.6). With these raw data, the first objective is to clearly identify to what object each one of the measurements belongs grouping them into subsets to finally get the B-spline curves that represent the portions of the detected objects as close as is possible.

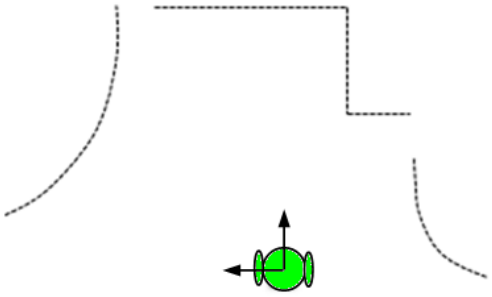

Figure 4.6 Measurements obtained with a laser scanning

As in section 3.3.2, the segmentation of the raw data will be performed using adaptive cluster method proposed by Dietmayer from which we will obtain measurements subgroups belonging to different objects. It is important to highlight at this point that unlike the method of Pedraza et al **[Pedraza et al. 2009]** recreated in Chapter 3, in our method the information obtained by the sensors is segmented just once to get groups of points belonging to the same object and not twice as is proposed by the authors in **[Pedraza et al. 2009]**. This because we have used methods from the area of digital images and artificial intelligence where high curvatures provide valuable information needed to perform a more efficient data association.

Once obtained the subsets or clusters, the measurements corresponding to each one of the detected objects will be approximated using unclamped B-spline curves of degree 3 as we have mentioned in section 3.3.1. Recall that to fit measurement points with B-Splines helps to reduce the noise contained in them. The choice of using ***unclamped B-Splines*** is because the use of ***clamped B-Splines*** would force to the extreme of the curve to pass directly from the start and end points of the set of noisy points that form it, obtaining a less accurate representation of the object that we are trying to model and therefore more difficult to associate (Figure 3.13).

Although the unclamped B-spline significantly reduce the noise in the measurements provoked by errors of the measurement system itself; one last smoothing of the curve must be performed using a Gaussian filter to guarantee that the process will not be affected by false information. For this, an evolved version Γ_σ of the curve Γ can be processed:

$$\Gamma_\sigma = \{x(u,\sigma), y(u,\sigma)\} \qquad (4.2)$$

Where

$$x(u,\sigma) = x(u) \otimes g(u,\sigma), \quad y(u,\sigma) = y(u) \otimes g(u,\sigma) \quad (4.3)$$

Here, $\otimes$ represents the convolution operator and $g(u,\sigma)$ denotes a Gaussian filter of width σ. This last is chosen in such way that only eliminates the noise but not valuable information on the curve so the value will be very low. Since our project works normally with open curves, a certain number of points proportional to the double of the Full width at half maximum (FWHM) for a Gaussian are symmetrically compensated at both extremes of the curve when it is smoothed.

Finally, in order that the localization process can run effectively, the invariability in the resolution of the curve must be assured. So, each discrete B-spline curve should be stored taking equidistant points on it with a distance ε between each point:

$$dist(\sum N_p(u)X, \sum N_p(u+1)X) \cong \varepsilon \quad (4.4)$$

Where $\sum N_p(u)X$ is the recursive formula Cox-de-Boor to obtain a curve B-Spline presented in the section 3.3.1. Of the previous we obtain a parametric vector containing the B-spline, where the parameter u represents a point on the curve.

In addition, a restriction over the length of the curve is applied because objects too small may not provide enough information and therefore it is not interesting to include them to the map. Also, although our method is designed to work in static environments, this restriction allows in some way to filter dynamic items (people par example) that will not be included in the map.

Once the B-Splines have been obtained and chosen, we can search specific feature contained in the curves that will be of great importance in the localization process. Essentially, two types of features will be searched in the curves:

- **Curvature Zero crossings**
- **Corners**

The Corner type is very common and does not require explanation. On the other hand, the concept of Curvature zero crossings in a very general way refers to the point on the curve where it passes from concave to convex or vice versa.

The process to obtain both features on the curve is based on the curvature scale space CSS [**Mokhtarian 1995**] which is used to recover invariant geometric features.

b) B-Spline curvature

The term "curvature" of a B-Spline curve is defined as the local measure which indicates how much a curve has moved away from a straight line. More formally, the curvature of a point $X_u = [x_u, y_u]$ in the b-spline, is defined as the amount equal to the inverse of the radius of the osculator circle at the point (the circle that touches tangentially to the curve at the point X_u) which means that while smaller is the radius ρ of this circle bigger will be the curvature at this point $1/\rho$.

The formula for computing the curvature can be expressed as:

$$k(u,\sigma) = \frac{\dot{x}(u,\sigma)\ddot{y}(u,\sigma) - \ddot{x}(u,\sigma)\dot{y}(u,\sigma)}{(\dot{x}(u,\sigma)^2 - \dot{y}(u,\sigma)^2)^{\frac{3}{2}}} \qquad (4.5)$$

Where according to the properties of convolution, the derivatives of each element can be easily calculated since we know the exact forms of the first and second derived of the Gaussian kernel used $\dot{g}(u,\sigma)$ and $\ddot{g}(u,\sigma)$. So:

$$\dot{x}(u,\sigma) = \frac{\partial}{\partial u}(x(u) \otimes g(u,\sigma)) = x(u) \otimes \dot{g}(u,\sigma) \qquad (4.6)$$

$$\ddot{x}(u,\sigma) = \frac{\partial^2}{\partial u^2}(x(u) \otimes g(u,\sigma)) = x(u) \otimes \ddot{g}(u,\sigma) \qquad (4.7)$$

$$\dot{y}(u,\sigma) = y(u) \otimes \dot{g}(u,\sigma) \qquad (4.8)$$

$$\ddot{y}(u,\sigma) = y(u) \otimes \ddot{g}(u,\sigma) \qquad (4.9)$$

b) Curve features search

With the list of curvatures k obtained from equation (4.2), the determination of the **curvature zero crossings** in the curve is carried out simply by looking in the list for points where a change of sign between two consecutive curvatures is found.

On the other hand, the search for **corners** requires a processing of the curvature more complex since the detector is based on local and global properties of the curvature [**He et al. 2008**]. In order that this new process can be applied correctly, the curvature obtained must contain only positive values therefore only the absolute value of k will be considered.

Initially all the local maxima of the curvature are seen as candidates to corner since is assumed that the true corners are included in this group. But given that within these local maxima could also be included curved segments that do not represent a true corner, two criteria have been established to eliminate these false corners from the list of candidates. The first of them refers to comparing the curvature of the candidates with a curvature adaptive threshold, and the second to evaluate the angles of the remaining candidate corners to eliminate those that provide trivial details.

In the first criterion, although the curvature of a round corner is a local maximum, the difference between this and the curvature of its neighbors may not be significant; then, in order to use the curvature of the neighbors to eliminate rounded corners, the concept of region of support (ROS) is introduced which is defined as the curve segment bounded by the two minimum curvatures closest to the actual corner in opposite senses to it (Figure 4.7).

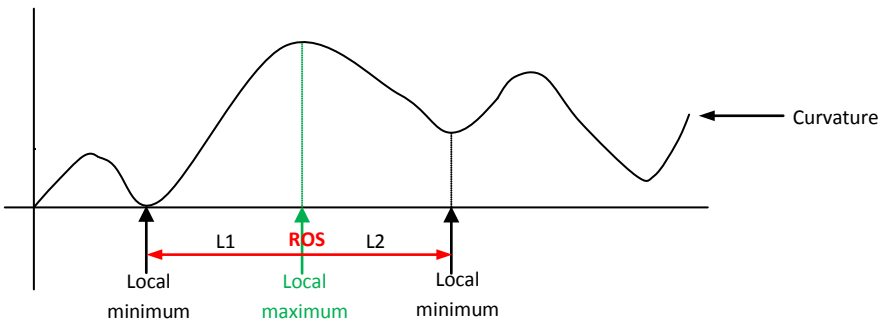

Figure 4.7 Region of support (ROS) for the elimination of round curves

The ROS of every corner will serve to obtain the adaptive threshold on which we will perform the first discrimination of candidate corners which is calculated as follows:

$$T(u) = Cx\bar{K} = 1.5x\frac{1}{L1+L2+1}\sum_{i=u-L1}^{u+L2}\left|k(i)\right| \qquad (4.10)$$

Where u is the candidate corner position on the curve, C is a coefficient (normally set at 1.5) , $L1 + L2$ is the size of the ROS ($L1$ and $L2$ is the length from the possible corner to closest the minimum local to the left and right respectively) and k is the curvature of the neighborhood.

With the adaptive threshold set, the curvatures of the possible corners are compared. If the curvature of the candidate corners is greater than this threshold, they are then declared as true corners; otherwise, they are removed from the list of candidates.

For the second criterion, an extended version of the ROS will be used, where this is now defined as the curve segment bounded by the two candidate corners to the left and right of the candidate corner that is currently being verified (Figure 4.8). For consistency, in open curves the initial and final extremes will be treated as corners for the determination of ROS.

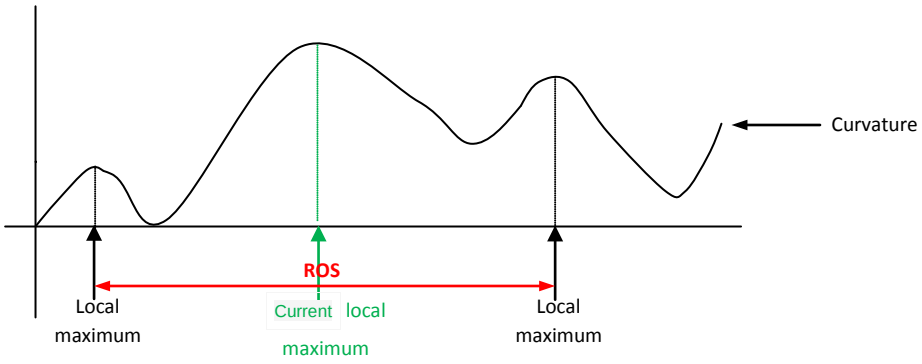

Figure 4.8 Region of support (ROS) for the elimination of false corners

Once redefined our ROS, the angle of the candidate corner will be obtained using tangents on the arms of the ROS (Figure 4.9). The calculation of these tangents is done by fitting a circle (that best fits) on the points of the arm of the ROS. Since the fit of this circle has not to be truly optimal, a three-step method is detailed down to find this circle.

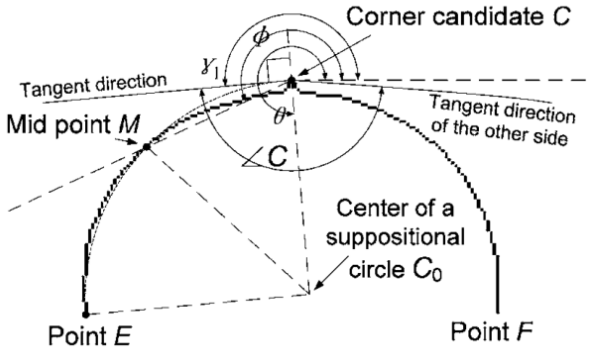

Figure 4.9 Definitions of angle of a corner. Figure taken from [He et al. 2008].

First three points are chosen on one arm of the ROS, being two of these the candidate corner that is being tested (C) and the neighboring corner to the left (E), the third point is the midpoint (M) of the curve segment bounded by the two previous points. If these points are collinear, the direction of this arm of the ROS is defined from C to E, else the center of the suppositional circle C_0 which has the same distance (radius of curvature of this ROS) of the three points is calculated using the following equation:

$$x_0 = \frac{(x_1^2 + y_1^2)(y_2 - y_3) + (x_2^2 + y_2^2)(y_3 - y_1) + (x_3^2 + y_3^2)(y_1 - y_2)}{2 \cdot [x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_2(y_3 - y_1)]} \quad (4.11)$$

$$y_0 = \frac{(x_1^2 + y_1^2)(x_2 - x_3) + (x_2^2 + y_2^2)(x_3 - x_1) + (x_3^2 + y_3^2)(x_1 - x_2)}{2 \cdot [y_1(x_2 - x_3) + y_2(x_3 - x_1) + y_2(x_3 - x_1)]} \quad (4.12)$$

Where $C = (x_1, y_1)$, $M = (x_2, y_2)$ and $E = (x_3, y_3)$.

Then a line is drawn between C and C_0 and the direction that form is stored in θ; this can be easily calculated by a four-quadrant inverse tangent function. In a similar way, the direction from C to M will be stored in ϕ. With these data, we can now calculate the value of the tangent of C to this side of ROS using the following equation:

$$\gamma_1 = \theta + sign(\sin(\phi - \theta)) \cdot \frac{\pi}{2} \quad (4.13)$$

The same procedure is performed to determine the tangent of the other arm (C to F) of the ROS which will be denoted by γ_2. With the two tangents determined, we get the angle of the corner that they form as:

$$\angle C = \begin{cases} |\gamma 1 - \gamma 2| & if\ |\gamma 1 - \gamma 2| < \pi \\ 2\pi - |\gamma 1 - \gamma 2| & Otherwise \end{cases} \qquad (4.14)$$

Finally, the corner $\angle C$ will be taken as a true corner if the angle formed is smaller than a certain obtuse angle threshold maximum θ_{obtuse}. Otherwise the corner will be set as invalid and is removed from the list of candidate corners.

c) Association of B-Splines

The data association process used for our topological SPLAM is not perform considering all the objects in the map built until the instant q_k, instead we use only the portions of the environment contained in the LSR of the last node built in the REG exploration method shown in section 3.1. This can be seen is in figure 4.10, where the yellow area indicates the portion of the environment on which the association will be performed and the green lines indicate obstacles that have not been seen yet. In this image we show that at the instant k the LSR has two objects $B_{RSL,1}$ and $B_{RSL,2}$ that will serve as reference objects until the moment in which the robot reaches the position q_{dest} and a new LSR is obtained.

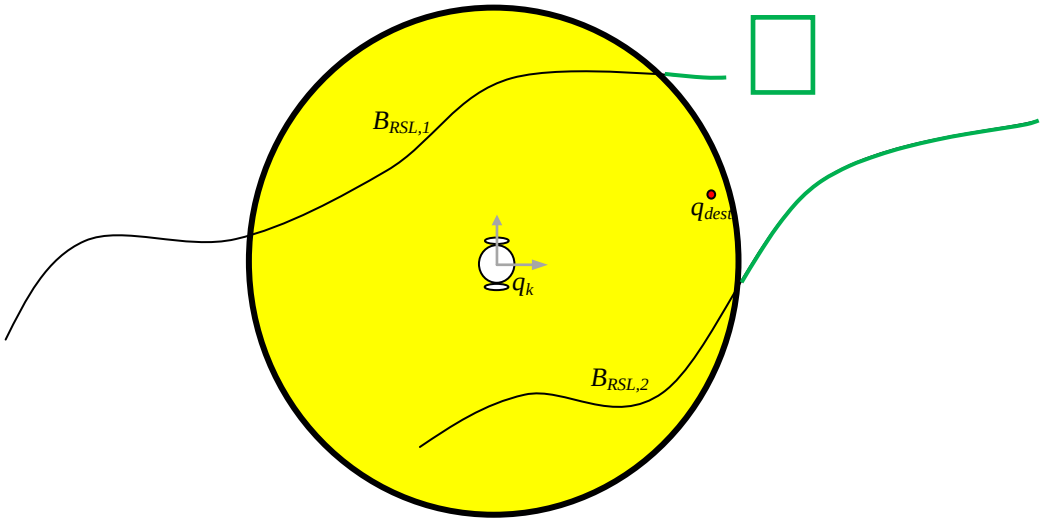

Figure 4.10 LSR of the robot at the instant *k*

Figure 4.11 shows to the evolution of the robot until the instant $k+s$ where three objects are $B_{O,1}$, $B_{O,2}$ and $B_{O,3}$ detected (shown in red) in the range of vision of the sensor (circumference in dotted line). In this figure, we can see that the odometric position is just an a priori approximation of the true position of the robot obtained using the robot's motion model since errors and sensory noise will lead the robot to a different real position that will not be the awaited one. For this reason, the objects $B_{O,1}$, $B_{O,2}$ and $B_{O,3}$ appear displaced and do not overlap exactly with the objects to which they belong.

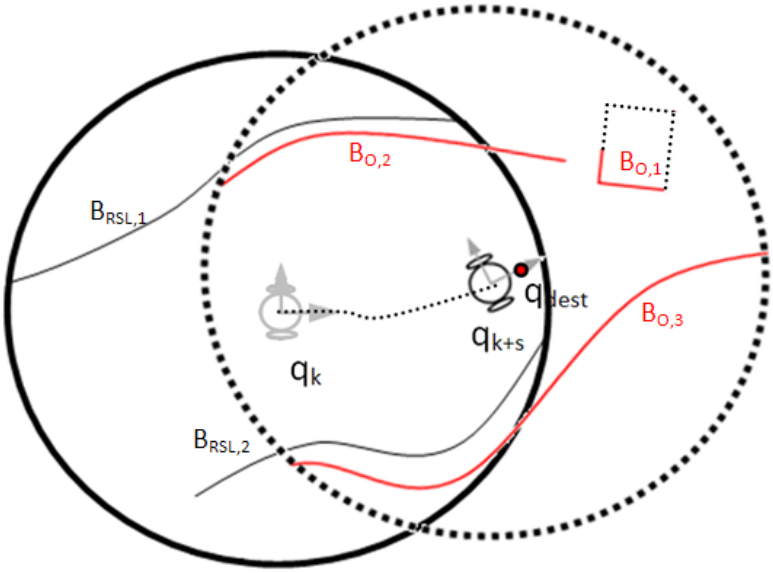

Figure 4.11 Robot in the odometric position q_{k+s} with three obstacles detected within its detection range

As in section 3.3.2, the B-Splines association will be performed initially considering only the points of the control polygon from which curves are obtained (Figure 4.12). In this step, the distances between the points of the control polygons of all the objects (those contained in the RSL on which we work and those observed in the position q_{k+s} are obtained associating the observed curves with the reference curves using the following criteria:

$$\min\left(dist\left(X_{RSL,i,}X_{O,j}\right)\right) \le d_{\min}\ ,\ \begin{cases} i=1\ldots n_m \\ i=1\ldots n_0 \end{cases} \qquad (4.15)$$

Where $X_{O,i}$ and $X_{RSL,I}$ are the control points of the spline observed and the splines in the current LSR respectively, $dist(X_{RSL,i}, X_{O,i})$ represents the Euclidean distance between two points and n_m and n_O are the number of control points of the splines in the LSR and of the observed splines respectively.

At the end of this first stage, the splines with a minimum number μ_{min} of related control points will be associated and other ones for which no relationship has been established will be marked as new curves to be added to the map once the robot's position has been updated by the localization method and if the robot has reached the new target position q_{dest} where it will continue the exploration.

In figure 4.12 we can see how the curves ($B_{RSL,1}$, $B_{O,2}$) y ($B_{RSL,2}$, $B_{O,3}$) have been associated since they have 5 and 7 points related of their control polygons respectively obtained from the equation (4.15). On the other hand, we see that the object $B_{O,1}$ has not been associated with any element and therefore will be marked as a new element to be added once the previously mentioned requirements are fulfilled.

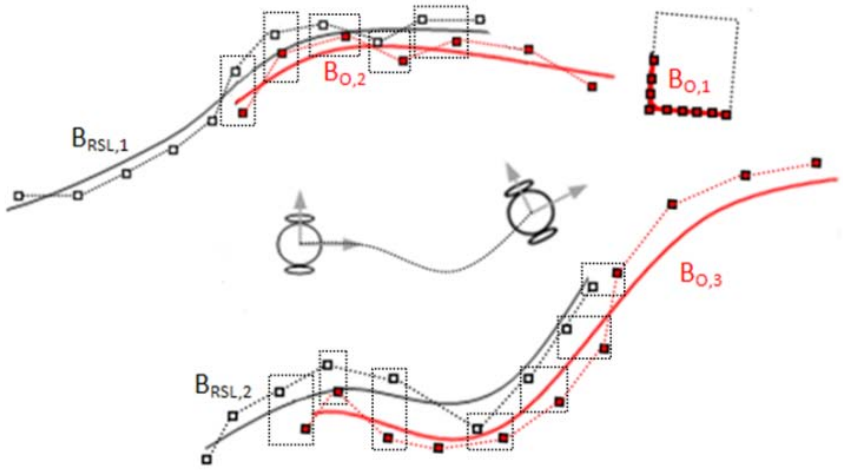

Figure 4.12 Rough association performs with the control points of the curves

If some curves have been associated at this point, the next step is to look for corners and Curvature Zero crossings contained in the related curves. If it has been possible to find some of the searched features, the elements found will be used to perform a precise association between each pair of curves ($B_{RSL,i}$,$B_{O,j}$). The information about the type of feature and of the curvature will be used to avoid errors in this step of association (figure 4.13). If in some of the related curves any elements have been found (lines or too smooth curves), the fine association process will be executed in a similar way to that described in Section 3.3.2.

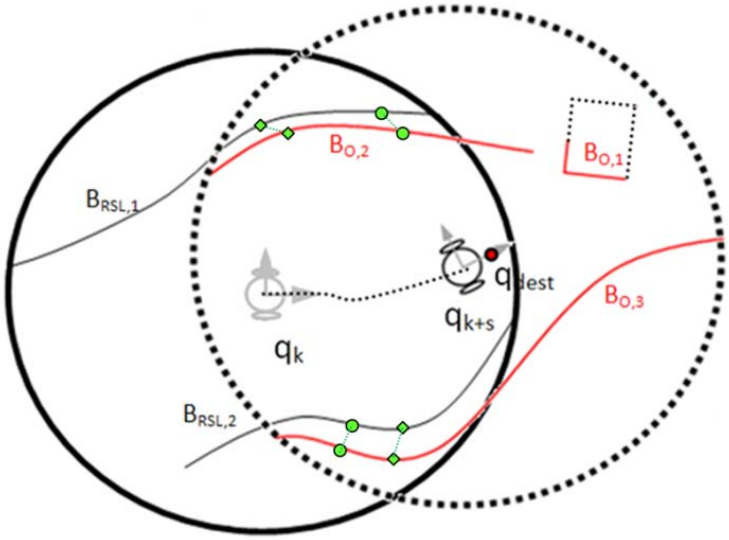

Figure 4.13 Association of curves zero crossing and corners between the RSL curves and curves observed. The green circles represent curves zero crossing while the diamonds represent corners

Once related all the elements, a final process is performed in order to find the initial and final points of the related curves. For this, taking the curvature zero crossing or the corners most extremes as starting points contained in both curves will be taken a number of continuous points on the parametric curve toward the end of this, where the number of points to take will be the maximum number of elements that can be taken in the curve segment of smaller length of the two related from the characteristic point most extreme toward the ends of the curve.

This can be seen in detail in Figure 4.14, where taking the curves related ($B_{RSL,2}$,$B_{O,3}$) we observe that the initial point represented by the blue circle was taken by choosing 6 elements of the parametric curve (shown in blue) from the curvature zero crossing represented by the green circle to the initial end of the curve since the length of the curve segment $B_{O,\ 3}$ since the beginning of the curve to the curvature zero crossing has a larger length than the curve segment $B_{RSL,\ 2}$ from the beginning of the curve to the curvature zero crossing and therefore the elements of the curve segment with shorter length surely will be contained in the other of larger length.

The same process is performed on the final end of the curve where 15 points of the parametric curve were taken from the corner represented by the green diamond given that the curve segment of shorter length from the corner to the end of the curve belonging to $B_{RSL,\ 2}$ contains only 15 points.

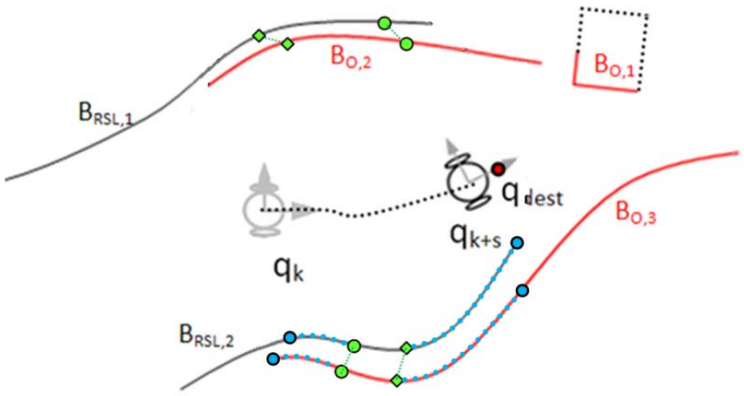

Figure 4.14 Example of how the start and end points of curve segments related are found

Finally, when the association process is finished, the related curves will have an appearance similar to the figure 4.15. Although the correction in the position of the robot will be performed using the start and end points obtained, the curvature zero crossing and the corners are not wasted since on them will be carry out the verification to determine if the localization process has been performed correctly.

