UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RENAN DE QUEIROZ MAFFEI

Segmented DP-SLAM

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Prof. Dr. Edson Prestes e Silva Junior
Advisor

Porto Alegre, May 2013

CIP – CATALOGING-IN-PUBLICATION

Maffei, Renan de Queiroz

Segmented DP-SLAM / Renan de Queiroz Maffei. – Porto Alegre: PPGC da UFRGS, 2013.

110 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2013. Advisor: Edson Prestes e Silva Junior.

1. SLAM. 2. Rao-BlackWellized Particle Filter. 3. Submap-based SLAM. I. Silva Junior, Edson Prestes e. II. Título.

Maxwell Smart, the agent 86 of C.O.N.T.R.O.L., and the famous Hawaiian detective Harry Hoo are in a crime scene investigation, when Hoo sees an ashtray with two cigarettes in it...

Hoo *- "Amazing!"*

Smart *- "You found something, Mr. Hoo?"*

Hoo *- "Notice please, ashtray contains two cigarettes."*

Smart *- "I see. Well, then that means there was someone here in the room with the victim when he was killed. Perhaps even the murderer."*

Hoo *- "Moment please."*

Smart *- "Moment please?"*

Hoo *- "Once again worthy C.O.N.T.R.O.L. agent leaps to obvious conclusion. If victim were non-smoker, could have been two smokers in room with him."*

Smart *- "Well, that's true."*

Hoo *- "Perhaps two smokers and one non-smoker."*

Smart *- "Moment please."*

Hoo *- "Moment please?"*

Smart *- "Perhaps there were two smokers and two non-smokers."*

Hoo *- "Perhaps two smokers and four non-smokers."*

Smart *- "Well, then what you are saying, Mr. Hoo, is that there could have been as many as fifty people in this room with the victim when he was killed. Provided that only two of them were smokers."*

Hoo *- "Exactly!"*

Smart *- "Boy, it must have been pretty crowded in here."*

— GET SMART - EPISODE 25: THE AMAZING HARRY HOO

AGRADECIMENTOS

Agradeço primeiramente à minha família. Aos meus pais e à Letícia, minha irmã, pela força que me deram enquanto eu fazia este trabalho e por tudo que me proporcionaram para que alcançasse este momento.

Agradeço ao professor Édson pela orientação, pois esta foi fundamental para que este trabalho alcançasse o resultado atual. Também agradeço à professora Sílvia, pois comecei minha vida acadêmica trabalhando com ela, e foi durante esse período que surgiu meu interesse na robótica.

Agradeço ao Vitor pela imensa colaboração no período final deste trabalho, onde ele foi o co-orientador não-oficial da minha dissertação. E ao Heinrich pela colaboração durante o período de mestrado.

Por fim, agradeço aos colegas Rodrigo e Pedro pela companhia diária nesses últimos dois anos.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

DP-SLAM	Distributed Particle SLAM
DP-Mapping	Distributed Particle Mapping
EKF	Extended Kalman Filter
MDP-SLAM	Modified Distributed Particle SLAM
PGS	Probabilistic Graph of Segments
PMT	Probabilistic Topological Maps
RBPF	Rao-Blackwellized Particle Filter
SDP-SLAM	Segmented Distributed Particle SLAM
SLAM	Simultaneous Localization and Mapping

LIST OF ALGORITHMS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Simultaneous Localization and Mapping (SLAM) is one of the most difficult tasks in mobile robotics, since there is a mutual dependency between the estimation of the robot pose and the construction of the environment map. Most successful strategies in SLAM focus in building a probabilistic metric map employing Bayesian filtering techniques. While these methods allow the construction of consistent and coherent local solutions, the SLAM remains a critical problem in operations within large environments. To circumvent this limitation, many strategies divide the environment in small regions, and formulate the SLAM problem as a combination of multiple precise metric submaps associated in a topological map.

This work proposes a SLAM method based on the Distributed Particle SLAM (DP-SLAM) and the Segmented SLAM (SegSLAM) algorithms. SegSLAM is an algorithm that generates multiple submaps for every region of the environment, and then build the global map by selecting combinations of submaps. DP-SLAM is a Rao-Blackwellized particle filter algorithm that uses an efficient distributed representation of the particles maps associated with an ancestry tree of the particles. The distributed characteristic of these structures favors the combination of locally accurate map segments, that can increase the diversity of global level solutions.

The algorithm proposed in this dissertation, called SDP-SLAM, segments and combines different hypotheses of robot trajectories to reconstruct the environment map. Our main contributions are the development of novel strategies for the matching of submaps and for the estimation of good submaps combinations. SDP-SLAM was evaluated through experiments performed by a mobile robot operating in real and simulated environments.

Keywords: SLAM, Rao-BlackWellized Particle Filter, Submap-based SLAM.

Segmented DP-SLAM

RESUMO

Localização e Mapeamento Simultâneos (SLAM) é uma das tarefas mais difíceis em robótica móvel, uma vez que existe uma dependência mútua entre a estimativa da localização do robô e a construção do mapa de ambiente. As estratégias de SLAM mais bem sucedidas focam na construção de um mapa métrico probabilístico empregando técnicas de filtragem Bayesiana. Embora tais métodos permitam a construção de soluções localmente consistentes e coerentes, o SLAM continua sendo um problema crítico em operações em ambientes grandes. Para contornar esta limitação, muitas estratégias dividem o ambiente em pequenas regiões, e formulam o problema de SLAM como uma combinação de múltiplos submapas métricos precisos associados em um mapa topológico.

Este trabalho propõe um método de SLAM baseado nos algoritmos DP-SLAM (*Distributed Particle SLAM*) e SegSlam (*Segmented SLAM*). SegSLAM é um algoritmo que cria múltiplos submapas para cada região do ambiente, e posteriormente constrói o mapa global selecionando combinações de submapas. Por sua vez, DP-SLAM é um algoritmo de filtro de particulas Rao-Blackwellized que utiliza uma representação distribuída eficiente dos mapas das partículas, juntamente com a árvore de ascendência das partículas. A característica distribuída destas estruturas é favorável para a combinação de diferentes segmentos de mapa localmente precisos, o que aumenta a diversidade de soluções.

O algoritmo proposto nesta dissertação, chamado SDP-SLAM, segmenta e combina diferentes hipóteses de trajetórias do robô, a fim de reconstruir o mapa do ambiente. Nossas principais contribuições são o desenvolvimento de novas estratégias para o casamento de submapas e para a estimativa de boas combinações de submapas. O SDP-SLAM foi avaliado através de experimentos realizados por um robô móvel operando em ambientes reais e simulados.

Palavras-chave: SLAM, Filtro de Partículas Rao-Blackwellized, SLAM baseado em submapas.

1 INTRODUCTION

Robotics can be defined as the science of perceiving and manipulating the physical world through computer-controlled mechanical devices (THRUN; BURGARD; FOX, 2005). In today's world, robotic systems are operating in environments increasingly unpredictable and less controllable. While in some situations, such as assembly lines, the predictability of the environment is high, in others, such as domestic environments or highways, the same cannot be said.

One of the major requirements of a wide range of robotic applications is the knowledge of the area where the robot is operating. If a robot is able to model a map of the environment from the sensors measurements, then it can perform tasks such as search and rescue operations, detection and mapping of landmines, transportation activities, and so on. In fact, robots that can acquire a precise model of the environment on their own fulfill a fundamental precondition of truly autonomous agents (STACHNISS, 2009). However, the construction of an accurate map requires the precise localization of the robot, that can be difficult to obtain. Thus, most of the time, both mapping and localization needs to be solved concurrently, in the so-called Simultaneous Localization and Mapping problem (SLAM).

Our focus in this work is in solving the SLAM problem in structured indoor environments, using a Pioneer 3-DX robot equipped with a laser rangefinder. Although numerous solutions have been proposed (STACHNISS, 2009), there is still room for improvement, as we will show in this work.

1.1 Motivation

The three main requirements for building truly autonomous robots is that they have the ability to solve the tasks of navigation, localization and mapping.

The **motion planning** or **navigation** problem consists in efficiently guide the robot to a desired location in the environment. The planning of the robot motion towards a goal position requires the existence of a precise environment map and the knowledge about the robot pose.

The **localization** problem requires that a mobile robot, using the readings of its sensors, is able to estimate its correct pose, consulting an exact representation of the environment. On some occasions, when the initial configuration of the robot is known, the localization can be considered an incremental problem, called *local localization*, because only the estimation of the subsequent pose of the robot are needed during the movement of the robot. A more complex version of the problem is the kidnapped robot situation, called *global localization*, in which a robot is placed inside an environment without any indication of its initial pose, and, by analyzing its surroundings, the localization must be estimated.

The **mapping** problem can be considered, in some way, the opposite to the localization. In this problem we assume that the robot has the exact knowledge of its pose, either because there are no odometry errors or because there is a perfect global positioning system. In both situations, there is no information about the environment, thus the robot needs to build an adequate representation that contain the information about obstacles (dynamic or static), free-space, or other entities (danger zones, objects, etc) that are needed to perform its task adequately. If there are errors on the sensors measurements, implying that observations of a same entity can differ, it is required an incremental analysis of the measurements from different positions in space, in order to enhance the estimation process over time.

That said, in most applications of mobile robotics, these three tasks cannot be resolved independently. Figure 1.1 shows the problems originated by the combinations of these tasks (MAKARENKO et al., 2002).

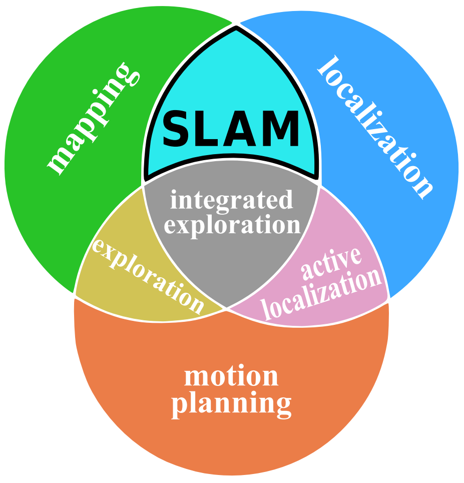


Figure 1.1: The three basic tasks of mobile robotics and their combinations (represented by overlapping areas), emphasizing the SLAM problem (focus of this work). Figure adapted from (MAKARENKO et al., 2002).

The **active localization** problem tries to guide the robot to places inside a known map where the estimation of the robot pose is improved. Conversely, the **exploration** problem defines trajectories that lead the robot to build the entire map, assuming that the robot pose is always known. The solution of these two problems consists in planning the robot motion to estimate an information (eg. the robot pose), considering the knowledge of the other information (eg. the map).

However, building a map of the environment simultaneously to the localization of the robot is a problem that cannot be decoupled. This problem, called **simultaneous localization and mapping (SLAM)**, is a hard task because the robot needs to know its precise pose to build a correct map and it needs a reliable environment map to calculate its pose. Figure 1.2 shows the importance of the SLAM when working in real environments, by comparing a map built using only odometry information to estimate the localization, which usually has a high degree of uncertainty associated, with the same map built using SLAM.

Finally, the intersection of the three fundamental tasks compose the **integrated exploration** problem. In integrated exploration, the robot must plan its navigation through

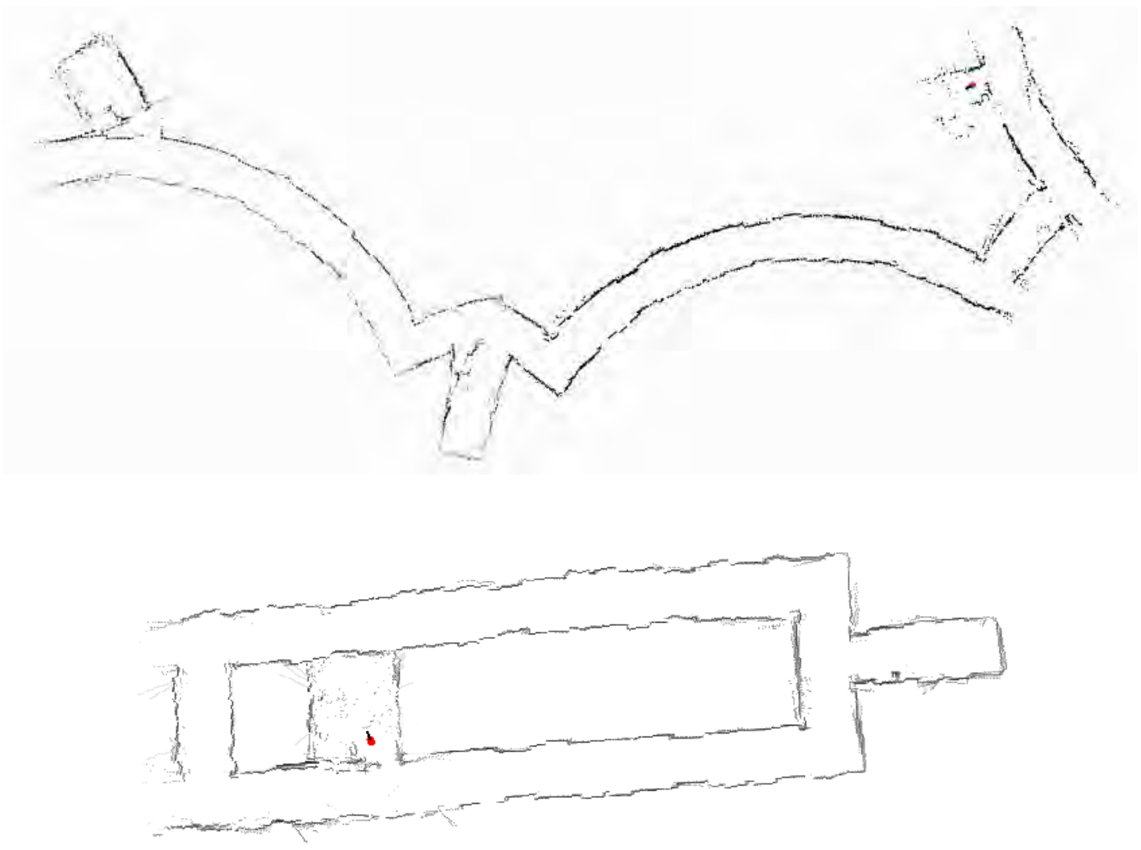


Figure 1.2: Comparison of a map built using only odometry information to estimate the robot pose (top), with the same map built using SLAM (bottom). The map is from a floor in a building of UFRGS Institute of Informatics.

the environment, at the same time that it estimates its precise pose and the map. The actions taken by the robot must enhance the quality of both the localization and the mapping process. Thus, a good integrated exploration strategy should balance the exploration of unknown regions, that advances the environment mapping, and the revisiting of known areas, that reduces the uncertainty on the robot localization.

The focus of our work is on SLAM. Only after possessing a SLAM strategy, that is possible to develop an integrated exploration approach. Therefore, solving the SLAM problem is a major requirement to the construction of real autonomous robots and indeed it has been the focus of much research over the last decades.

The literature presents several approaches to solve the problem of SLAM, and can be divided into two distinct groups: online SLAM, where the estimations over current robot pose and map are increased always that new measurements become available, and full SLAM, where the estimations over the full robot trajectory are obtained from the full set of measurements. The first, and most popular, group consists mainly of strategies based on Bayesian filtering, like Extended Kalman filters (EKF) (SMITH; SELF; CHEESEMAN, 1990), while the second group is grounded on least-square error minimization techniques, usually performed offline, like graph-based methods (NI; STEEDLY; DELLAERT, 2007) (GRISETTI et al., 2010). A popular incremental full SLAM approach is the Rao-Blackwellized particle filter (RBPF) (MONTEMERLO et al., 2002) (HAHNEL et al., 2003) (ELIAZAR; PARR, 2003), that separates the estimation of the robot path from the estimation of the map, through a process called Rao-Blackwellization.

Although many SLAM algorithms have been successfully presented, the SLAM problem remains particularly challenging when working in large scale domains. Over the last years, the size of the environments has increased by several magnitudes, and hence the time required by those methods to obtain adequate solutions has also increased. To circumvent the computational complexity of a large scale environment, many approaches reduce the problem into solving low-level instances of SLAM (inside limited regions of the map) and later globally adjusting the individual solutions into a complete map (ESTRADA; NEIRA; TARDOS, 2005; ELIAZAR; PARR, 2006; NI; STEEDLY; DELLAERT, 2007; PAZ; TARDOS; NEIRA, 2008; FAIRFIELD; WETTERGREEN; KANTOR, 2010; LEE; LEE; OH, 2011).

These methods, referred as submap-based SLAM, usually combine a topological map with metric maps, such as occupancy grids. In most strategies, the set of individual limited regions of the environment are represented by different small metric maps that are refined independently using an ordinary SLAM algorithm. Each one of these regions is a node of a topological map, whose edges define the relationships between the submaps. Given the reduced size of submaps, the problem of large-scale environments is bypassed, yet new challenges arise, such as the matching of neighboring submaps, the decision of when segmenting the environment, and the re-entry of already existent submaps to close loops.

1.2 Objectives

In this work, we propose a submap-based particle filter algorithm called Segmented Distributed Particle SLAM (SDP-SLAM), which combines an optimized data structure to store the maps of the particles with a probabilistic map of segments, representing hypotheses of submaps topologies.

Our method is based on the Segmented SLAM (SegSLAM) algorithm (FAIRFIELD; WETTERGREEN; KANTOR, 2010) and on the Distributed-Particle SLAM (DP-SLAM) algorithm (ELIAZAR; PARR, 2003). SegSLAM is a method that allows partitioning the map into several regions containing various submaps and, later, combine these submaps to generate global solutions. DP-SLAM, one of the earliest Rao-Blackwellized Particle Filter methods, defines an optimized joint representation of the particles maps, that reduces the storage space required, and improves the performance of the map updating process.

We perceived that this distributed aspect of DP-SLAM could be adapted to support the segmentation of the environment in multiples submaps. Thus, with some modifications over the data structure, we extended DP-SLAM to a submap-based approach that easily allows the combination of map segments from different particles, and consequently, generates multiples hypotheses of coherent trajectories of the robot (i.e. continuous trajectories, in which the local errors respect the robot motion model). With this process, the diversity of solutions of the particle filter is very high, even maintaining only a set of few active particles.

The idea of combining submaps of different particles to increase the diversity of hypotheses was introduced by the SegSLAM algorithm (FAIRFIELD; WETTERGREEN; KANTOR, 2010). However, SegSLAM does not focus on the construction of a global map of the environment, instead its goal is to build locally reliable associations between segments, that produces good representations of the environment surrounding the robot, but can lead to inconsistencies in regions far away from the robot. In our approach, we developed an estimation process with a second particle filter, that uses a probabilistic topological map to guide the searching of segments combinations, aiming at the generation of precise global solutions.

Another contribution of our method is a new approach of submaps matching, used to evaluate the quality of segments combinations. Our approach performs the matching of points extracted from the free-space, instead of points extracted from the obstacles, as in SegSLAM. This strategy reduces the ambiguities in the matching, because, for example, it eliminates the doubts of what regions are inside and what regions are outside the walls that the robot detected.

1.3 Organization

This dissertation is divided as follows:

Chapter 2 discusses the SLAM problem with further details. First, it presents a formal derivation of the problem, based on Bayesian filtering. Then, it discusses the main approaches to solve the SLAM problem, like graph-based strategies, EKF methods and particle filters. Next, it focus on the Rao-Blackwellized particle filter (RBPF) approach, detailing quintessential RBPF methods. At the end, it discusses some of the most influential improvements on RBPF SLAM, focusing on the submap-based approaches.

Chapter 3 starts describing our initial approach of SLAM, which was an important step to the construction of our method. Then, it describes our proposed method of submap-based SLAM, called SDP-SLAM. It presents an overview of SDP-SLAM algorithm, followed by the formal derivation of the method. Later, it discusses the individual steps of SDP-SLAM, beginning with the segmentation process, passing by the map matching process and concluding with the topology estimation using a probabilistic graph of segments.

Chapter 4 presents experimental results of the application of our method to solve SLAM. It shows the solutions of experiments in both simulated and real environments, focusing on large nested loops configurations, which are particularly challenging for online SLAM approaches. It also compares SDP-SLAM with others RBPF and submap-based strategies, showing the good performance of our method.

Finally, Chapter 5 presents our conclusions and proposes ideas for future work.

2 THEORETICAL FOUNDATION

This chapter discusses the Simultaneous Localization and Mapping problem, starting by the presentation of a formal derivation of the problem, based on Bayesian filtering. Next, the main approaches to solve the SLAM problem are discussed, such as graph-based strategies, EKF methods, and particle filters. We focus on the Rao-Blackwellized particle filter (RBPF) approach, group to which our algorithm belongs, presenting its formal definition and some of the most influential methods. In particular, we present in detail the DP-SLAM algorithm, which served as basis for our method. Then, we discuss the disadvantages of DP-SLAM and the others traditional RBPF algorithms, while presenting some of the most influential improvements over RBPFs. Finally, we discuss submap-based approaches, focusing on SegSLAM.

2.1 Simultaneous Localization and Mapping (SLAM)

In many practical cases in Robotics, requirements as the existence of an accurate map or the exact location of a robot may not be provided. Therefore, one of the robot activities is to build a map of an unknown environment, based only on proprioceptive and exteroceptive sensors measurements. Since the mapping process requires the knowledge about the precise robot pose, and the exact localization is only obtained through a pre-defined map, the solution is to perform two estimations simultaneously, or as it is known, solve the SLAM problem.

Before presenting the definition of the SLAM problem, we introduce the nomenclature used in this dissertation, that are exemplified in Figure 2.1:

- $\mathbf{x}_t$ is the position and orientation of the robot at the instant t. Likewise, the trajectory developed by the robot during its course is given by $\mathbf{x}_{1:t} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t\} = \{\mathbf{x}_{1:t-1}, \mathbf{x}_t\}$

- $\mathbf{u}_t$ is the control vector applied at instant $t-1$ that takes the robot to the pose $\mathbf{x}_t$ at instant t. It is generally given by the odometry measurement between the instants $t-1$ and t. The history of control commands is represented by $\mathbf{u}_{1:t} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_t\} = \{\mathbf{u}_{1:t-1}, \mathbf{u}_t\}$

- $\mathbf{m}_i$ is the position of the landmark i, that is fixed during the whole SLAM process. The set of all landmarks is given by $\mathbf{m} = \{\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_n\}$

- $\mathbf{z}_{i,t}$ is the observation about the position of the landmark i, made by the robot at instant t. $\mathbf{z}_t = \{\mathbf{z}_{0,t}, \mathbf{z}_{1,t} \cdots, \mathbf{z}_{n,t}\}$ is the set of all observations made by the robot at the same instant of time, while $\mathbf{z}_{1:t} = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_t\} = \{\mathbf{z}_{1:t-1}, \mathbf{z}_t\}$ represents the history of all observations.

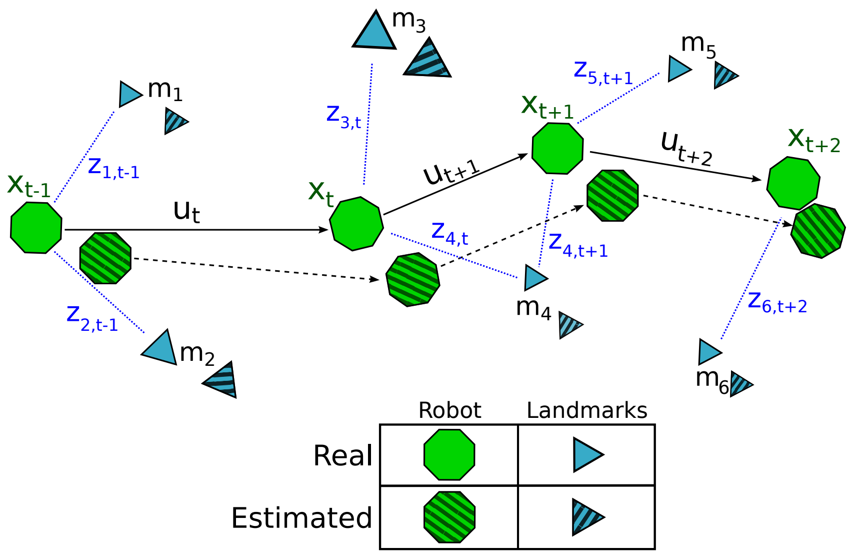


Figure 2.1: Exemplification of the SLAM problem. The real pose of the robot $\mathbf{x}$ and the position of the landmarks in the environment $\mathbf{m}$ are unknown. Through the information of control defined by the odometry measurements $\mathbf{u}$ and the observations made by the robot $\mathbf{z}$, the localization and the mapping are estimated.

In the example of Figure 2.1, the robot at time $t-1$ is observing the landmarks $\mathbf{m}_1$ and $\mathbf{m}_2$. Based on the robot observations $\mathbf{z}_{1,t-1}$ and $\mathbf{z}_{2,t-1}$, the positions of $\mathbf{m}_1$ and $\mathbf{m}_2$ are estimated. Then, with the action $\mathbf{u}_t$, the robot moves from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$. The process is repeated in the subsequent steps. At instant t, another two landmarks, $\mathbf{m}_3$ and $\mathbf{m}_4$, are observed. Later, at $t+1$, the robot observes $\mathbf{m}_5$ and, once again, $\mathbf{m}_4$. Finally, at $t+2$, $\mathbf{m}_6$ is observed.

We can formulate the SLAM problem as a Dynamic Bayesian Network, shown in Figure2.2. The observed variables are the actions $\mathbf{u}_{1:t}$ and observations $\mathbf{z}_{1:t}$ made by the robot, while the hidden variables (ie. variables that must be estimated from the observed variables) are the robot pose $\mathbf{x}_t$ and the map $\mathbf{m}$.

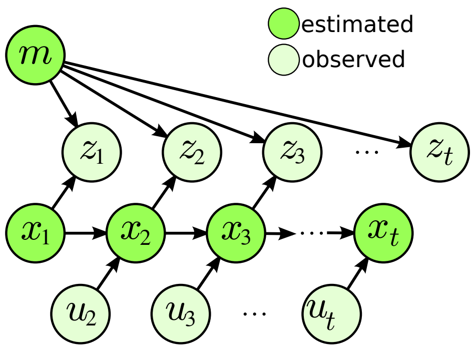


Figure 2.2: Formulation of SLAM as a Dynamic Bayesian Network. Observed states (measurements $\mathbf{z}$ and odometry $\mathbf{u}$) are dependent on the hidden states (robot pose $\mathbf{x}$ and map $\mathbf{m}$).

There are two different approaches to SLAM, analyzing from a probabilistic perspective (THRUN; BURGARD; FOX, 2005): the online SLAM and the full SLAM. In **online SLAM**, it is estimated the posterior probability that, at instant t, the robot is at pose $\mathbf{x}_t$ and has the map $\mathbf{m}$, given the sets of observations $\mathbf{z}_{1:t}$ and odometry measurements $\mathbf{u}_{1:t}$

$$p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{2.1.1}$$

In contrast, the **full SLAM** estimates the posterior probability over the entire path traversed by the robot $\mathbf{x}_{1:t}$, instead of just the current robot pose, besides the map $\mathbf{m}$

$$p(\mathbf{x}_{1:t}, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{2.1.2}$$

The online SLAM posterior can be described as the result of successive integrations of poses from the full SLAM problem

$$p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \int\int \cdots \int\int\int p(\mathbf{x}_{1:t}, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \, d\mathbf{x}_1 \, d\mathbf{x}_2 \, d\mathbf{x}_3 \cdots d\mathbf{x}_{t-2} \, d\mathbf{x}_{t-1} \tag{2.1.3}$$

By integrating the SLAM posterior over the previous pose estimates, we obtain the marginal probability for $\mathbf{x}_t$, given all possible values of $\mathbf{x}_{1:t-1}$. In online SLAM, it does not matter which path the robot took to get into the pose $\mathbf{x}_t$, what matters is that the current robot pose and its map are $\mathbf{x}_t$ and $\mathbf{m}$.

The integrations in the Equation 2.1.3 are typically performed incrementally, as shown below.

$$\begin{aligned}
p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) &= \int\int \cdots \int\int p(\mathbf{x}_{2:t}, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \, d\mathbf{x}_2 \, d\mathbf{x}_3 \cdots d\mathbf{x}_{t-2} \, d\mathbf{x}_{t-1} \\
p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) &= \int\int \cdots \int p(\mathbf{x}_{3:t}, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \, d\mathbf{x}_3 \cdots d\mathbf{x}_{t-2} \, d\mathbf{x}_{t-1} \\
&\vdots \\
p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) &= \int\int p(\mathbf{x}_{t-2:t}, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \, d\mathbf{x}_{t-2} \, d\mathbf{x}_{t-1} \\
p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) &= \int p(\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \, d\mathbf{x}_{t-1}
\end{aligned}$$

The main advantage of online SLAM strategies is that they operate incrementally, discarding sensors readings and odometry measurements insofar as they are processed. Full SLAM approaches, on the other hand, usually rely on batch optimization techniques, that operate on the complete set of observations. These methods are very powerful, but have a high-cost associated, which is a strong constraint for their use in real-time operations. In recent years, this constraint has been minimized due to the advances in computational power. Nevertheless, incremental SLAM strategies still are the most popular and used by the robotic community.

2.2 Online SLAM derivation using Bayesian Filtering

The posterior distribution of the SLAM problem at time t can be defined through the recursive computation of the distribution at time $t - 1$ (MONTEMERLO; THRUN, 2007). The vast majority of methods of online SLAM are based on this recursive update rule, which is a kind of probabilistic Markov chain, called Bayesian filtering.

In this approach, the current pose of the robot $\mathbf{x}_t$ is given by a probabilistic function of the pose in the previous instant $\mathbf{x}_{t-1}$ and the odometry $\mathbf{u}_t$. This function, called **motion model**, describes how the movement should be performed by the robot and what particular amount of uncertainty must be inserted in the motion estimation.

$$p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \mathbf{u}_t) \tag{2.2.1}$$

Additionally, the **measurement model** (or observation model), which describes the behavior and uncertainty of the robot sensors, is a probabilistic function associated with the measurement $\mathbf{z}_t$, made it when the robot is in the position given by $\mathbf{x}_t$ and has the map given by the set of landmarks $\mathbf{m}$.

$$p(\mathbf{z}_t \,|\, \mathbf{x}_t, \mathbf{m}) \tag{2.2.2}$$

With the definition of these two functions, we can start the derivation of the recursive formula for the posterior distribution. First, we rewrite the posterior distribution of SLAM, from the equation 2.1.1, using the Bayes rule.

$$p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{x}_t, \mathbf{m}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) / p(\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

$$p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \overbrace{p(\mathbf{z}_t \,|\, \mathbf{x}_t, \mathbf{m}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})\, p(\mathbf{x}_t, \mathbf{m}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})} / p(\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

$$p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{z}_t \,|\, \mathbf{x}_t, \mathbf{m}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})\, \overbrace{p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})} / p(\mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

As seen in equation 2.2.2, of the measurement model, $\mathbf{z}_t$ depends only on $\mathbf{x}_t$ and $\mathbf{m}$, so we can rewrite the above equation as:

$$p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{z}_t \,|\, \mathbf{x}_t, \mathbf{m})\, p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})\, \underbrace{p(\mathbf{z}_{1:t-1} \,|\, \mathbf{u}_{1:t}) / p(\mathbf{z}_{1:t} \,|\, \mathbf{u}_{1:t})}_{\text{constant}}$$

$$p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta\, p(\mathbf{z}_t \,|\, \mathbf{x}_t, \mathbf{m})\, p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \tag{2.2.3}$$

where η is a constant of normalization.

In relation to the last term of the equation 2.2.3, the Theorem of Total Probability can be applied to condition the distribution of $\mathbf{x}$ into a function of $t-1$.

$$p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \int p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})\, p(\mathbf{x}_{t-1} \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) d\mathbf{x}_{t-1} \tag{2.2.4}$$

After, we apply the definition of conditional probability to the first term on the right side of 2.2.4

$$p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = p(\mathbf{x}_t \,|\, \mathbf{m}, \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})\, p(\mathbf{m} \,|\, \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})$$

obtaining

$$p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \int p(\mathbf{x}_t \,|\, \mathbf{m}, \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})\, p(\mathbf{m} \,|\, \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})\, p(\mathbf{x}_{t-1} \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) d\mathbf{x}_{t-1}$$

Now, we can replace the first term inside the integral by the motion model defined on the equation 2.2.1, since $\mathbf{x}_t$ only depends on $\mathbf{x}_{t-1}$ and $\mathbf{u}_t$. It is also possible to combine the last two terms into one. In this case, we also exclude $\mathbf{u}_t$ from the set $\mathbf{u}_{1:t}$, since the odometry at instant t does not affect the $\mathbf{x}$ or $\mathbf{m}$ at the instant $t-1$.

$$p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \int p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \mathbf{u}_t)\, p(\mathbf{x}_{t-1}, m \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) d\mathbf{x}_{t-1} \tag{2.2.5}$$

Finally, replacing the equation 2.2.5 in equation 2.2.3, we obtain a recursive formula for the posterior distribution of $\mathbf{x}_t$ and $\mathbf{m}$, which is given by the combination of the observation model with the motion model and the posterior distribution at instant $t-1$.

$$p(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta\, p(\mathbf{z}_t \,|\, \mathbf{x}_t, \mathbf{m}) \int p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \mathbf{u}_t)\, p(\mathbf{x}_{t-1}, m \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) d\mathbf{x}_{t-1} \tag{2.2.6}$$

Generally, it is not possible to compute the equation 2.2.6 in its closed form, since the SLAM problem has a high dimensional state space. This space encompasses not only the robot position but also the complete information about the map, which can contain thousands of features. Thus, approximate algorithms have been used to deal with this problem.

2.3 Main approaches to solve the SLAM problem

As stated before, SLAM methods are usually divided in two major groups: full SLAM and online SLAM approaches.

The main advantage of full SLAM approaches, such as **Graph-Based methods**, is that they are very powerful to find solutions to large-scale maps when compared to Bayesian filtering methods. However, while in filtering approaches the estimation process is incremental, in a full SLAM strategy the full set of measurements is required, at each instant, to process a solution. Therefore the process must be performed offline or over limited-sized data sets.

Full SLAM methods, such as the GraphSLAM algorithm (THRUN; MONTEMERLO, 2006), adapt the posterior distribution of SLAM into a graphical network, where the nodes are robot poses and the edges are transitions estimates between consecutive robot poses. Typically, these algorithms apply variable elimination techniques to reduce the complexity of the graph, and afterwards, refine the estimations with optimization methods, like least-squares minimization techniques. Since the processing of full slam techniques can be slow when compared to online approaches, some graph-based methods hierarchically divide the process in subregions to speed-up the performance of the optimization algorithms (NI; STEEDLY; DELLAERT, 2007) (KIM et al., 2010) (MCDONALD et al., 2011).

2.3.1 EKF-SLAM

A pioneer and one of the most popular online SLAM approach is the use of **Extended Kalman Filter (EKF)** (SMITH; SELF; CHEESEMAN, 1990). The simple Kalman filter, proposed by Rudolf Kalman in the late 1950s, is a recursive method for estimating the state of a dynamic system corrupted by noise. At each instant, a measurement of the state is performed considering the presence of Gaussian and independent error sources. The modeling of the robot pose and the sensors measurements according to the previous notation are given by the following equations (CHOSET et al., 2005)

$$p(\mathbf{x}_{t+1} \,|\, \mathbf{x}_t, \mathbf{u}_t) : \quad \mathbf{x}_{t+1} = \mathbf{F}_t\mathbf{x}_t + \mathbf{G}_t\mathbf{u}_t + \mathbf{v}_t \tag{2.3.1}$$

$$p(\mathbf{z}_t \,|\, \mathbf{x}_t) : \quad \mathbf{z}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{w}_t \tag{2.3.2}$$

where $\mathbf{F}_t$ is a matrix of size $n \times n$ encoding the dynamics of the system (n is the dimension of $\mathbf{x}_t$), $\mathbf{G}_t$ is a matrix of size $n \times m$ encoding the implication of control entries on the dynamics (m is the dimension of $\mathbf{u}_t$), and $\mathbf{H}_t$ is a matrix of size $p \times n$ encoding how state vectors are mapped into outputs (p is the dimension of $\mathbf{z}_t$). $\mathbf{v}_t$ (the process noise) and $\mathbf{w}_t$ (the observation noise) are Gaussian white noises of zero mean and covariance matrix $\mathbf{V}_t$ and $\mathbf{W}_t$.

The simple Kalman filter provides excellent results for linear systems. However, the assumptions of linear transition state and linear observations are not common in practice. The Extended Kalman Filter (EKF), in turn, disregards the need for linearity. It is assumed that the posterior probabilities of the state **x** and observations z are governed by nonlinear functions f and h

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, t) + \mathbf{v}_t \tag{2.3.3}$$

$$\mathbf{z}_t = h(\mathbf{x}_t, t) + \mathbf{w}_t \tag{2.3.4}$$

In this case, however, it is not possible to compute an exact solution to the problem. So, the idea behind EKF is to approximate the linearization of the nonlinear functions with Gaussians, for example, by using Taylor Series expansions.

On EKF-SLAM, the map of the environment consists of a set of landmarks, i.e., features corresponding to distinct objects in the physical world. While this map representation requires much lower storage space than grids, it demands a mechanism for feature extraction that can be computationally cost. Also, the knowledge about landmark correspondences is very important to the estimation process, so that, the lower the ambiguity between the landmarks, the better are the results obtained.

Furthermore, there are two main disadvantages in using Extended Kalman Filter in SLAM. First, whenever a new observation is made, the updating of all the covariance matrices requires a number of calculations that grows quadratically with the number of landmarks. Second, the sensitivity to failures in the data association. Since the EKF maintains a single association hypothesis per observation, if a failure occurs in the data association, the effect of the incorporation of an erroneous observation cannot be removed. In fact, if many failures occur, leading to many wrong associations between observations and landmarks, the algorithm will diverge (MONTEMERLO; THRUN, 2007).

A widely studied approach for reducing the complexity of the update step is to partition the map update. Some examples of this approach are the Compressed EKF (CEKF) (GUIVANT; NEBOT, 2001), which operates locally in regions of the map, maintaining the global coordinates; the Postponed EKF (KNIGHT; DAVISON; REID, 2001), which delays the updating of the full map by maintaining a separated set of data that can be used to generate updates in places not yet observed; among others (TARDOS et al., 2002; FRESE, 2006; PAZ; TARDOS; NEIRA, 2008).

Problems associated to the approximation of non-linear functions are the focus of the Unscented Kalman Filter method (UKF) (JULIER; UHLMANN, 1997). When the prediction and update functions are highly non-linear, the first-order approximations made by EKF introduce large errors that can lead to divergence of the filter. So, UKF uses a sampling technique known as Unscented Transformation to accurately process the estimations of means and covariances.

In addition to these problems, many other issues have been addressed in the literature, such as the improvement of the EKF consistency (BAILEY et al., 2006; HUANG; MOURIKIS; ROUMELIOTIS, 2008, 2010), or the landmarks extraction (NIETO; BAILEY; NEBOT, 2006; KANG et al., 2010; LEE; SONG, 2010).

2.3.2 Particle filters

More recently, a new form of approach that become very popular was the use of **Rao-Blackwellized Particle filters (RBPF)** (MURPHY, 1999). RBPF methods are based on the idea that the correlations between landmarks arise from the uncertainty in the robot pose, so that, if the robot path is known, the estimations of the landmarks positions can be made independently (MONTEMERLO; THRUN, 2007). While EKF-based approaches use a parameterized model of Gaussians to represent probability distributions, RBPF algorithms use a set of particles (samples) distributed in the environment to estimate the posterior distribution over the robot path. Then, for each particle, a map is estimated as a function of the path defined by the particle.

$$p(\mathbf{x}_{1:t}, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{x}_{1:t} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})\, p(\mathbf{m} \,|\, \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{2.3.5}$$

Since the map estimation considers that the robot path is known, the probability distribution of each landmark can be computed separately, because in this case observations of a landmark do not affect the estimation of others landmarks.

$$p(\mathbf{x}_{1:t}, m \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{x}_{1:t} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_i p(\mathbf{m}_i \,|\, \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{2.3.6}$$

A major problem related to RBPF SLAM approaches is the problem of particle depletion. Due to the limited number of particles, the resampling step of the filter is essential to the functioning of the filter, yet it eventually reduces the diversity of solutions. Over time, the particles tend to share the same ancestors, which can be a big problem in the long term, if a good local solution become a bad globally solution in the future (STACHNISS, 2009).

Many works focus on ways to reduce this problem, as is the case of the second version of Fast-SLAM (MONTEMERLO et al., 2003), whose idea is to build the proposal distribution based not only on the control information, but also on the current observations.

An important advantage of RBPF methods is that they can be based both in features and in occupancy grids. In this case, instead of estimating the position of landmarks, it is estimated the occupancy of cells of the environment, whose positions are fixed and known. The maintenance of occupancy grids is very simple, however the storage cost of occupancy grids is considerably larger than when using landmarks. Nevertheless, there is no need for specific mechanisms of extraction of features, enabling the treatment of completely arbitrary maps.

Two of the earliest RBPF methods based on occupancy grids are the GridSLAM (HAHNEL et al., 2003) and the DP-SLAM (ELIAZAR; PARR, 2003). The GridSLAM efficiently combines the RBPF SLAM algorithm with laser scan-matching techniques. The principle behind the algorithm is to transform sequences of range measurements in odometry measures that will supply the particle filter. Proposed simultaneously with the GridSLAM, the DP-SLAM algorithm also applies a particle filter over a occupancy grid, however, its differential is the optimization of the environment representation. The basic principle of the DP-SLAM consists in storing in a single occupancy grid, shared by all particles, the differences observed in the environment by particles of a same lineage. The method also maintains an ancestry tree with the lineage of the particles, to be able to reconstruct individual maps.

Besides these quintessential RBPF solutions, numerous improvements have been proposed, such as the selective resampling technique (GRISETTI; STACHNISS; BURGARD, 2005), methods for detection of cycles in the environment (STACHNISS; HAHNEL; BURGARD, 2004), techniques of recovery from particle depletion (STACHNISS; GRISETTI; BURGARD, 2005), among others.

In the next section, we will detail the Rao-Blackwellized particle filter approach, which is the approach followed by the proposed work. We start with the definition of the basic RBPF algorithm and then we present some of the fundamental methods, such as the Fast-SLAM algorithm (MONTEMERLO et al., 2002, 2003) and the GridSLAM (HAHNEL et al., 2003). We will also detail the DP-SLAM method (ELIAZAR; PARR, 2003), that served as the base for our method.

2.4 Rao-Blackwellized Particle Filters

While approaches based on Extended Kalman Filter represent probability distributions through a parameterized model of Gaussians, particle filters represent the distributions through a finite set of states samples. In a particle filter, the particle density of a region corresponds to the probability of the robot being in that region. Using a sufficient number of particles it is possible to approximate complex multi-modal distributions, in other words, we can implement the Bayesian filtering through simple sampling procedures.

The Rao-Blackwellized particle filter (RBPF) was proposed by Murphy (MURPHY, 1999) as an algorithm for Dynamic Bayesian Networks (DBN), based on the factorization of variables of a probability distribution, a technique known as Rao-Blackwellization. The principle behind a RBPF is to perform a sampling process on some of the system variables and estimate the remaining regarding these samples. Originally the filter was applied to model non-stationary regressions in neural networks and, in robotics, to build maps using discrete values (DOUCET et al., 2000).

The FastSLAM algorithm proposed the use of the Rao-Blackwellized particle filter to solve the problem of SLAM (MONTEMERLO et al., 2002). The method applies the Rao-Blackwellized factorization to separate the trajectory estimation from the mapping estimation. The idea behind this approach is to apply a particle filter to estimate the trajectory of the robot and use the obtained paths to compute the map.

Fig. 2.3 shows an example of the behavior of a RBPF SLAM approach, depicting possible trajectories (blue) of a robot (red) in a cyclic environment. In (a), the trajectories encoded by the particles are a bit close since the uncertainty about the robot pose is still low. As the robot moves (b)-(e), this uncertainty increases and consequently the trajectories represented by the particles are more distant from each other. When the robot reaches a region already visited (f), the uncertainty about its pose diminishes, since it is possible to compare what the robot should expect to see at that position with what it is really observing.

A major problem of Rao-Blackwellized particle filters is the complexity associated to the number of particles required for built precise maps. Very high amounts of particles makes the method impractical. However, the reduction of the number of particles entails problems such as the particle depletion.

2.4.1 FastSLAM: the RBPF strategy for SLAM

The principle behind the use of Rao-Blackwellized particle filters in SLAM is the factorization of the posterior probability distribution $p(\mathbf{x}_{1:t}, m \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$, taking advantage of the conditional independence among the landmarks of the environment. If the path of the robot is known, an observation of a landmark does not influence on the position of

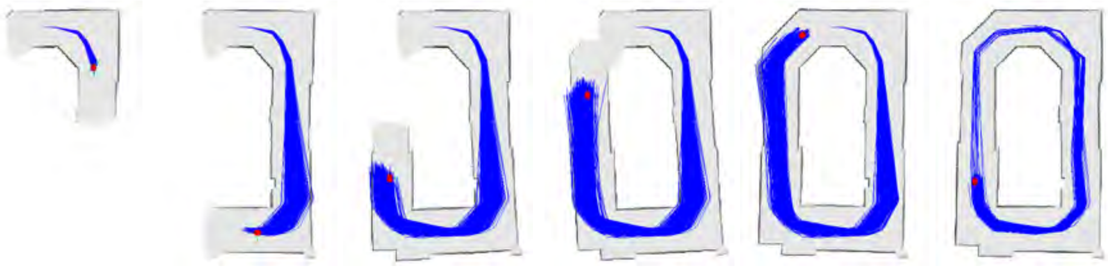


Figure 2.3: Example of the particles dispersion of a RBPF-based SLAM. In (a)-(f), the robot is shown in red, and bluelines are the particles paths.

another landmark, in other words, given a specific path, the estimation of each landmark is independent. The factorization can be written as:

$$p(\mathbf{x}_{1:t}, m \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{x}_{1:t} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})\, p(\mathbf{m} \,|\, \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{2.4.1}$$

$$p(\mathbf{x}_{1:t}, m \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{x}_{1:t} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_i p(\mathbf{m}_i \,|\, \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{2.4.2}$$

This factorization, called Rao-Blackwellization, ensures that the posterior distribution of SLAM can be efficiently obtained by multiplying the posterior probability of the robot path by the posterior of the landmarks, given the knowledge of the path.