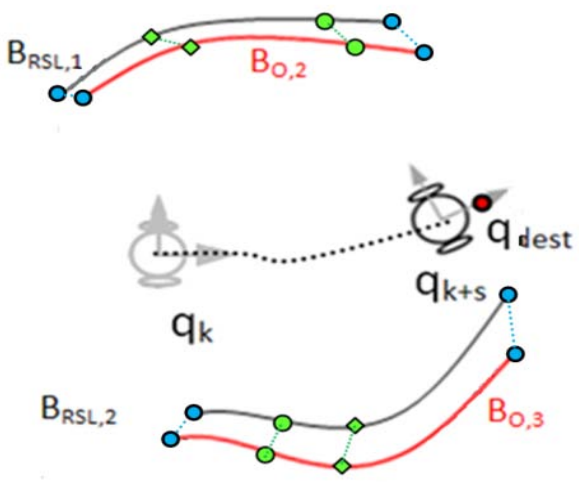

Figure 4.15 Segments of curves related with the process described. The blue circles represent the extremes of the related curves and the green circles and diamonds represents the curvature zero crossings and the corners respectively.

4.2.2 Topologic localization with B-Splines

So far, we have spoken of the tool necessaries for the representation of the environment as well as the tools for the data association. In this section we present the topological localization method developed which forms part of our strategy SPLAM and whose algorithm is shown in Figure 4.16.

TOPOLOGICAL_LOCALIZATION$(\hat{q}, q_{act}, Amb\_S)$

1. $q_{est} \leftarrow q_{act} + \hat{q}.inc$
2. $D \leftarrow$ **SENSOR_DATA**(q_{est})
3. Ss $\leftarrow$ **DATA_SEGMENTATION**(D)
4. PRelAmb,PRelEst,Num_Related_Curves $\leftarrow$ **DATA_ASSOCIATION**(Ss,AMB_S)
5. Localization_Succesful $\leftarrow$ False
6. While(Localization_Succesful=False) and (Iteration $\leq$ Num_Related_Curves)
7. CoefA,CoefE $\leftarrow$ **ANGULAR_COEFFICIENTS**(GA,GE)
8. $e_\theta \leftarrow$ **ANGULAR_ERROR**(CoefA,CoefE)
9. CoefA,CoefE $\leftarrow$ **POSITION_COEFFICIENTS**(GA,GE)
10. $e_x, e_y \leftarrow$ **POSITION_ERROR**(CoefA,CoefE)
11. q_{act}.THETA $\leftarrow q_{est}$.THETA $+ e_\theta$
12. $q_{act}.X \leftarrow q_{est}.X + e_x$
13. $q_{act}.Y \leftarrow q_{est}.Y + e_y$
14. Localization_Succesful $\leftarrow$ **VERIFY_LOCALIZATION**
15. end
16. *return* q_{act}

Figure 4.16 Topological localization algorithm

Every time the robot moves from a position q_k to a position q_{k+1} the estimation of the robot's new position is obtained by adding to the last localized position the increase in x, y and θ obtained from the odometric information provided by the robot. As we know, this new estimate position will be only an approximation to the true position given that the noise in the sensors of the robot causes that the information provided is inaccurate and therefore the estimated position also will be.

With the new estimated position, a new perception of the environment will be made where the information obtained will be placed spatially with regard to this position (function **SENSOR_DATA**). Then, the measurements obtained will be segmented with the function **DATA_SEGMENTATION** using the Dietmayer's adaptive cluster criterion to obtain subgroups of measurements belonging to different objects as we have shown in section 3.3.2.

The next step of the method is to find the relationship between the observed objects contained in the subgroups found and objects contained in the current RSL (section 4.2.1c). This task will be performed by the function **DATA_ASSOCIATION** and will

be the vital center on which will be made the corrections on the estimated position to obtain the real position of the robot.

Once the curve segments have been associated and their respective initial and final points have been obtained, this information will be used to correct the error in the estimated position in two stages (first angular and then in translation x, y) considering the following:

$$\begin{aligned} X_{curr} &= \hat{X} + e_X \\ Y_{curr} &= \hat{Y} + e_Y \\ \theta_{curr} &= \hat{\theta} + e_\theta \end{aligned} \qquad (4.16)$$

With $(x_{curr}, y_{curr}, \theta_{curr})$ as the real actual position of the robot, $(\hat{x}, \hat{y}, \hat{\theta})$ the estimated position and (e_X, e_Y, e_θ) are the errors in x, y and θ. Thus, the angular correction is performed by taking the associated curve segments belonging to the LSR (that will serve as a reference) on which the start and end points of each one of them will be joined by line segments from which the function **ANGULAR_COEFFICIENTS** will get a vector α^{ref} that will contain the angular coefficient of each one of these (line 6) which is calculated as follows:

$$\alpha_{RSL,i} = \arctan \frac{p_{i,F} \sin(\varphi_{i,F} + \theta) - p_{i,I} \sin(\varphi_{i,I} + \theta)}{p_{i,F} \cos(\varphi_{i,F} + \theta) - p_{i,I} \cos(\varphi_{i,I} + \theta)} \qquad (4.17)$$

Where θ is the angle of the robot at the instant when the current RSL was obtained, φ is the angle of the extreme point (initial and final) of the curve segment related to the frame of reference of the robot and p is the distance from the extreme point (initial and final) of the curve segment to the reference frame of the robot. At the same time, the function **ANGULAR_COEFFICIENTS** use the same process over the associated curve segments belonging to the last observation and clearly will give a different vector α^{curr} due to the presence of errors in the estimation (figure 4.11). The equation (4.17) is used in this case taking as reference the estimated position and to $\theta_{curr} = \hat{\theta} + e_\theta$ instead of θ so this equation depends only on e_θ.

Obtained the angular coefficients (Figure 4.17), the function **ANGULAR_ERROR** use the equation (4.18) to find the angular error that will be used to correct angular position. The idea is then to correct the estimated orientation in such a way that the

norm of the difference between the two vectors is minimized in a least-squares sense.

$$e_\theta^* = \arg\min_{e_\theta} \sum_{i=1}^{Obj\_R} Ci(\alpha_i^{ref} - \alpha_i^{curr})^2 \qquad (4.18)$$

The weight C_i in equation (4.18) depends on the reliability of the pair of features (lines and curves where have been found curvature zero crossing or corners will have more confidence than smooth curves).

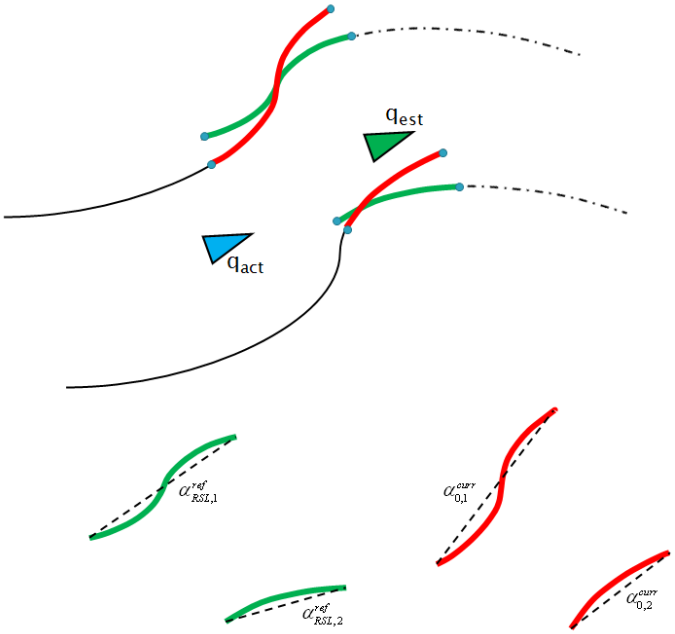

Figure 4.17 Acquisition of angular coefficients

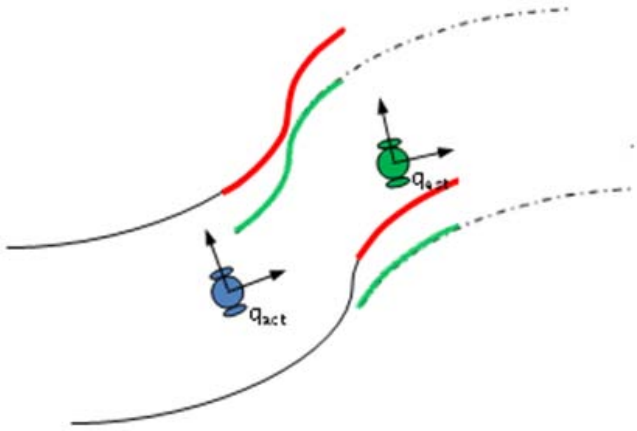

Figure 4.18 Angular correction performed

After the angular correction e_θ^* has been obtained (figure 4.18), the corrections of translation e_x and e_y must be processed. For this correction no special treatment is required, the function **POSITION_COEFFICIENTS** just will take the coordinates of the initial and final points of each pair of related segments (X^{ref}, Y^{ref}) and (X^{curr}, Y^{curr}) with these last depending only on e_x and e_y respectively. Therefore, the function **POSITION_ERROR** obtains the best estimates of these corrections of translation as:

$$e_x^* = \arg\min_{e_x} \sum_{i=1}^{Obj\_R} Ci(X_i^{ref} - X_i^{curr})^2 \qquad (4.19)$$

$$e_y^* = \arg\min_{e_y} \sum_{i=1}^{Obj\_R} Ci(Y_i^{ref} - Y_i^{curr})^2 \qquad (4.20)$$

The errors obtained with equations (4.19) and (4.20) will be added to the estimated position to obtain the real position of the robot (Figure 4.19).

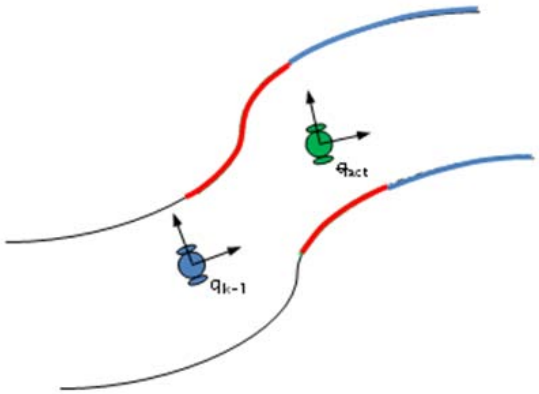

Figure 4.19 Localization process finished

Finally, a verification process is performed by the function **VERIFY_LOCALIZATION**. This process will ensure that the localization process has been successful by measuring the distance between the initial and final related curves and also between the corners and the curvature zero crossing (if exists). If these distances are less than a certain threshold Ψ, the localization process will be considered successful. On the other hand, if any of the measured distances is greater than the threshold, the localization process will be reported as fail and restarted but this time considering only individual curves instead of the whole set. If even considering individual curves the process fail, the position will be reported as unsafe and therefore unusable to extend the map.

Considering our localization process, we can consider the following question. Why use all the related elements and not just one of them?.

The reason of this phenomenon is that although we could effectively use only one curve related to correct the position of the robot, the use of the full set allows to obtain redundant information which allows to increase the robustness of the correction procedure in presence of conflicting data (Figure 4.20c). This is, some times and due to possible associations errors, the use of a single curve would result in a incorrect localization that in absence of more data will not be able to be corrected (Figure 4.20b).

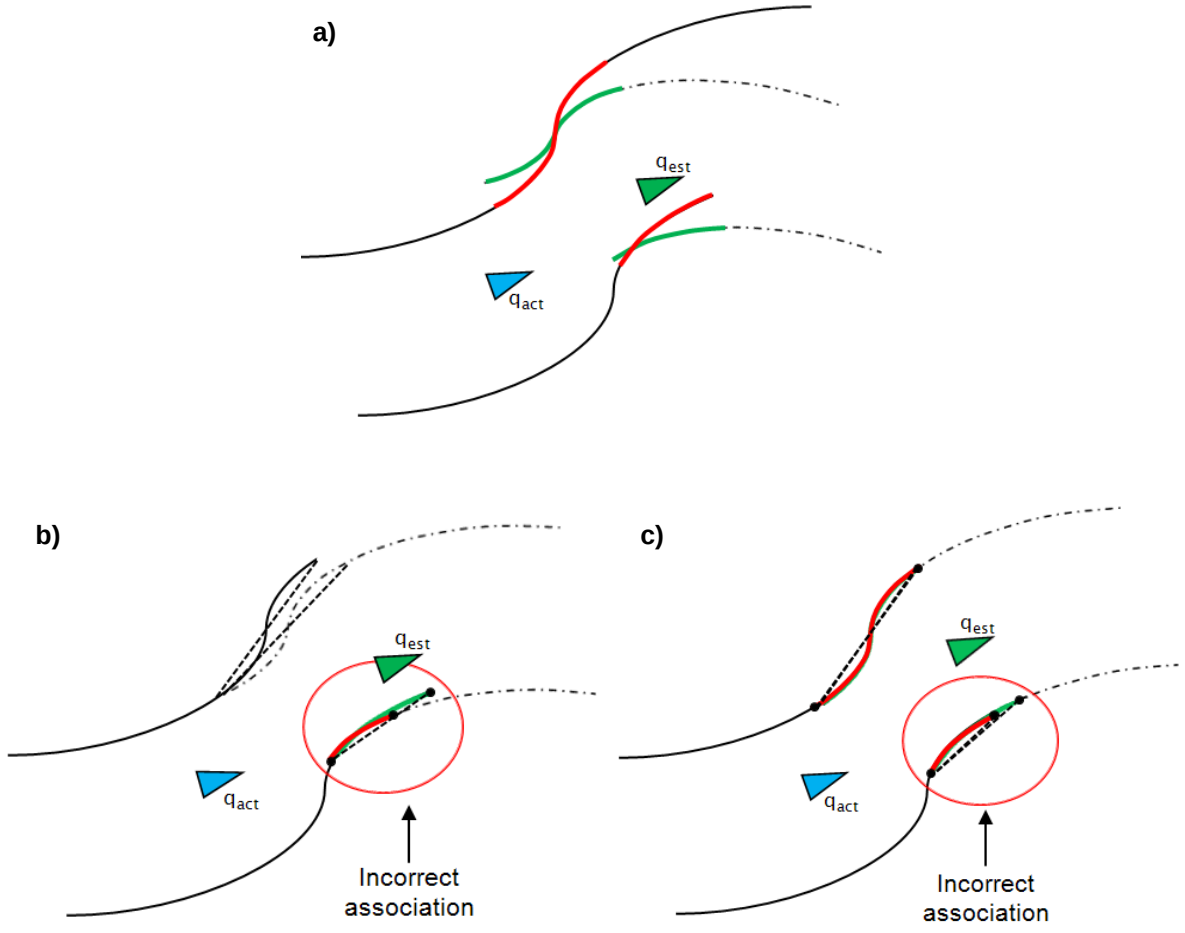

Figure 4.20 Localization process with incorrect data association. a) Curves associates. b) Localization using only on curve that has been incorrectly associated. c) Compensation of the localization when more of one curve is used, even if one of them is incorrectly associated

4.2.3 Extension of the map

As we know, the environment map is incrementally built each time that the robot reaches a predefined position with the intention of exploring a free frontier (Section 4.1). The objects perceived as in section 3.3.4 can be classified after the association process required for the localization of this new position as:

- **New objects** if they have not been associated with any object belonging to the previous RSL.

- **Partially associated objects** whose information will be used to extend objects in the map

a) Adding new objects to the map

The first type does not present any problem, when a new observation is not associated with any of the splines in the map, these are considered as new elements and the spline that defines its geometry is added to the map. When we talk of add one spline, we refer to add the control points of the spline to the vector that represents the explored map. This is done as follows:

$$Map = \begin{cases} O_1 \\ O_2 \\ \vdots \\ O_n \\ NO \end{cases} \tag{4.21}$$

Where Map is the vector containing the information of the map constructed so far, O_j represents the curves of variable size and NO represents the new object that is being added which is a vector containing the control points that form the new curve.

b) Extending objects in the map

Regarding the second type, the process to follow is different to that presented in section 3.3.4. This because in our project we work with unclamped B-Splines which makes that the extension process is different. The technique used for the extension of objects will be explained with the image 4.21.

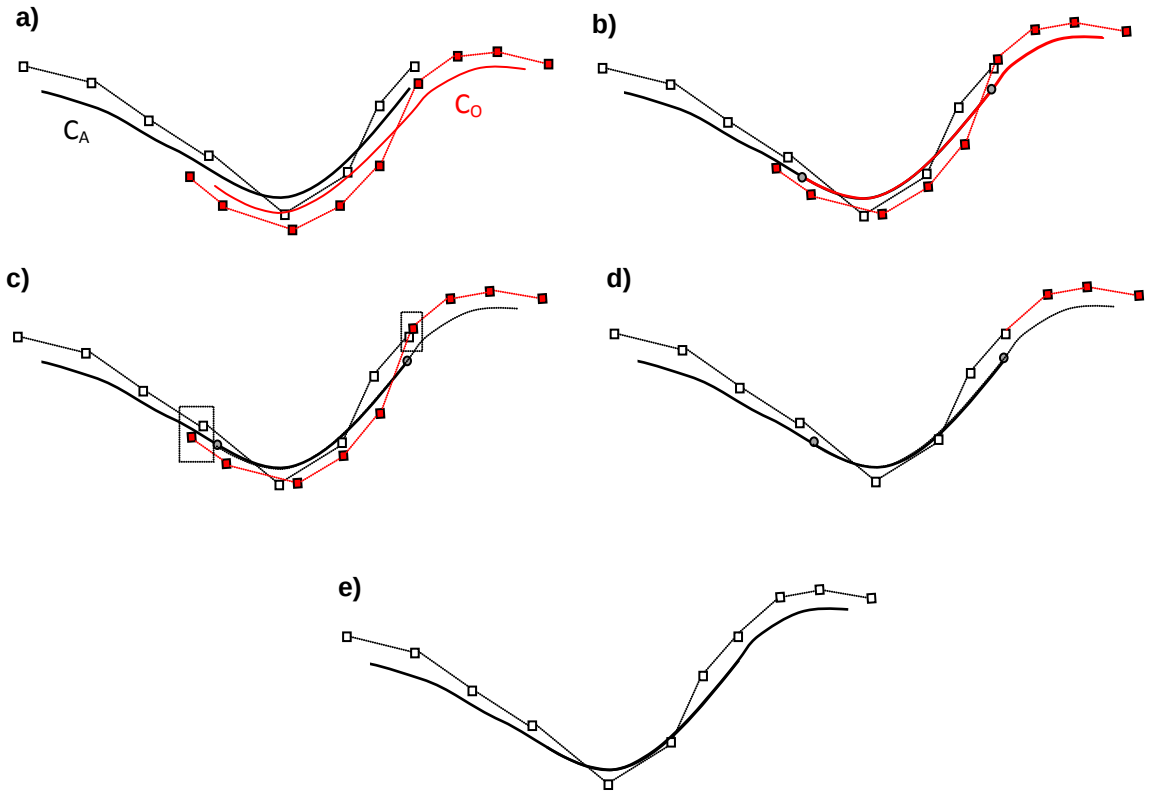

Figure 4.21 Extension of the spline of the map with new data. a) Initial situation before the location. b) Configuration of the curves after location. c) Search for the control points belonging to the association points of the curves. d) Elimination of the control points that describe the curve segment already in the map. e) Extended curve and their control points.

Given two curves associated CA and CO (Figure 4.21a) whose geometric representation in the associated segments is, if not equal at least similar, the objective is to eliminate in some way the control points of the curve observed representing the overlapping area between the curves. For this, once the position of the curves has been corrected through the localization method (figure 4.21b), we look for the control points in the new curve belonging to the initial and final points of association of the curve segments associated (gray circles in Figure 4.21b) whose description is already in the map. With these points obtained (Figure 4.21c) the next step is to eliminate these two control points and the control points that are contained in this range to then connect the control points of the curve belonging to the map with the remaining control points of the new curve (figure 4.21d):

$$X_t = X_e \cup X_n \qquad (4.22)$$

Where X_t represents the control points of the extended curve, X_e represents the control points of the curve in the map and X_n represents the control points of the observed curve. At the end of the process the control points of the new extended curve (Figure 4.21e) will be added to the vector of the map in the same position where it was the original curve.

4.3 Kidnapping

The kidnapping is one of the hardest problems to solve in the SLAM field. To kidnap a robot means to take it in the course of its work and to move it quickly to any point in the environment, without telling it that it has been kidnapped. It is similar than knocking a human and moving him to a different place. This problem differs from the global localization problem in that the robot might firmly believe to be somewhere else at the time of the kidnapping.

Even when the kidnapping problem is an event that rarely occurs, it may happen naturally due major landslides in the work area or even the robot itself. However, the kidnapping problem also arises as a form to check the robustness of the localization method and its ability to recover from catastrophic localization failures.

The approach proposed to solve the kdnapping problem is described in detail below.

4.3.1 Collection and management of marks

The most important process for any kidnapping method is the collection and processing of certain hallmarks that will help the system to relocate the robot once it has been kidnapped.

a) Handling characteristics

The exploration process allows us to find distinctive features (characteristics) of the environment and store them in a list. These features will undergo a treatment that will simplify their use in our approach.

First, the distances of the characteristic features of the position q_{curr} are obtained and sorted in ascending order. At the same time, the longest non-recurring distance is selected and the other features are rotated in order to get a 0 degrees angle with it. Then, the angles between the characteristics are obtained. If the distances between

two different characteristics have the same length, they will be sorted according to the obtained angles. It is important to note that the rotation should be performed in a not repeated distance. This process is described in figures 4.22 and 4.23.

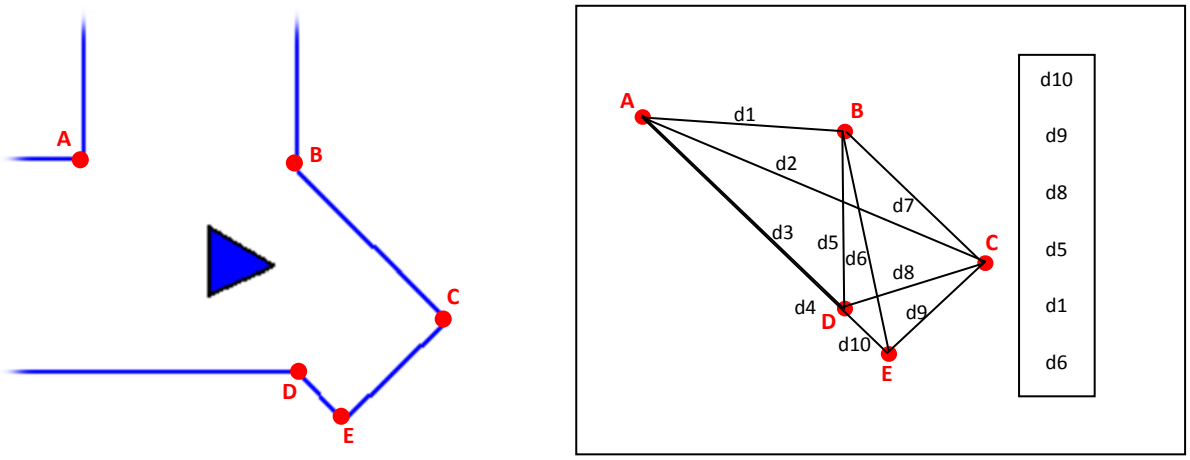

Figure 4.22 Characteristic marks found

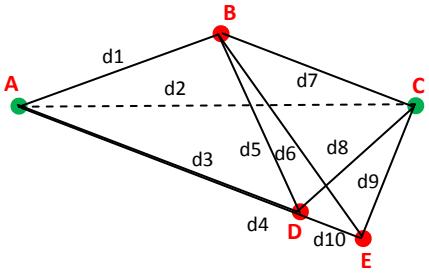

Figure 4.23 Points rotated to get a 0 degrees angle for the slope d2

b) Generation of the code

A code that provides support in the relocation time is generated, after the data of distances and angles are obtained. The code contains the number of marks included N, the ordered distances, D_n, and the ordered angles of each of the lines connecting the marks (denoted as A_n). Besides, the code keeps the marks position in the map built before the kidnapping (X_i,Y_i, i = 1 ... n).