However, as in EKF-based SLAM approaches, the FastSLAM method suffers from the problem of data association regarding landmarks observations. The solution adopted by FastSLAM is to use a maximum likelihood estimator to determine which landmarks are observed at every moment. As this problem can be studied aside of the particle filter, this section will consider only the case that the correspondences are known.

The pioneering RBPF algorithm for SLAM is the FastSLAM (MONTEMERLO et al., 2002). The method uses a SIR particle filter (Sampling Importance Resampling) to determine the robot pose, similarly to Monte Carlo localization (MCL) (DELLAERT et al., 1999), but keeping a map for each individual sample. Then, for every particle map, the position of each landmark is estimated using an EKF. Since each EKF has only one landmark, the associated cost is small and fixed in size.

The FastSLAM algorithm (Algorithm 2.1) is composed by four steps, that merges a SIR (Sampling Importance Resampling) particle filter with a map estimation step using an EKF for each landmark. Next, each of the four steps of the algorithm will be explained in details.

Sampling

The sampling step must generate a new set of particles S_t from the previous generation of particles S_{t-1}, based on the observations $\mathbf{z}_t$ and the odometry $\mathbf{u}_t$. Yet, we cannot sample from the exact posterior distribution, since this distribution is the goal of SLAM. So, we use a simpler distribution to generate new particles, the so-called *proposal distribution*, considering only the odometry $\mathbf{u}_t$. Later, these particles will be evaluated by comparisons between the proposal and the posterior distributions, considering then the observations $\mathbf{z}_t$.

Assuming that the particles S_{t-1} are distributed according to $p(\mathbf{x}_{1:t-1} \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})$, we can define that the new set of particles S_t, generated using the probabilistic motion

Algorithm 2.1: Rao-Blackwellized Particle Filter algorithm

```
1 Sampling - The particles representing hypothesis over the robot pose are
  propagated according to a motion model, indicated by a proposed distribution $\pi$.
2 Weighting - An individual weight is assigned to each particle, by comparing the
  current sensors measurements with the momentaneous map of each particle.
3 Resampling - Low weighted particles are discarded, while particles with high
  weights are replicated. This is necessary to increase the diversity of good particles
  in the filter, however, in long term, it can lead to the particle depletion problem.
4 Map Estimation - Update of the particles maps according the observation model
  using EKFs.
```

model $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})$, follow this proposal distribution

$$p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \tag{2.4.3}$$

The motion model used by the algorithm assumes that the errors associated with components v_t (translational) and w_t (rotational) of the control $\mathbf{u}_t$ are given by Gaussians. This does not imply that the motion model itself is a Gaussian, but instead, a non-linear function of controls and control noise.

Figure 2.4, extracted from (MONTEMERLO; THRUN, 2007), shows an example of sampling of 250 particles, by applying the motion model on a curved trajectory. In this case, it was considered a small translational error and a large rotational error.

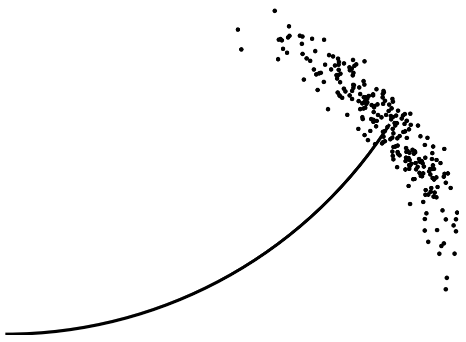


Figure 2.4: Example of sampling based on the probabilistic motion model. Figure extracted from (MONTEMERLO; THRUN, 2007).

Weighting

The way to correct the difference between the proposal distribution $p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})$ and the target posterior distribution $p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ is through the resampling of weighted particles. Therefore, it is necessary to compute the weight of each particle by evaluating the quality of their maps.

The importance weight of each particle $[m]$ in FastSLAM is obtained by the ratio between the posterior and the proposal distribution

$$w_t^{[m]} = \frac{\text{posterior distribution}}{\text{proposal distribution}} = \frac{p(\mathbf{x}_{1:t}^{[m]} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})}{p(\mathbf{x}_{1:t}^{[m]} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})} \tag{2.4.4}$$

We can separate $\mathbf{z}_t$ from $\mathbf{z}_{1:t}$, and rewrite the equation applying the Bayes Rule

$$w_t^{[m]} = \frac{p(\mathbf{x}_{1:t}^{[m]} \mid \mathbf{z}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}{p(\mathbf{x}_{1:t}^{[m]} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})} = \frac{p(\mathbf{x}_{1:t}^{[m]}, \mathbf{z}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}{p(\mathbf{z}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_{1:t}^{[m]}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}$$

$$w_t^{[m]} = \frac{p(\mathbf{z}_t \mid \mathbf{x}_{1:t}^{[m]}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}{p(\mathbf{z}_t \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}$$

The denominator is a constant, and it can be eliminated since the particle weights are normalized before the resampling step.

$$w_t^{[m]} = p(\mathbf{z}_t \mid \mathbf{x}_{1:t}^{[m]}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \tag{2.4.5}$$

The weights are computed according to an innovation degree, i.e., the difference between the real observation $\mathbf{z}_t$ and the predicted observation $\hat{z}_t$. In FastSLAM, the landmarks estimations are made with EKF, and the sequence of innovations in the EKF is a Gaussian with zero mean and covariance matrix $\mathbf{z}_t$. Thus, the importance weight is equal to the probability of the innovation $\mathbf{z}_t - \hat{z}_t$ being generated by the Gaussian of the EKF. Its calculation spends a constant time to be made, since only depends on the size of the observation, which is fixed.

$$w_t^{[m]} = \frac{1}{\sqrt{|2\pi \mathbf{z}_t|}} \exp\{-\frac{1}{2}(\mathbf{z}_t - \hat{z}_t)^T [\mathbf{z}_t]^{-1} (\mathbf{z}_t - \hat{z}_t)\} \tag{2.4.6}$$

Resampling

The resampling step is required in a Rao-Blackwellized particle filter to approximate the particles distribution to the SLAM posterior, before the next sampling step is performed. After the particles weights are calculated, a method is applied to randomly select particles with replacement from the current set. A simple technique that can be chosen is the Roulette algorithm. This technique, commonly used in genetic algorithms, sets the probability of selection of each particle in proportion to its weight. In general, particles with higher weights will be more likely to be replicated than those with lower weights, yet there is a chance that some particles with low weights may survive, to maintain the diversity of the filter.

Figure 2.5 illustrates the necessity of the resampling step in a RBPF. In (a), the target distribution is depicted in green, while the proposal distribution, represented by a Gaussian, is depicted in red. Since the sampling step, in (b), is performed according to the proposal distribution, it is necessary to weight the samples (black lines) to approximate them to the target distribution. The weighting is performed, in (c), and samples of regions in which the target distribution is larger than the proposal distribution receive higher weights (lines with larger length) than regions where the proposal distribution is overestimated. In the resampling step, (d), the probability of selecting particles from a "region" of the distribution is proportional to the summation of the particles weights of this "region" (presented in a reduced scale). The relation between the number of samples from a region and the weight of those samples is a good approximation to the behavior of the target distribution, as shown in (e). In fact, insofar as the number of particles grows, the distribution will properly represent the SLAM posterior.

Map Estimation

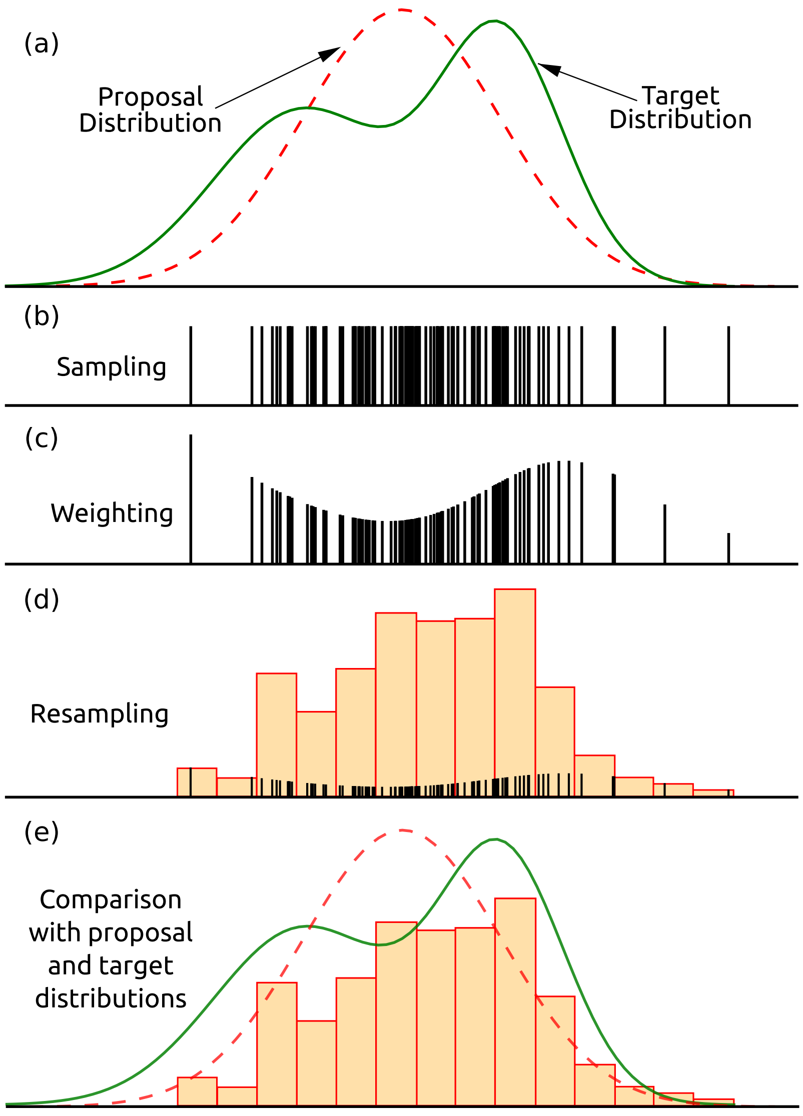


Figure 2.5: Example of the sampling, importance weighting and resampling in a Rao-Blackwellized particle filter. Figure adapted from (MONTEMERLO; THRUN, 2007).

As said before, the estimates of the landmarks positions $p(\mathbf{m} \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ are obtained by EKFs.

For each of the particles, a EKF is required to estimate each landmark, totaling $M \times N$ EKFs, where M is the amount of landmarks in the environment and N is the number of particles. However, the cost of each EKF is fixed, since only one landmark is treated by filter. Thus, the execution time of the map estimation step is also constant.

2.4.2 FastSLAM 2.0: Improvements on the proposal distribution

The resampling step of RBPF SLAM is essential to the functioning of the method, specially for closing loops in the environment. When a robot explores an unknown region, all particles have similar weights, since the last action taken by the robot is consistent with the map. However, when a robot returns to a previously visited area, only the maps of some particles will be consistent. Thus, several particles can be discarded and the uncertainty of the process decreases.

Nevertheless, the resampling contributes to a major problem associated with particle filters, which is the impoverishment of solutions over time, also known as particle depletion. Resampling particles eventually reduces the diversity of hypotheses by replicating the best ones at the expense of the momentarily worst ones. As the filter progresses, the particles tend to share the same ancestors, which is an enormous problem in long term.

In FastSLAM, the particles diversity is linked to the relation between the accuracy of the proposal distribution and the elimination of bad particles. The problem arises when there is a large disparity between the odometry errors and the observation errors, and consequently between the proposal and the posterior distribution. It is possible to notice this situation in Figure 2.6. Since the uncertainty embedded in the motion model is large, due to the errors in the robot odometry, the particles (black points) end up spreading too much. However, the sensors of the robot, such as sonars and, specially, lasers, are much more accurate than the odometry. So, eventually, the weighting step will give high weights only to a small group of particles (points inside the ellipse) and neglect the others. If this occurs very often, then the filter will probably diverge.

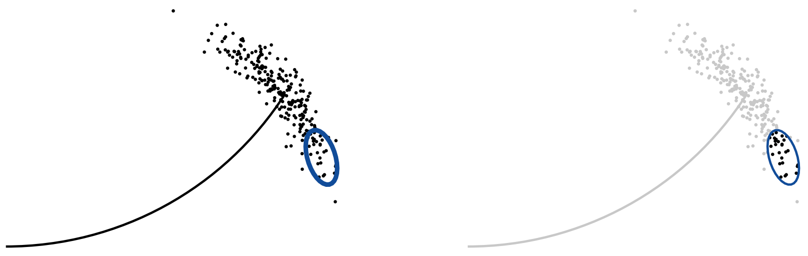


Figure 2.6: The large difference between the proposal distribution and the posterior distribution, causes the elimination of many hypotheses at each iteration, which is the main factor for the particle depletion problem. Figure extracted from (MONTEMERLO; THRUN, 2007).

A modification of FastSLAM, called FastSLAM 2.0 (MONTEMERLO et al., 2003), was presented to improve the proposal distribution and reduce the incidence of this problem. The idea of the algorithm is to construct the proposal distribution based on not only the odometry information $\mathbf{u}_t$, but also on the observations of the sensors $\mathbf{z}_t$.

If the motion model of the robot is not linear, the observations cannot be incorporated into the proposal distribution in closed form. The solution is to linearize the motion model by applying an EKF. The sampling of particles in FastSLAM 2.0 takes the three following steps (MONTEMERLO; THRUN, 2007):

1. Project the previous robot pose $\mathbf{x}_{t-1}^{[m]}$ according to a linear motion model. The estimated pose $\hat{x}_t$ will have uncertainty in the form of a Gaussian.

2. This Gaussian will correspond to the prior distribution of an EKF that estimates the robot pose. The observation $\mathbf{z}_t$ will be incorporated to the EKF.

3. Generate a sample $\mathbf{x}_t^{[m]}$ from the resulting posterior distribution.

The step of importance weighting is also changed, since the proposal distribution has been modified to $p(\mathbf{x}_{1:t-1}^{[m]} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})p(\mathbf{x}_t^{[m]} \mid \mathbf{x}_{1:t-1}^{[m]}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$.

$$w_t^{[m]} = \frac{\text{posterior distribution}}{\text{proposal distribution}} = \frac{p(\mathbf{x}_{1:t}^{[m]} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})}{p(\mathbf{x}_{1:t-1}^{[m]} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})p(\mathbf{x}_t^{[m]} \mid \mathbf{x}_{1:t-1}^{[m]}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})} \tag{2.4.7}$$

We separate $\mathbf{x}_t$ from $\mathbf{x}_{1:t}$ in the numerator, and apply the definition of conditional probability,

$$w_t^{[m]} = \frac{p(\mathbf{x}_t^{[m]}, \mathbf{x}_{1:t-1}^{[m]} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})}{p(\mathbf{x}_{1:t-1}^{[m]} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) p(\mathbf{x}_t^{[m]} \mid \mathbf{x}_{1:t-1}^{[m]}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})}$$

$$w_t^{[m]} = \frac{p(\mathbf{x}_t^{[m]} \mid \mathbf{x}_{1:t-1}^{[m]}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) p(\mathbf{x}_{1:t-1}^{[m]} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})}{p(\mathbf{x}_{1:t-1}^{[m]} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) p(\mathbf{x}_t^{[m]} \mid \mathbf{x}_{1:t-1}^{[m]}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})} = \frac{p(\mathbf{x}_{1:t-1}^{[m]} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})}{p(\mathbf{x}_{1:t-1}^{[m]} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})}$$

As in the derivation of FastSLAM, presented in the previous section, we now separate $\mathbf{z}_t$ from $\mathbf{z}_{1:t}$, and rewrite the equation applying the Bayes Rule

$$w_t^{[m]} = \frac{p(\mathbf{x}_{1:t-1}^{[m]} \mid \mathbf{z}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}{p(\mathbf{x}_{1:t-1}^{[m]} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})} = \frac{p(\mathbf{x}_{1:t-1}^{[m]}, \mathbf{z}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})}{p(\mathbf{z}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_{1:t-1}^{[m]}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})}$$

$$w_t^{[m]} = \frac{p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}^{[m]}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}{p(\mathbf{z}_t \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}$$

The denominator is constant and it can be dropped, since the weights are normalized in the filter.

$$w_t^{[m]} = p(\mathbf{z}_t \mid \mathbf{x}_{1:t}^{[m]}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \tag{2.4.8}$$

Although quite similar to the original FastSLAM, the equation for calculating the weight of the particles differs, as well as the choice of the proposal distribution. Yet, the remainder of the algorithm, such as the resampling step and the map estimation, is the same.

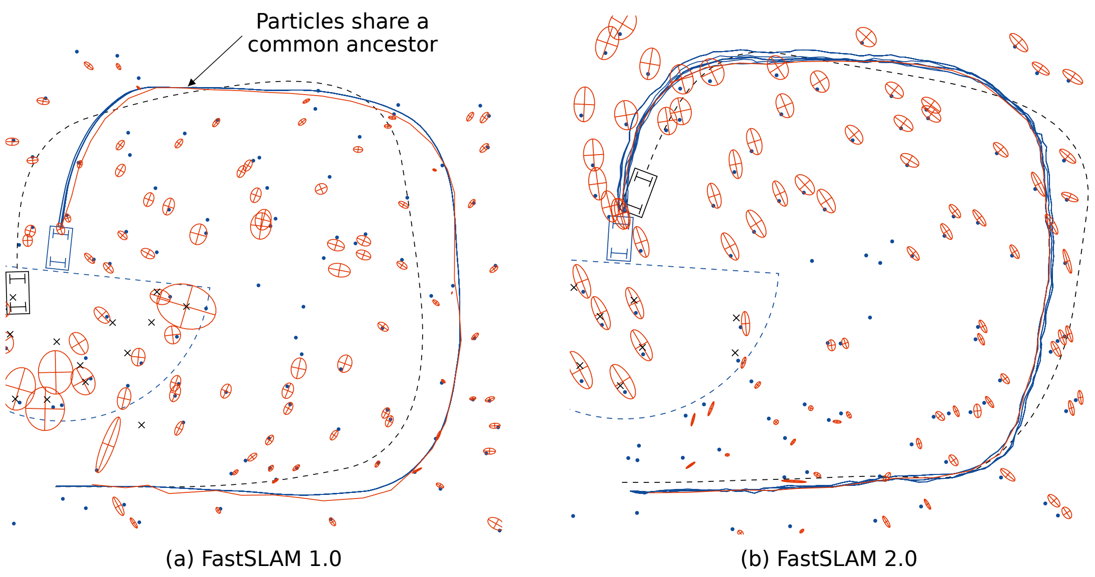


Figure 2.7: Comparison of the diversity of particles trajectories between the FastSLAM 1.0 and FastSLAM 2.0. Figure extracted from (MONTEMERLO; THRUN, 2007).

An important advantage of this version of FastSLAM is that, while in the standard particle filter, the resampling is the only step that considers the observations, in FastSLAM 2.0 this is also done in the sampling process. Therefore the probability of divergence is reduced. Figure 2.7 shows a comparison between the methods FastSLAM 1.0 and

FastSLAM 2.0. In (a), the particles of FastSLAM 1.0 share a common ancestor at some point not far from the robot. In this case, all diversity prior to this point has been lost, and the solution will diverge from the real path. On the other hand, with FastSLAM 2.0, in (b), there is still some diversity of solutions even in the beginning of the path.

2.4.3 GridSLAM: RBPF SLAM using occupancy grids

Some of the main difficulties of the FastSLAM method, as well as methods based on Kalman filters, are the problems of data association and features extraction related to the use of landmarks. Two methods, the GridSLAM (HAHNEL et al., 2003) and DP-SLAM (ELIAZAR; PARR, 2003), adapt the RBPF algorithm for environments based on occupancy grids. The great advantage of these algorithms is that arbitrary types of environments can be modeled with high accuracy by volumetric dense maps. This section covers the method GridSLAM, while the next will describe DP-SLAM.

The GridSLAM method is an efficient algorithm for the problem of SLAM that combines a Rao-Blackwellized particle filter with techniques of scan-matching using laser sensors. The principle behind the algorithm is to transform sequences of laser readings into information about the robot odometry that will supply the particle filter. Since this estimation is an order of magnitude more accurate than the encodings of the robot wheels movements, the proposal distribution is more precise than the simple RBPF SLAM.

The scan matching technique used by the algorithm performs the alignment between different laser readings, through the construction of a occupancy grid. To avoid the time consuming ray-tracing operation, required to the obtention of the likelihood of a measurement $p(z|x)$, the method applies an approximation that considers only the endpoint of a beam, so that $p(z|x)$ can be computed efficiently.

Figure 2.8 shows the difference between a map purely obtained with the odometer readings, where the accumulated error eventually generates large distortions in the environment, and a map obtained with the GridSLAM, both using 500 particles.

The results presented by GridSLAM show the flexibility of using Rao-Blackwellized particle filter with occupancy grids. However, even with the improved proposal distribution, the particle depletion problem, although attenuated, persists. Thus, closing loops in large environments still stands as a very difficult problem in SLAM.

2.4.4 DP-SLAM: Improvements on the data management

Another RBPF method based on occupancy grids, presented in the same year that GridSLAM was proposed, is the DP-SLAM algorithm (ELIAZAR; PARR, 2003). DP-SLAM is a RBPF-based SLAM that uses an optimized structure to store the maps of the particles. In the traditional RBPF-based SLAM, the particles resampling process requires the copy of multiple instances of the map. This process has complexity $O(MP)$, where M is the map size and P is the number of particles. Considering that the robot scans an area of size A, where $A \ll M$, the variation of the map of each particle, at two consecutive instants, occurs at most in A cells. DP-SLAM takes advantage of this observation to introduce an improved map representation. Basically, it merges all particles maps into only one map containing the differences observed by each particle, through a process called DP-Mapping.

DP-Mapping uses two efficient data structures: an ancestry tree and a modified grid map. The ancestry tree is the family tree for all active particles of the filter. The leaves of the ancestry tree represent the active particles, while the other internal nodes are the ancestrals of these particles, i.e., the particles from which they have derived. The second

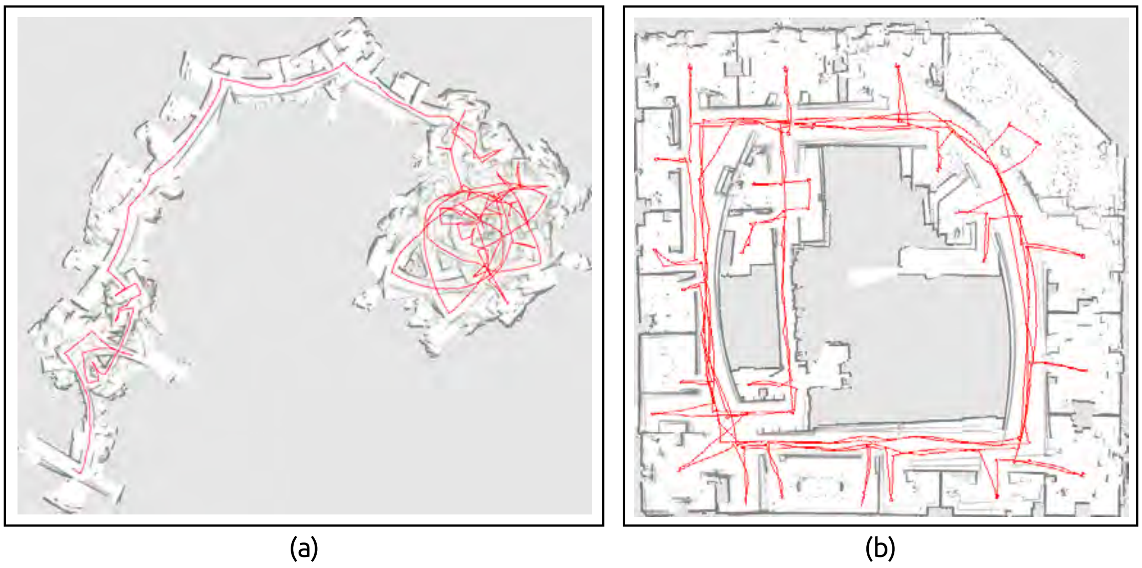


Figure 2.8: Comparison between a map generated using only raw odometry data (left) and a map generated by the GridSLAM method using scan-matching with occupancy grids (right). Figure extracted from (HAHNEL et al., 2003).

structure, called distributed particle map (DP-Map), is a grid that lists all observations for every cell (using trees to reduce the management costs). If a particle gets an observation for a cell that is different from those made by its ancestrals, then the tree of the cell is updated, adding the information and observation of that particle. Therefore, to obtain the full map associated to a particle it is necessary to consult the observations made by the particle and by its ancestrals.

To ensure that the ancestry tree does not grow indefinitely, a pruning process is performed. When a particle does not generate a child, it is removed from the tree since its information will not be inherited. In addition, a particle that generates a single child has its information updated with the information of its child, and then both are merged to prevent the creation of branches without ramifications. The number of internal nodes and the depth D of the ancestry tree are limited by P. In practice, $D = O(\log P)$, due to the particle resampling that implicates in almost balanced ancestry trees (ELIAZAR; PARR, 2003).

The DP-SLAM algorithm is described in Algorithm 2.2 (the computational cost of each step is indicated on the right) (ELIAZAR; PARR, 2004).

The map representation on DP-SLAM is an occupancy grid, updated through a probabilistic model of laser scan. It defines $P_c(g, \rho) = 1 - e^{-g/\rho}$ as the cumulative probability that a laser cast will have been interrupted after traveling through g grid squares with opacity ρ. Thus, the cumulative probability that the laser cast is interrupted by squares up to n is

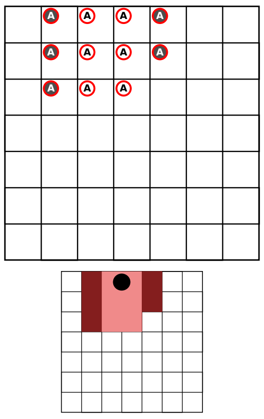
$$P_c(g^{1:n}, \rho^{1:n}) = \sum_{i=1}^{n} P_c(g_i, \rho_i) \prod_{j=1}^{i-1} (1 - P_c(g_j, \rho_j))$$

The map updating is simple, because the mean of the exponential distribution with opacity ρ is simply ρ. In practice, for each square, the method maintains the sum of the total distance, d_τ, travelled by laser scans through the square and the number of times, h, that the laser is presumed to have stopped in the square. The estimate of ρ is $\rho = d_\tau / h$.

Algorithm 2.2: DP-SLAM algorithm

```
1 Sampling of particles according to the proposal distribution - $O(P)$.
2 Insertion of the new particles into the ancestry tree - $O(P)$.
3 Computation of the new particle weights - $O(ADP \log P)$. For each particle
  ($P$), sweep the area reached by the laser sensor ($A$) and verify the cell occupation
  $O(D \log P)$.
4 Normalization of the particle weights - $O(P)$.
5 Update of each particles maps - $O(AP \log P)$. For each particle ($O(P)$), sweep
  the area reached by the laser sensor ($O(A)$) and update the cell occupation
  ($O(\log P)$).
6 Particles resampling according to the computed weights - $O(P)$.
7 Prune of the ancestry tree - $O(AP \log P)$. First, remove the particles that do not
  generated children $O(P)$. Then, merge "only children" with their parents
  $O(AP \log P)$: for each "only child" particle ($O(P)$), removes the child from the
  observations trees, and insert child's observations in parent particle ($O(A \log P)$).
```

Now, we present a detailed example of the DP-SLAM method using three particles, to improve the understanding of the DP-Mapping strategy. In the start of the SLAM process, represented in Figure 2.9, the initial position of the robot is the same for all particles, and equal to (0,0). Therefore, only one particle is needed to represent the estimate about the robot pose. This particle, referred as 'A', is the only particle in the ancestry tree. In the top, we show the distributed particle map, that contains the information about obstacles (dark circles) and free space (light circles). In the bottom, we show the momentary grid map of the particle, where the robot is the black circle.



(a) particle A

Figure 2.9: Example of the DP-Mapping process - step 1: At the start, all particles are equal, so only one is maintained by the filter. Distributed particle map (top), map of the particle (bottom).

From the second generation of particles onwards, it is possible to perceive the advantages of DP-Mapping. In Figure 2.10, the particle 'A' generated three children - 'B', 'C' and 'D' - whose robot paths and observations differ slightly. In (a), we can see the ancestry tree and the distributed particle map updated with the three sets of observations. In (b), (c) and (d) we show the maps of the three active particles. To build each individual map, the ancestry tree is consulted to define which sets of observations should be considered. In this case, the maps of 'B', 'C', and 'D' are respectively composed by the observations of A-B, A-C and A-D.

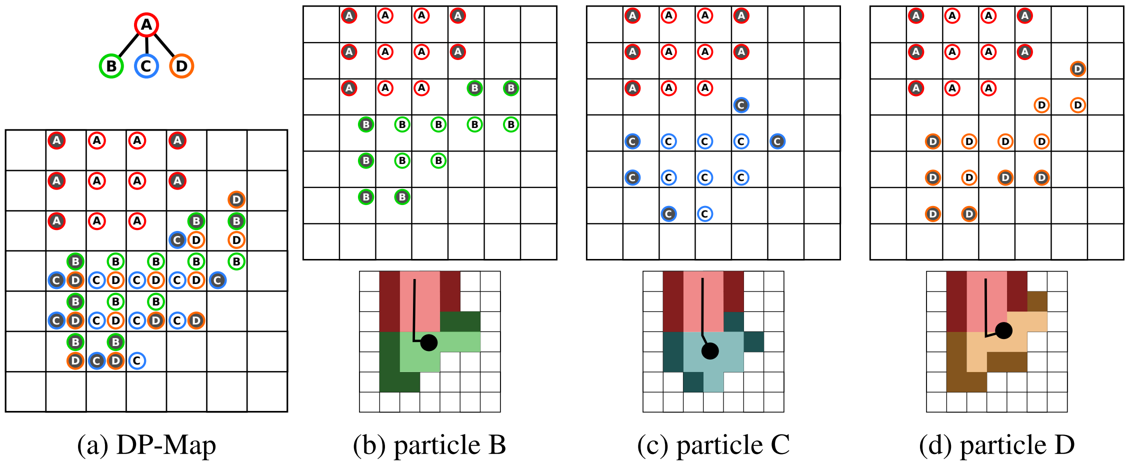


Figure 2.10: Example of the DP-Mapping process - step 2: The observations of each active particle are inserted into the DP-Map. To build the map of a particle, sets of observations must be combined.

With the processing of the particle filter, particles with better maps will have more chance of being replicated, in the resampling, than the others particles. In this example, as we see in Figure 2.11, the particle 'B' generated two children ('E' and 'F'), particle 'C' generated one ('G'), and the particle 'D' do not generated children. Now, the maps of the active particles 'E', 'F', and 'G' are the combinations of the observations A-B-E, A-B-F and A-C-G.

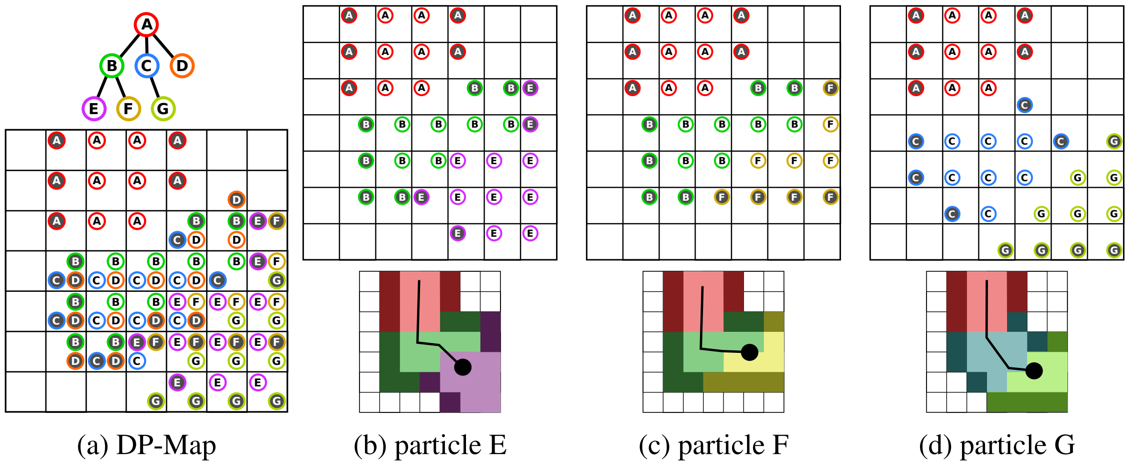
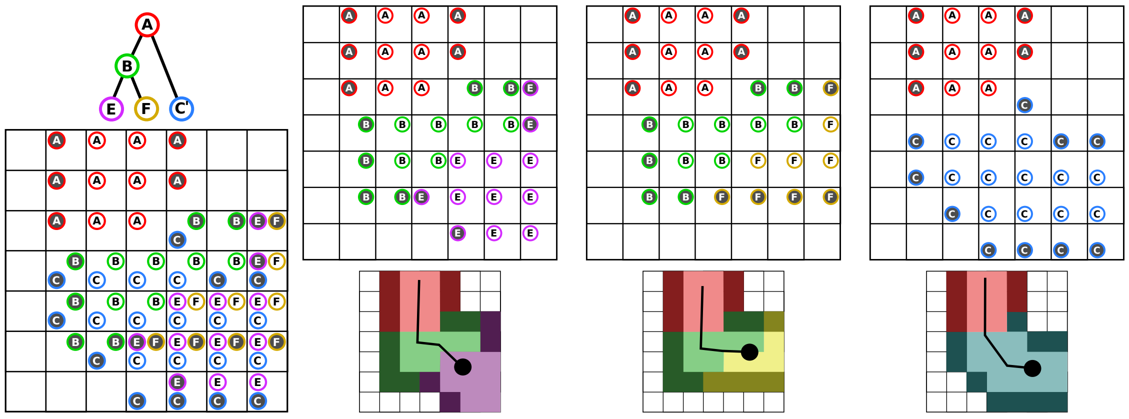


Figure 2.11: Example of the DP-Mapping process - step 3: Over time, some particles generate more than one children, while others remain childless. This situation highlights the importance of the DP-Map to avoid the data replication.

When the lineage of a particle is interrupted after the resampling step, the information regarding that particle, in both DP-Map and ancestry tree, becomes useless. As we see in Figure 2.12, all the information about the particle 'D' is removed from the filter to free up space. Also, since the particle 'G' is the only child of 'C' (as shown in Figure 2.11), the information regarding 'G' can be merged into 'C', because, in the future, all particles will always consult the information of C and G, and not separately. So, after the prune and merge processes, the active particles are 'E', 'F' and 'C'.



(a) DP-Map (b) particle E (c) particle F (d) particle C

Figure 2.12: Example of the DP-Mapping process - step 4: A prune and merge process is applied to the filter to remove unnecessary informations.

After knowing the advantages of DP-SLAM, we need to analyze the cost associated with the method. The cost of the original version of the DP-SLAM algorithm is $O(ADP \log P)$. In large environments, where $A \ll M$, this method can be preferable than a traditional method with cost $O(MP)$. Eliazar and Parr developed two improved versions of DP-SLAM (ELIAZAR; PARR, 2004) (ELIAZAR; PARR, 2006), that introduced, respectively, the addition of a batch processing technique and the creation of caches to reduce the access cost of the particles maps. These modifications result in an algorithm with computational cost of $O(AP)$. Despite all the space optimization, DP-SLAM still requires a very large number of particles to obtain good results, therefore we decided to search for alternative ways to improve the method, taking advantage of its characteristics.

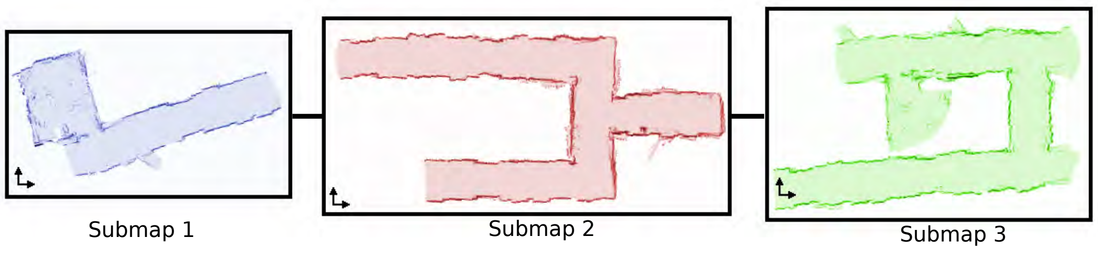
2.5 Submap-based SLAM

Several works have introduced modifications on the SLAM strategy based on particle filters, to improve the proposal distribution (GRISETTI; STACHNISS; BURGARD, 2005) or the the resampling step. We can point out the GRR algorithm (Geometric relation resampling) (KIM et al., 2009), that uses a KD-Tree to extract relations between particles and improve the resampling, and the Selective Resampling technique (GRISETTI; STACHNISS; BURGARD, 2005), that uses a particle dispersion measure to decide when the resampling is required.

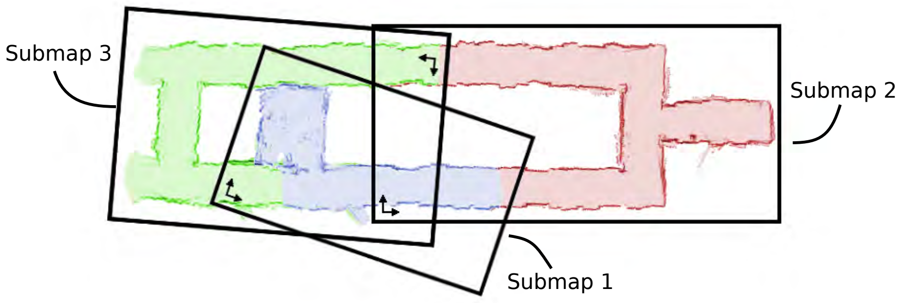
Other modifications focus on the loop closure problem, which is of great importance and difficulty in RBPF-based SLAM strategies. One of the most influential approaches introduces a simple method for detecting loops in the environment using topological maps proposed by Stachniss et al.(STACHNISS; HAHNEL; BURGARD, 2004). Another method for detecting loops, shown by Granstrom (GRANSTROM et al., 2009), uses machine learning to match patterns of sensors readings. Tungady and Kleeman (TUNGADI; KLEEMAN, 2009) presented a method based on Voronoi diagrams for closing loops.

As mentioned in Section 2.4, one of the main difficulties of RBPF approaches is the particle depletion problem, that causes a lack of variability of particles to close loops in the environment. To minimize this effect, a solution is to divide the environment map in several submaps, which are locally consistent, and connect them using map matching techniques. Methods that use this approach are commonly called **submap-based strategies**.

A prominent submap-based approach is the use of hybrid maps, that combine topological and metric maps (BOSSE et al., 2003) (LEE; LEE; OH, 2011) (ESTRADA; NEIRA; TARDOS, 2005) (BLANCO; FERNANDEZ-MADRIGAL; GONZALEZ, 2008). Each node of the topological map corresponds to a metric submap, while the edges of the topological map correspond to connections between these submaps. Figure 2.13 shows an example of a global map reconstruction from multiple local maps. The topological map, shown in (a), is a graph with three nodes representing three different submaps. It is important to note that, usually, each local map is described in a different coordinate system. In (b), the matching between the submaps is performed by transforming the coordinate systems of each submap. The resulting global map is presented in (c).



(a) Topological map with three nodes associated to submaps of the environment.



(b) Submaps matching through transformations in the coordinate systems.

(c) Resulting global map.

Figure 2.13: Example of a global map reconstruction from multiples submaps described in different coordinated systems.

One of the earliest works with hybrid maps was the Atlas framework (BOSSE et al., 2003). In Atlas, the uncertainties of each submap are modelled according to its own coordinate system. Connections between submaps are detected by a matching process, and refined always that the submaps uncertainties decrease. Global maps are reconstructed through an off-line alignment process based on least squares optimization.

The Hierarchical SLAM (ESTRADA; NEIRA; TARDOS, 2005), although very similar to the Atlas method, introduces a loop closing method that imposes consistency at the global level, and therefore increases the precision of the resulting map. When a loop is detected in the topological map, the algorithm constraints the composition of coordinate system transformations to zero considering each of the submaps along the loop. Then, a non-linear least-squares optimization technique is applied to estimate the correct alignment of the submaps.

While in the Hierarchical SLAM only a single topological hypothesis is supported, Ranganathan et al. (RANGANATHAN; MENEGATTI; DELLAERT, 2006) propose a method called Probabilistic Topological Maps (PTM). This method performs a Bayesian inference through a sample-based representation, that approximates the posterior distribution over the possible topologies with multiples hypotheses.

Blanco et al. (BLANCO; FERNANDEZ-MADRIGAL; GONZALEZ, 2008) propose the Hybrid Metric-Topological SLAM (HMT-SLAM), which performs, with a single particle filter, an unified estimation of the robot path in both metrical and topological map. The HMT-SLAM introduces a posterior distribution over both the metric and the topological part of the robot path. Therefore the particles propagation is based not only on the motion and observation models of the robot (like traditional RBPF strategies), but also on a transition model associated to the topological map.

There are some methods that do not specifically use hybrid maps, but also divide the process in small regions. Eliazar and Parr propose a new technique for hierarchical slam (ELIAZAR; PARR, 2006), that process the SLAM problem using particle filters in two levels. On the low level, the DP-SLAM algorithm is executed inside a small portion of the map to obtain locally accurate paths. Then, the best resulting trajectory of the low level process is passed to the high-level SLAM, which is another instance of DP-SLAM. Since the input of the high level SLAM is already a refined segment of trajectory, a smaller number of particles is required to obtain good results.

Another method based on submaps is the SegSLAM algorithm (FAIRFIELD; WETTERGREEN; KANTOR, 2010), which introduces the idea of using segments, instead of single submaps. A segment represents a limited region of the environment that can be described by multiple submaps. Hence, different combinations of submaps (one for each segment), produce different solutions, i.e., different maps.

As our approach uses the basic principles of SegSLAM, we dedicate the next subsection to discuss it in details.

2.5.1 SLAM based on segments

SegSLAM (FAIRFIELD; WETTERGREEN; KANTOR, 2010) is a SLAM approach based on particle filters for 3D environments, that extends the estimation step of the filter to decide when to make the environment segmentation. Its main distinction to RBPF SLAM is that, while RBPF particles describe complete trajectories hypothesis, SegSLAM particles are only responsible for generating submaps of the environment, which are stored in a structure called Segmented Map or SegMap. This structure maintains the connections between the submaps using a graph, and thus, to reconstruct a possible trajectory it is necessary to concatenate compatible consecutive segments.

Each SegSLAM particle $\mathbf{s}_t = \{\mathbf{x}_t, \theta_i\}$ consists of a submap θ_i, from the SegMap

Θ, and a pose $\mathbf{x}_t$ described in the coordinate system of the submap. The particles are propagated according to the motion model $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$ and to the segment transition model $p(\theta = \{\theta_{same}, \theta_{new}, \theta_{match}\}|s_{t-1}, \Theta)$. This transition model estimates when the particle must create a new segment (θ_{new}), stay on the same segment (θ_{same}) or return to a previously defined segment (θ_{match}).

The first step of the segment prediction stage is done by sampling paths from the SegMap graph. Then the combinations of possible paths are weighted by a technique of submaps matching. Later, a list of potentially matchable segments is generated, with their respective positions transformations. In the end, particles choose matches from this list, or create new segments if a good matching has not been found.

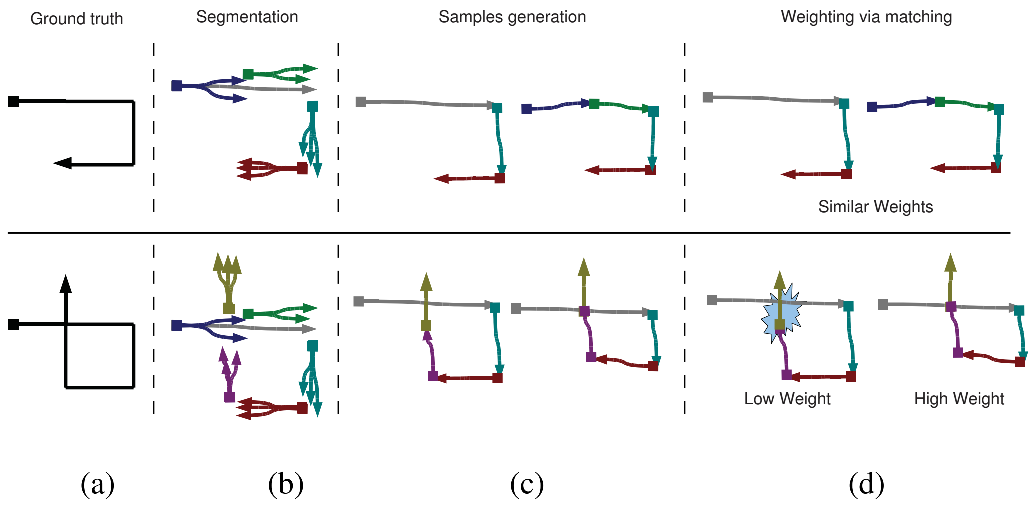


Figure 2.14: Segments estimation in SegSLAM, comparing two examples of trajectories. While the robot does not revisit known regions (top), the samples of segments combinations have similar weights. After closing loops (bottom), samples with more alignment erros have lower weights than others. Figure adapted from (FAIRFIELD; WETTERGREEN; KANTOR, 2010)

Fig. 2.14 shows the segment estimation step of SegSLAM, using as example the two trajectories displayed in (a). (b) presents the two momentarily SegMaps, from where the combinations of path segments, in (c), were sampled. Then (d), the matching of compatible segments is made and the sampled combinations are weighted. In the example at the top, the robot does not return to previously visited areas, therefore the matching results are not significant and the samples present similar weights. In the example at the bottom, some samples have higher weights than others, because some combinations are more misaligned than others.