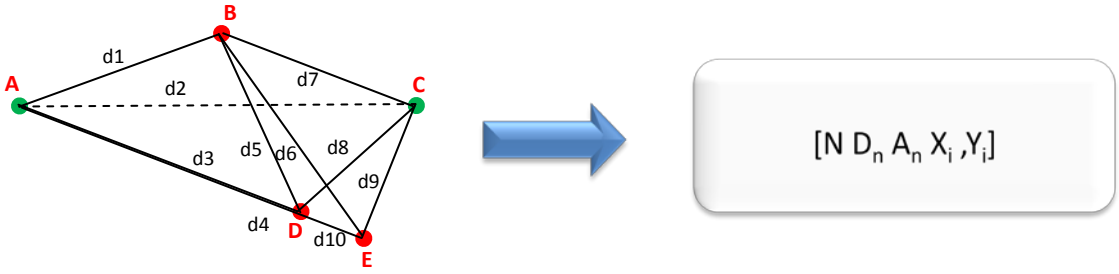

Figure 4.24 Code or signature generated with the process described

4.3.2 Kidnapped robot

An important part of the kidnapping is the knowledge of the exact moment when the robot has been moved from the original position to an unknown location. This thanks to an inherent characteristic the methodology of exploration presented in section 4.1, the local safe region (LSR).

If at any time during the scanning and location processes, the robot can not associate any of the curves found in the LSR at the exploration time, with the curves obtained in its current position, q_{curr}, this means that the robot has been kidnapped. At that time, the robot will enter in a kidnapping state, after which the system will try to find and relate some of the characteristics stored in memory with its new features.

To maximize the information and minimize the exploration time, it is made an auxiliar structure that can store the environment built up to the moment and restart the construction of the new environment, considering the new position as $(0,0,0)$. It means that no matter the last position of the robot in the previous map, after the kidnapping is identified, the system will reset its position at coordinates $(0,0,0)$. As mentioned, the above map is not removed, this is stored with the intention of finding a known area, and to adjust the new map to this position and merge both maps to get only one.

It should be mentioned that if any characteristic zone or with high information content is detected before the kidnapping, the robot will not be relocated on the map being created and therefore the kidnapping will not be resolved. In this case the previous map is removed and it will be considered that there is no kidnapping.

4.3.3 Recovery of the kidnapping

Otherwise, if the robot is kidnapped and there are several codes of the areas already explored, the process continues as described above, building a new map, until a distinctive area, that allows the following procedure, is found:

A code considering the described steps is created. The stored list is verified to look for codes with the same number of marks than the marks found. If the number matches, the elements corresponding to the distance and angles will be used to look for a coded that contains similar data, deferring at most in 0.01 m for the distances and in 0.01 radians for the angles (these heuristic values are set after a series of experiments).

If at this moment, there is a candidate mark, the coordinates of the stored marks are obtained to analyze which marks corresponds to the newly found, using a cross-reference as follows: The ordered distances lists that contains the marks that make up each distance (considering the stored and the new marks lists) are used to relate them in two possibly ways, as illustrated in the figure 4.25:

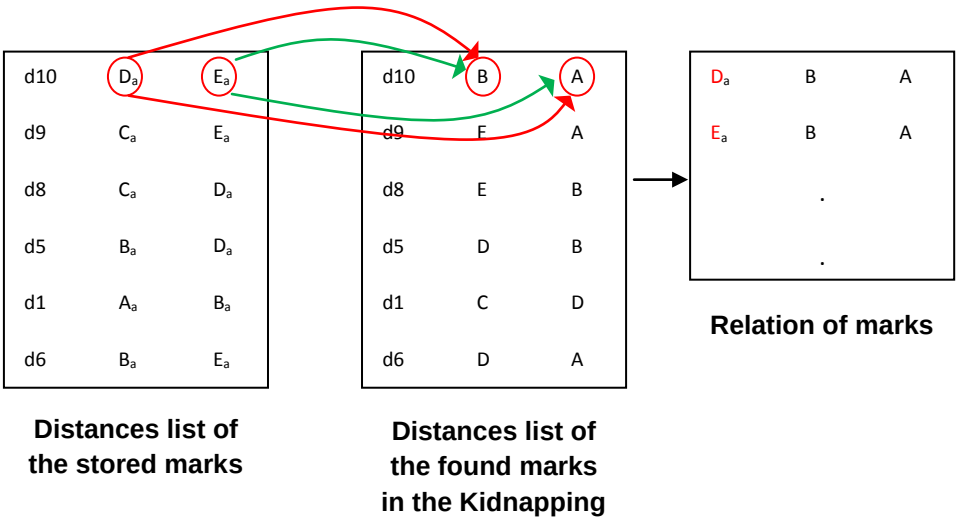

Figure 4.25 Mark relation process

Once the relationship matrix is obtained, it is sorted according to the first column of the nodes stored in the list. Once ordered, the elements in the first column are extracted to check which element appear $N\text{-}1$ times in columns 2 and 3, where N is the number of marks contained in the code. Finally, this element will be the mark

corresponding to the stored mark. This method is repeated for all the different elements of the first column (Figure 4.26).

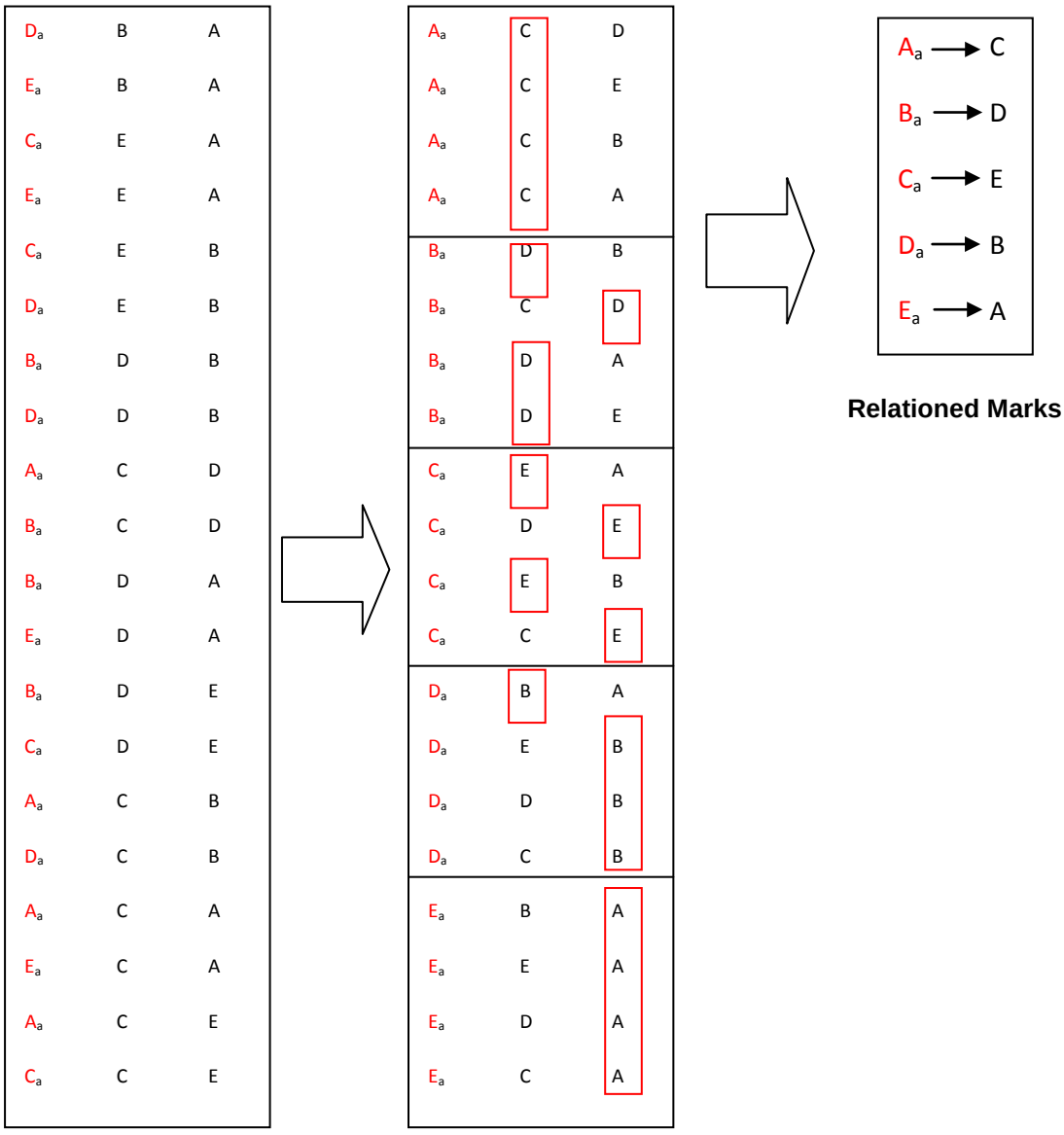

Figure 4.26 Relationship found

When the relationship process is finished, we know exactly which characteristic node of the observable area corresponds to the same node in the stored area. The above is illustrated in figure 4.27.

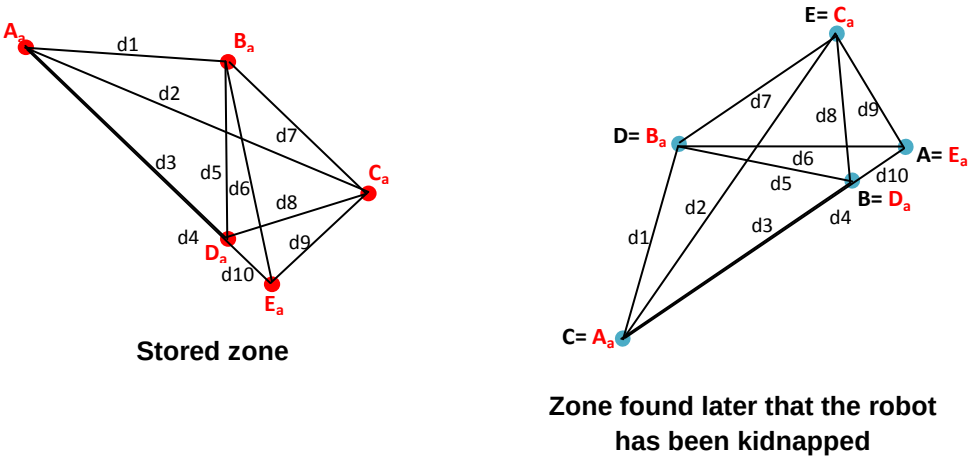

Figure 4.27 Associated points

Once the zones and the nodes have been linked, the next step is to make a correction, first by finding the angle and then the displacement. This process will allow us to find a correlation between the stored and the explored areas. In other words, the rotation and the translation take place in the new constructed map, after the kidnapping, in order to merge it with the stored information.

The correction is performed similarly than in the proposed location method. First, a complete graph with the marks is used, because the relationship between the nodes of the two graphs is known and there is no need to look for the lines that must be associated. The task now is to find the difference of angles between the lines, allowing angular correction with the same formula used in the localization process.

After the angular correction e_θ is made, the translation corrections e_x and e_y are processed, using again the full characteristics graph built before.

Once the translation and angular corrections are made, they are applied to the structure built during the kidnapping. Finally, to ensure a good merging, a last global localization process is performed. Curves segments from the new environment whose partner are close enough to the stored curves in the map built before the kidnapping are selected if this association between the maps is correct. Then, the global localization process has been carried out successfully.

Finally, in order to store only a structure that corresponds to the complete merged map. The nodes in the new environment with a distance less or equal to a certain

threshold with respect to the stored map nodes are discarded (as they are considered repeated). In the same way the nodes with a distance greater than this threshold, but within the LSR of some node, will be added as child nodes of the node that owns the LSR. The exploration will continue in the leaf nodes of the new structure.

4.4 Conclusion

This chapter has shown the development of a series of tools that allow an efficient exploration of environments.

First we have developed an extension of the basic exploration method SRT called REG which transforms the tree structure used into a graph, this allows to robot to traveler in a more efficient way since it can take shortcuts to go from one place to another.

The REG exploration approach also use a simplified criterion to find the next position to explore which allows to stay into a specific region until this has been fully explores. The main advantage of this is that the robot will travel short distances until the nearest position that should be explored. In this way, we avoid that the robot travels long distances in which there will be no gain of information. At the same time, this strategy prevents the robot has to return to zones where it had already been.

The topological-SLAM presented, show a new method of data association using B-Splines. Here, all the information contained in the curve is used to perform a more accurate data association that as we know, represents a very important step for the robot localization since incorrect or inexact associations between the curves will cause divergences that might unleash in an erroneous map that will not be able to be used and therefore in an overall system failure.

With respect to the localization method, we have presented an approach based on unclamped B-Splies that use the structure built for the REG method. The main advantage of this is that e robot use just information local of a global frame to performs the localization. This limits the calculations only to the interpretation and association of objects contained in the current LSR of the current node.

Finally, we have presented a mapping method that performs in a very easy and intuitive way, the extension and addition of curves representing the objects in the environment to the map.

The results and comparisons with other methods will be presented in Chapter 5.

Chapter 5. Experimental Results

In this section we have carried out numerous experiments with both real and simulated data in order to verify and validate the procedures and algorithms proposed in this thesis. The data obtained with simulation experiments have allowed to verify the accuracy and consistency properties of the algorithms by comparing them with existing procedures and algorithms. On the other hand, experiments with different data from real environments have allowed to verify the applicability and effectiveness of these techniques.

Section 5.1 presents some experiments that help to validate the usefulness and efficiency of the method of exploration proposed showing the versatility that offers the transformation of the exploration tree into an exploration graph and the advantage of having a frontier control.

Section 5.2 presents some experiments to help understand the technique of approximation of data acquired through a laser sensor showing the importance of order and type of restraint at the time of constructing the curves.

Section 5.3 presents the results of the proposed solution for the problem of SPLAM which are compared with results obtained using SPLAM methods based on the extended Kalman filter showing the properties of the algorithm from the point of view of accuracy and consistency.

To end with the proofs, section 5.4 show the maps constructed with data acquired in real environments.

Finally, the chapter closes with some conclusions collected in Section 5.5.

5.1 Exploration Methods

As we have mentioned repeatedly, the control of movements (known as exploration in this area) for the task of SPLAM, plays a very important role in the performance of the strategy and for the acquisition time of the map. Unlike the localization and mapping necessary in the SPLAM task, the efficiency of the exploration method can be analyzed and tested individually. Therefore, in this section are performed comparative tests between the SRT exploration method that is used as a base, and the proposed exploration method developed from it.

The tests carried out to the methods were made using a simulated and real diferential robot Pioneer P3DX (Figure 5.1) equipped with a laser sensor Hokuyo URG-04lx (Figure 5.2) which has a detection range of 0.02 to 4 meters approximately with a typical deviation (σ_L) of 1% of the measure, an angular resolution of 0.36 degrees and a scan angle of 240 degrees. Furthermore, the robot has a ring of 16 ultrasonic sensors of which 6 of them positioned in the rear are used to obtain information from the environment in the 120 ° where the laser sensor cannot see.

Figure 5.1 Robot Pioneer P3DX

Figure 5.2 Laser sensor Hokuyo URG-04lx

The environments used are part of the installations of the Laboratory of computer sciences, Robotics and Microelectronics of Montpellier (LIRMM) and they are shown in the images 5.3 and 5.4.

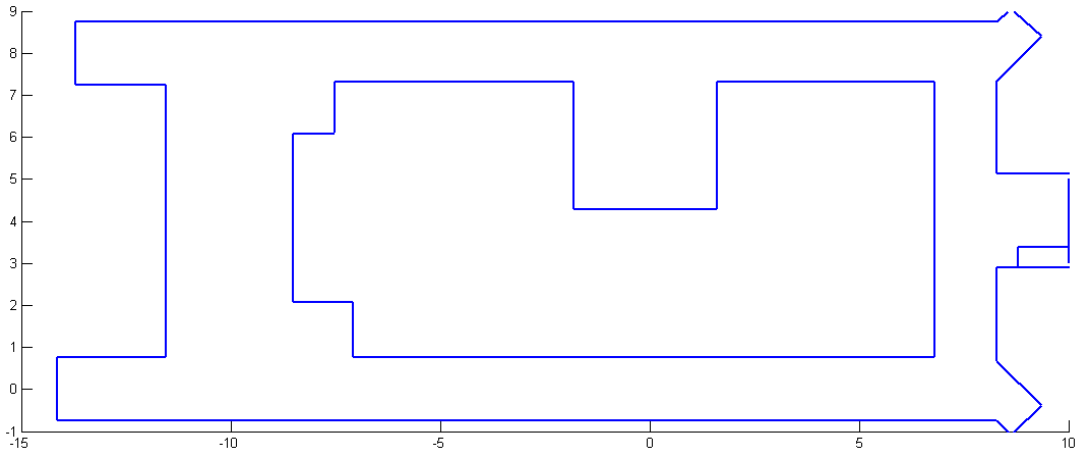

Fig 5.3 LIRMM Office Environment

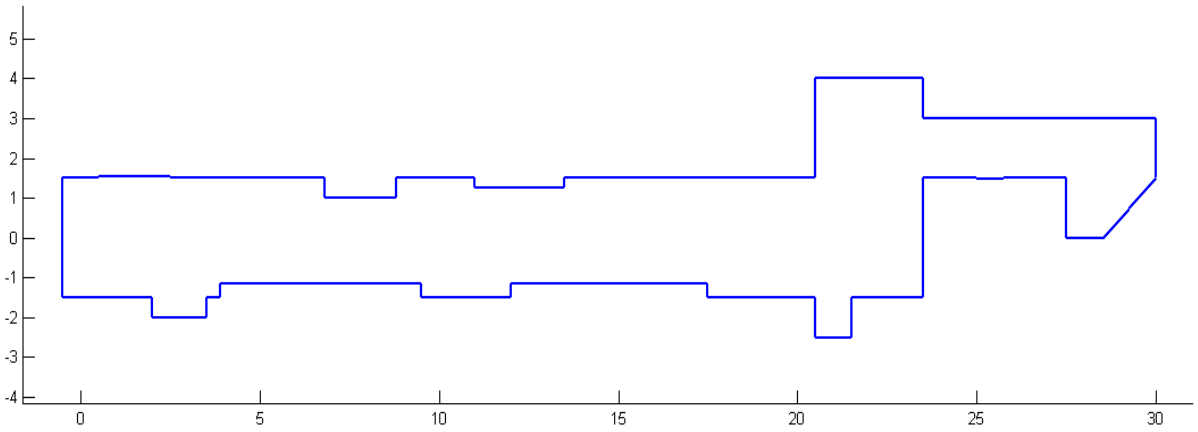

Fig. 5.4 LIRMM Corridor Environment

As we have mentioned, both the method of exploration of unknown environments SRT developed by Oriolo et al. [**Oriolo et al 2004**] And the REG method developed in this thesis, generate a data structure that determines the paths by which our robot can travel.

For the case of SRT, the generated tree structure and the lack of border control to indicate what nodes have not been fully explored force the robot to travel 2 times the navigation structure generated (Figure 5.5 and 5.6) to complete the task; therefore, the time required and the length of the path to cover all the environment are completely dependent on the number of nodes that contain the structure. Also, the shape of the structure used in this method would result particularly inappropriate if it is conserved as a reference for future navigations; this because in some cases the

robot will be forced to retreat toward a parent node when the destination node is in another branch even if it can be access directly from the current position.

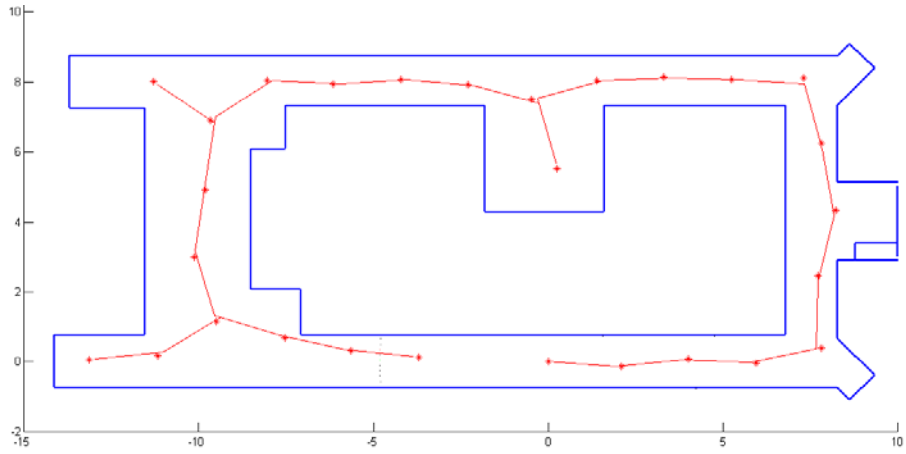

Figure 5.5 Exploration tree obtained with the SRT method on the LIRMM office environment

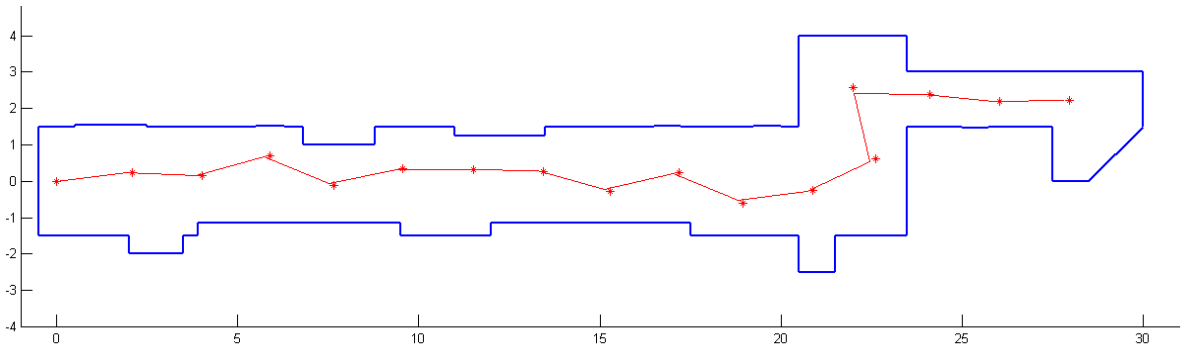

Figure 5.6 Exploration tree obtained with the SRT method on the LIRMM corridor environment

On the contrary, the graph structure used by the REG method (Figure 5.7 and 5.8) as the integrated concept of frontier control allows an exploration much more versatile and efficient since the method knows exactly where to direct to the robot in order to continue the exploration, while the structure allows to find an efficient route to this new position thus allowing a shorter path and a lower time needed for the exploration of the environment.