Even though the increase in the diversity of solutions, resulting from the numerous possibilities of segments combinations, is beneficial to circumvent the problem of particle depletion, it considerably increases the search space. In SegSLAM, the sampling step is made considering only a local analysis, i.e., the algorithm chooses the best samples from chains of few segments, not samples of the entire trajectory. It is difficult to find a good combination of submaps within the total space of solutions without a well-directed search, so numerous samples must be generated, which becomes unfeasible as the number of segments grows. Locally submaps adjustments can lead to inconsistent global maps, therefore, SegSLAM uses an offline optimization approach to search in the space of possible solutions. It is noteworthy that the goal claimed by them was not to build a global map, but only allow a proper navigation and localization.

Nevertheless, the addoption of a strategy of topological map learning in conjunction with the idea of segments of maps, could generate a new SLAM strategy which avoids both the particle depletion problem and the unrestricted growth of the search space of solutions. For example, while SegSLAM circumvent the particle depletion, a method like PTM or HMT-SLAM (seen in Section 2.5), guides the search for good topologies.

2.5.2 Matching of submaps: the ICP technique

A key component of submap-based approaches is the matching of submaps. In the SLAM based on segments the matching process evaluates the quality of the maps generated from segments combinations. One of the most popular methods for matching two sets of points is the ICP (iterative closest point) (BESL; MCKAY, 1992; CHEN; MEDIONI, 1992), which is the technique used by SegSLAM. The ICP is a iterative algorithm that basically consists of two steps: first, it computes the nearest neighbors between the points of a set **Q** to the points of another set **P**, and then it obtains a transformation for **Q** that minimizes the square error between the associated points.

It is possible to obtain the nearest neighbors in the most simplistic way, which is testing all points between themselves. However, a better strategy is to use KD-Trees (BENTLEY, 1975). The complexity to construct a KD-Tree is $O(n \log n)$, and the average cost to find the nearest neighbor of a point is $O(\log n)$. Therefore, since the matching is usually performed multiple times, using a KD-tree is advantage.

The second stage of ICP is to find a transformation that, applied to all the points of **Q**, minimizes the error between the associated points of $\mathbf{P} = \{\mathbf{p}_i\}_{i=1}^n$ and $\mathbf{Q} = \{\mathbf{q}_i\}_{i=1}^n$, where $\mathbf{p}_i = (p_i x, p_i y)^T$ and $\mathbf{q}_i = (q_i x, q_i y)^T$. In other words, we want to minimize the following expression

$$E = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{p}_i - \mathbf{R}\mathbf{q}_i - \mathbf{t}||^2 \tag{2.5.1}$$

where $\mathbf{t} = (t_x, t_y)^T$ is the translational component of the transformation and $\mathbf{R} = \begin{vmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{vmatrix}$ is the rotational component of the transformation, relative to an angle α. We can rewrite E as

$$E = \frac{1}{n}\sum_{i=1}^{n}[(p_i x - \cos\alpha.q_i x + \sin\alpha.q_i y - t_x)^2 + (p_i y - \sin\alpha.q_i x - \cos\alpha.q_i y - t_y)^2]$$

To minimize E, we have to find the values where the derivatives of E with respect to t_x, t_y and α are equal to zero.

Starting with the derivative with respect to t_x

$$\frac{dE}{dt_x} = \frac{1}{n}\sum_{i=1}^{n} -2(p_i x - \cos\alpha.q_i x + \sin\alpha.q_i y - t_x) = 0$$

$$\sum_{i=1}^{n}(p_i x - \cos\alpha.q_i x + \sin\alpha.q_i y) - n.t_x = 0$$

$$t_x = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{i=1}^{n} p_i x - \cos\alpha.\sum_{i=1}^{n} q_i x + \sin\alpha.\sum_{i=1}^{n} q_i y\right) \tag{2.5.2}$$

Now, deriving E with respect to t_y

$$\frac{dE}{dt_y} = \frac{1}{n}\sum_{i=1}^{n} -2(p_iy - \sin\alpha.q_ix - \cos\alpha.q_iy - t_y) = 0$$

$$\sum_{i=1}^{n} -2(p_iy - \sin\alpha.q_ix - \cos\alpha.q_iy) - n.t_y = 0$$

$$t_y = \frac{1}{n}\left(\sum_{i=1}^{n} p_iy - \sin\alpha.\sum_{i=1}^{n} q_ix - \cos\alpha.\sum_{i=1}^{n} q_iy\right) \tag{2.5.3}$$

Finally, deriving E with respect to α

$$\frac{dE}{d\alpha} = \frac{1}{n}\sum_{i=1}^{n}[2(p_ix - \cos\alpha.q_ix + \sin\alpha.q_iy - t_x)(\sin\alpha.q_ix + \cos\alpha.q_iy) + \\ 2(p_iy - \sin\alpha.q_ix - \cos\alpha.q_iy - t_y)(-\cos\alpha.q_ix + \sin\alpha.q_iy)] = 0$$

Expanding the multiplications and discarding the values that cancel themselves, we get

$$\sum_{i=1}^{n}(\sin\alpha.p_ix.q_ix + \cos\alpha.p_ix.q_iy - \sin\alpha.t_x.q_ix - \cos\alpha.t_x.q_iy - \\ \cos\alpha.p_iy.q_ix + \sin\alpha.p_iy.q_iy + \cos\alpha.t_y.q_ix - \sin\alpha.t_y.q_iy) = 0$$

Now, we isolate t_x and t_y,

$$\sin\alpha\sum_{i=1}^{n}(p_ix.q_ix) + \cos\alpha\sum_{i=1}^{n}(p_ix.q_iy) - \cos\alpha\sum_{i=1}^{n}(p_iy.q_ix) + \sin\alpha\sum_{i=1}^{n}(p_iy.q_iy) + \\ t_x(-\sin\alpha\sum_{i=1}^{n} q_ix - \cos\alpha\sum_{i=1}^{n} q_iy) + t_y(\cos\alpha\sum_{i=1}^{n} q_ix - \sin\alpha\sum_{i=1}^{n} q_iy) = 0$$

and replace both values with the expressions from Equations 2.5.2 and 2.5.3

$$\sin\alpha\sum_{i=1}^{n}(p_ix.q_ix) + \cos\alpha\sum_{i=1}^{n}(p_ix.q_iy) - \cos\alpha\sum_{i=1}^{n}(p_iy.q_ix) + \sin\alpha\sum_{i=1}^{n}(p_iy.q_iy) + \\ \frac{1}{n}(\sum_{i=1}^{n} p_ix - \cos\alpha.\sum_{i=1}^{n} q_ix + \sin\alpha.\sum_{i=1}^{n} q_iy)(-\sin\alpha\sum_{i=1}^{n} q_ix - \cos\alpha\sum_{i=1}^{n} q_iy) + \\ \frac{1}{n}(\sum_{i=1}^{n} p_iy - \sin\alpha.\sum_{i=1}^{n} q_ix - \cos\alpha.\sum_{i=1}^{n} q_iy)(\cos\alpha\sum_{i=1}^{n} q_ix - \sin\alpha\sum_{i=1}^{n} q_iy) = 0$$

Expanding the multiplications (with some fast-forwarding), we get

$$\sin\alpha\sum_{i=1}^{n}(p_ix.q_ix) + \cos\alpha\sum_{i=1}^{n}(p_ix.q_iy) - \cos\alpha\sum_{i=1}^{n}(p_iy.q_ix) + \sin\alpha\sum_{i=1}^{n}(p_iy.q_iy) + \\ -\frac{\sin\alpha}{n}\sum_{i=1}^{n} p_ix\sum_{i=1}^{n} q_ix - \frac{\cos\alpha}{n}\sum_{i=1}^{n} p_ix\sum_{i=1}^{n} q_iy + \frac{\cos\alpha}{n}\sum_{i=1}^{n} p_iy\sum_{i=1}^{n} q_ix - \frac{\sin\alpha}{n}\sum_{i=1}^{n} p_iy\sum_{i=1}^{n} q_iy = 0$$

Isolating both $\sin\alpha$ and $\cos\alpha$

$$\sin\alpha\left[\sum_{i=1}^{n}(p_ix.q_ix) + \sum_{i=1}^{n}(p_iy.q_iy) - \frac{1}{n}\sum_{i=1}^{n} p_ix\sum_{i=1}^{n} q_ix - \frac{1}{n}\sum_{i=1}^{n} p_iy\sum_{i=1}^{n} q_iy\right] +$$

$$\cos\alpha\left[\sum_{i=1}^{n}(p_ix.q_iy)-\sum_{i=1}^{n}(p_iy.q_ix)-\frac{1}{n}\sum_{i=1}^{n}p_ix\sum_{i=1}^{n}q_iy+\frac{1}{n}\sum_{i=1}^{n}p_iy\sum_{i=1}^{n}q_ix\right]=0$$

Thus, we can obtain α through the arctangent of $\sin\alpha/\cos\alpha$

$$\alpha=atan\left(-\frac{[\sum_{i=1}^{n}(p_ix.q_iy)-\sum_{i=1}^{n}(p_iy.q_ix)-\frac{1}{n}\sum_{i=1}^{n}p_ix\sum_{i=1}^{n}q_iy+\frac{1}{n}\sum_{i=1}^{n}p_iy\sum_{i=1}^{n}q_ix]}{[\sum_{i=1}^{n}(p_ix.q_ix)+\sum_{i=1}^{n}(p_iy.q_iy)-\frac{1}{n}\sum_{i=1}^{n}p_ix\sum_{i=1}^{n}q_ix-\frac{1}{n}\sum_{i=1}^{n}p_iy\sum_{i=1}^{n}q_iy]}\right) \tag{2.5.4}$$

From these results (Equations 2.5.2, 2.5.3 and 2.5.4), to obtain the transformation that minimizes the alignment error between the two sets of points, we only need to calculate all summations, which has a linear cost in relation to the number of points of the two sets. Finally, for each matching instance, both the nearest neighbor searching and the error minimization are performed repeatedly until convergence, or until a specific number of times is reached.

3 SDP-SLAM: A SUBMAP-BASED DP-SLAM

Previous chapter showed that DP-SLAM has a degraded performance in large scale environments. It works well on limited spaces, but suffers with the problem of particle depletion.

In this chapter, we first describe some modifications of DP-SLAM, made at the beginning of our work. We applied to DP-SLAM resampling step, an elitism strategy in the particles disposal, along with the selective resampling technique proposed by Grisetti et al. (GRISETTI; STACHNISS; BURGARD, 2005). The experimental results obtained by this modified DP-SLAM (that we will call MDP-SLAM) were good, however the number of particles required by the filter, although smaller, remained high.

As shown in the previous chapter, a strategy to reduce the complexity of RBPF methods is partitioning the process in submaps. DP-SLAM has appealing characteristics to a submap-based approach that can circumvent the impoverishment problem, and reduce the number of particles required. In particular, DP-SLAM presents some similarities with the SegSLAM method, described on Section 2.5.1. For instance, while the first needs to consult the particles history to construct a map, the second consults a graph of segments to build trajectories and combine submaps.

The description of our method combining aspects from DP-SLAM and SegSLAM, that we called Segmented DP-SLAM (SDP-SLAM), is the core of this chapter. SDP-SLAM is a submap-based SLAM approach composed of two levels of particle filters. The bottom level filter generates several accurate local maps using our modified DP-SLAM (MDP-SLAM), while the top level filter estimates combinations of submaps to compose global maps. By using multiple instances of MDP-SLAM to process small regions, besides gaining the space optimization from DP-SLAM, we reduce the particle depletion problem. Also, we increase the diversity of solutions by keeping various hypotheses of submaps for each segment of the map, as in SegSLAM. Nevertheless, unlike SegSLAM, we restrain the search space for solutions by using our top level particle filter. Additionally, our method introduces a submaps matching technique that reduces ambiguities that may occur in the SegSLAM matching process.

3.1 Our first approach: DP-SLAM with the modified resampling

DP-SLAM has an optimized storage space to represent the environment in an efficient way. However, since it was developed in the beginning of the RBPF SLAM popularization, all the improvement made up to now on the field was not incorporated on it. For instance, the method suffers a lot with the particle depletion problem. In fact, the resampling step of the filter makes the particles converge very fast to a local minima, and thus the method requires a large number of particles to work properly (tens of thousands of particles), which is extremely costly.

Our first approach to improve the quality of DP-SLAM was to extend it incorporating an exploration process. In our method the robot was guided to areas of high *localizability*[1] to improve the map quality. By combining DP-SLAM with a Boundary Value Problem (BVP)-based exploration (PRESTES; ENGEL, 2011), we generate a new integrated exploration strategy. We will not describe in details our navigation strategy, since this is out of the scope of this dissertation, but the principle was to maintain the robot in regions with high localizability (for example, regions near walls) and to often revisit known locations to reduce the uncertainty about its localization.

Apart from the combination with the exploration strategy, we made modifications on DP-SLAM resampling step by incorporating the *selective resampling* technique, developed by Grisetti (GRISETTI; STACHNISS; BURGARD, 2005). This method uses a measure, introduced by Liu (LIU, 1996), called *number of effective particles* (N_{eff})

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w_t^{[i]})^2}$$

where $w_t^{[i]}$ is the weight of a particle i from the set of N particles. N_{eff} estimates how well the current particle set represents the true SLAM posterior. The worse the approximation, the higher the variance of the importance weights. If all samples are good, and their weights are equally distributed, the value of N_{eff} is N. On the other hand, if a single sample concentrates all the propability distribution, the value of N_{eff} is 1.

The idea of selective resampling is to avoid performing the resampling step while N_{eff} keeps high, because in such situation the particles are well approximated to the target distribution. Only when N_{eff} drops below a certain threshold Th (e.g. $Th = N/2$), the resampling is performed. This strategy reduces the number of resampling operations, and as a result, decreases the risk of discarding good particles.

Besides the selective resampling, we used an elitism strategy that replaces only a subset of the worst particles at each resampling step, instead of the entire particles set. The size of this subset of particles is represented by N_{wp}. This strategy aims to keep the filter diversity, preserving some non-optimal particles and avoiding fast filter convergence.

Our modified DP-SLAM algorithm, called MDP-SLAM in this dissertation, differs only in the resampling step from DP-SLAM (Algorithm 2.2 in Section 2.4.4). The resampling step of MDP-SLAM is described in Algorithm 3.1. The Th and N_{wp} values were empirically defined as $N/8$ and $N/4$, respectively.

Algorithm 3.1: Modified Resampling step

```
1 Compute $N_{eff}$
  if $N_{eff}$ < $Th$ then
2     Eliminate the set of $N_{wp}$ worse particles.
3     Sample $N_{wp}$ new particles from the remaining set of $N - N_{wp}$ particles,
      with probabilities according to their weights.
  end
```

[1]*Localizability* is the degree to which a vehicle, at a certain location, can *relocalize* itself given the current map (MAKARENKO et al., 2002).

In Chapter 4, we present the results of experiments comparing MDP-SLAM with DP-SLAM. We also present results obtained with our integrated exploration strategy. As we will observe, the SLAM still presents opportunities for improvement. For instance, although the number of particles required by MDP-SLAM is smaller than in DP-SLAM, this number still is high, augmenting in large environments. The same can be said for the integrated exploration method.

Therefore, we decided to look DP-SLAM from another perspective. DP-SLAM has appealing characteristics to the application of a submap-based approach, that can reduce even more the number of particles required and increase the diversity of solutions. So, we propose a submap-based SLAM approach, called SDP-SLAM.

3.2 The SDP-SLAM algorithm

Our proposed method called Segmented DP-Slam, or simply SDP-SLAM, combines maps based on segments, from SegSLAM, to the particles ancestry tree, from DP-Slam. The idea is to capture the high diversity of solutions, ie. global maps generated by numerous possibilities of submaps combinations, without increasing the size of the space for data storage.

SDP-SLAM is a SLAM approach composed of two levels of particle filters, like the hierarchical SLAM methods shown in Section 2.5 (ELIAZAR; PARR, 2006; LEE; LEE; OH, 2011). At the bottom level, the operation of the particle filter is similar to the filter in SegSLAM, where the particles are only responsible to generate locally accurate submaps. The key difference lies in the way the maps and trajectories are constructed, using the DP-Mapping process. At the top level, a particle filter is used to estimate good combinations of submaps of the environment.

A probabilistic graph with weighted connections between submaps is used to direct the sampling step of the top level particle filter. This graph, called Probabilistic Graph of Segments (PGS), represents a Monte Carlo approximation of the probability distribution of all possible topologies that can represent the environment. A segment of the environment contains multiple submaps, each one built by a different particle. Those submaps are represented by nodes of the PGS, which are grouped in levels representing the segments. The connections between submaps of adjacent segments are the edges of the PGS.

The SDP-SLAM algorithm is presented in Algorithm 3.2. The first step (line 1) is the initialization of both particle filters. At the bottom level, all particles start at the same position with empty submaps. At the top level, all particles start with empty submaps combinations. The PGS starts with only one segment, since the environment was not segmented yet.

In the main loop, the first step (line 2) is the acquisition of the odometry and sensors measurements. Next, the update of the bottom level filter is performed to build hypotheses of local maps. The initial five steps are the same performed by our modified DP-SLAM, described in last section. First, particles are propagated inside the current segment and weighted (lines 3-4). Then, our modified resampling is made (line 5), the ancestry tree is updated through a process of prune and merge (line 6), and the particles observations are updated into the Distributed Particle Map (line 7).

Next, occurs the segmentation decision (line 8). We chose to perform a periodic segmentation, therefore we only check if it is the right time to segment the environment. We also experimented a segmentation based on the particles dispersion. Both strategies will be properly detailed later in this chapter.

Algorithm 3.2: SDP-SLAM algorithm

```
1  Initialization
   while the robot is navigating do
2      Read odometry and sensors measurements.
       Bottom level process:
       begin
3          Sampling    // particles propagation inside current segment
4          Weighting    // particles evaluation
5          Resampling    // particles replication/discard through our
           modified resampling
6          Ancestry tree update    // prune/merge of the ancestry tree and
           insertion of new particles into the tree
7          Submaps estimation    // update of the particles local maps
8          Segmentation decision    // verification of the need for
           segmentation
           if a segmentation occurs then
9              Ancestry tree anchorage    // fixation of the current
               ancestry tree information
10             Particles restart    // generation of a new submap for each
               particle
           end
       end
       Top level process:
       begin
           if a segmentation occurs then
11             Insertion of a new level of nodes in the PGS
           end
12         Estimation of submaps combinations    // update of the set of
           combinations hypotheses, consulting the PGS
13         Weighting of submaps combinations    // hypotheses evaluation
           through a process of submaps matching
14         Update of the PGS    // update of the weights of connections
           between adjacent submaps
       end
   end
```

Whenever a segmentation occurs, the bottom level particle filter is stopped and the ancestry tree section regarding the last segment is anchored, so it can not be modified later (line 9). The current set of particles is restarted to allow the construction of new independent submaps (line 10). At the top level, a new set of nodes representing the submaps of the new segment is inserted into the PGS. Among the information stored in those nodes are the identification of the particle, required for queries on ancestry tree; and the initial and final transformations of the submap (the first and the last robot poses of the submap), used to combine submaps in a same coordinate system.

The next step is the update of the top level particles (line 11), that are responsible for estimating hypotheses of submaps combinations. We adopt an elitism strategy similar to our modified resampling, described in last section. Only a small set of the worst particles are eliminated, while the best particles are maintained. The sampling of new particles is made consulting the Probabilistic Graph of Segments.

Then, the evaluation of the top level particles is performed through the matching between overlapping submaps of each sample (line 12). As in SegSLAM, the matching process is made with the ICP (Iterative Closest Point) algorithm (FAIRFIELD; WETTERGREEN, 2009). ICP is a very simple and fast method, but requires that the two sets of points being compared have a strong association, otherwise, the method might converge to a a local minimum or even not converge. Besides, ICP does not utilize the information about empty spaces implied by the use of range sensors, because the data is reduced to point clouds. Hence, our matching select points from free-space regions that have already been visited, as we will see in Section 3.4.2. Using this type of points, we can reduce ambiguities and produce more reliable maps.

The last step of our method is the update of the Probabilistic Graph of Segments (line 13). The graph is updated through a Monte Carlo process, in which a set of combinations samples is regularly generated and evaluated with the matching technique. The measured quality of each sample is used to compute the weights of the connections between each adjacent segment in the sample. As more samples are generated during the process, more connections are updated, and more precise is the probability distribution of the weights in the graph. In the end, combinations of segments that have high matching measures will have higher probability to be chosen than combinations that have low matching measures.

Figure 3.1 shows an example of the functioning of SDP-SLAM, emphasizing the PGS. In (a), three submaps of a same segment are depicted in the map. The probabilistic graph at this point only have the three nodes (submaps 1A, 1B, 1C) of the first segment, as shown in (b). When the method is processing the second segment, in (c), the particles of the bottom level are generating a new set of submaps. At this moment, in (d), the graph of segments contains two levels of nodes, representing both segments. Then, as shown in (e), it is possible to combine the submaps of the two segments and evaluate the sampled combinations. The weight of each connection between submaps is updated in the graph of segments, as indicated by the w values in (f). These estimated weights will be used during the sampling step of the top-level particle filter. Finally, (g), a possible trajectory is reconstructed by combining two submaps ($1A$ and $2C$). The global map is built by consulting the observations made by each particle of the selected submaps. As shown in (h), a transformation T must be applied to put both submaps in the same coordinate system. This transformation is a composition of the final pose from the first submap (T_e1A) with the initial pose from the second submap (T_s2C).