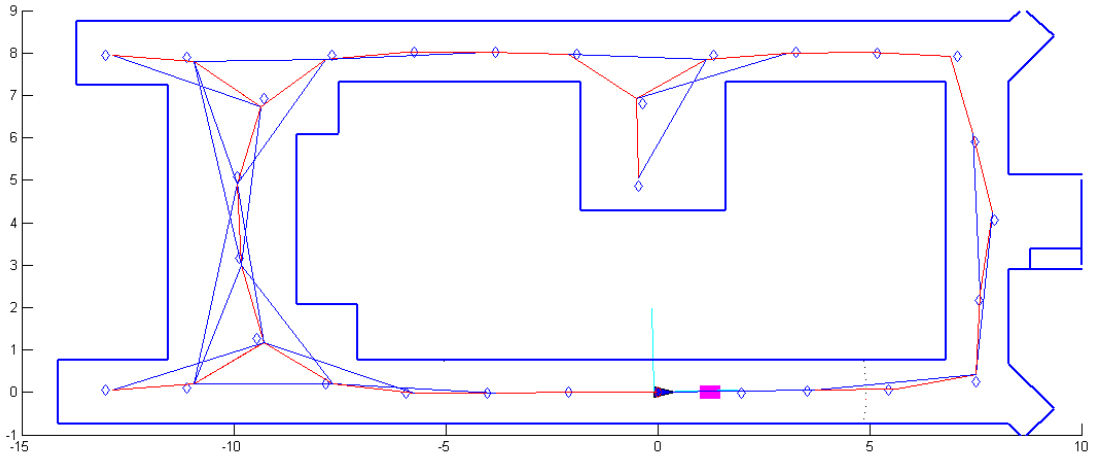

Figure 5.7 Exploration graph obtained with the REG method on the LIRMM office environment

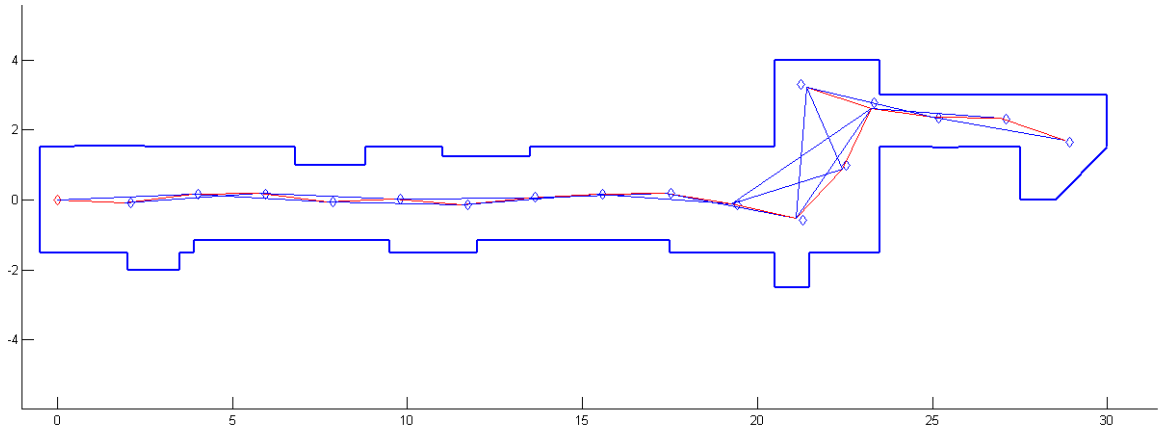

Figure 5.8 Exploration graph obtained with the REG method on the LIRMM corridor environment

The above is verified in Figure 5.9, 5.10 and 5.11 where the data obtained with the two methods are confronted to verify the assumptions that we have made.

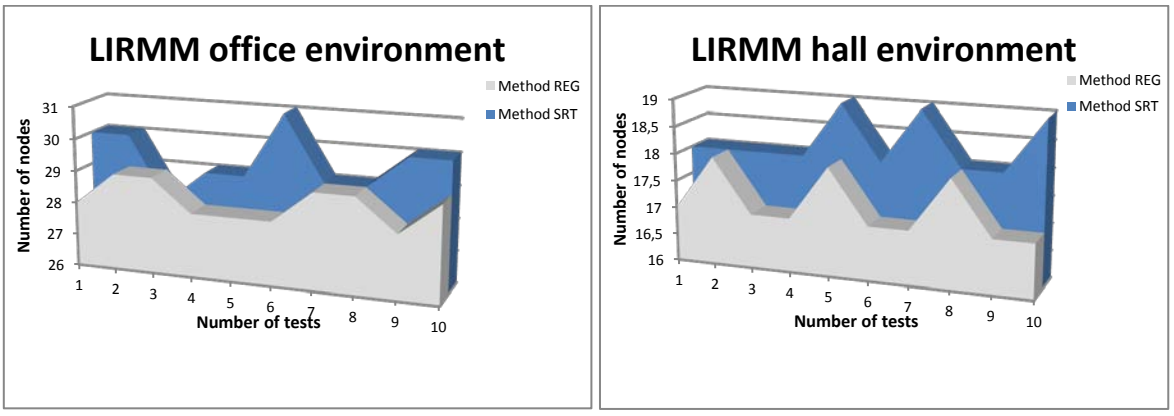

Figure 5.9 Nodes needed to cover the LIRMM office and corridor environments respectively on the basis of 10 tests

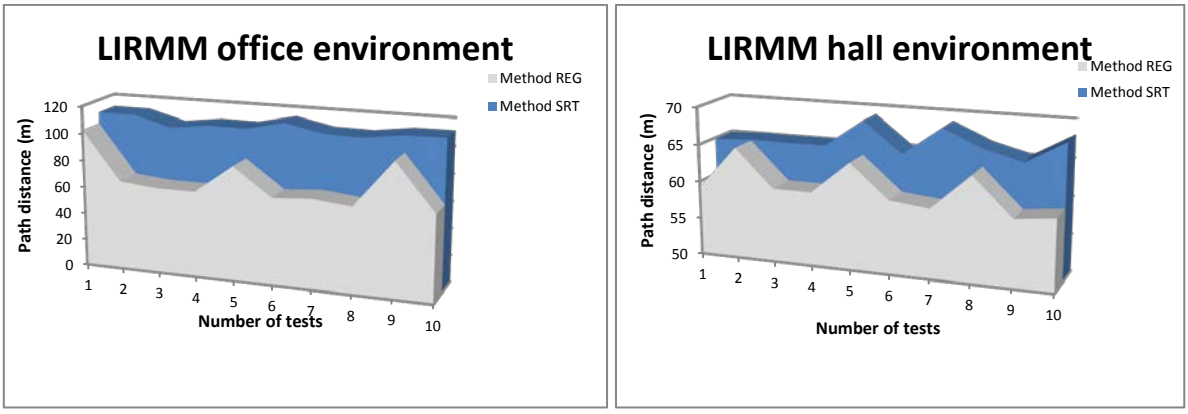

Figure 5.10 Path distance traveled to cover the LIRMM office and corridor environments respectively on the basis of 10 tests

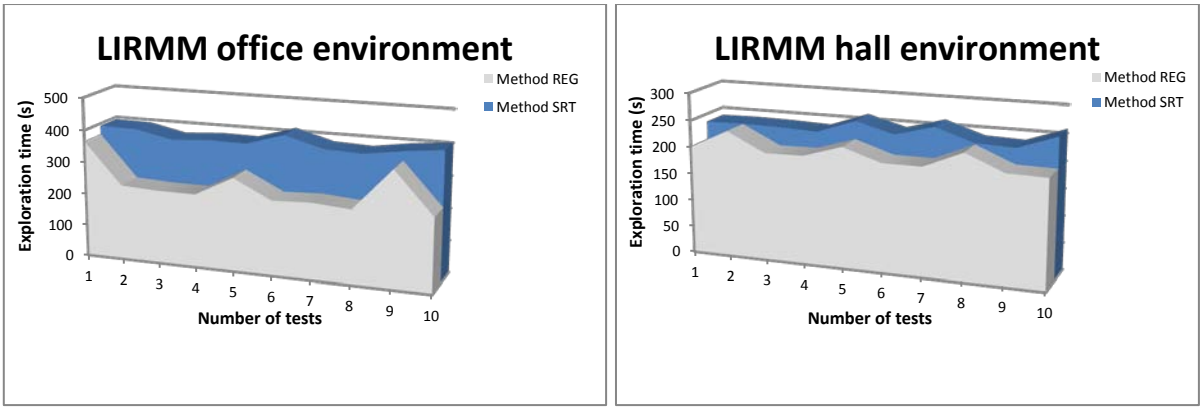

Figure 5.11 Time needed for the exploration of LIRMM office and corridor environments respectively on the basis of 10 tests

Finally, and based on the information presented in the previous graphs we can conclude the following:

- It is noted that the nodes needed for the exploration (Figure 5.9) in most cases is lower in the REG method than in the SRT method, this because the REG method attempts to obtain in every border of each node as much information as is possible and so the robot is always directed to the point over the randomly chosen frontier where this goal can be reached. On the contrary, the SRT method works on the basis of finding a valid random direction (in a free frontier) regardless of the information gain.

The above deduction is valid to some extent, because in hall type environments with too narrow corridor will not be found a too marked difference between the two methods with respect to the number of nodes needed to complete the task.

- Another element that must be considered is the length of the traveled route in the exploration of environments. In Figure 5.10 we can see that traveled distance in the REG method is in most cases less than the distance traveled by the SRT method. This is because the graph structure allows to use shortcuts when the robot has to move within the structure something that within the rigid structure of the tree is not allowed.

  Another important aspect that affects the traveled route is the knowledge of the nodes that have not been fully explored; thanks to the introduced concept of frontier control, the REG method knows what nodes should be revisited once the node in which it currently is has been fully explored; this, unlike the methodology used in the SRT, prevents that the robot have to revisit unnecessary nodes without possibility of exploration.

- Finally, in Figure 5.11 we see that in most cases the execution time for the exploration of environments is lower in the REG method than in the SRT method; phenomenon that is closely linked just as in the previous point to the versatility of the graph structure that allows the path planning using shortcuts and also to the fact that the robot knows exactly where to go without the need of revisit and examine nodes that cannot provide more information (frontier control).

5.2 SLAM Method

The SLAM strategy developed in this thesis as well as any other solution proposed in this field is validated using as criteria the computational performance, map quality and consistency of the algorithm. However, unlike methods whose environmental representation is based on specific geometries and where much of the information acquired by the sensor is wasted, our approach attempts to exploit the maximum amount of information possible provided by the sensor thus avoiding dangerous simplifications.

Therefore, in addition to the mentioned tests to validate our SLAM method, we also include a section dedicated to the treatment of information provided by the sensor and the impact that this management has on our method.

5.2.1 Approximation of data points

One of the main priorities of this thesis is to exploit the maximum amount of information possible provided by the laser sensor which will be used to obtain the representation of the environment using B-spline curves as was seen in Chapter 3 and 4.

To achieve this goal some basic elements in the acquisition and processing of information should be considered such as:

- **Angular resolution of the laser**. While greater it is the resolution of the laser sensor greater will be the amount of information and therefore the details of the environment making it possible to identify more easily characteristic traits.

- **Degree of the curve**. While more elevated it is the used degree smoother will be the curve.

- **Time-to-curve**. The treatment time should be considered at every moment time since if it is too high the SPLAM method cannot be executed in real time.

In this way, in this sub section we will try to briefly study the mentioned parameters and the influence that they have on the quality of the curve and over the modeling time. For this, we will use data from a real environment with different laser sensor angular resolutions (called clusters) and from which we obtain B-spline curves of different degrees.

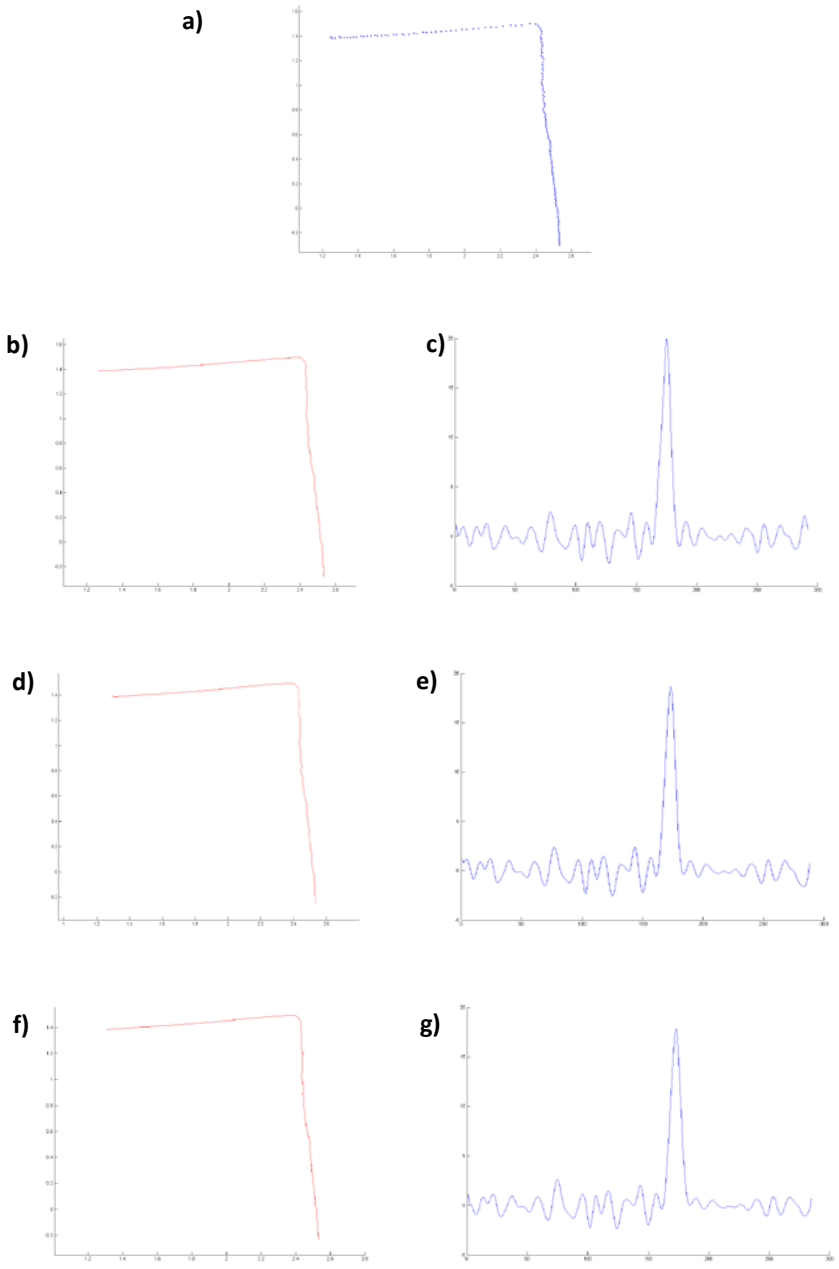

Figure 5.12 Segment acquired with a laser sensor resolution of 0.36° . a) Environmental data points obtained with the laser sensor. b) y c) Curve of degree 3 obtained from the laser data and its curvature respectively, obtained in a time t=0.038101. d) y e) Curve of degree 6 obtained from the laser data and its curvature respectively, obtained in a time t= 0.039121. f) y g) Curve of degree 9 obtained from the laser data and its curvature respectively, obtained in a time t= 0.04255

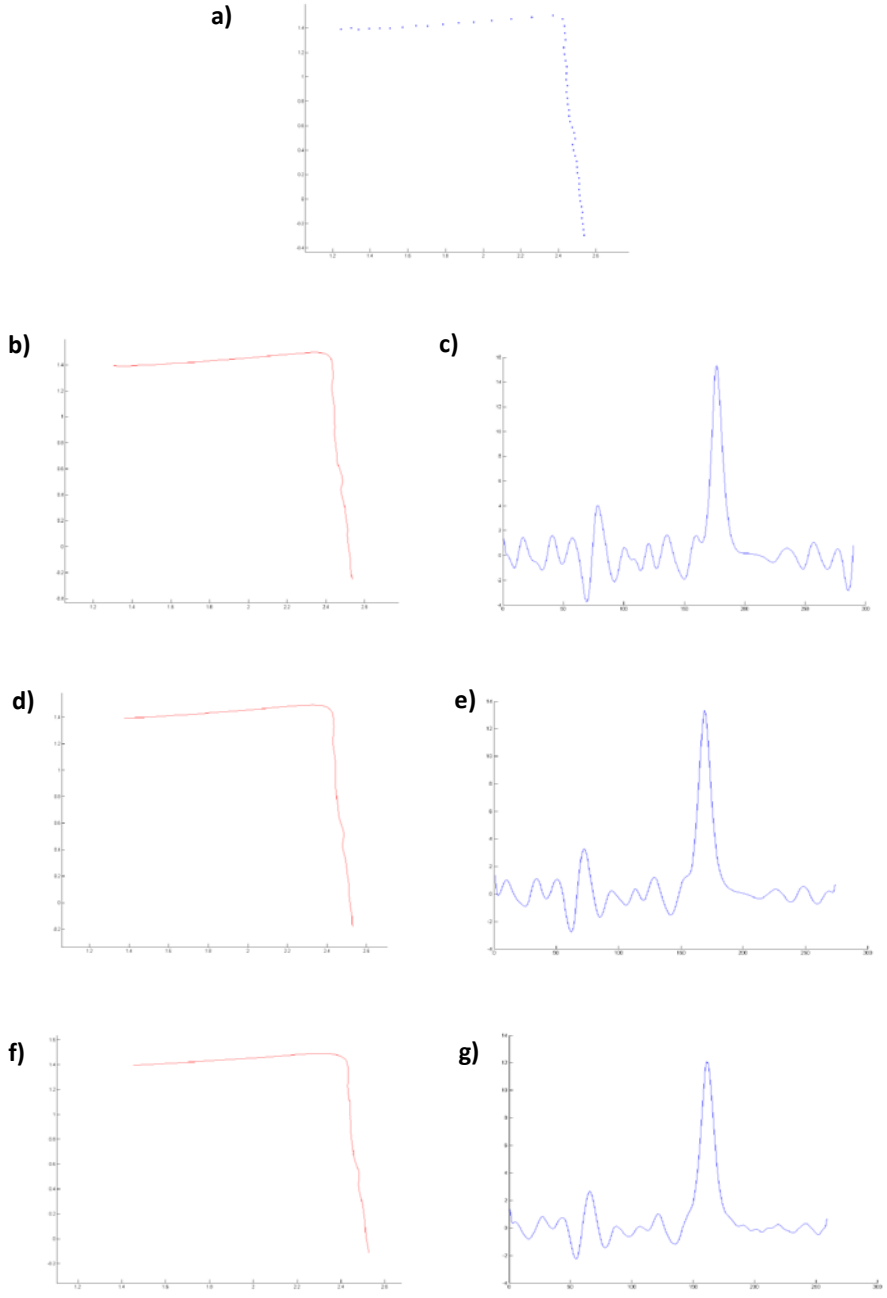

Figure 5.13 Segment acquired with a laser sensor resolution of 1.08° . a) Environmental data points obtained with the laser sensor. b) y c) Curve of degree 3 obtained from the laser data and its curvature respectively, obtained in a time t= 0.0361902. d) y e) Curve of degree 6 obtained from the laser data and its curvature respectively, obtained in a time t= 0.038916. f) y g) Curve of degree 9 obtained from the laser data and its curvature respectively, obtained in a time t= 0.043407

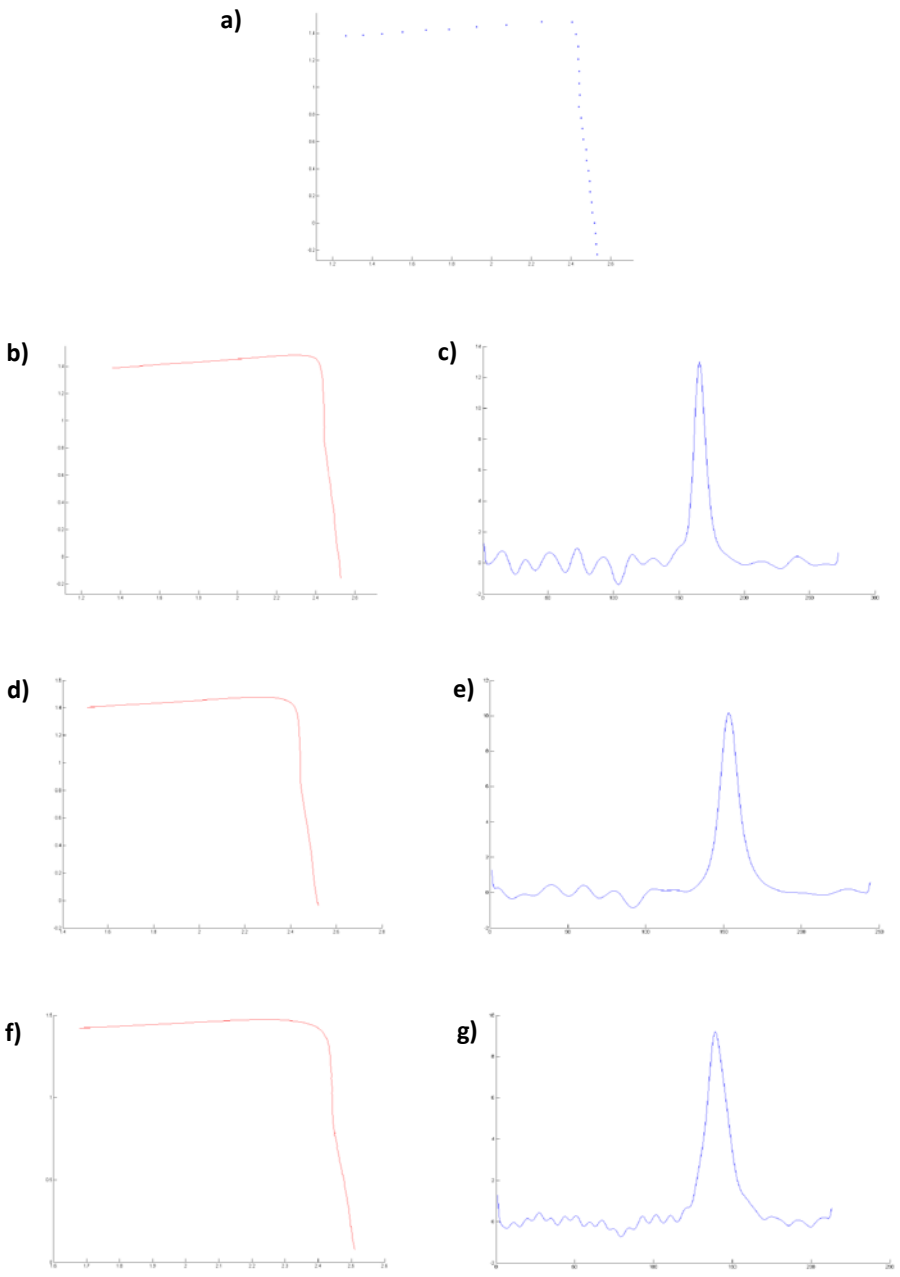

Figure 5.14 Segment acquired with a laser sensor resolution of 1.8° . a) Environmental data points obtained with the laser sensor. b) y c) Curve of degree 3 obtained from the laser data and its curvature respectively, obtained in a time t= 0.026312. d) y e) Curve of degree 6 obtained from the laser data and its curvature respectively, obtained in a time t= 0.031838. f) y g) Curve of degree 9 obtained from the laser data and its curvature respectively, obtained in a time t= 0.032502

Figures 5.12, 5.13 and 5.14 show the results of approximating a segment of environment using different cluster or resolutions over the laser that we use. The total length of the curve that we pretend to approximate is 11.6 meters.

Figure 5.12a shows the result of obtaining the data points of the environment with a sensor angular resolution of 0.36 °. In this case we have obtained 620 points along the length of the object. Figure 5.12b shows the effect of approximating these 620 points with a B-spline curve of degree 3. At its side, Figure 5.12c shows the calculated curvature belonging to the B-spline of degree 3. Similarly, the figures 5.12d and 5.12e show the result approximate the data points with a B-spline curve of degree 6 and the calculated curvature associated with it. Finally, the figures 5.12f and 5.12g show the approximation of the sensor data with a B-Spline curve of degree 9 and its associated curvature.

The images 5.13 and 5.14 just as in the previous paragraph shows the data obtained with an angular resolution of 1.08 ° and 1.9 ° degrees from which we obtain 208 and 124 data points respectively. In these, just as in the image 5.12, we show the effect of approximating curves using different degrees of curves which is reflected in the calculation of the curvature.

As it can be appraised, the result is perfectly logical, as the object to represent compte with a smaller number of data, we obtain smoother curves and the degree of the curve affects them more. On the contrary, the curves with too much information require higher order B-Splines if we want to get smoother curves. This can be verified in the curvature graphs that are found to the right of each approximated curve. In them we can see that when the degree of the curve increases its curvature decreases; at the same time, we can observe in these graphs that having fewer data points becomes more difficult to determine the exact position of the point with greater curvature because the features are not as accentuated due to lack of information.

Finally, based on data obtained we can conclude that it must be found a compromise between the resolution of the laser to obtain the greater amount of information available from the laser sensor and the degree of the curve to eliminate as much noise as possible on the measures without removing valuable information, given that a high degree will smooth the curve in such a way that useful information would be discarded. For this reason and because our method is based on using as much information as is possible for the analysis of the curves, we have decided for the

higher angular resolution that can provide our device from which we get an average of 700 readings and that allow us to exploit the potential of our the method. Thus, we will used cubic B-Splines (which are the most commonly used) to make the description of objects, since as we have seen, the effect of the degree of the curve over the smoothing only results evident when using a high degree for the case of the higher resolution which can affect the time of analysis making the method not feasible for its use in real time.

5.2.2 Accuracy of the algorithms

As in section 5.2, we have experimented with simulated data environments (Figures 5.3 and 5.4) in order to evaluate the accuracy and effectiveness of the algorithms. Although the shown environments pose no challenge to traditional geometric SLAM algorithms based on the use of segments as a descriptive entity of the environment, the use of B-Splines as a way of representation allows to take larger data segments without the need of perform a segmentation of the information into smaller pieces and even allows to represent straight line segments because these are a particular type of curve. From the above we find that the B-Splines modeled extend the capabilities to more general situations.

Figures 5.15, 5.16, 5.17, 5.18, 5.19 and 5.20 show the results of SLAM experiments for the case of the classical extended Kalman filter, for the most recent strategy based on B-Splines and on Extended Kalman Filter and finally for the strategy that we have developed and which is also based on the use of B-splines for the representation of the environment.

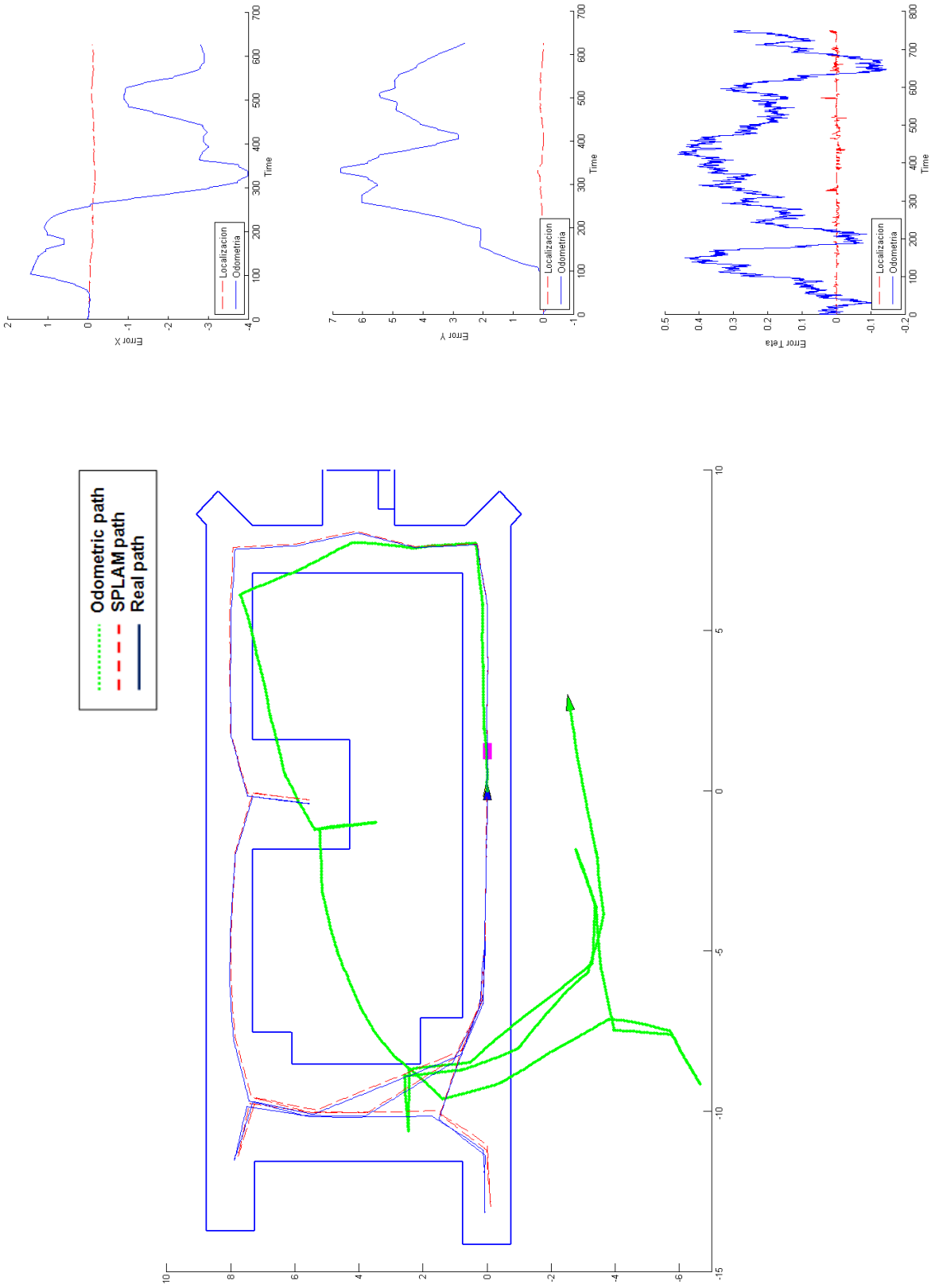

Figure 5.15 Accuracy and consistency experiment for the Classical EKF-SPLAM method on the office environment shown in Figure 5.3

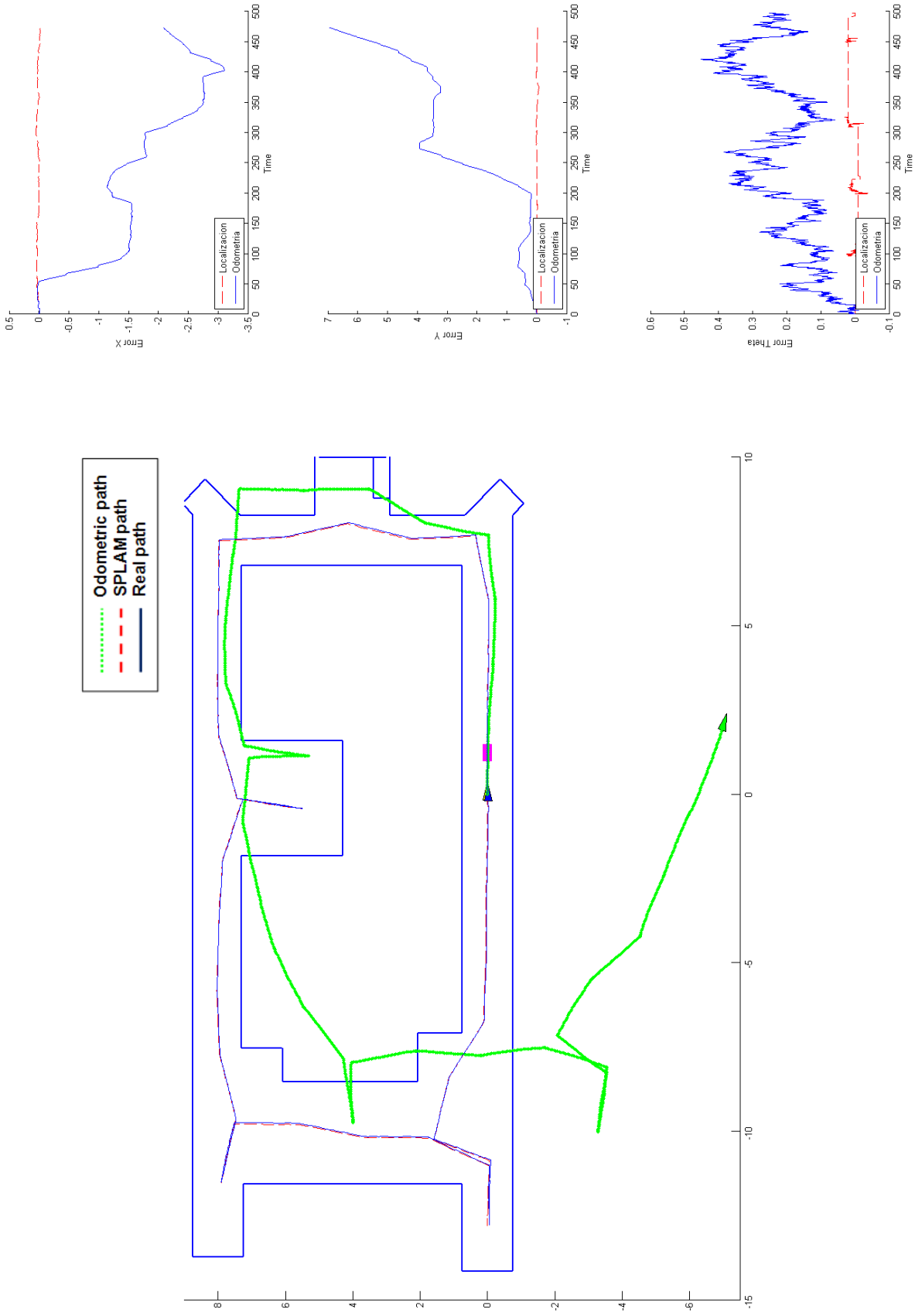

Figure 5.16 Accuracy and consistency experiment for the B-Splines based EKF-SPLAM method on the office environment shown in Figure 5.3

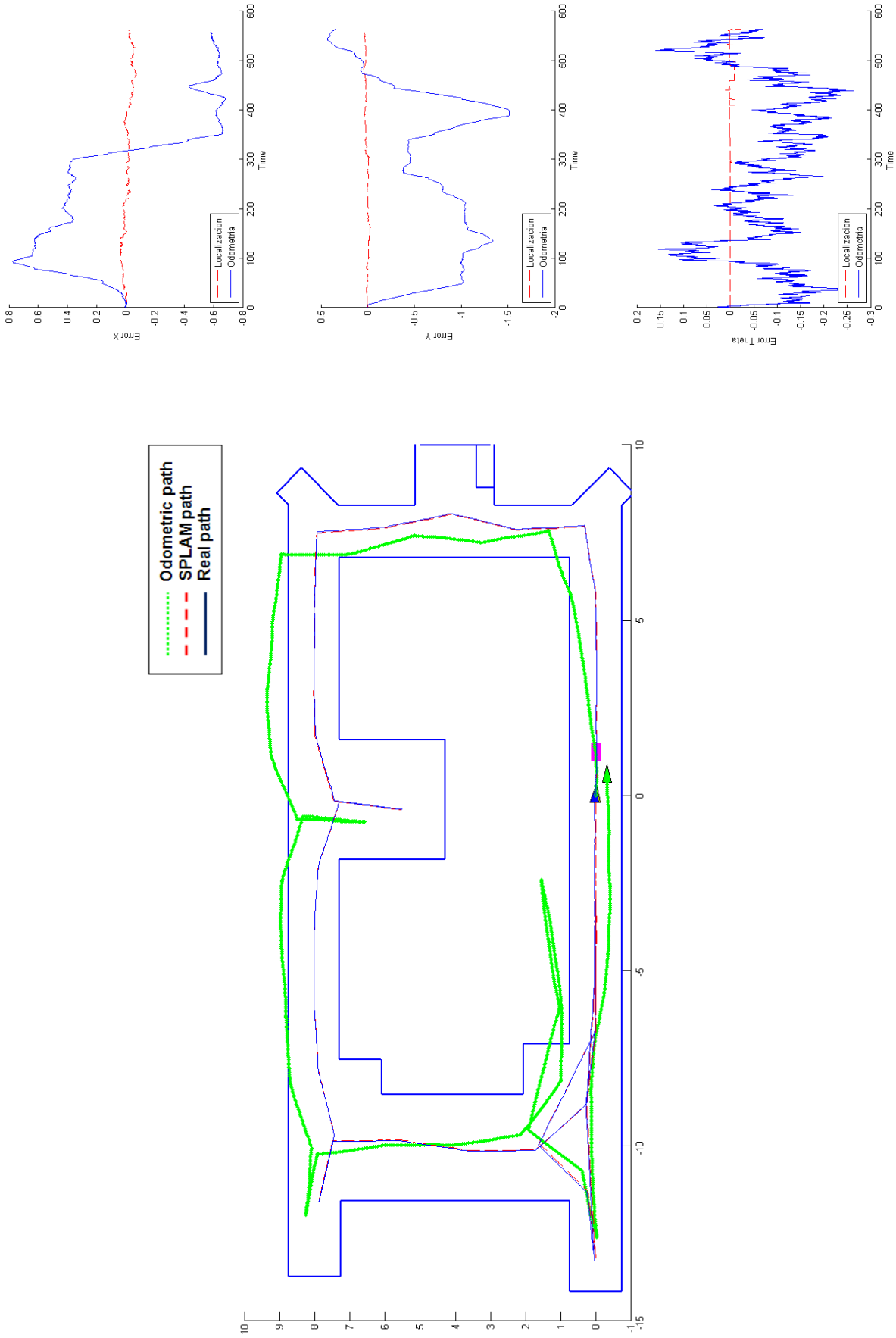

Figure 5.17 Accuracy and consistency experiment for the B-Spline based Topologic-SPLAM method on the office environment shown in Figure 5.3

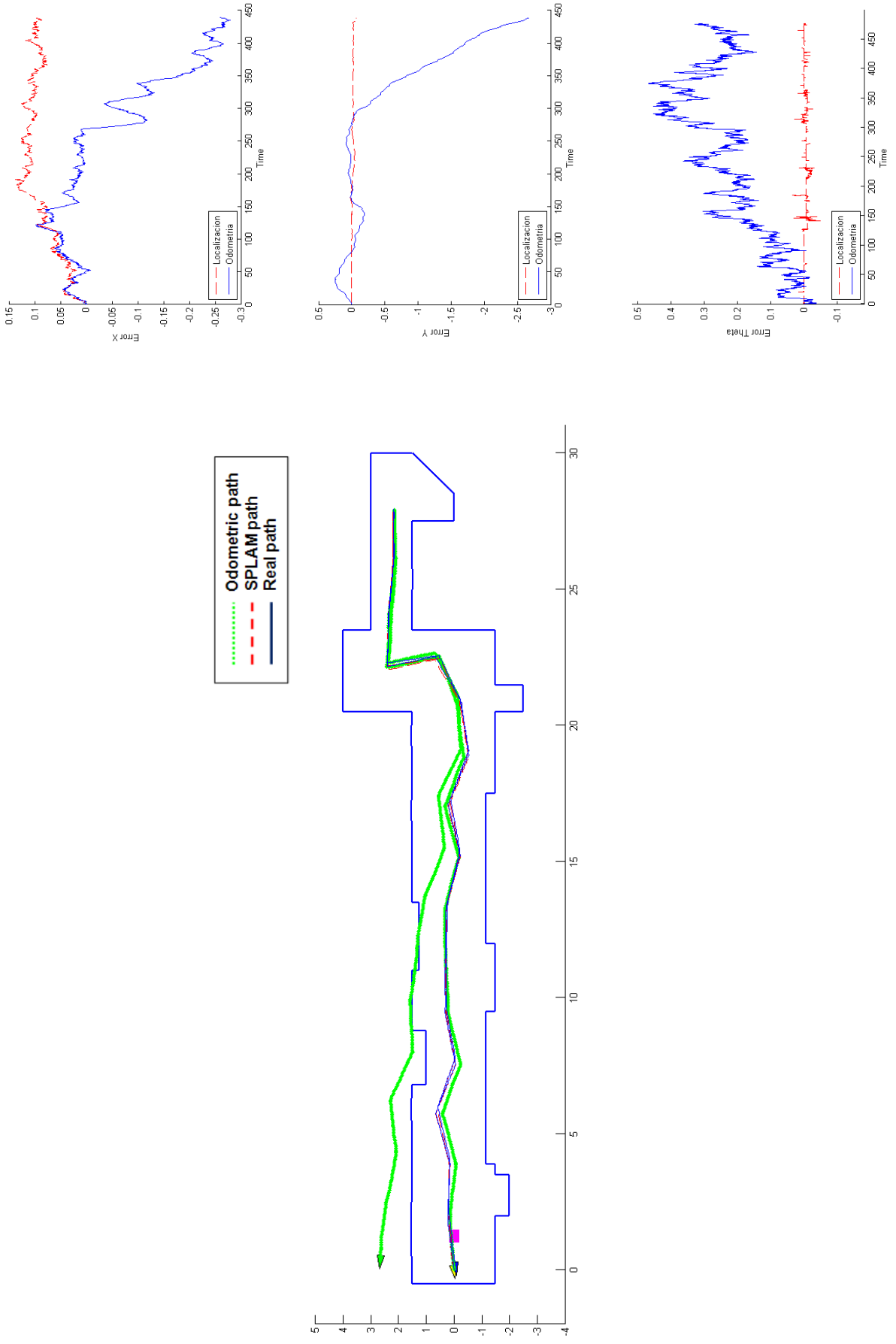

Figure 5.18 Accuracy and consistency experiment for the classical EKF-SPLAM method on the corridor environment shown in Figure 5.4

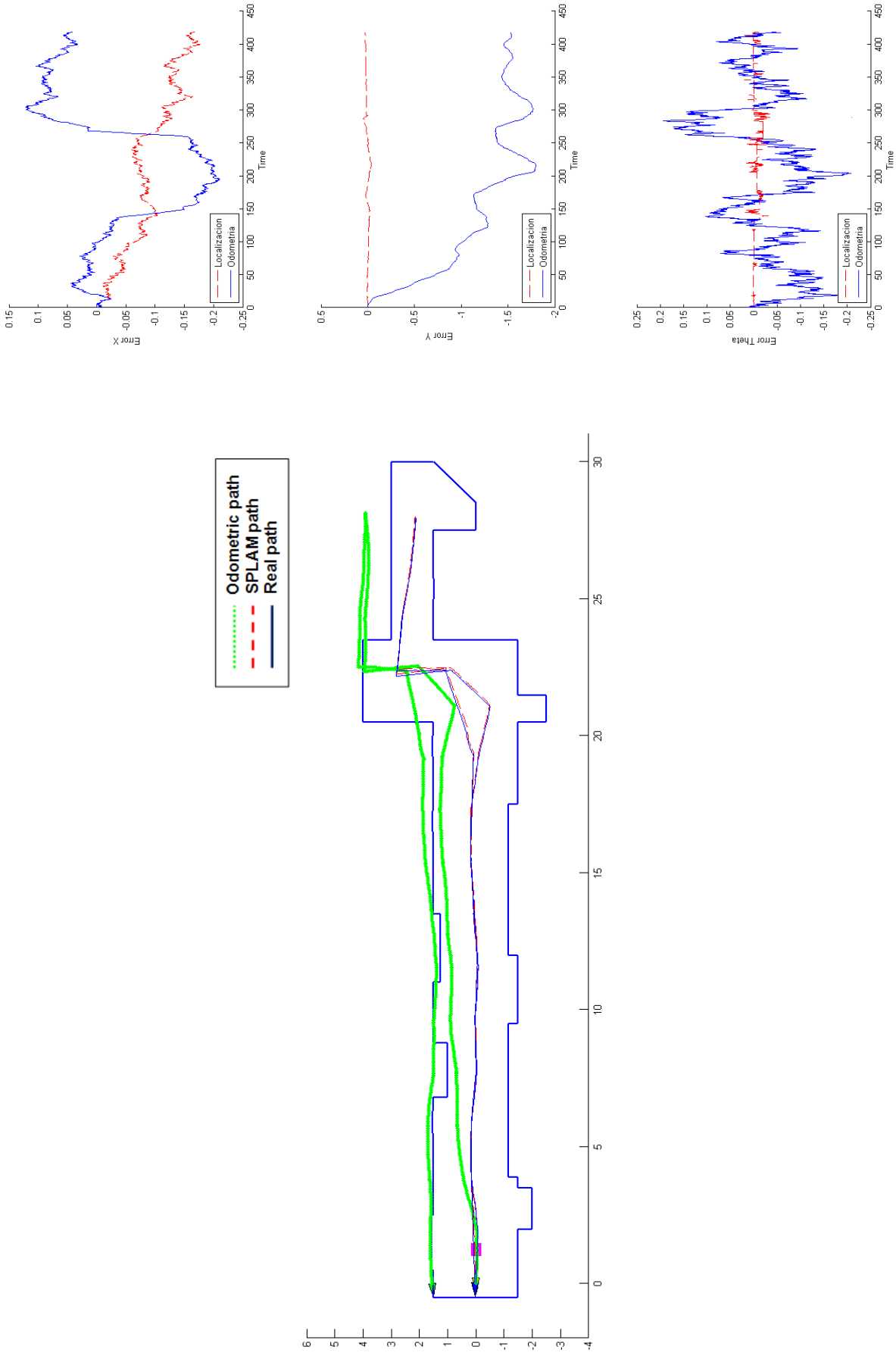

Figure 5.19 Accuracy and consistency experiment for the B-Splines based EKF-SPLAM method on the corridor environment shown in Figure 5.4

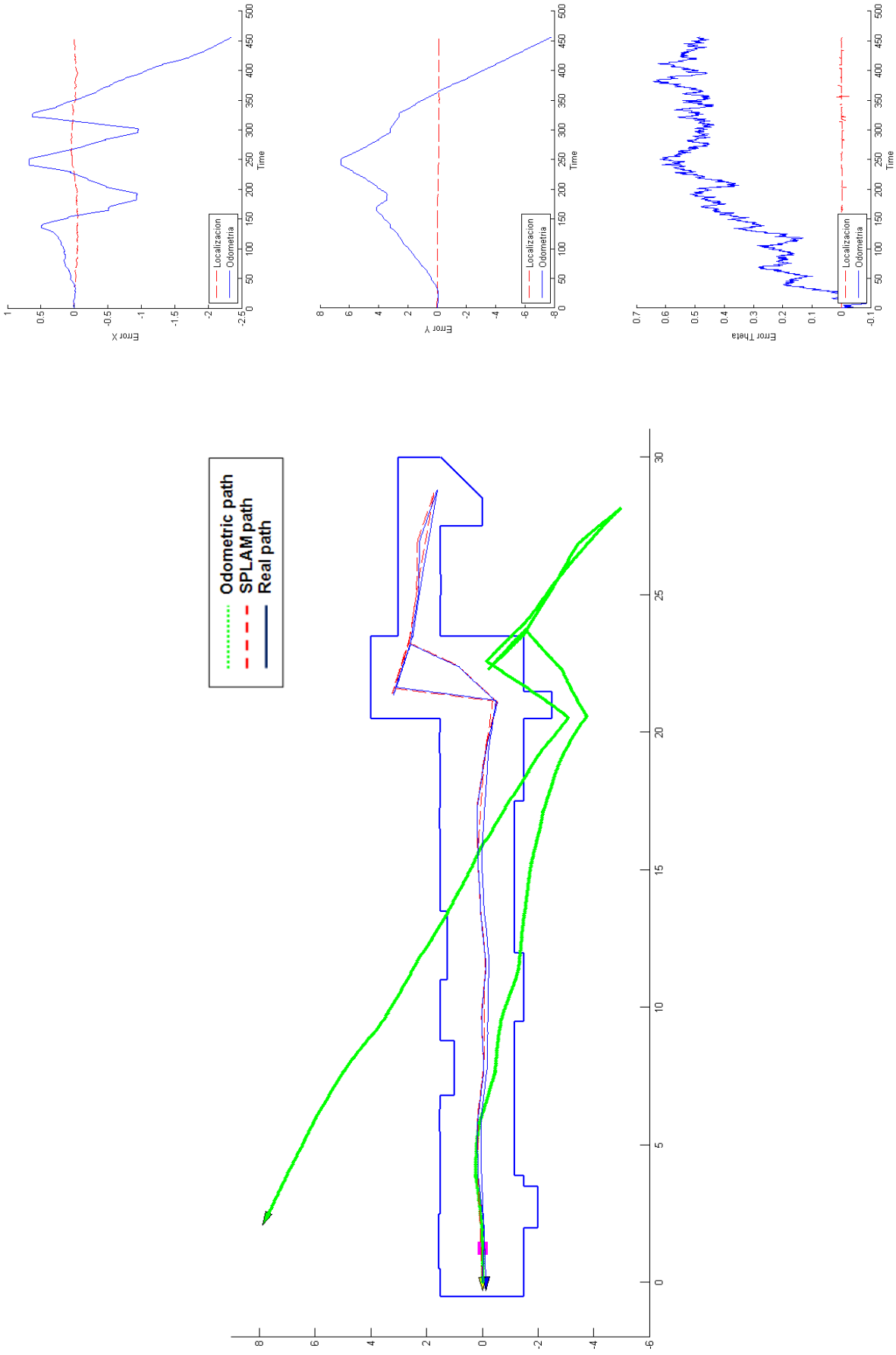

Figure 5.20 Accuracy and consistency experiment for the B-Spline based Topologic-SPLAM method on the corridor environment shown in Figure 5.4

The 6 figures above show in its left half, the actual path of the robot (blue continuous line), the odometric trajectory (green dotted line) and the trajectory obtained by the method of SPLAM (red dashed line). On the other hand, when there is real certainty about the actual path of the robot (as in the simulations) it is possible perform some checks to yield an idea of the quality of the algorithms from the point of view of its consistency. For this reason it has been possible to include in the right half of each figure the representation of the odometric error (blue line) in X, Y and Theta as well as localization errors (red dashed line).