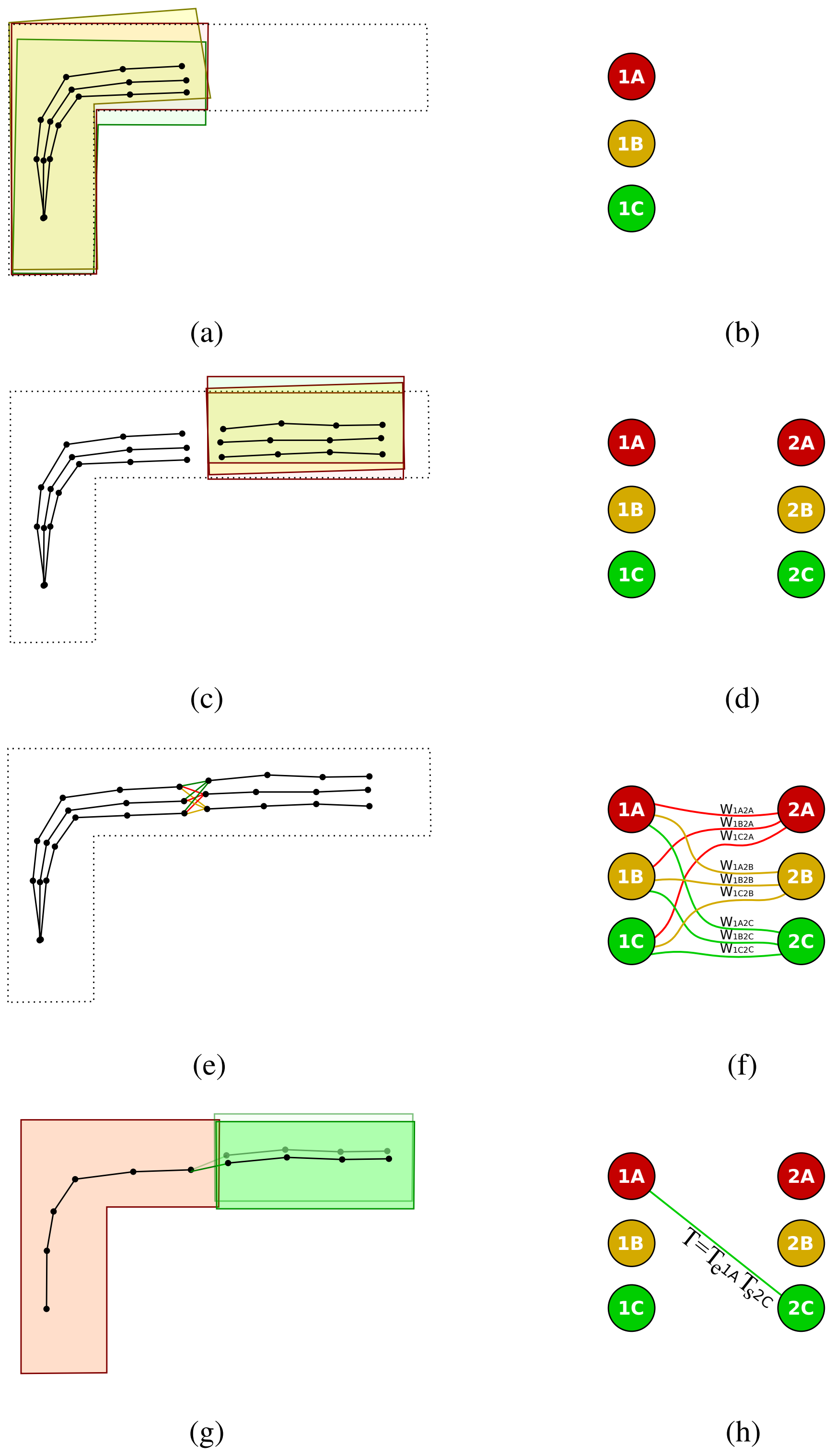


Figure 3.1: Example of functioning of SDP-SLAM, showing the submaps (left) and the PGS (right). (a) First generation of submaps. (b) PGS with one level. (c) Second generation of submaps. (d) PGS with two levels. (e) Creation of the relationships between submaps. (f) Weighted connections between submaps (g) Reconstruction of a possible trajectory, applying a transformation to the second submap (green). (h) Composition of the transformation.

Figure 3.2 shows the ancestry evolution for the example presented in Figure 3.1. In (a), the ancestry tree contains only the particles of the first segment. In (b), the tree contains the particles of both first and second segments. In (c), we show the resulting tree after the pruning and merging process. In (d), a possible path is selected, combining two particles from different branches. A transformation is required to connect the two paths in the same coordinate system.

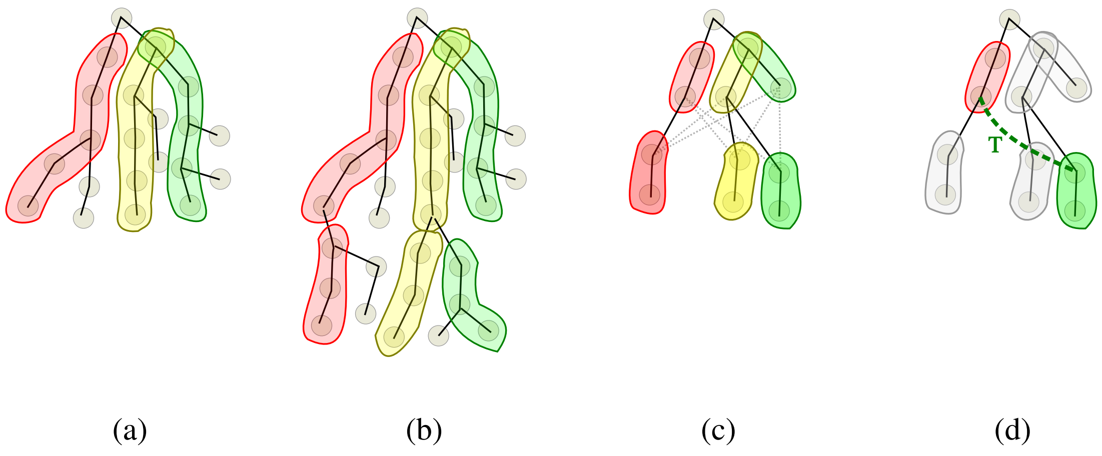


Figure 3.2: Example of the ancestry tree evolution. (a) During the first segment. (b) During the second segment. (c) Exemplification of the prune/merge of ancestry tree. (d) Reconstruction of a possible trajectory, emphasizing the transformation required to connect the two paths.

3.3 Probabilistic Foundations of SDP-SLAM

The proposed method is based on the fact that in large environments, observations within a small region are highly related to one another, while observations from distant regions are not, and therefore, the segmentation of the environment in submaps is feasible. This is the same principle behind the works of Blanco et al. (BLANCO; FERNANDEZ-MADRIGAL; GONZALEZ, 2008), Fairfield et al. (FAIRFIELD; WETTERGREEN; KANTOR, 2010), and others submap-based approaches. Figure 3.3 shows the conditional independence between two segments, given the initial pose of each segment. The odometry **u** and the observations **z** made by the robot are the observed variables, while the variables to be estimated (ie. the hidden variables) are the robot state **s**, the global map Θ (composed by submaps θ_i) and the transformations between submaps T.

Following the formalizations of HMT-SLAM (BLANCO; FERNANDEZ-MADRIGAL; GONZALEZ, 2008) and SegSLAM (FAIRFIELD; WETTERGREEN; KANTOR, 2010), we define the global map of the environment as

$$\Theta = \langle \{\theta_i\}_{i \in V(\Upsilon_t)} \, , \, \{T_{a,b}\}_{a,b \in V(\Upsilon_t)} \rangle \tag{3.3.1}$$

where $\theta_i = \{\theta_i^{(1)}, \theta_i^{(2)}, \cdots, \theta_i^{(p)}\}$ is a set of nodes from the graph of segments Υ_t, and it represents the set of p metric submaps associated to the segment i. $T_{a,b}$ is a set of edges from Υ_t, that represents the coordinate transformations between submaps of two adjacent

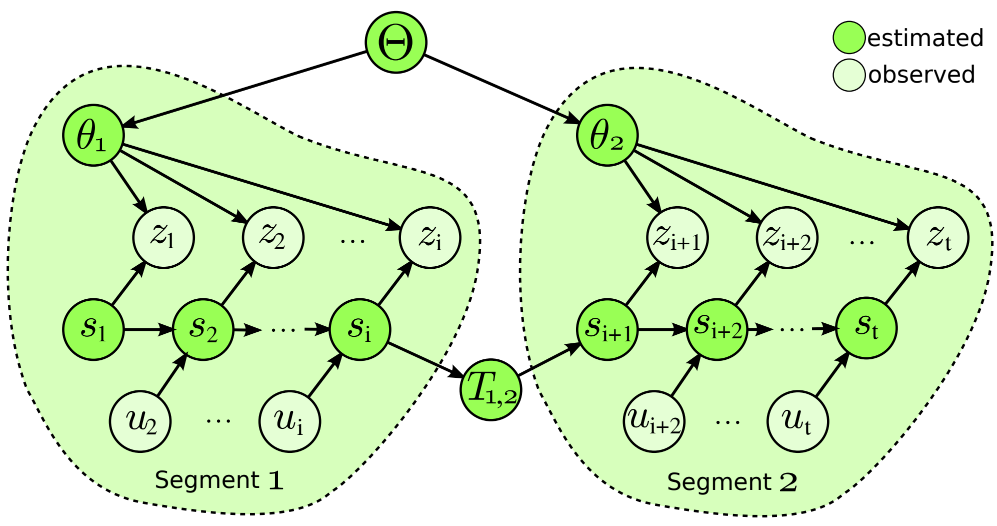


Figure 3.3: Graphical model of SDP-SLAM. The transform T_{ab} is the only relation between the conditionally independent segments.

segments a and b. In fact, for each consecutive pair of segments there are transformations $T_{a,b} = \{T_{a,b}^{(1,1)}, T_{a,b}^{(1,2)}, \cdots, T_{a,b}^{(1,p)}, T_{a,b}^{(2,1)}, \cdots, T_{a,b}^{(p,p)}\}$ associating the submaps generated by all particles.

The state $\mathbf{s}_t$ of the robot pose at instant t is given by

$$\mathbf{s}_t = \langle \mathbf{x}_t \, , \, \gamma_t \rangle \tag{3.3.2}$$

where $\mathbf{x}_t$ represents the metric robot pose, while γ_t indicates to which segment the robot pose is associated.

The posterior distribution of SLAM, considering $\mathbf{s}_{1:t}$ and Θ is defined as

$$p(\mathbf{s}_{1:t}, \Theta \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{3.3.3}$$

By applying the Rao-Blackwellized factorization, as defined in Chapter 2, we obtain

$$p(\mathbf{s}_{1:t}, \Theta \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{s}_{1:t} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) p(\Theta \,|\, \mathbf{s}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{3.3.4}$$

Considering the estimation of the state of the robot, we can rewrite the posterior over $\mathbf{s}$ using the definition of conditional probability

$$p(\mathbf{s}_{1:t} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{x}_{1:t}, \gamma_{1:t} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{x}_{1:t} \,|\, \gamma_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) p(\gamma_{1:t} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{3.3.5}$$

Now, the posterior of the robot trajectory $\mathbf{x}_{1:t}$ can be converted into the product of the estimations over trajectories segments

$$p(\mathbf{x}_{1:t} \,|\, \gamma_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \prod_{k,k'}^{t} p(\mathbf{x}_{k:k'} \,|\, \gamma_{k:k'}, \mathbf{z}_{k:k'}, \mathbf{u}_{k:k'}) \tag{3.3.6}$$

where k and k' represent the instants when two consecutive segmentations occur. This is possible because, in a submap-based approach, the segments of the robot trajectory are defined in different coordinate systems, so they can be considered independent variables.

Also, every pose inside a same segment has the same information about γ, thus the solution to the posterior is exactly the same as the traditional SLAM, defined in the last chapter. Note that the information associating consecutive robot poses in the boundaries of segments is not lost, because they are inserted in the transformations between submaps. Therefore, the entire estimation process over the robot position can be tought as the joint of multiple instances of the SLAM problem in limited regions of the environment.

Regarding the map estimation, we can rewrite the posterior over Θ as

$$p(\Theta \,|\, \mathbf{s}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\{\theta_i\}_{i\in V(\Upsilon_t)}\,,\ \{T_{a,b}\}_{a,b\in V(\Upsilon_t)} \,|\, \mathbf{s}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{3.3.7}$$

The metric sub-maps θ_i and the coordinate transformations $T_{a,b}$ are conditionally independent, due to the distributed nature of the robot path, so the posterior over Θ can be factorized

$$p(\Theta \,|\, \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\{\theta_i\}_{i\in V(\Upsilon_t)} \,|\, \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})\; p(\{T_{a,b}\}_{a,b\in V(\Upsilon_t)} \,|\, \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{3.3.8}$$

The same principle used on the definition of the posterior over the robot path (Eq. 3.3.6) can be applied to factorize both the joint posterior of all submaps and the posterior over the transformations

$$p(\{\theta_i\}_{i\in V(\Upsilon_t)} \,|\, \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \prod_{i\in V(\Upsilon_t)} p(\theta_i \,|\, \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{3.3.9}$$

$$p(\{T_{a,b}\}_{a,b\in V(\Upsilon_t)} \,|\, \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \prod_{a,b\in V(\Upsilon_t)} p(T_{a,b} \,|\, \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{3.3.10}$$

Now, we can estimate the various segments of robot trajectories and their associated submaps and transforms by performing multiple individual runs of particle filters. The propagation of the particles generates the segments of robot trajectories, while for each segment a submap and a transformation is estimated. In our approach, we use the MDP-SLAM method, described in the Section 3.1 to do this step.

Finally, we estimate the posterior distribution over the combinations of submaps, represented by the chains of segments $\gamma_{1:t} = \{\gamma_1, \gamma_2, \cdots, \gamma_t\}$. The values of $\gamma_1, \gamma_2, \cdots, \gamma_t$ are selected from the following sets

$$\gamma_1 \in \{\gamma_{1a}, \gamma_{1b}, \gamma_{1c}, ...\gamma_{1p}\}$$

$$\gamma_2 \in \{\gamma_{2a}, \gamma_{2b}, \gamma_{2c}, ...\gamma_{2p}\}$$

$$\vdots$$

$$\gamma_t \in \{\gamma_{ta}, \gamma_{tb}, \gamma_{tc}, ...\gamma_{tp}\}$$

where $\gamma_{ia}, \gamma_{ib}, \gamma_{ic}, ...\gamma_{ip}$ are the identifications of the p distinct submaps in the i-th segment. For instance, the combination $\gamma_{1:3} = \{\gamma_{1a}, \gamma_{2b}, \gamma_{3a}\}$ is the chain of the submaps indicated by γ_{1a} (first segment), γ_{2b} (second segment), and γ_{3a} (third segment).

Our method uses a particle filter in a global level that tries to find good combinations of segments. The particle filter approximates the posterior distribution using a set of weighted particles

$$p(\gamma_{1:t} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \approx \sum_{i=1}^{n} w^{(i)} \gamma_{1:t}^{(i)} \tag{3.3.11}$$

where $[\gamma_{1:t}^{(1)}, \gamma_{1:t}^{(2)}, \cdots, \gamma_{1:t}^{(n)}]$ is the set of n particles with weights $[w^{(1)}, w^{(2)}, \cdots, w^{(n)}]$, at instant t. The particles are sampled from the previous set of particles according to a transition model

$$\gamma_{1:t} = q(\gamma_{1:t} \,|\, \gamma_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{3.3.12}$$

while the weights are computed through a map matching technique that evaluates the quality of submaps combinations. The matching technique and the particles sampling process are properly detailed in Sections 3.4.2 and 3.4.3.

3.4 Detailing the SDP-SLAM strategy

The SDP-SLAM structure overview is presented on Figure 3.4. SDP-SLAM is composed by two levels of particle filters: the low level filter is responsible to estimate the segments of the robot path and the associated local maps, while the high-level process is responsible to estimate the topology of the global map, in other words, find the best combinations of submaps.

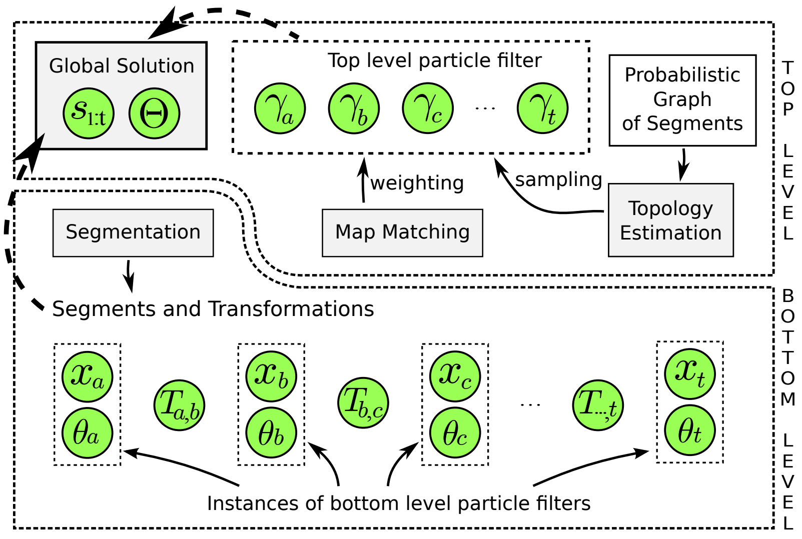


Figure 3.4: SDP-SLAM structure overview. At the bottom level, the segmentation is performed to partition the SLAM estimation. Bottom level particle filters estimate the submaps and the segments of the robot path. At the top level, a particle filter is used to estimate the global map. Two tasks are performed: the topology estimation (sampling), that is based on the PGS, and the matching of submaps (weighting).

Since the bottom level particle filters correspond to the modified DP-SLAM, already explained in Section 3.1, it remains to be detailed three important steps of SDP-SLAM: the segmentation of the environment, that indicates when to start or stop the local instances of SLAM; the matching of submaps, that evaluates the quality of the maps built by chains of segments; and the topology estimation, that searches for the combinations of segments that generate the best global maps.

3.4.1 Segmentation

The segmentation process divides the environment into regions that are independent and locally consistent. The independence between submaps arises from the fact that they are built in different coordinate systems, defined by the initial position of the robot in the submap. As each local process depends only on the sensor readings and the odometry of the robot during a specific period of time, the correlations between features of different regions are not computed in the bottom level (these correlations are intrinsically treated by the high level process responsible by the topology estimation). Therefore, considering that the size of the segments is limited, the cost to construct a submap is constant, regardless of the size of the global map.

The local consistency of a map is directly associated at how much of the map the robot can see at each moment. In small regions, the map description is accurate because the accumulated odometry errors from the robot are small.

Most of submap-based SLAM approaches performs the environment segmentation practically on a regular time interval. Methods such as the hierarchical approaches, of Eliazar and Parr (ELIAZAR; PARR, 2006) and Lee et al. (LEE; LEE; OH, 2011), always divide the environment at a certain number of steps. In the ATLAS framework the segmentation is done according to the accumulated error of the robot pose (BOSSE et al., 2003). In turn, Estrada et al. does the partitioning whenever the robot observes a maximum number of new features, to maintain constant the complexity to build submaps (ESTRADA; NEIRA; TARDOS, 2005).

Some methods propose more complex strategies for segmentation, as is the case of the HMT-SLAM (BLANCO; FERNANDEZ-MADRIGAL; GONZALEZ, 2008) and the SegSLAM (FAIRFIELD; WETTERGREEN; KANTOR, 2010). The former divides the environment into contiguous regions, through positions that minimize the overlap between submaps. On the other hand, the latter uses a predictive score metric combined with a localization error metric (using a limited amount of look-ahead data), that estimates how well the current map of the robot predicts future measurements.

Our approach in this aspect is quite simple. We tested two modes of segmentation: the first, segmenting the environment regularly, based on the number of steps processed by the filter, and the second, segmenting considering the particles dispersion, that is, a segmentation is performed whenever the uncertainty of the filter grows too much.

While the periodic segmentation is straightforward, the segmentation considering the particles dispersion requires a more in-depth explanation. Given a set of P particles, a measure of the particles dispersion N_{pd}, at instant t, is the standard deviation of the particles positions $\mathbf{x}_t$

$$N_{pd} = \sqrt{\frac{1}{P}\sum_{i=1}^{P} |\mathbf{x}_t^{(i)} - \bar{\mathbf{x}}_t|^2} \tag{3.4.1}$$

The idea behind this strategy is to segment the environment always that N_{pd} reaches a threshold, because this means that the uncertainty about the robot pose is getting too high.

Figure 3.5 shows how this type of segmentation works, in an experiment in a real environment. In (a), the robot starts to move from its initial pose, beginning the environment mapping. The uncertainty about the robot pose grows as the robot moves. This uncertainty can be represented by the dispersion between the position of the particles, whose trajectories are depicted by the blue lines. In (b), a segmentation occurs after the particles of the first segment reach a maximum value of dispersion. At this point, the particle filter of the second segment starts to build new submaps. The uncertainty is small once again, so the particles dispersion is low (the particles trajectories are now represented by dark green lines). In (c), the particles dispersion of the second segment continues to grow until being above the threshold. Then, the segmentation is applied again.

Another aspect of the segmentation based on the particles dispersion can be analyzed in Figure 3.6. In (a), when the robot performs some local revisits (ie. passing more than once by the same place within a segment), the particles dispersion drops, because it is possible to eliminate some particles with poor local maps. In such cases, the dispersion takes longer to reach the stipulated threshold, increasing the size of the segment. However, as shown in (b), revisits to regions mapped by different segments do not imply in the reduction of the particles dispersion. This happens because the bottom level particle filters are independent, so the association between overlapping submaps is only performed at the top level process. This can be verified in the larger loop on the right, where there were no significant changes in the size of the segments, even though that area had been completely revisited.

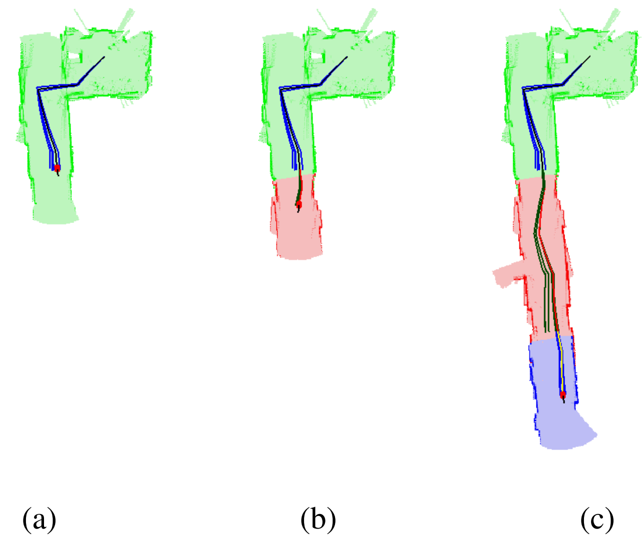


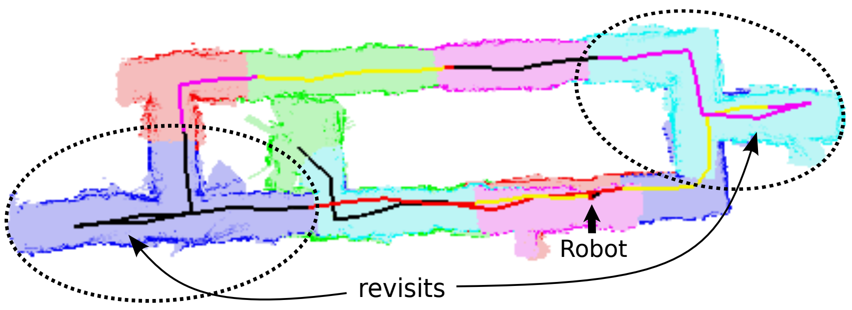
Figure 3.5: Segmentation based on the dispersion of the particles of the bottom level, whose trajectories are represented by blue and dark green lines.

3.4.2 Matching

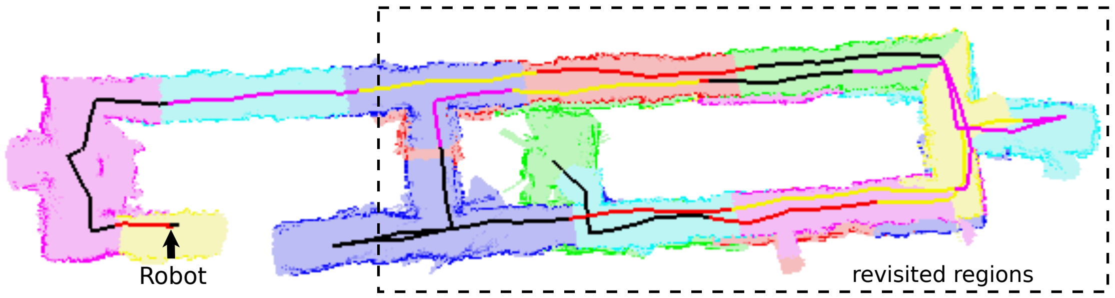
The matching process evaluates the quality of the maps built from combinations of segments. This operation is done in SegSLAM (FAIRFIELD; WETTERGREEN; KANTOR, 2010) by a variant of the ICP method (described in Section 2.5.2) for 3D environments based on octrees (FAIRFIELD; WETTERGREEN, 2009).