From these data and taking as reference the errors shown by the methods based on Kalman filter, we conclude that the method proposed in this thesis maintains similar levels of error and in some cases better than those shown by other methods (Figure 5.21).

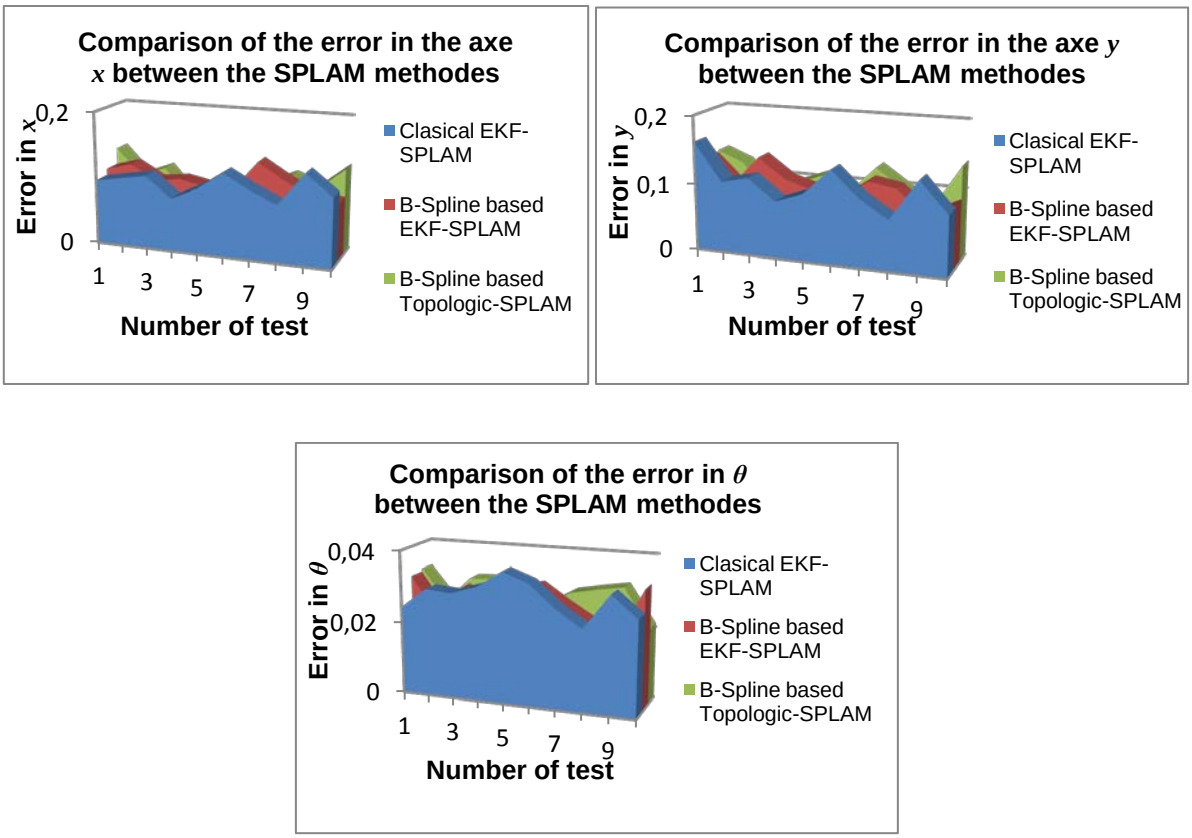

Figure 5.21 Errors obtained with the SPLAM strategies

The maps obtained with the three strategies are shown in the Figures 5.22 and 5.23; on them we can see the qualities of the maps obtained with the strategies but also the differences in the continuity of the line segments (Figure 5.22a and 5.23a) and curves (Figures 5.22b, 5.22c, 5.23b, 5.23c), where the first thing that stands out is that in the EKF- lines based strategy and in the EKF B-Splines based the segments

that form the corners (extended areas in the images) are discontinuous segments while in our strategy (Figure 4.15c) the highlighted environmental portion consists of a single continuous curve.

In the first case this is obvious because a corner represented by segments of lines always consist of two of them. In the second case, the methodology proposed by Pedraza et al. [Ref] forbids the use of too closed curves or corners, so in order to to represent them we must use several segments curve. In our case, the hypothesis used is that curves with highest curvatures are more easily associated and in consequence it will be more accurate will be the association of data and therefore the quality SPLAM method.

For us, the used hypothesis is that while more form or curvature has the curve segments, more exact it could be the association of data and therefore the quality of the SPLAM method.

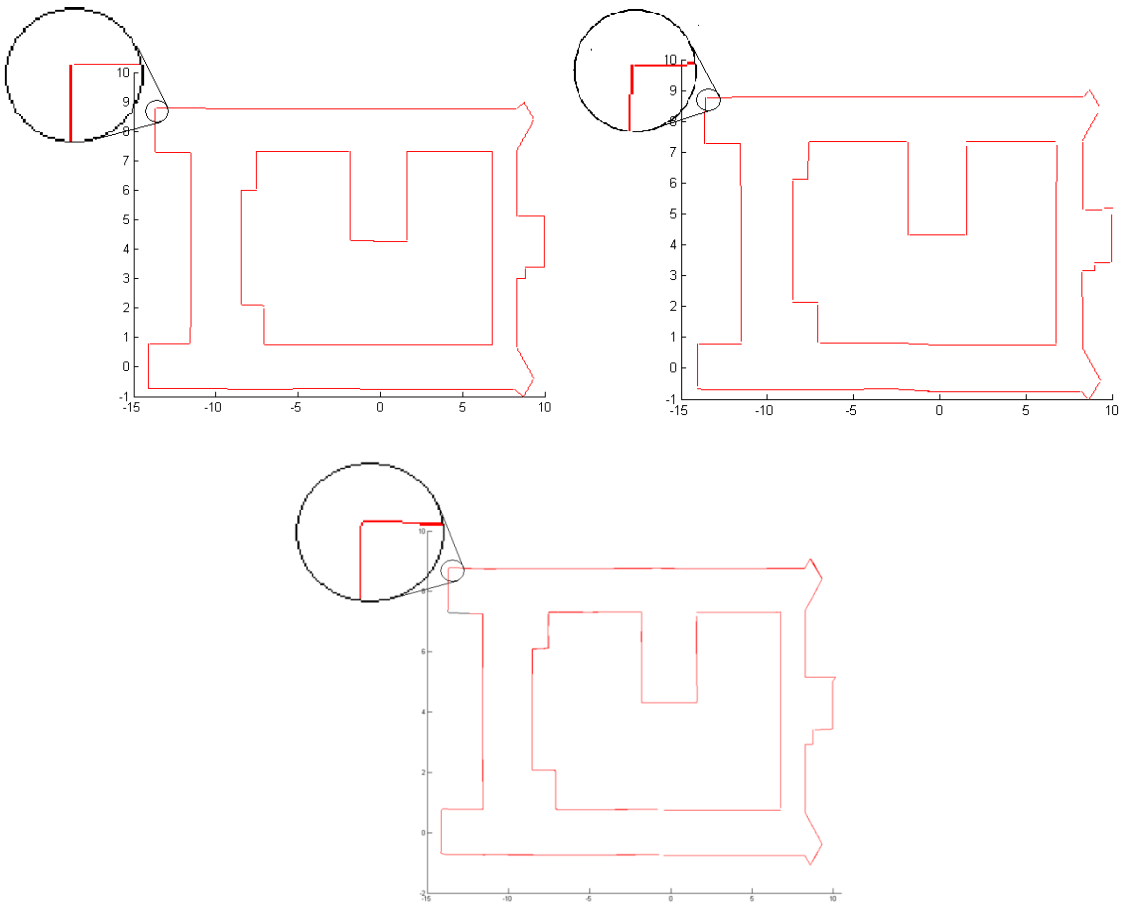

Figure 5.22 LIRMM offices environment maps. a) office environment map obtained with the method of classical EKF. b) Map obtained by EKF approach with B-Splines. c) Map obtained with the strategy of SPLAM proposed in this work.

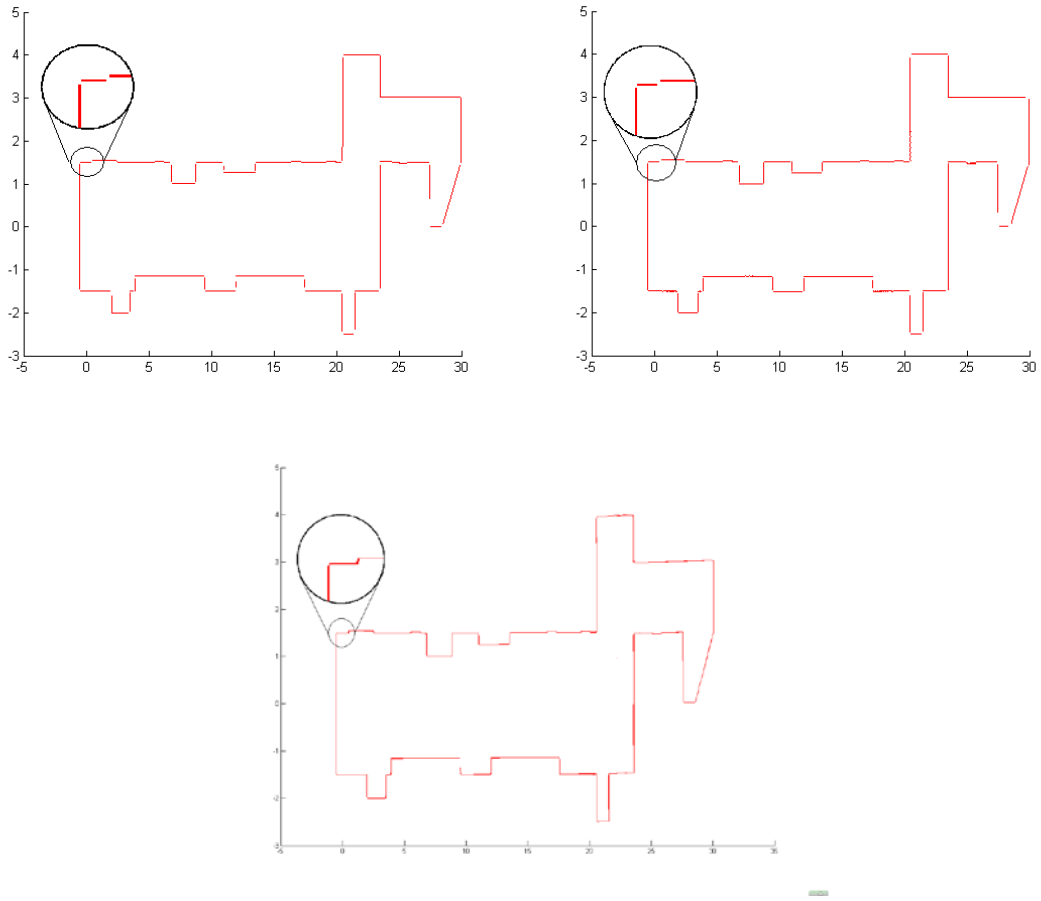

Figure 5.23 LIRMM corridor environment map. a) Corridor environment map obtained with the method of classical EKF. b) Map obtained by EKF approach with B-Splines. c) Map obtained with the strategy of SPLAM proposed in this work.

5.3 Kidnapping

As we have said in chapter 4, the kidnapping is one of the hardest problems to solve in the SLAM field. Although under normal conditions is very unlikely to find this problem, in this section we will show the operation of the proposed solution which provide more robustness to our strategy SPLAM.

Using the environment shown in Figure 5.3 we will validate the strategy of kidnapping proposed. Figure 5.24 shows the kidnapping of the robot from the position q11 in which it was operating to an unknown position. At this point, the robot was built an exploration graph with 11 nodes and it had collected 3 digital signatures from recognizable zones.

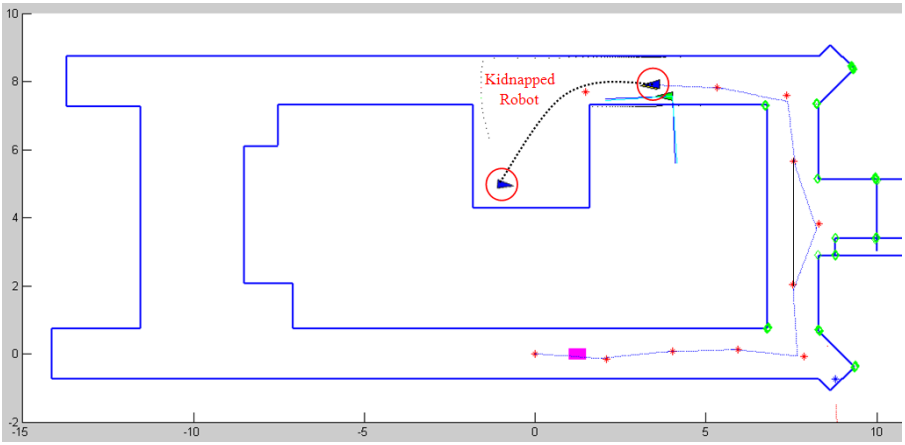

Figure 5.24 Kidnapped robot

In this example, the robot easily recognizes that it has been kidnapped because the RSL of the region where it was does not match with the observations performed and therefore it goes into a kidnapping mode.

Being aware of your current situation, the map created so far is stored and the robot reset its memory considering now that the position where it is placed is the initial position. This can be seen in Figure 5.25, where the position of kidnapping is now taken as the initial position and therefore the spatial change in the position of the environment due to the change of the reference frame (Red dashed lines map).

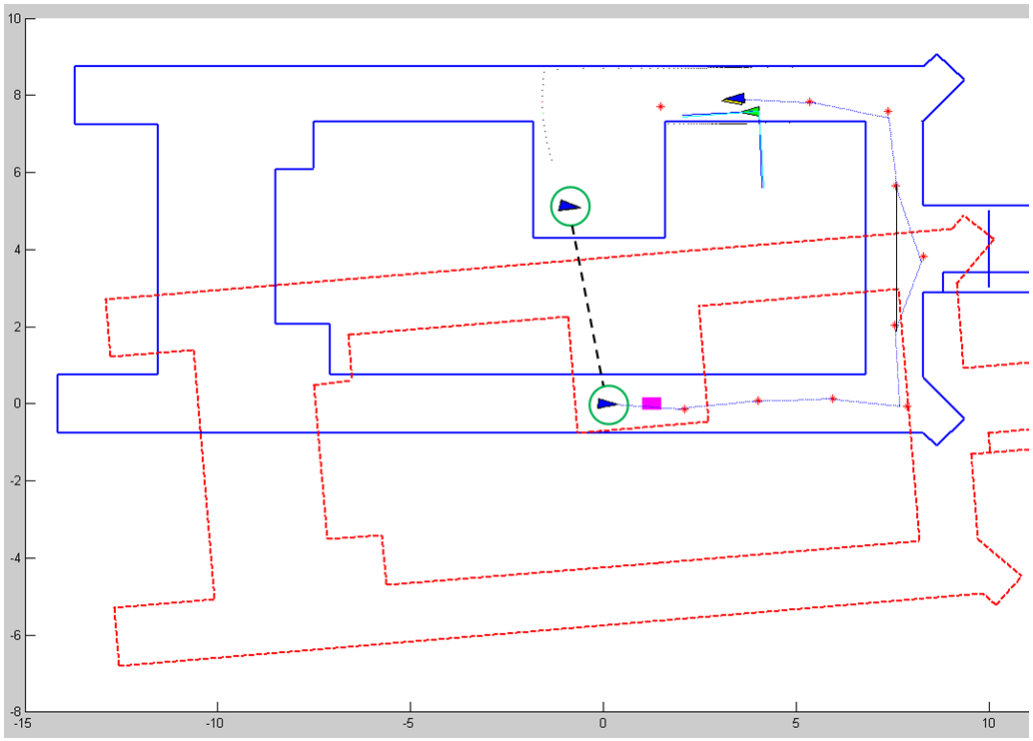

Figure 5.25 New environment position after the kidnapping

In this point, the robot starts the construction of a secondary exploration graph. The process continues until the digital signature of a distinguishable area is recognized and identified (Figure 5.26).

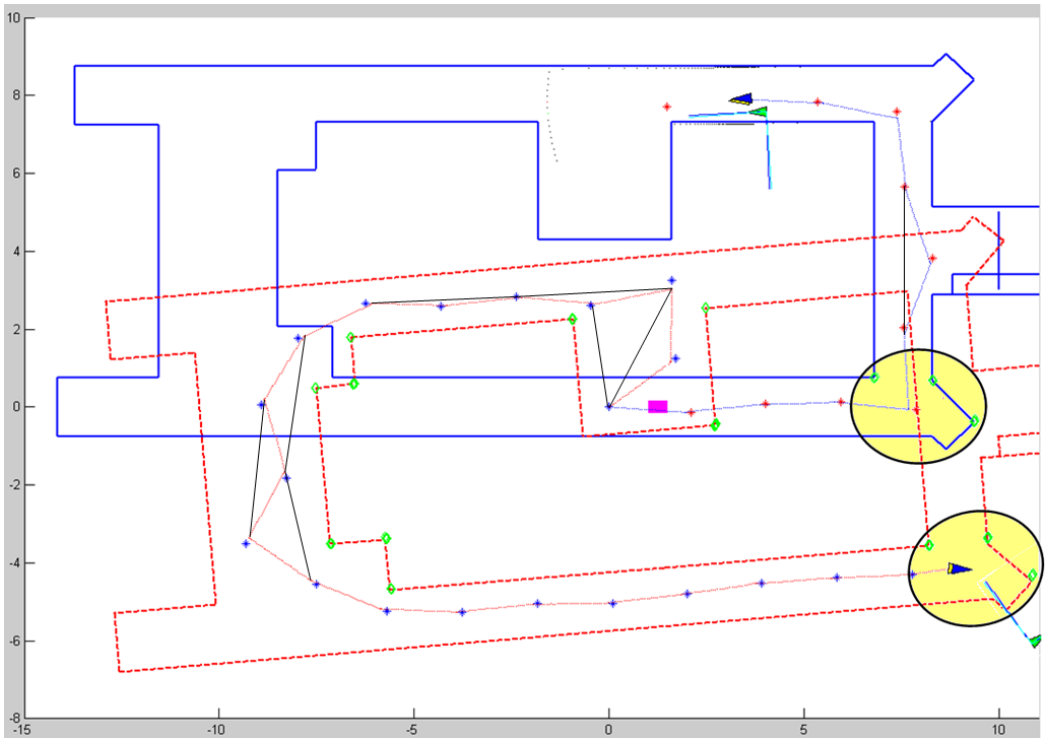

Figure 5.26 Areas with similar digital signature

Once compared and identified the digital signature of the area, the robot adjusts its position and the position of all nodes in the auxiliary graph to the reference frame of the recognized area stored in the first exploration graph.

With the corrected position and the position of the auxiliary graph updated, we merge the structures of the main graph and of the auxiliary graph so as to obtain a single structure with which we will continue (if after the merger still remain free frontiers) the exploration (Figure 5.27).

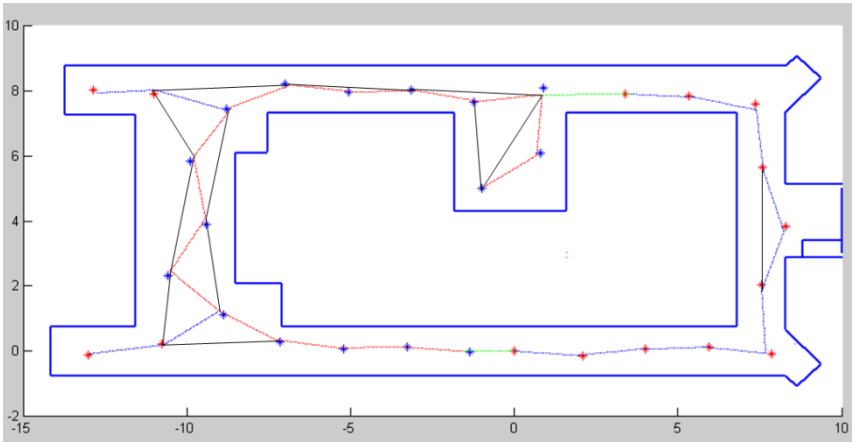

Figure 5.27 Random exploration graphs merged

Finally, as with the graph structure, the auxiliary map created after kidnapping (figure 5.28) is updated with the new corrected position of the robot and merged with the partial map created before kidnapping (figure 5.29). With this last action we obtain a complete map of the environment after solving the kidnapping problem (Figure 5.30).

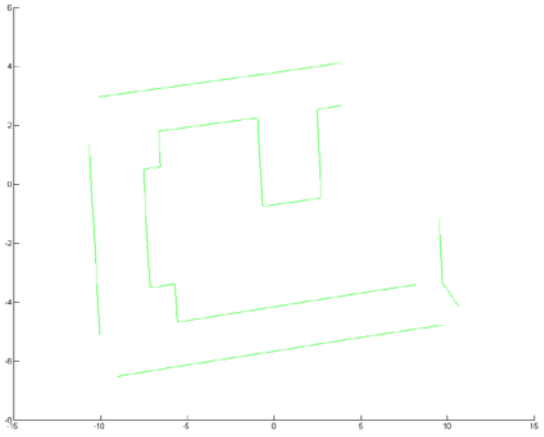

Figure 5.28 Map constructed during kidnapping

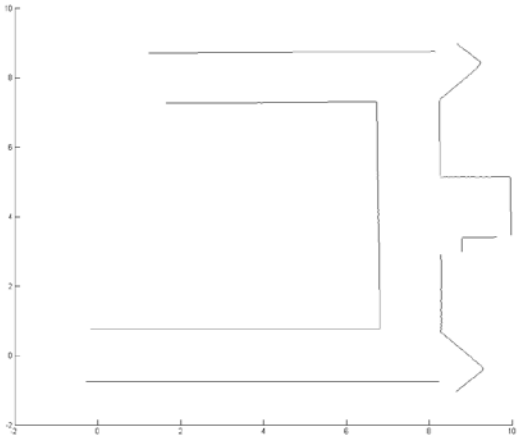

Figure 5.29 Map constructed before the kidnapping

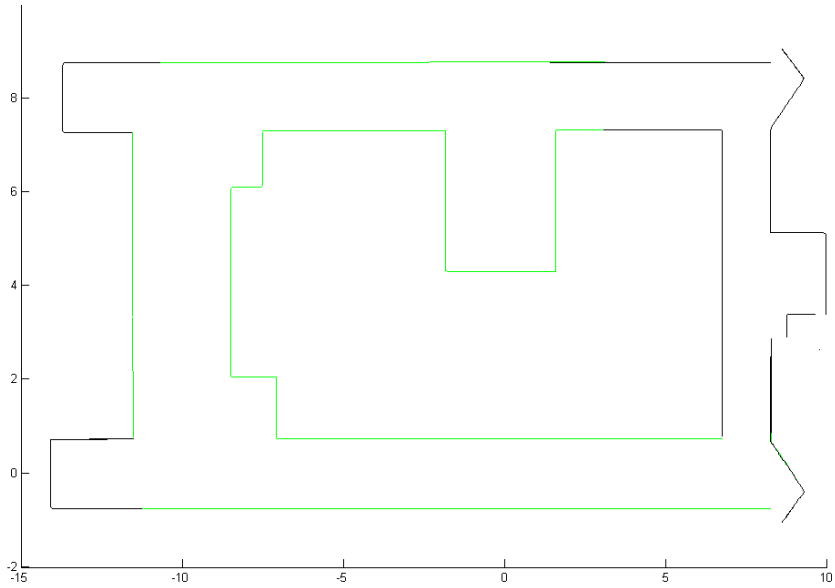

Figure 5.30 Before and after kidnapping maps fused

Figure 5.31, 5.32 and 5.33 show the errors obtained during the simulation. Here, we can clearly see the moment in which the kidnapping occurred as a big leap in the graphics error. Also, we observe that once the method recovers of the kidnapping, it maintains similar levels of error that in **cases without kidnapping.**

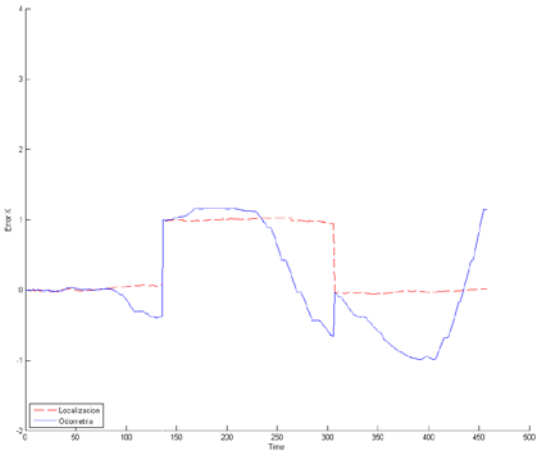

Figure 5.31 Error in X during the kidnapping simulation

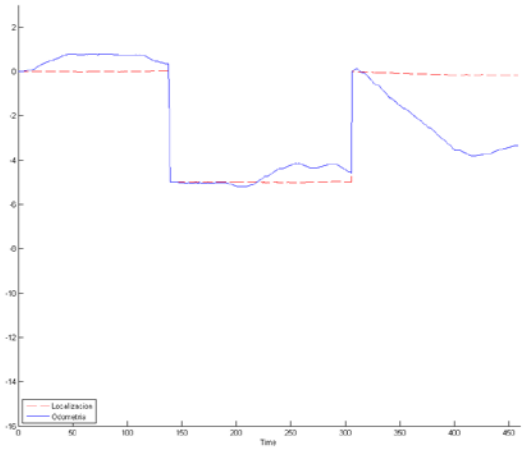

Figure 5.32 Error in Y during the kidnapping simulation

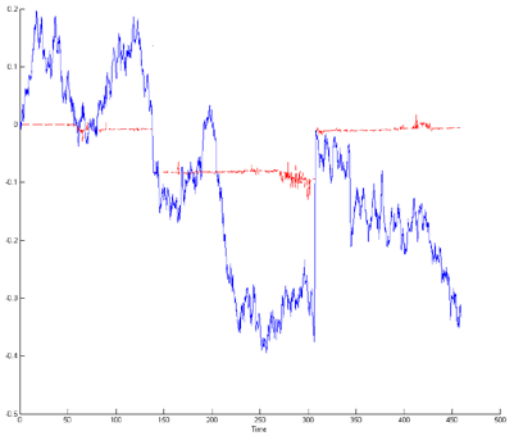

Figure 5.33 Error in Theta during the kidnapping simulation

5.4 Experiments with real data

Finally in this section, we present experiments with real data in real environments to validate the results presented. In all the tests will be shown the maps obtained considering only the odometric information reported by the robot and maps obtained after applying the SPLAM method proposed in this thesis.

Figure 5.34 shows the real office environment used for the tests and figure 5.35 show the map of this environment form which was obtained the simulated environment shown in Figure 5.3. This environment has been built using 58 B-splines curve segments of degree 3 which are used as modeling tool and which is defined by 1754 data points; the time required for the exploration of this environment was 463 seconds. Figure 5.35 shows the map obtained after the application of our SPLAM method while Figure 5.36 shows the map constructed using only the robot's odometric information.

Figure 5.34 Real office environment used for tests

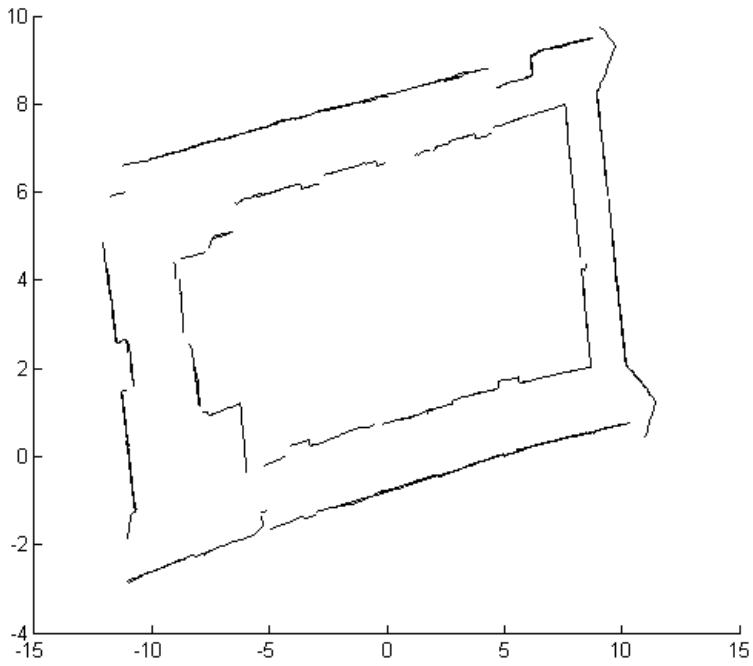

Figure 5.35 Real office environment acquired with the SPLAM proposed method

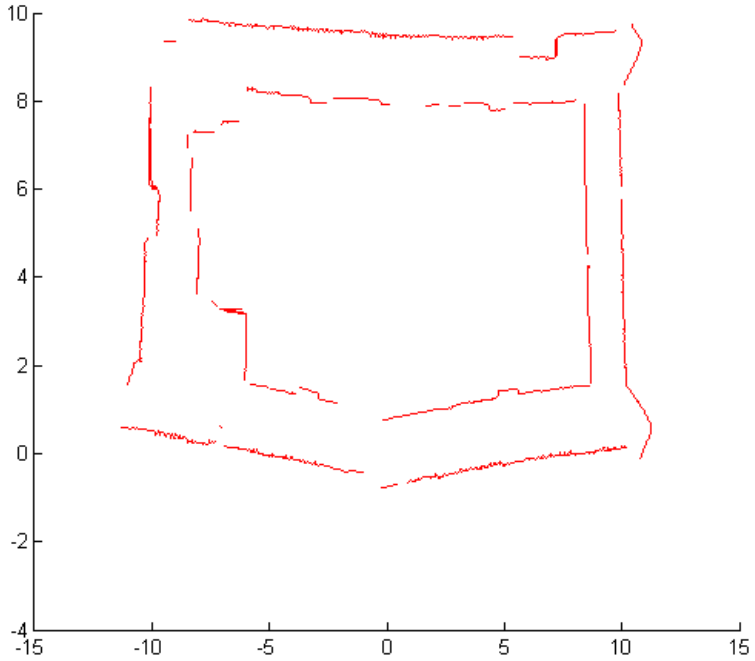

Figure 5.36 Real office environment acquired using only odometric information

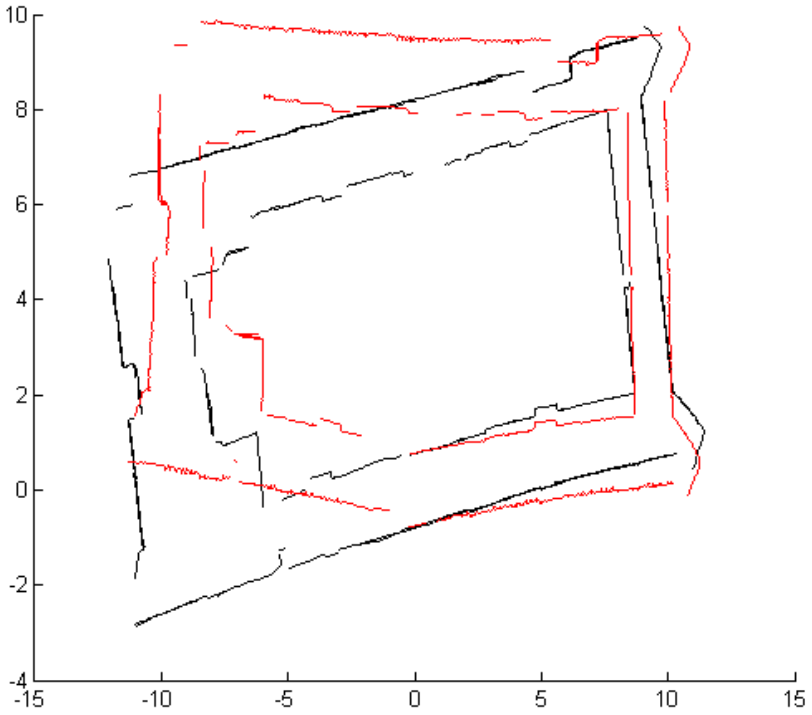

Figure 5.37 Difference of maps with and without the use of the SPLAM method

In the same way, figure 5.38 show the real corridor environment . The map obtained with our SPLAM approach is shown in figure 5.39 while the odometric map obtained using just odometric information is shown in figure 5.40.

Figure 5.38 Real corridor environment used for tests

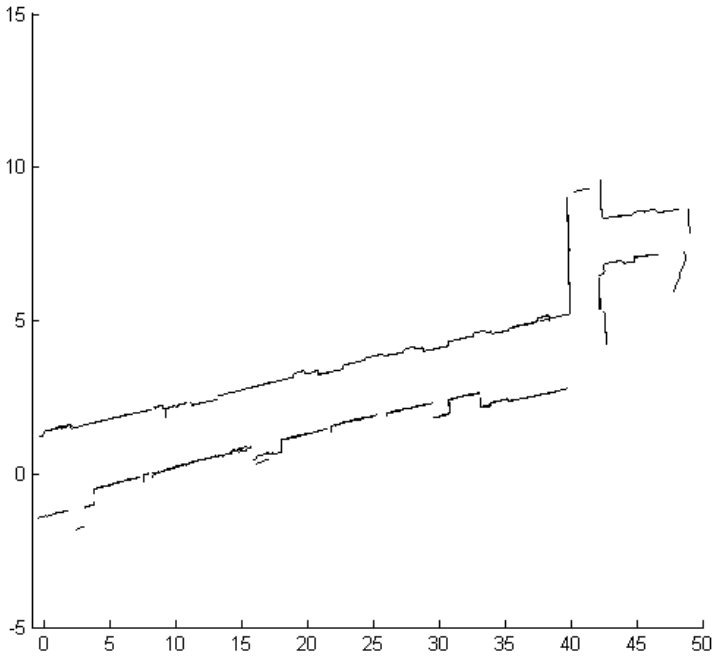

Figure 5.39 Real corridor environment acquired with the SPLAM proposed method

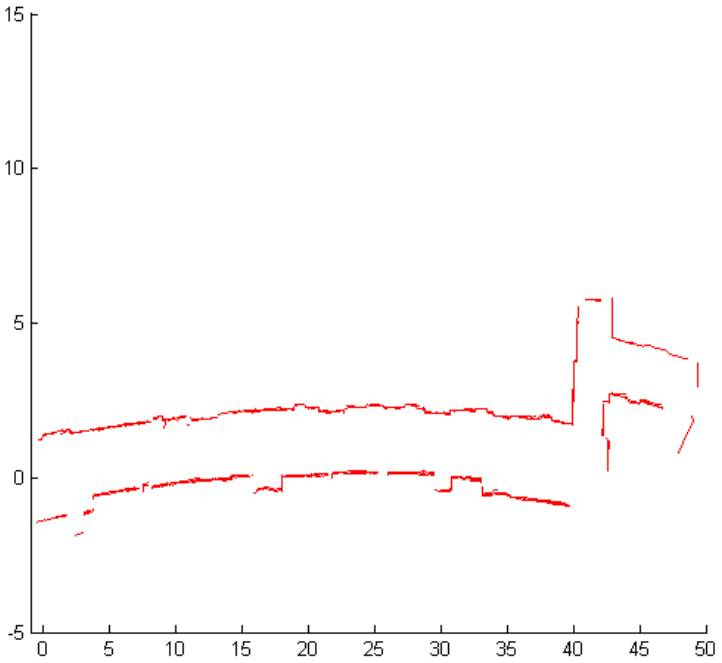

Figure 5.40 Real corridor environment acquired using only odometric information

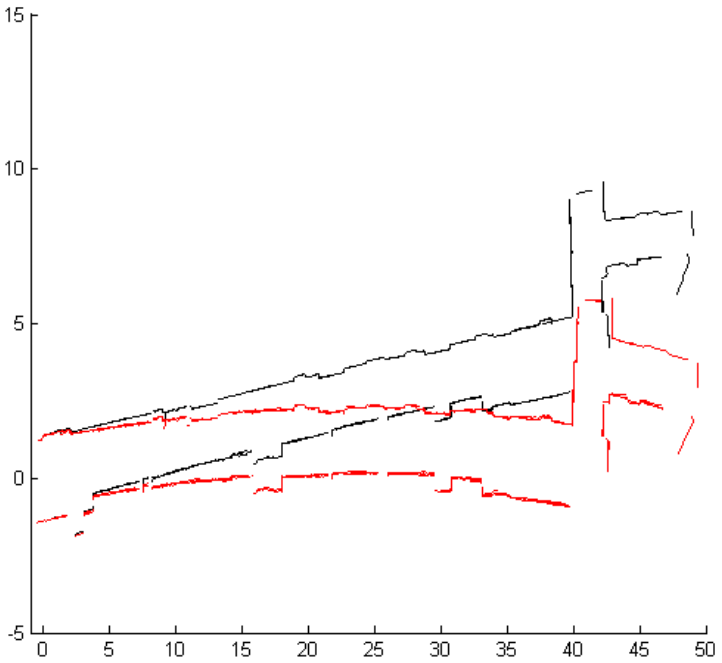

Figure 5.41 Difference of maps with and without the use of the SPLAM method

Finally, a last example is presented using the real environment called “the extension” (Figure 5.42). An in the other two environments, the figure 5.43 presents the map obtained with our SPALM approach. In the other hand, figure 5.44 shows the map obtaines using only odometric information

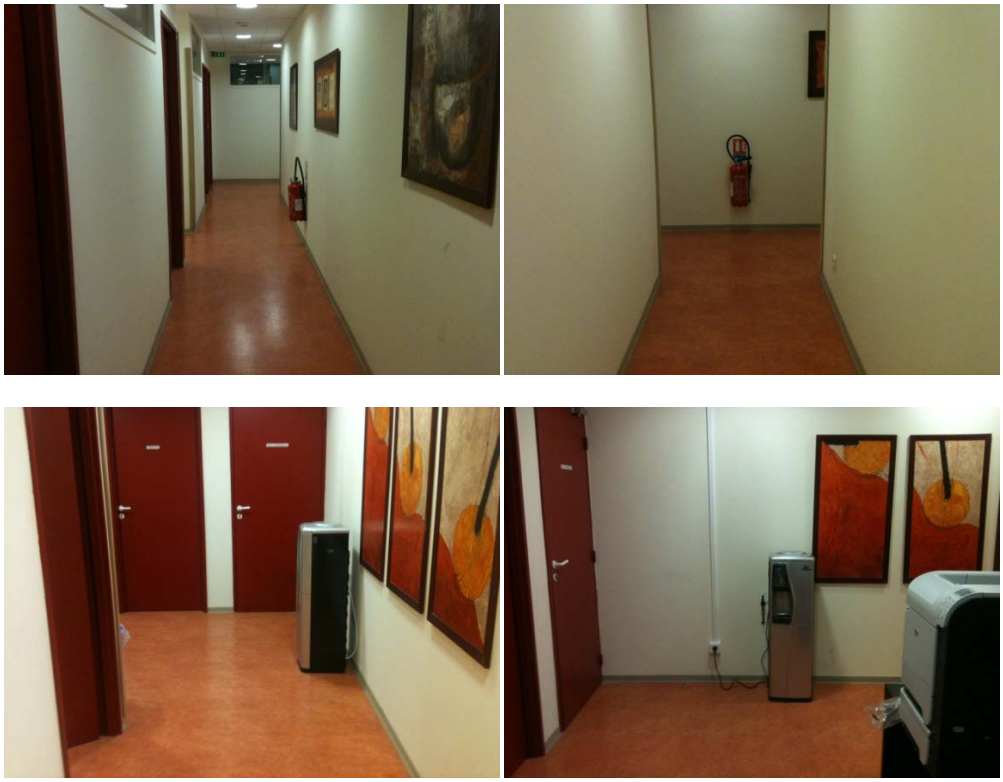

Figure 5.42 LIRMM’s extension corridor used for tests

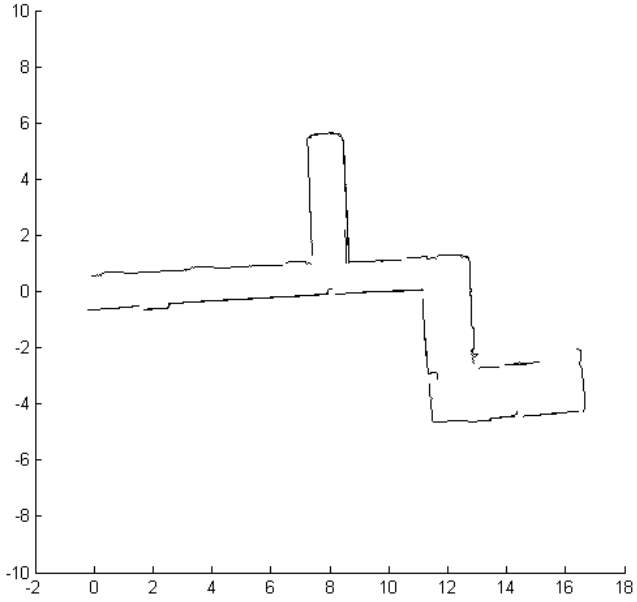

Figure 5.43 Real LIRMM’s extension corridor environment acquired with the SPLAM proposed method

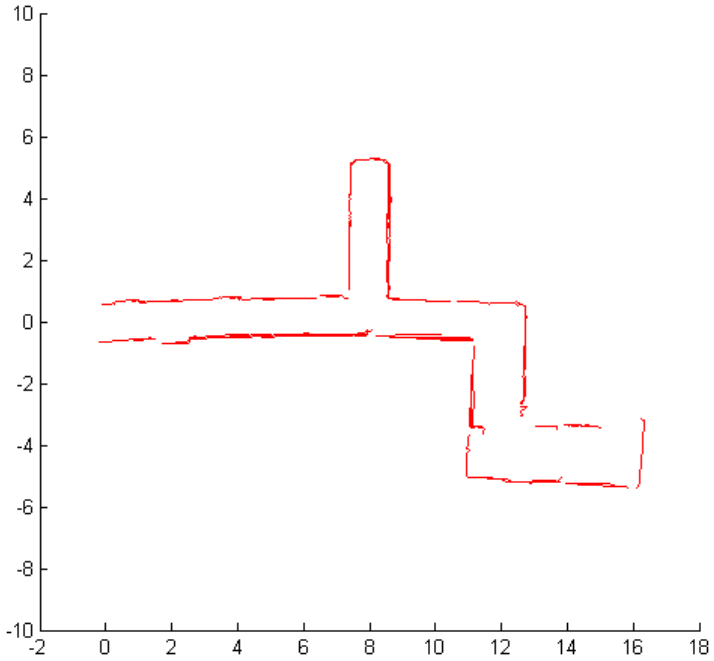

Figure 5.44 Real LIRMM's extension corridor environment acquired using only odometric information

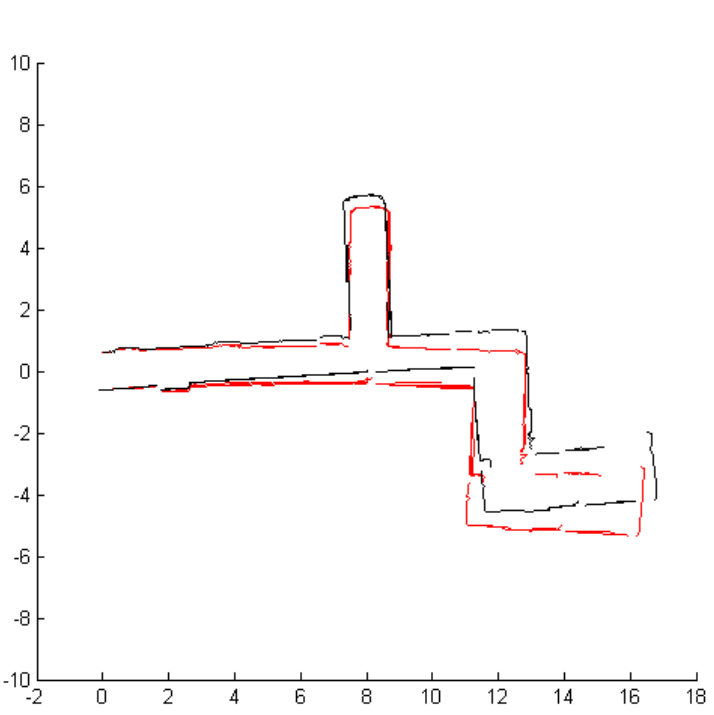

Figure 5.45 Difference of LIRMM's extension corridor maps obtained with and without the use of the SPLAM method

5.5 Conclusions

In this chapter we have shown and validated experimentally the properties of the algorithms presented in this thesis.

Experiments with simulated data have allowed evaluating the properties of the estimation algorithm from the point of view of its consistency. The confrontations of our methodology with other algorithms developed by other researchers have revealed that the results obtained by our method remain within the error tolerance range accepted or obtained by other methodologies. It has also shown that the use of B-spline curves to represent the environment offers new possibilities to extract geometric information furthermore offer the possibility to represent complex environments which would be impossible to model geometric tools.

Finally, experiments with real data provide important results that have allowed us to verify the applicability of the techniques developed in our thesis for the complex problem of SPLAM.

Chapter 6. Conclusions and Future Work

In previous chapters we have presented the construction of SPLAM tools implementing and developing methods in the fields of SLAM and of exploration of environments. With the tests and results obtained, in this chapter are summarized the main conclusions highlighting the contributions and contributions that have been made to the current state of the art of integrated Exploration or SPLAM.

6.1 Main contributions

In this thesis we have presented a new methodology for the process of exploration of environments based on the construction of a graph of exploration where each node represents a robot position with its respective segments of environment explored. Although the use of such structures has already been used to solve this kind of problems, our solution fully exploits the functionality of the structure to make navigation on the portion of the known environment regardless of the purpose. Also, unlike other graph-based solutions, our solution preserves the random nature since this kind of solutions have proven greater effectiveness in the area of exploration environments where we cannot have a certainty of which will be the next best position to be explored.

Also, we have developed a B-spline curves based SLAM strategy to represent environments. Although this is not the first job where it has been used this type of representation, we have used mathematical algorithms used in the area of pattern recognition (that until where our knowledge arrives had not been used before in the SLAM area) in order to obtain the greater amount of information contained in this representation and in this way to propose an innovative data association method to SLAM.

In the same way, the methodology applied for the correction of the robot's position and to adjust the environment based on local information using the structure of the exploration method, represents an important contribution since in this way the method does not waste time and energy trying to associate data out of range.

We have presented a method that gives solution to the problem of kidnapping, where even though it is a simple solution to the problem, it is an efficient solution when the robot works on environments with high content of distinguishable information (areas of high curvatures).

Finally, despite the apparent difficulty that the symbiosis of the methods developed could present, we have achieved a harmonious cooperation which fuses the properties of the method of exploration of environments with the properties of the SLAM method based on B-Spline curves.

As a general summary, we discuss briefly the results of each chapter pointing in each one of them the contributions presented.

- Chapter 2 gives an overview of current state of the art in the field of SPLAM which was taken as a reference and motivation for the development of this thesis. In it, we analyze the contributions made so far first in the field of SLAM, then in the field of exploration of environments and finally of the strategies developed for the SPLAM problem where many of the solutions found (as would be expected) are a fusion of the two previous elements

- Chapter 3 presents a SPLAM tool in which motion control is perform using the SRT random exploration tool and the SLAM task is performed using the extended Kalman filter in its classical version and also in the version presented by Pedraza et al in [Ref]. This chapter also collected fundamental aspects of B-spline curves which are used for the representation of environments. The theory found in this chapter provides an understanding of these curves as tools in the representation of maps and also of their use in the well-known EKF-SLAM tool.