Despite its speed and simplicity, one problem with ICP is that its convergence is not guaranteed, in fact, it depends on how good is the association between points in the sets been compared. If many points of a set do not have corresponding points in the other set, the method will not operate properly.

Figure 3.7 shows an example of a common error in the association of points, that occurs when the submaps to be matched are far apart. This is the result of an experiment using ICP over points extracted from obstacles. The different colors in the environment represent different submaps, the sets of dark green and red points represent the points matched with ICP, and the black lines represent the association between the points of both sets. The figure shows a combination of submaps with a large misalignment, sampled after the robot traverse a large loop. In the matching process, the points of the region where the robot is currently located (indicated by the number 9) were associated only to the outside



(a) Local revisits delay the segmentation process



(b) Revisited regions in different segments does not alter the segmentation

Figure 3.6: Example of the segmentation based on the particle dispersion.

walls of the corridor that is being revisited (region indicated by the number 4). Thus, the matching result was completely misleading, causing the wrong evaluation of the quality of the sample. Situations like this often occur when small sets of points are used. Hence, this selection approach may compromise the functioning of the topology estimation.

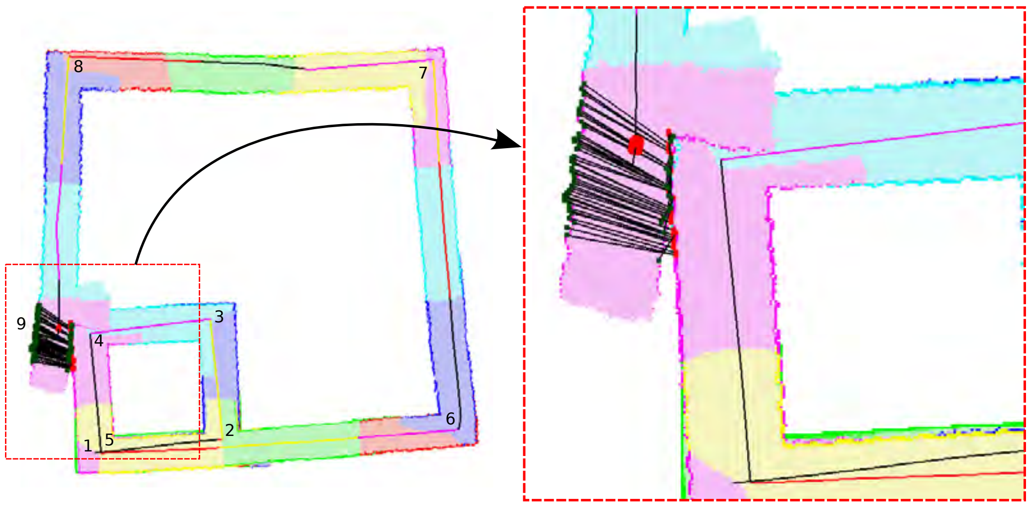
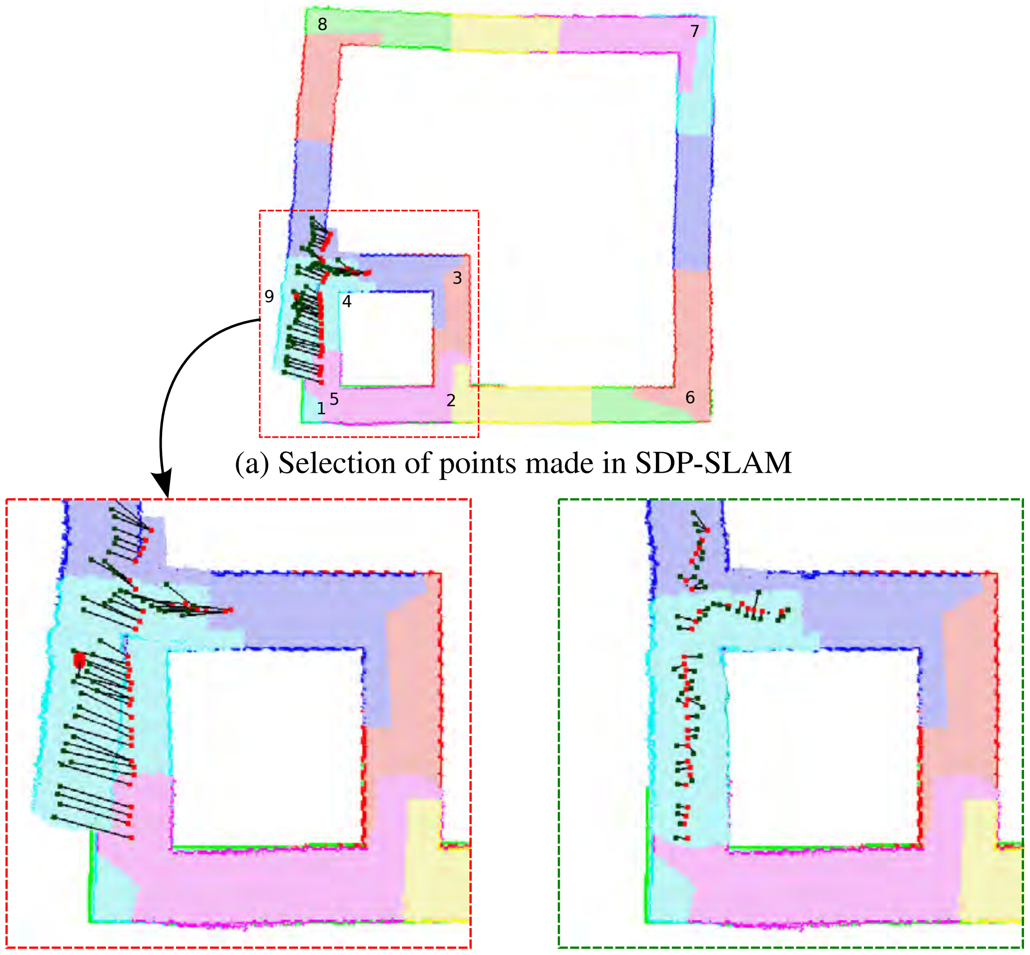


Figure 3.7: Example of a wrong association of points extracted from obstacles.

Many methods, such as SegSLAM, perform the ICP over points extracted from obstacles in the environment, like in the example above. Those strategies disregard relevant information about empty spaces obtained by range finder sensors. This information can be used to improve the matching, since it indicates which part of the environment has been already visited. A main problem is how to choose points from the free-space environment to use in the ICP. In a recent work of SLAM with multiple robots (SAEEDI et al., 2012), a probabilistic voronoi diagram is used in the matching process to capture the information about free-space.

Similarly, our method provides to ICP [2] only points selected from the middle of the corridors. These points are extracted through the skeletonization of the free-space.

Figure 3.8 shows the result of the matching operation in SDP-SLAM. As shown in (a), the tested combination is similar to that analyzed in the experiment with points extracted from obstacles. In (b), we show in details the association between points of the two sets of submaps. As can be seen, both sets of points were extracted from the interior of corridors. In (c), after applying the transformation calculated by ICP to the set of points in dark green, we can see the convergence of the associated points. The idea of the matching process is that the better the association of points is, the better will be the convergence between the two sets of points, and more precise will be the weight given to the submaps combination that is being tested.



(a) Selection of points made in SDP-SLAM

(b) Ampliation of the association of points

(c) Convergence within ICP

Figure 3.8: Example of a matching made by SDP-SLAM.

[2]Our ICP implementation uses the ANN library of Mount and Arya (MOUNT; ARYA, 1997) to do the computation of the nearest neighbors using KD-Trees (this is the same library used by the SegSLAM developers)

Algorithm 3.3: Matching process of SDP-SLAM

```
1 Selection of two submaps with the best overlap in the sample.
2 Extraction of two sets of points from the free-space environment of the
  submaps.
3 ICP processing with the two sets of points.
4 Computation of the sample weight, using the mean ICP nearest neighbor
  error.
```

The matching process of SDP-SLAM is composed by four steps, as presented in Algorithm 3.3. First (line 1), a quick analysis is performed to select which sets of submaps should be compared. The submap where the robot is currently located is always chosen. If the current submap is too small, the prior visited submap is also selected. Then, another submap should be chosen to be compared to the already selected submaps. To define what submap to choose, each one maintains a bounding box indicating its limits. When a particular chain of submaps is selected, the bounding boxes are translated according to the transformations between segments. Then, a search comparing the bounding boxes is made, from the first submap to the last ones, to decide which submaps have the best overlap. As before, we can select one or two submaps, depending on the size of the overlapped area of the submaps tested. When this size is below a given threshold the ICP is not applied, saving computational time.

After choosing the submaps to be compared, we need to extract the central points of free-space (line 2 of Algorithm 3.3). This is done through a skeletonization process, where a morphological thinning of the free space is processed. The idea is to remove all contour cells, until the free-space is shrunk to at most one cell from the boundaries. This also can be viewed as a potential field process, where the potential to be a central cell is larger, as the farther from the obstacles the cell is.

Figure 3.9 shows an example of the point selection process. In (a), the occupancy grid of the submap is shown, with obstacles in black, free-space in white, and unknown cells in grey. Then, it is computed the Manhattan distance from each free-space cell to the nearest obstacles. This is made computing the distances in the four directions: (b) from left to right, (c) top down, (d) from right to left, (e) bottom up. The value of a cell is updated always that the new computed distance is smaller than the current one. The cells updated at each pass are highlighted in yellow. In the end, we choose the cells (highlighted in green) that have no neighbor cell more distant from obstacles than themselves.

After extracting the two sets of points $\mathbf{P} = \{\mathbf{p}_i\}_{i=1}^{n}$ and $\mathbf{Q} = \{\mathbf{q}_i\}_{i=1}^{n}$, the ICP is performed (line 3 of Algorithm 3.3), as described in Section 2.5.2. Finally (line 4 of Algorithm 3.3), the weight $w_t^{(i)}$ of the tested sample $\mathbf{s}_t^{(i)}$ is obtained from the mean square error between the association of points in ICP. In fact, $w_t^{(i)}$ is proportional to the inverse of the ICP error, because the weight of a sample must be higher, as smaller are the alignment errors in its map.

$$w_t^{(i)} \propto \frac{1}{E_{ICP}} = \frac{1}{\frac{1}{n}\sum_{i=1}^{n} ||\mathbf{p}_i - \mathbf{q}_i||^2} \tag{3.4.2}$$

Figure 3.10 shows a matching example in SDP-SLAM, emphasizing the submaps selection and the extraction of points from free-space. Figure (a) shows the region where the overlapping between submaps occurs. As in the previous experiment, the robot is returning to close a large loop, after exploring the environment. In (b), the submaps to be compared are highlighted in red and green. Then, points (in yellow) are selected from the center of the free-space from the two submaps, as shown in (c) and (d). In the figure, the closest to the walls a cell is located, the darker is the tone of its color representation. Finally, Figure (d) shows the association without ambiguities made between the two sets of points selected in the previous step. The points extracted from the submap in (c) are green, while the points extracted from the submap in (d) are red.

3.4.3 Topology Estimation and Map Reconstruction

In the top level process of SDP-SLAM, a particle filter is used to estimate good submaps combinations to build an environment map. The propagation of the particles is made by the particles transition model, which is a function of the prior set of particles and the set of all actions and observations made by the robot:

$$\gamma_{1:t} = q(\gamma_{1:t} \mid \gamma_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

This transition model uses the Probabilistic Graph of Segments (PGS), that contains weighted connections between submaps of adjacent segments. Thus, before explaining the particles propagation, it is necessary to formally present the PGS.

The Probabilistic Graph of Segments (PGS)

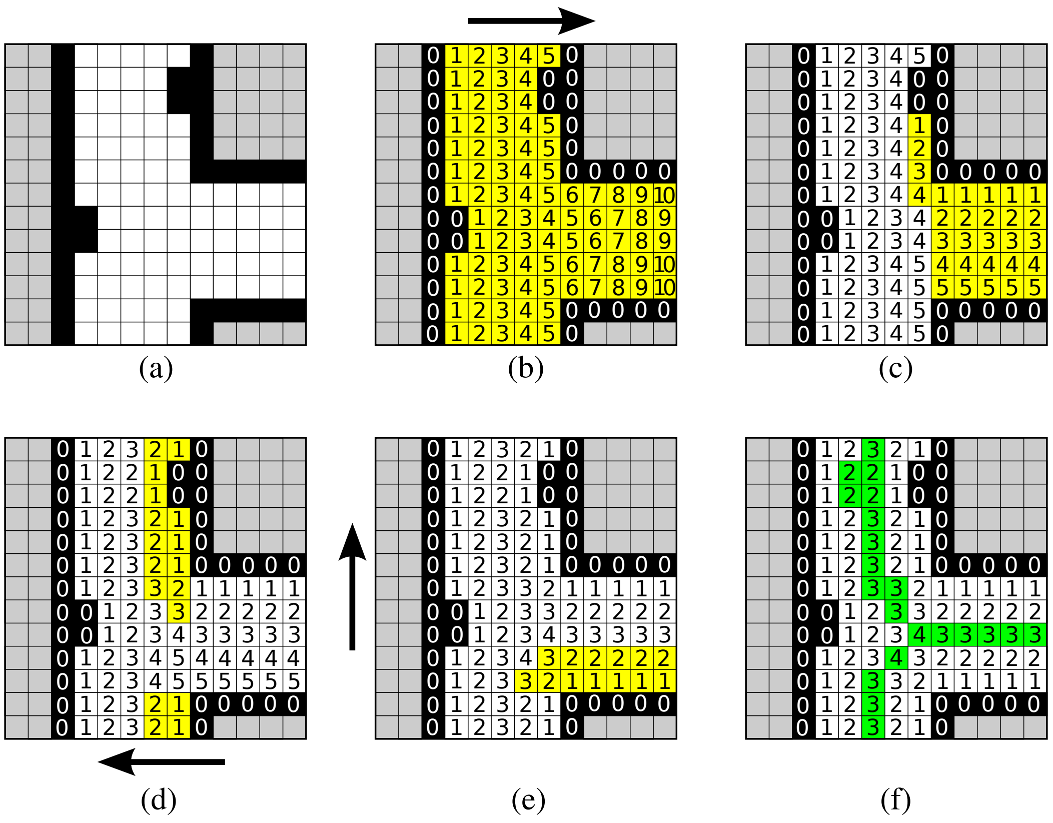


Figure 3.9: Extraction of points from free-space. (a) Submap occupancy grid. (b-e) Computing the distance from each cell to nearest obstacles in four passes (yellow cells are the updated cells of each step): (b) from left to right, (c) top down, (d) from right to left, (e) bottom up. (f) Resulting grid, highlighting the cells farther from obstacles.

The principle behind the PGS is to estimate the probabilities of pairs of adjacent submaps to compose good global solutions. For example, chains of submaps that are often present in good combinations will have high weights, and hence, a higher priority to be selected than chains of submaps from combinations with low weights. These probabilities estimations are made by a Monte Carlo process, based on the accumulation of matching results of numerous samples of submaps combinations.

PGS estimates the probability of choosing each submap from each segment, given that a specific submap was chosen in the prior segment. Despite the fact that there is p^n hypotheses of submaps combinations, where p is the number of submaps per segment and n is the number of segments, PGS estimates only $n \cdot p^2$ probabilities, since p^2 probabilities are computed for each pair of consecutive segments, as shown below

$$\text{Segments } 1,2 : \{p(\gamma_{2a}|\gamma_{1a}), p(\gamma_{2b}|\gamma_{1a}), \cdots, p(\gamma_{2p}|\gamma_{1a}), p(\gamma_{2a}|\gamma_{1b}), \cdots, p(\gamma_{2p}|\gamma_{1p})\}$$

$$\text{Segments } 2,3 : \{p(\gamma_{3a}|\gamma_{2a}), p(\gamma_{3b}|\gamma_{2a}), \cdots, p(\gamma_{3p}|\gamma_{2a}), p(\gamma_{3a}|\gamma_{2b}), \cdots, p(\gamma_{3p}|\gamma_{2p})\}$$

$$\vdots$$

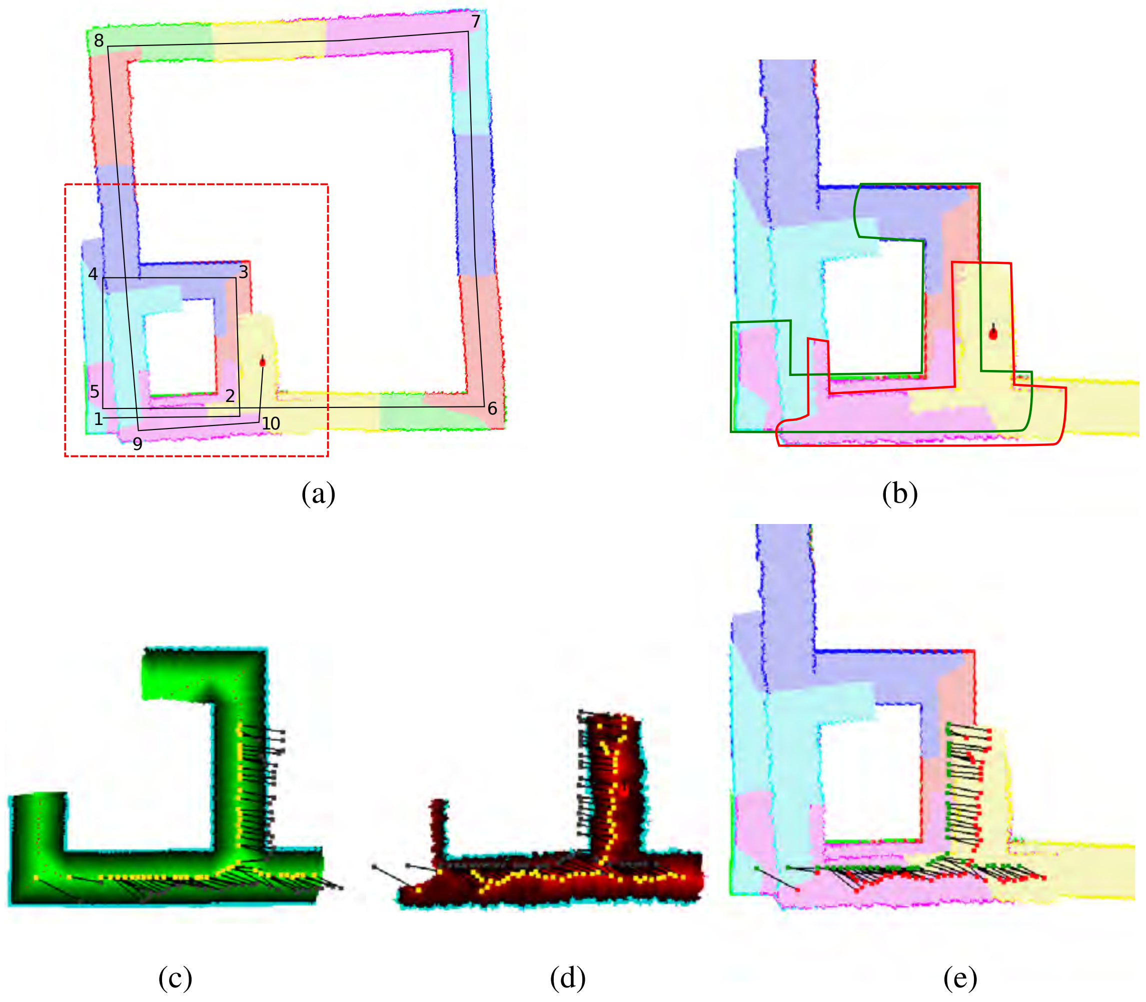


Figure 3.10: Matching of points made by SDP-SLAM. (a) Map showing a revisited region. (b) Submaps selected for the matching. (c-d) The two sets of selected points (in yellow). (d) The association of points made by ICP.

Segments $t-1, t$: $\{p(\gamma_{ta}|\gamma_{t-1,a}), p(\gamma_{tb}|\gamma_{t-1,a}), \cdots, p(\gamma_{tp}|\gamma_{t-1,a}), p(\gamma_{ta}|\gamma_{t-1,b}), \cdots, p(\gamma_{tp}|\gamma_{t-1,p})\}$

Algorithm 3.4: PGS maintenance process

```
1 Randomly generation of samples of submaps combinations.
2 Evaluation of the samples by the ICP error computed in the matching process.
3 Update of the accumulated error of each connection between adjacent
  submaps.
4 Computation of the quality of each connection.
5 Normalization of the quality of each connection.
6 Update of the conditional probabilities of each connection.
```

The PGS maintenance process is presented in Algorithm 3.4. Systematically, samples of combinations are randomly generated (line 1) and evaluated by the matching process using ICP (line 2). The result of the matching is a measure of the ICP error.

Next (line 3), the measured error of each sample is added to the accumulated errors of the connections between submaps that compose that sample. For example, the accumulated error E_{1b2a} of the connection between γ_{1b} and γ_{2a} is the sum of the errors from all sampled combinations having γ_{1b} and γ_{2a}

$$E_{1b2a} = \sum E_{ICP}(\gamma_{1b}, \gamma_{2a}, \gamma_{3i}, \gamma_{4j}, \cdots, \gamma_{tl}), \qquad \forall i, j, \cdots, l \tag{3.4.3}$$

The idea is that, over time, connections with low accumulated errors possess great chance to compose good solutions.

Yet, the quality of a connection between submaps is inversely proportional to its accumulated error. Thus, we compute how much of the total accumulated error is caused by each connection (line 4). Using the same example of the connection between γ_{1b} and γ_{2a}, we define the quality function f as

$$f_{1b2a} = \frac{E_{total}}{E_{1b2a}} \tag{3.4.4}$$

where E_{total} is the accumulated error of all sampled combinations.

Now (line 5), to compute the probability of each connection to compose a good solution, we must normalize f.

$$p(\gamma_{1b}, \gamma_{2a}) = \frac{f_{1b2a}}{\sum_{i,j=1}^{p} f_{1i2j}} \tag{3.4.5}$$

Finally (line 6), we compute the conditional probability of choosing γ_{2a} given that γ_{1b} was chosen

$$p(\gamma_{2a}|\gamma_{1b}) = \frac{p(\gamma_{1b}, \gamma_{2a})}{p(\gamma_{1b})} = \frac{p(\gamma_{1b}, \gamma_{2a})}{\sum_{j=1}^{p} p(\gamma_{1b}, \gamma_{2j})} = \frac{f_{1b2a}}{\sum_{j=1}^{p} f_{1b2j}} \tag{3.4.6}$$

Figure 3.11 shows an example of the probabilities computation in PGS. SDP-SLAM was performed for a certain time, and four segments were created with 2 submaps each. In (a), it is shown the accumulated errors (in meters) of every combination of adjacent

(1-2)	γ_{2a}	γ_{2b}	
γ_{1a} γ_{1b}	$E_{1a2a} = 378.0$ $E_{1b2a} = 209.4$	$E_{1a2b} = 432.3$ $E_{1b2b} = 134.2$	$\sum_i E_{1a2i} = 810.3$ $\sum_i E_{1b2i} = 343.6$
			$E_{total} = \sum_{i.j} E_{1i2j} = 1153.9$
(2-3)	γ_{3a}	γ_{3b}	
γ_{2a} γ_{2b}	$E_{2a3a} = 269.6$ $E_{2b3a} = 382.4$	$E_{2a3b} = 317.8$ $E_{2b3b} = 184.1$	$\sum_i E_{2a3i} = 587.4$ $\sum_i E_{2b3i} = 566.5$
			$E_{total} = \sum_{i.j} E_{2i3j} = 1153.9$
(3-4)	γ_{4a}	γ_{4b}	
γ_{3a} γ_{3b}	$E_{3a4a} = 357.7$ $E_{3b4a} = 286.5$	$E_{3a4b} = 294.3$ $E_{3b4b} = 215.4$	$\sum_i E_{3a4i} = 652.0$ $\sum_i E_{3b4i} = 501.9$
			$E_{total} = \sum_{i.j} E_{3i4j} = 1153.9$

(a) Accumulated error of each combination of adjacent submaps.