  Although the individual usage of these tools is not innovative, to where our knowledge arrives, these never before had been combined to obtain a strategy of SPLAM based on the probabilistic control of movements and on the extended Kalman filter and even less based on splines for the representation of the environment. Despite this, the results obtained with the work made in this chapter only will be used to verify the effectiveness and to validate the algorithms presented in Chapter 4.

- Chapter 4 describes the main contributions of this thesis which have been thought for the ultimate goal of obtaining an effective tool for the SPLAM problem in which the modeled of the environment is performed using B-spline curves. Thus the contributions of this chapter are listed below.

  1. We have developed an exploration strategy that creates graph type structure where each node in it represents a robot position with a portion of environment associated with it. In this method, the concept introduced of frontier control (which represents one of the main contributions for the method) allows to have complete control over the exploration avoiding to travel fully explored areas and and revisiting those who still have the possibility for exploration. Once more, this concept allows to reach the goal of completeness basic on all the exploration methods

On the other hand, the developed method fully exploits the graph structure generated allowing the use of all the connections contained in the structure when the robot needs to travel from a position to another. Finally, the criterion of choosing the node with possibility of exploration with the lower measured path from the current position allows the method to stay and explore nearby areas until they are fully explored without having to travel from one end to another of the map.

2. We have proposed a data association method based on the analysis of the curvatures of the curves related. Here, we use the CSS digital imaging techniques, curvature zero crossing and corners extraction techniques used in the field of pattern recognition. This mechanism of association not only allows establishing a robust correspondence between the observations realised by robot and the objects contained in the current working node but also facilitates the parametric correspondence between each pair of representative elements associated.

3. The correction in the position of the robot and the associated data is done topologically (scan matching) using the distinguishing elements mentioned in point 2 correcting the information first in angular sense and later in translation. Obviously, this type of location is novel given that the representation used in it has been recently presented.

4. We present a simple and novel algorithm to lengthen the objects contained in the map. This is possible and is natural after considering the form in how the association of data is performed.

5. Finally, the use of sub areas of environment contained in the nodes of the structure of the exploration method combined with the SLAM method, allows that the run time of the SPLAM tool remains within the acceptable limits to use our tool in real time. Thus, the construction of the environment can be done both online and offline since that with the information of the exploration graph we know what areas correspond to what part of the environment.

- Finally, Chapter 5 presents the results that evaluate, demonstrate and validate the algorithms presented in previous chapters.

We start with a series of tests made to the SRT and REG exploration methods validating its effectiveness with respect to the path traveled, number of nodes needed to complete the exploration and time spent to complete the task. We continue with an analysis that relates the angular resolution of the laser with the degree of the spline and how these affect the curvature of the spline from which the information is obtained for the association and elongation of the environment

We also show a series of simulation tests performed on the SLAM methods. On them we show the accuracy obtained with each one of the methods and with which we validated the SPLAM method that we have proposed since the error levels remain within the limits obtained with the EKF-based methods.

At the end, we show the maps obtained with real data that allow to evaluate the practical applicability of the proposed methods.

6.2 Future Work

The algorithms and methodologies presented in this thesis represent (from our point of view) an interesting and complete form to deal q_{ith} the SPLAM problem as for environments with geometries defined as for environments with complex forms.

Taking this into consideration, we are sure that this proposal opens the door to an endless of future applications and developments between which we can emphasize the following:

- Improve the data segmentation mechanism in which a more sophisticated mechanism will determine more accurately the individual objects present in a sensor observation in a more robust way.

- Explore the use of alternative representations for data fitting. In this section we have thought about the use of another type of parametric curves (NURBS [**Fisher et al. 2004**], X-Splines [**Blanc et al. 1995**], Beta-Splines [**Joe et al. 1990**], or curves of variable resolution) as modeling tools with which we could obtain better results in the quality of fit, the automatic selection of the knot vector and the ability to represent singular points (such as corners) using a smaller amount of information to that needed with the use of B-Splines.

- Deepen in the methods recently introduced in the field of pattern recognition which could get more interesting and effective solutions for data association process using parametric curves.

- To improve the recovery process of the exploration when a robot kidnapping has occurred not only exploiting unique features of the environment but also invariant relations between the segments that form the environment. Although the purpose of this point is not to go into details, we call invariant relations to the comparison of the key metrics that define each type of relationship between two found elements in the same area of the environment (Example: high-curvature curve associated with a segment of straight line, curve with high curvature associated with a point, two straight line segments, a point and a straight line, etc.).

- Extend the presented approach to the multi-robot case in which, if used an appropriate distribution of the robot in the environment, the exploration time would be reduced dramatically depending on the number of robots that will operate in the environment.

- Extend the obtained maps to the three-dimensional space using B-Splines surfaces which could be used by a large number of types of robots since so far the use of our maps is limited to the use of terrestrial robots.

Bibliography

Bennet et al. 2000 → A. Bennet and J. J. Leonard. “A behavior-based approach to adaptive feature mapping with autonomous underwater vehicles”. IEEE J. of Oceanic Engineering, 25(2):213-226, 2000.

Bishop 2006 → C. M. Bishop. “Pattern Recognition and Machine Learning”. Springer, 2006.

Blanc et al. 1995 → C. Blanc, C. Schlick. “X-splines: a spline model designed for the end-user”, Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. 1995.

Bosse et al. 2004 → M. C. Bosse, P. M. Newman, J. J. Leonard, and S. Teller. “Simultaneous Localization and Map Building in Large-Scale Cyclic Environments Using the Atlas Framework”. The International Journal of Robotics Research, 23(12):1113–1139, 2004.

Bourgault et al. 2002 → F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky and H. F. Durrant-Whyte. “Information based adaptive robotic exploration”. In IEEE/RSJ Int. Conf. on intelligent Robots ans Systems, pages 540-545. 2002.

Burgard et al. 2005 → Burgard, W., Moors, M., Stachniss, C., Schneider, F., “Coordinated multi-robot exploration”. IEEE Transactions on Robotics 21 (3), 376–386. 2005.

Castellanos et al. 1997 → J.A. Castellanos, J.D. Tardos, and G. Schmidt. “Building a global map of the environment of a mobile robot: The importance of correlations”. In IEEE International Conference on Robotics and Automation, pages 1053–1059, 1997.

Choset et al. 2001 → H. Choset and K. Nagatani. “Topological Simultaneous Localization and Mapping (SLAM): Toward Exact Localization Without Explicit Localization”. Transactions on Robotics and Automation Vol. 17. 2001.

Choset et al. 2004 → H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun. "Principles of Robotic Motion: Theory, Algorithms, and Implementation". MIT Press, Cambridge, MA, 2004.

Davison et al. 2002 → A. J. Davison and D. W. Murray. "Simultaneous localisation and map-building using active vision". IEEE Trans. Pattern Anal. Machine Intell., 24(7):865–880, 2002.

de Boor 1978 → C. de Boor, "A Practical Guide to Splines", Springer-Verlag, New York, 1978.

Dellaert et al. 1999 → F. Dellaert, D. Fox, W. Burgard, and S. Thrun. "Monte Carlo localization for mobile robots". In IEEE International Conference on Robotics and Automation, pages 1322– 1328, 1999.

Dellaert et al. 2006 → F. Dellaert and M. Kaess. "Square root SAM: Simultaneous localization and mapping via square root information smoothing". Int. J. Robotics Research, 25(12), December 2006.

Devy et al. 1995 → M. Devy, R. Chatila, P. Fillatreau, S. Lacroix and F. Nashashibi. "On autonomous navigation in a natural environment". Robotics and Autonomous Systems, vol. 16, no. 1, pages 5-16, 1995.

Di Marco et al. 2001→ M. Di Marco, A. Garulli, S. Lacroix and A. Vicino. "Set membership localization and mapping for autonomous navigation". International Journal of Robust and Nonlinear Control. 2001.

Dietmayer et al. 2001 → K. C. J Dietmayer, J. Sparbert, D. Streller. "Model based object classification and object tracking in traffic scenes from range images". Proceedings of the IV IEEE Intelligent Vehicles Symposium. 2001.

Doucet et al. 2000 → A. Doucet, N. de Freitas, K. Murphy, and S. Russell. "Rao-blackwellised particle Filtering for dynamic bayesian networks". Proceedings of Uncertainty in AI (UAI), 2000.

Durrant-Whyte et al 2006. → Hugh Durrant-Whyte and Tim Bailey, "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms". IEEE Robotics and Automation Magazine 2006.

Espinoza et al. 2007 → J. Espinoza, A. Sánchez , M.A. Osorio. “Exploring unknown environments with mobile robots using SRT-Radial”. IROS 2007.

Estrada et al. 2005 → C. Estrada, J. Neira, and J. D. Tardós. “Hierarchical SLAM: Real-Time Accurate Mapping of Large Environments”. IEEE Transactions on Robotics, VOL. 21, NO. 4, 2005.

Eustice et al. 2006 → R. M. Eustice, H. Singh, and J. J. Leonard. “Exactly sparse delayed-state filters for view-based SLAM”. IEEE Trans. on Robotics, 22(6):1100-1114, Dec 2006.

Feder et al. 1999 → H.J.S. Feder, J.J. Leonard, and C.M. Smith. “Adaptive mobile robot navigation and mapping”. International J. of Robotics Research, 18(7):650–668, 1999.

Feder et al. 1999 → H. Feder, J. Leonard, and C. Smith. "Adaptive mobile robot navigation and mapping", International J. of Robotics Research, vol. 18, no. 7, pp. 650-668, 1999.

Fisher et al. 2004 → J. Fisher, J. Lowther, C. Shene. ” If you know b-splines well, you also nnow NURBS!”. Proceedings of the 35th SIGCSE technical symposium on Computer science education. 2004.

Franchi et al. 2007 → Franchi, A., Freda, L., Oriolo, G., Vendittelli, M., 2007. “A randomized strategy for cooperative robot exploration”. Proceedings of the IEEE International Conference on Robotics and Automation, Roma, Italy. 2007.

Franchi et al. 2009 → A. Franchi, L. Freda, G. Oriolo, M. Vendittelli. “The Sensor-based Random Graph Method for Cooperative Robot Exploration”. Transactions on Mechatronics, Vol. 14, NO. 2, 2009.

Frese 2006 → U. Frese. “Treemap: An O(log n) algorithm for indoor simultaneous localization and mapping”. Autonomous Robots, 21(2):103-122, September 2006.

Frese 2006b → U. Frese. “Using Treemap as a Generic Least Square Backend for 6-DOF SLAM”. Procedings of the Spatial Cognition V Workshop Robotic 3D Environment Cognition. 2006.

Garrido et al. 2008 → S. Garrido, L. Moreno and D. Blanco. "Exploration of a cluttered environment using Voronoi transform and fast marching". Robotics and Autonomous Systems 56 (12), 1069–1081. 2008.

Gonzalez et al. 2002 → H. H. González-Baños and J.C. Latombe. "Navigation strategies for exploring indoor environments". Int. J. Robotic Res. 2002.

Grisetti et al. 2005 → G. Grisetti, C. Stachniss, and W. Burgard. "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling". Proceedings of the IEEE International Conference on Robotics and Automation, 2005.

Grisetti et al. 2007 → G. Grisetti, C. Stachniss, and W. Burgard. "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters". IEEE Transaction on Robotics, 23(1):34-46, 2007.

He et al. 2008 → X. C. He and N. H. C. Yung. "Corner detector based on global and local curvature properties". Optical Engineering. 2008.

Howard et al. 2006 → A. Howard, G. S. Sukhatme, and M. J. Mataric. "Multi-robot mapping using manifold representations". Proceedings of the IEEE - Special Issue on Multi-robot Systems, 2006.

Howard et al. 2006b → A. Howard, L. E. Parker, and G. S. Sukhatme. "Experiments with large heterogeneous mobile robot team: Exploration, mapping, deployment and detection". International Journal of Robotics Research, 25(5):431–447, 2006.

Hu et al. 2002 → S. Hu, C. Tai and S. Zhang. "An extension algorithm for B-splines by curve unclamping". Computer-Aided, 2002.

Ishida 1997 → J. Ishida. "The general B-spline interpolation method and its application to the modification of curves and surfaces". Computer-Aided Design, Volume 29, Issue 11, Pages 779–790, 1997.

Jaulin 2009 → L. Jaulin. "A Nonlinear Set Membership Approach for the Localization and Map Building of Underwater Robots". IEEE Transactions on Robotics, 2009.

Joe et al. 1990 → B. Joe. "Knot insertion for Beta-spline curves and surfaces". ACM Transactions on Graphics, 1990.

Julia et al. 2008 → M. Juliá, A. Gil, L. Payá and O. Reinoso. “Local minima detection in potential field based cooperative multi-robot exploration”. International Journal of Factory Automation, Robotics and Soft Computing 3. , 2008.

Julia et al. 2010 → M. Juliá, O. Reinoso, A. Gil, M. Ballesta and L. Payá. “A hybrid solution to the multi-robot integrated exploration problem”. Engineering Applications of Artificial Intelligence . 2010.

Julia et al. 2011 → M. Juliá, O. Reinoso, A. Gil, M. Ballesta and L. Payá. “Behaviour Based Multi-Robot Integrated Exploration”. International Journal of Innovative Computing, Information and Control. 2011.

Julier et al. 2001 → S. J. Julier and J. K. Uhlmann. “A Counter Example to the Theoryof Simultaneous Localization and Map Building”. In 2001 IEEE Int. Conf. on Robotics and Automation, pages 4238-4243, Seoul, Korea, 2001.

Kaess et al. 2007 → M. Kaess, A. Ranganathan, and F. Dellaert. “iSAM: Fast incremental smoothing and mapping with efficient data association”. In IEEE Int. Conf. Robot. Automat., pages 1670– 1677, Rome, 2007.

Khatib 1986 → 0. Khatib. "Real-Time Obstacle Avoidance for Manipulator and Mobile Robots". The Int. JRobotics Research, vol. 5, no. 1, pp. 90-98, 1986.

Kim et al. 2008 → C. Kim, R. Sakthivel and W. K. Chung. “Unscented FastSLAM: A Robust and Efficient Solution to the SLAM Problem”. IEEE Transactions on Robotics, Vol. 24, 2008.

Kuhn 1995 → H. W. Kuhn. "The Hungarian method for the assignment problem". Naval Research Logistics Quarterly, **2**:83–97. 1955.

Kuipers et al. 1991 → B. Kuipers and Y.T. Byun. “A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations”. Journal of Robotics and Autonomous Systems, 8:47–63, 1991.

Kuipers et al. 1991 →B. Kuipers and Y.T. Byun. “A robot exploration and mapping strategy based on asemantic hierarchy of spatial representations”. Journal of Robotics and Autonomous Systems, 8:47–63, 1991.

Lau 2003 → H. Lau. “Behavioural approach for multi-robot exploration”. Proceedings of the Australasian Conference on Robotics and Automation, 2003.

LaValle 1998 → S. M. LaValle. “Rapidly-exploring random trees: A new tool for path planning”. TR 98-11, Computer Science Dept., Iowa State University, 1998.

Le Bars te al. 2010 → F. Le Bars, A. Bertholom, J. SLIWKA and L. JAULIN. “Interval SLAM for underwater robots; a new experiment”. 8th IFAC Symposium on Nonlinear Control Systems. 2010.

Leonard et al. 1991 → J.J. Leonard and H.F. Durrant-Whyte. “Simultaneous map building and localization for an autonomous mobile robot”. In IEEE International Workshop on Intelligent Robots and Systems, pages 1442–1447, 1991.

Leonard et al, 1992 → J. Leonard and H. Durrant-Whyte. “Directed Sonar Sensing for Mobile Robot Navigation”. Boston: Kluwert Accademic Publisher. 1992.

Leonard et al. 2000 → J. J. Leonard and H. J. S. Feder. “A computationally e±cient method for large-scale concurrent mapping and localization”. In D. Koditschek and J. Hollerbach, editors, Robotics Research: The Ninth International Symposium, pages 169-176, Snowbird, Utah, 2000.

Leonard et al. 2003 → J. J. Leonard and P. M. Newman. “Consistent, convergent and constant-time SLAM”. In Int. Joint Conf. on Artificial Intelligence, 2003.

Leung et al. 2008 → C. Leung, S. Huang, and G. Dissanayake. “Active SLAM for structured environments”. In IEEE Int. Conf. Robot. Automat., pages 1898–1903, 2008.

Lisien et al. 2005 → B. Lisien, D. Morales, D. Silver, G. Kantor, H. Rekleitis, and I. Choset. “The hierarchical atlas”. IEEE Transactions on Robotics and Automation, 21:473–481, 2005.

Lu et al. 1997 → F. Lu and E. Milios. “Globally consistent range scan alignment for environment mapping”. Autonomous Robots, 4:333–349, 1997.

Lu et al. 1997 → F. Lu and E. Milios. “Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans”. Journal of Intelligent and Robotic Systems 18: 249–275, 1997.

Maimone et al. 2004 → M. Maimone, A. Johnson, Y. Cheng, R. Willson, and L. Matthies. “Autonomous navigation results from the mars exploration rover (mer) mission”. In 9th International Symposium on Experimental Robotics (ISER), 2004.

Makarenko et al. 2002 → A. A. Makarenko, S. B. Williams, F. Bourgault, H. F. Durrant-Whyte, "An experiment in integrated exploration", IEEEIRSJ Int. Conf. on Intelligent Robots and System, Vol 1, pp. 534-539, 2002.

Mokhtarian 1995 → F. Mokhtarian. “Silhouette-based isolated object recognition through curvature scale space”. IEEE Pattern Analysis and Machine Intelligence 17(5): 539–544. 1995.

Montemerlo et al. 2002 → M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. “FastSLAM: A factored solution to the simultaneous localization and mapping problem”. In Proceedings of the AAAI National Conference on Artificial Intelligence, 2002.

Montemerlo et al. 2003→ M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, “FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges”. Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI), 2003.

Mount et al. 2007 → D. M. Mount, N. S. Netanyahuy, K. Romanikz, R. Silvermanx, A. Y. Wu. “A Practical Approximation Algorithm for the LMS Line Estimator”. Computational Statistics & Data analysis. 2007.

Newmant et al. 2003 → P. M. Newman, M. Bosse, and J. J. Leonard. “Autonomous feature-based exploration”. In IEEE Int. Conf. Robot. Automat., pages 1234–1240, 2003.

Nieto et al. 2003 → J. Nieto, J. Guivant, E. Nebot and S. Thrun. “Real time data association for fastslam”. IEEE International Conference on Robotics and Automation, 2003.

Oriolo et al 2004 → G. Oriolo, M. Vendittelli, L. Freda and G. Troso. “The SRT Method: Randomized strategies for exploration”. IEEE International Conference on Robotics and Automation, 2004.

Paskin et al. 2003 → M. A. Paskin. « Thin Junction Tree Filters for Simultaneous Localization and Mapping". Int. Joint Conf. Artificial Intelligence, pages 1157-1164, 2003.

Pavlidis et al. 1974 → T. Pavlidis, S. L. Horowitz. "Segmentation of plane curves". IEEE Transactions on Computers, 1974.

Paz et al. 2007 →L. M. Paz, P. Jensfelt, J. D. Tardos and J. Neira. "EKF SLAM updates in O(n) with Divide and Conquer SLAM". In Proc. IEEE Int. Conf. Robotics and Automation, 2007.

Pedraza et al. 2007 → L. Pedraza, G. Dissanayake, J. Valls Miro, D. Rodriguez-Losada and F. Matia. "BS-SLAM: Shaping the World". Proceedings of Robotics: Science and Systems, 2007.

Paz et al. 2008 → L. M. Paz, J. D. Tardos, and J. Neira. "Divide and conquer: EKF SLAM in O(n)". IEEE Trans. on Robotics, 2008.

Pedraza et al. 2009 → L. Pedraza, D. Rodriguez-Losada, F. Matia, G. Dissanayake and J.V. Miro. "Extending the Limits of Feature-Based SLAM with B-Splines", Robotics, IEEE Transactions on, 2009.

Rogers 2001 → D.F. Rogers. "An introduction to NURBS with historical perspective". 2001.

Schultz et al. 1999 → A.C. Schultz, W. Adams, B. Yamauchi, and M. Jones. "Unifying exploration, localization, navigation and planning through a common representation". In IEEE International Conference on Robotics and Automation, pages 2651–2658, 1999.

Sim 2005 → R. Sim, "Stable Exploration for Bearings-only SLAM", Proceedings of the IEEE International Conference on Robotics and Automation, 2005.

Sim 2005b → R. Sim. "Stabilizing information-driven exploration for bearings-only SLAM using range gating". In IEEE/RSJ Int.Conf.on Intelligent Robots and Systems, pages 3396–3401, 2005.

Sim et al. 2005 → R. Sim and N. Roy. "Global A-optimal robot exploration in SLAM". In IEEE Int. Conf. Robot. Automat., pages 673–678, 2005.

Simons et al. 2000 → R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun and H. Younes. “Coordination for multi-robot exploration and mapping”. Proceedings of the AAAI National Conference on Artificial Intelligence, 2000.

Smith *et al.* 1987 → R. Smith, M. Self, and P. Cheeseman. “A stochastic map for uncertain spatial relationships”. In 4th International Symposium on Robotics Research. MIT Press, 1987.

Sola et al. 2008 → J. Sola, A. Monin, M. Devy & T. Vidal-Calleja. “Fusing Monocular Information in Multicamera SLAM”. IEEE Transactions on Robotics, vol. 24, no. 5, pages 958-968, 2008.

Stachniss et al. 2003 → C. Stachniss and W. Burgard. “Exploring unknow environments with mobil robots using coverage maps”. In int. Joint Conf. Artifitial intelligence, pages 1127-1134, 2003.

Stachniss et al. 2006 → C. Stachniss, O.M. Mozos and W. Burgard. “Speeding-up multi-robot exploration by considering semantic place information”. Proceedings of the IEEE International Conference on Robotics and Automation, 2006.

Tardos et al. 2002 → J. D. Tardos, J. Neira, P. Newman, and J. Leonard. “Robust mapping and localization in indoor environments using sonar data”. Int. J. Robotics Research, 21(4):311-330, 2002.

Tardos et al. 2002b → Juan D. Tardós. “Data Association in SLAM”. Summer School on SLAM. 2002.

Thrun et al. 2005 → S. Thrun, W. Burgard, and D. Fox. “Probabilistic Robotics”. The MIT Press, 2005.

Vidal-Calleja et al. 2006 → T. Vidal-Calleja, A. J. Davison, J. Andrade-Cetto, and D.W. Murray. “Active control for single camera SLAM”. In IEEE Int. Conf. Robot. Automat., pages 1930–1936, 2006.

Walter et al. 2007 → M. R. Walter, R. M. Eustice, and J. J. Leonard. “Exactly sparse extended information filters for feature-based SLAM”. Int. J. Robotics Research, 26(4):335-359, 2007.

Williams et al. 2002 → S. B. Williams, G. Dissanayake, and H. Durrant-Whyte. “An efficient approach to the simultaneous localization and mapping problem”. In IEEE Int. Conf. on Robotics and Automation, Vol. 1, pages 406-411, 2002.

Williams et al. 2004 → S. Williams and I. Mahon. “Simultaneous localization and mapping on the great barrier reef”. In Proceedings of the IEEE International Conference on Robotics and Automation, 2004.

Wurm et al. 2008 → K.M. Wurm, C. Stachniss and W. Burgard. “Coordinated multi-robot exploration using a segmentation of the environment”. Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems, 2008.

Xiaoping et al. 1997 → Y. Xiaoping and T. Ko-Cheng. “A wall-following method for escaping local minima in potential field based motion planning”. Proceedings of the International Conference on Advanced Robotics, 1997.

Yamauchi 1997 → B. Yamauchi. “A frontier based approach for autonomous exploration”. Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, 1997.

Yamauchi et al. 1998 → B. Yamauchi, A. Schultz, and W. Adams. “Mobile robot exploration and map-building with continuous localization”. In IEEE International Conference on Robotics and Automation, pages 3715–3720, 1998.

Zhang et al. 2000. → L. Zhang and B.K. Ghost. “Line segment based map building and localization using 2D laser range finder”. IEEE International conference on robotics and Automation. Vol. 3. 2000.

Zunino et al. 2001. → G. Zunino and H.I. Christensen, “Simultaneous Localization and Mapping in domestic Environments”. International Conference on Multisensor Fusion and Integration for Intelligent Systems. 2001.