(1-2)	γ_{2a}	γ_{2b}	
γ_{1a} γ_{1b}	$f_{1a2a} = E_{total}/E_{1a2a} = 3.053$ $f_{1b2a} = E_{total}/E_{1b2a} = 5.510$	$f_{1a2b} = E_{total}/E_{1a2b} = 2.669$ $f_{1b2b} = E_{total}/E_{1b2b} = 8.600$	$\sum_i f_{1a2i} = 5.722$ $\sum_i f_{1b2i} = 14.110$
			$\sum_{i,j} f_{1i2j} = 19.832$
(2-3)	γ_{3a}	γ_{3b}	
γ_{2a} γ_{2b}	$f_{2a3a} = E_{total}/E_{2a3a} = 4.279$ $f_{2b3a} = E_{total}/E_{2b3a} = 3.018$	$f_{2a3b} = E_{total}/E_{2a3b} = 3.631$ $f_{2b3b} = E_{total}/E_{2b3b} = 6.269$	$\sum_i f_{2a3i} = 7.910$ $\sum_i f_{2b3i} = 9.286$
			$\sum_{i,j} f_{2i3j} = 17.197$
(3-4)	γ_{4a}	γ_{4b}	
γ_{3a} γ_{3b}	$f_{3a4a} = E_{total}/E_{3a4a} = 3.226$ $f_{3b4a} = E_{total}/E_{3b4a} = 4.028$	$f_{3a4b} = E_{total}/E_{3a4b} = 3.920$ $f_{3b4b} = E_{total}/E_{3b4b} = 5.358$	$\sum_i f_{3a4i} = 7.146$ $\sum_i f_{3b4i} = 9.385$
			$\sum_{i,j} f_{3i4j} = 16.532$

(b) Quality measure of each combination of adjacent submaps.

(1-2)	γ_{2a}	γ_{2b}	
γ_{1a} γ_{1b}	$p(1a, 2a) = f_{1a2a}/\sum_{i,j} f_{1i2j} = 0.154$ $p(1b, 2a) = f_{1b2a}/\sum_{i,j} f_{1i2j} = 0.278$	$p(1a, 2b) = f_{1a2b}/\sum_{i,j} f_{1i2j} = 0.135$ $p(1b, 2b) = f_{1b2b}/\sum_{i,j} f_{1i2j} = 0.434$	$p(1a) = 0.289$ $p(1b) = 0.711$
(2-3)	γ_{3a}	γ_{3b}	
γ_{2a} γ_{2b}	$p(2a, 3a) = f_{2a3a}/\sum_{i,j} f_{2i3j} = 0.249$ $p(2b, 3a) = f_{2b3a}/\sum_{i,j} f_{2i3j} = 0.175$	$p(2a, 3b) = f_{2a3b}/\sum_{i,j} f_{2i3j} = 0.211$ $p(2b, 3b) = f_{2b3b}/\sum_{i,j} f_{2i3j} = 0.365$	$p(2a) = 0.460$ $p(2b) = 0.540$
(3-4)	γ_{4a}	γ_{4b}	
γ_{3a} γ_{3b}	$p(3a, 4a) = f_{3a4a}/\sum_{i,j} f_{3i4j} = 0.195$ $p(3b, 4a) = f_{3b4a}/\sum_{i,j} f_{3i4j} = 0.244$	$p(3a, 4b) = f_{3a4b}/\sum_{i,j} f_{3i4j} = 0.237$ $p(3b, 4b) = f_{3b4b}/\sum_{i,j} f_{3i4j} = 0.324$	$p(3a) = 0.432$ $p(3b) = 0.568$

(c) Estimated probability of each combination of adjacent submaps to compose good solutions.

(1-2)	γ_{2a}	γ_{2b}
γ_{1a} γ_{1b}	$p(2a\|1a) = f_{1a2a}/\sum_i f_{1a2i} = 0.534$ $p(2a\|1b) = f_{1b2a}/\sum_i f_{1b2i} = 0.390$	$p(2b\|1a) = f_{1a2b}/\sum_i f_{1a2i} = 0.466$ $p(2b\|1b) = f_{1b2b}/\sum_i f_{1b2i} = 0.610$
(2-3)	γ_{3a}	γ_{3b}
γ_{2a} γ_{2b}	$p(3a\|2a) = f_{2a3a}/\sum_i f_{2a3i} = 0.541$ $p(3a\|2b) = f_{2b3a}/\sum_i f_{2b3i} = 0.325$	$p(3b\|2a) = f_{2a3b}/\sum_i f_{2a3i} = 0.459$ $p(3b\|2b) = f_{2b3b}/\sum_i f_{2b3i} = 0.675$
(3-4)	γ_{4a}	γ_{4b}
γ_{3a} γ_{3b}	$p(4a\|3a) = f_{3a4a}/\sum_i f_{3a4i} = 0.451$ $p(4a\|3b) = f_{3b4a}/\sum_i f_{3b4i} = 0.429$	$p(4b\|3a) = f_{3a4b}/\sum_i f_{3a4i} = 0.549$ $p(4b\|3b) = f_{3b4b}/\sum_i f_{3b4i} = 0.571$

(d) Conditional probability of choosing a submap, given the prior selected submap.

Figure 3.11: Example of the probabilities computation in PGS.

submaps, after the generation of numerous random samples. The quality measures of the submaps combinations are shown in (b), and the values after a normalization, in (c). The conditional probabilities associating adjacent submaps are presented in (d). In this example, we observe that the pair with the lowest accumulated error is $(\gamma_{1b}, \gamma_{2b})$, as shown in (a). Hence, this is the combination with the best probability to compose a good solution, as shown in (c). Also, Figure (a) shows that the error associated to the submap γ_{1b}, given by the sum of E_{1b2a} and E_{1b2b}, is much smaller than the error associated to the submap γ_{1a}. This fact implies that the probability of a combination of submaps containing γ_{1b} to build a good map is much higher than the probability of combination containing γ_{1a}. Finally, by the table of conditional probabilities, shown in (d), we observe that after selecting the submap γ_{1b}, the probability to choose γ_{2b} is higher than the probability to choose γ_{2a}. However, this is not the largest difference between conditional probabilities, which occurs when we must choose between γ_{3a} and γ_{3b}, after selecting γ_{2b}.

An example of the progression of the PGS is shown in Figure 3.12. The graph has three segments each one with three submaps. The rows of the figure represent three updates of the graph. The columns of the figure represent, from left to right: the PGS current state, where the width of the edges is associated to the weight of the submaps connections; the set of weighted samples that will be used to update PGS; the edges of PGS that must be altered at each update; and the update of the connections weights. At first, (a), all the connections in PGS have the same weight. Then, (b), four samples of submaps combinations are generated. The values inside the ellipses correspond to the submaps selected by the samples, and the weight of each sample is depicted in the bar graph. In (c), the connections that compose these four samples are highlighted in the graph. In (d), the width of each connection is increased according to the weight of the sample that it belongs. The resulting graph after the first round of updates is shown in (e).

The second and third rounds of updates, respectively shown in (e-h) and (i-l), are performed the same way as the first round. The final graph after the three rounds of updates is shown in (l). It is possible to see that the PGS starts to show some important characteristics. The connection between the submaps $2A$ and $3B$ seems promising, because more than one time it was a component of samples with high weights (this connection is present in the first sample of Figure (b) and in the first sample of Figure (f)). On the other hand, the connection between the submaps $2B$ and $3C$ seems to be a poor contributor to combinations samples, since, it was present twice in samples with low weights (this connection is present in the third sample of Figure (b) and in the second sample of Figure (f)).

In the example in Figure 3.12, only three iterations were performed (to facilitate the understanding of the process), thus every new sample made a great impact on the weights distribution of PGS. Nevertheless, in a long term operation, the impact of new samples is much smaller. To contour this, we force the weights of the graph to decline slowly with time, by multiplying, at each iteration, a decay rate κ of 0.99. Given that a sample s contributed with an initial error $E_0^{(s)}$ to the PGS accumulated error, after t iterations the sample contribution $E_t^{(s)}$ is

$$E_t^{(s)} = \kappa^t * E_0^{(s)} = 0.99^t * E_0^{(s)} \tag{3.4.7}$$

For instance, using this value of κ, after 10, 50, 100, and 500 steps the contribution of a sample to PGS is respectively decreased in 9.5%, 39.5%, 63.4%, and 99.3%.

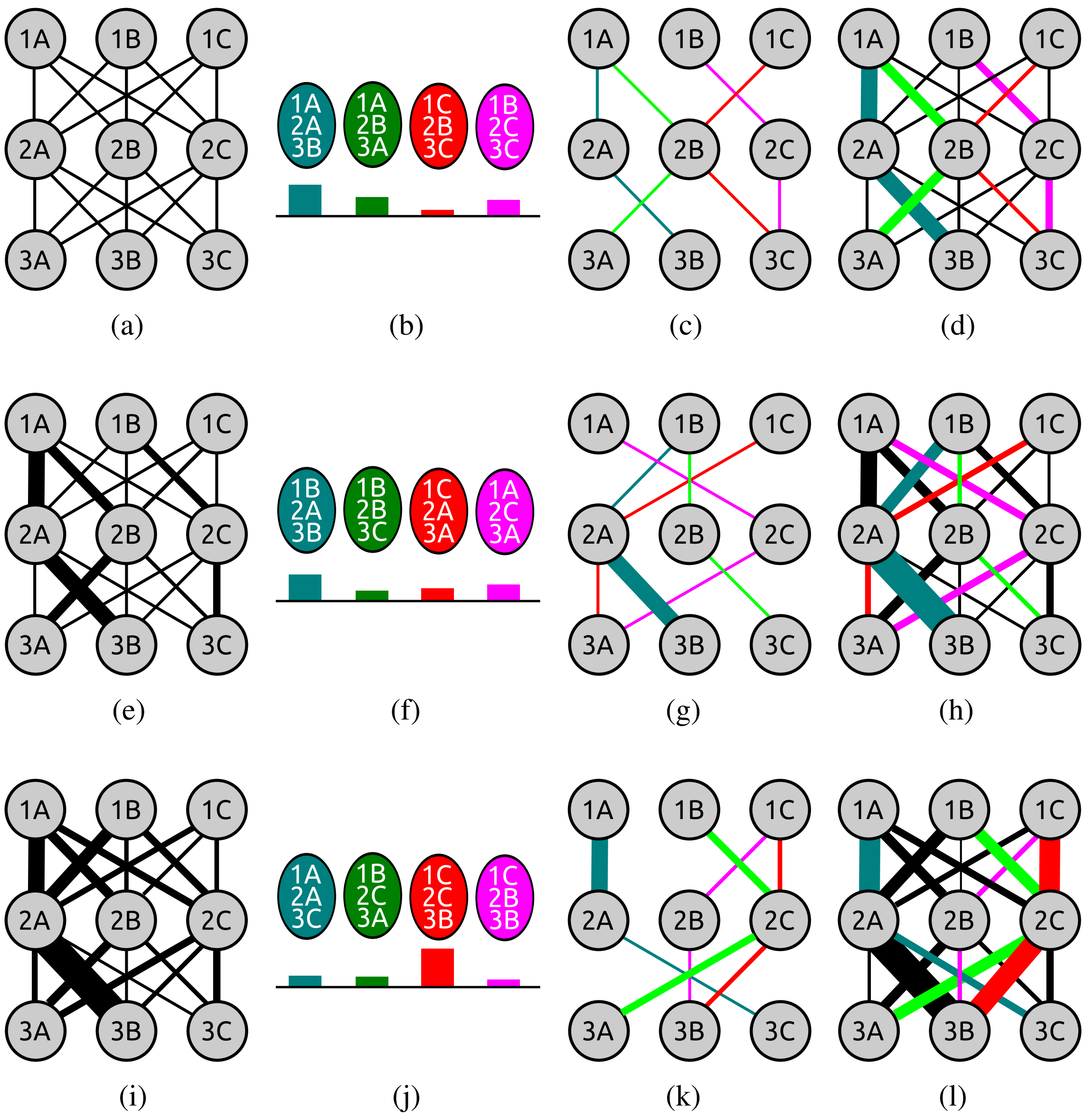


Figure 3.12: Example of the PGS progression. The three rows represent three update iterations. The four columns represent the momentary PGS state; the sets of samples used to update PGS; the edges to be altered at each update; and the updated PGS after each iteration.

SDP-SLAM top level particles propagation

After presenting the Probabilistic Graph of Segments, we can define the SDP-SLAM top level particles propagation. The Algorithm 3.5 describes how this process works. First (line 1), the set of particles is sorted by the particles weights and divided into three sets, with sizes N_{bp}, N_{mp} and N_{wp}. The first set is a small set of particles with the best current combinations. We defined the size N_{bp} of this set to be $1/10$ of the total of particles. These particles are maintained unchanged (line 2), to avoid the sudden loss of good solutions that can occur in a resampling.

Algorithm 3.5: SDP-SLAM top level particles propagation

```
1 Division of the particles in three sets of sizes $N_{bp}$, $N_{mp}$ and $N_{wp}$, according to
  their weights.
2 Maintenance of the set of $N_{bp}$ best particles.
3 Replacement of the set of $N_{wp}$ worst particles, through the resampling of
  particles.
4 Update of the remaining set of $N_{mp}$ particles, based on the PGS.
```

On the other hand, the set of the worst particles is discarded from the filter. The size N_{wp} of this set corresponds to $3/10$ of the total of particles. To replace these particles, the SDP-SLAM performs a resampling step, where particles with higher weights have more chance to be replicated than particles with lower weights.

The last set of particles, with size N_{mp} corresponding to $6/10$ of the total, is composed of those particles that are not the best and nor the worst ones. These remaining particles undergo a mutation process. Actually, the initial portion of each particle chain of submaps is maintained, whereas the end portion is discarded. In place of the discarded portion, a new chain of submaps is sampled from PGS. The point to partition the submaps chain of each particle is based on the particle weight. The best particles of this set will preserve most of their submaps chains unchanged, while the worst particles will alter the majority of their chains. After sorting the particles in a list according to their weights, in ascending order, we can use the index $idx^{(i)}$ of each particle to compute the partitioning point $pp^{(i)}$ of each particle submaps chain:

$$pp^{(i)} = idx^{(i)} \frac{N_{mp}}{n_s} \tag{3.4.8}$$

where n_s is the number of segments.

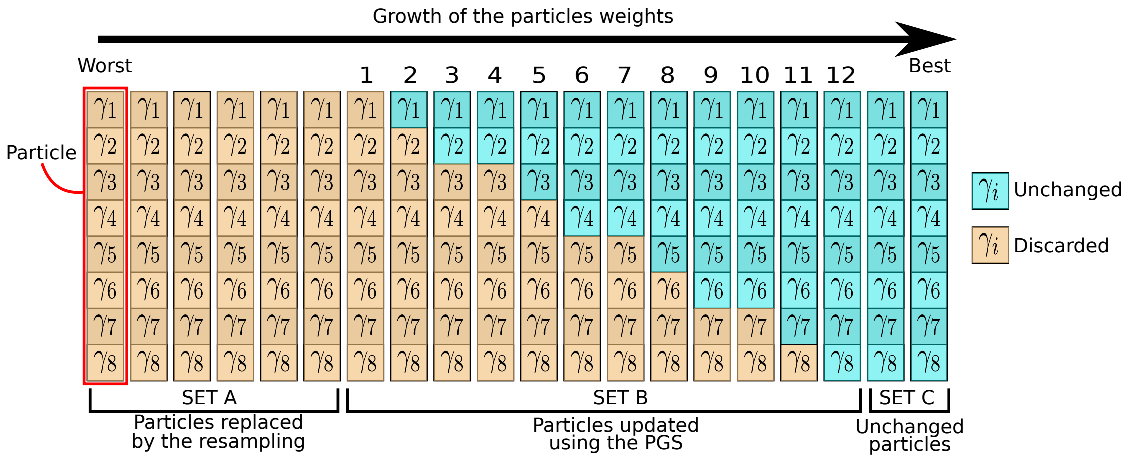

Growth of the particles weights
Worst
Best
1 2 3 4 5 6 7 8 9 10 11 12
Particle
γ_1 γ_2 γ_3 γ_4 γ_5 γ_6 γ_7 γ_8
γ_i Unchanged
γ_i Discarded
SET A
Particles replaced by the resampling
SET B
Particles updated using the PGS
SET C
Unchanged particles

Figure 3.13: Example of the particles propagation.

An example of the particles propagation is shown in Figure 3.13, using 20 particles to estimate combinations of eight submaps. The particles are sorted by their weights from left to right. The first six particles (Set A) are the worst ones and they will be discarded to perform the resampling. The last two particles (Set C) are the best ones and they will be kept unchanged. The particles from the middle set (numerated from 1 to 12) are the particles that will be updated using the PGS. In this example, $pp^{(i)} = idx^{(i)} \cdot 12/8 = idx^{(i)} \cdot 1.5$.

Knowing the partitioning point of the particle submaps chain, we compute the particle update according to the Algorithm 3.6. First (line 1), we set the start of the estimation process to the segment following the partitioning point $pp^{(i)}$. If $pp^{(i)}$ is zero, the process must start the sampling by the first segment. Thus (line 2), γ_1 is sampled from the set of probabilities $\{p(\gamma_{1a}), p(\gamma_{1b}), \cdots, p(\gamma_{1p})\}$. After this the method passes to the choice of the second submap (line 3). When a particle has a submaps chain with at least one submap, the sampling of the following submaps is made from the distribution of conditional probabilities. Therefore, each remaining submap is sampled incrementally, based on the choice of the prior submap (line 4).

Algorithm 3.6: Sampling from PGS

```
1 start = $pp^{(i)} + 1$
  if start = 1 then
2     Sample $\gamma_1$ from the probabilities set of $\{p(\gamma_{1a}), p(\gamma_{1b}), \cdots, p(\gamma_{1p})\}$
3     start = start+1
  end
  for i from start to n do
4     Sample $\gamma_i$ from the probabilities set of
      $\{p(\gamma_{ia}|\gamma_{i-1}), p(\gamma_{ib}|\gamma_{i-1}), \cdots, p(\gamma_{ip}|\gamma_{i-1})\}$
  end
```

The next section presents the experiments performed with the SDP-SLAM, justifying the choices made in the implementation of the method and showing comparisons with other methods.

4 EXPERIMENTS

In this chapter, we present some results of experiments in simulated and real environments obtained with a Pioneer 3-DX robot. This robot, developed by Adept Mobilerobots, is assembled with embedded controller, motors with 500-tick encoders, $19cm$ wheels, and 8 forward-facing ultrasonic (sonar) sensors. Our robot is also equipped with a SICK LMS-200 laser rangefinder, with resolution of $10mm$, scanning range of $80m$, field of view of 180°, and angular resolution from 0.25° up to 1°. The robot embedded controller performs velocity control and provides information about the robot state and the range sensing data, which is used by our client program that runs remotely. The robot is shown in Figure 4.1.

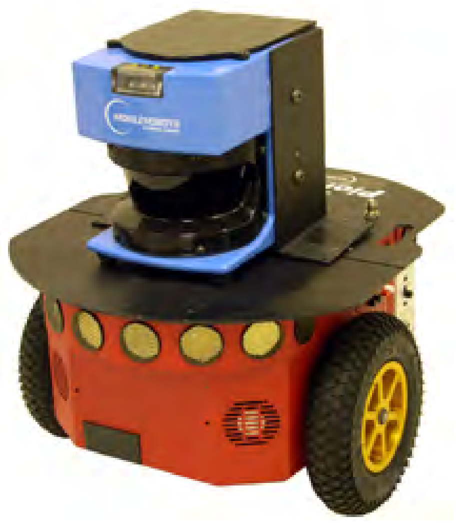


Figure 4.1: Robot used in the experiments: a Pioneer 3-DX equipped with a SICK LMS-200 laser rangefinder.

All the experiments are performed in real and simulated indoor environments of different sizes. The simulated experiments were performed using the MobileSim software from MobileRobots. Since we do not have real large size environments, we preferred to simulate them by constraining the laser range to $3m$. The maps acquired by the robot have grid cell with size of $5cm \times 5cm$ and are selected from the best particle (sample) remaining after the full process.

4.1 Evaluation of the Modified DP-SLAM

We start with the evaluation of our initial approach: the Modified DP-SLAM (MDP-SLAM). Since MDP-SLAM was originally planned as a component from an integrated exploration strategy, as seen in Section 3.1, we show some results obtained with this strategy. The simulated environment where the experiments were performed is shown in Figure 4.2. The size of the environment is $23m \times 10m$.

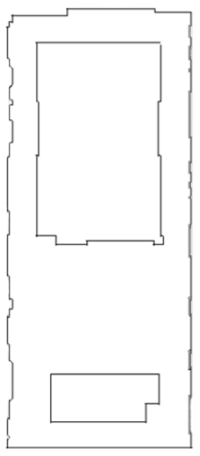
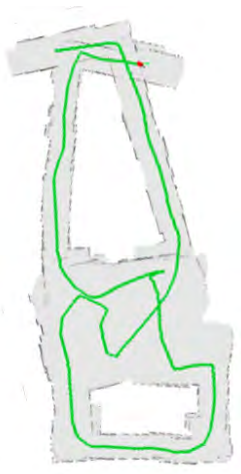


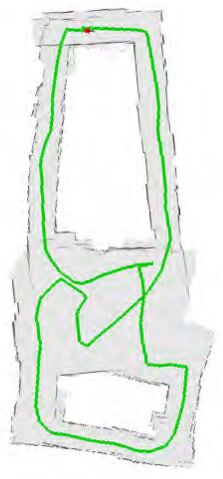
Figure 4.2: Ground truth of the environment mapped in the experiments of MDP-SLAM.

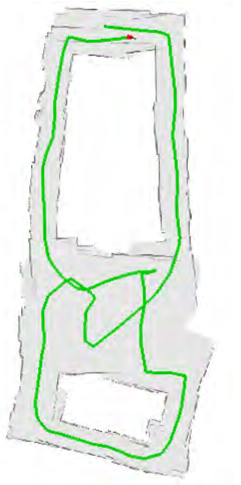
Three different strategies were compared: a greedy exploration approach using the original DP-SLAM (G-DP); a greedy exploration approach using the modified DP-SLAM (G-MDP), and our integrated exploration approach based on BVP and MDP-SLAM (BVP-MDP). This integrated exploration approach guides the robot to unexplored regions near to walls (where the localizability is high), and promotes revisits to known regions aiming to reduce the uncertainty of the particle filter.

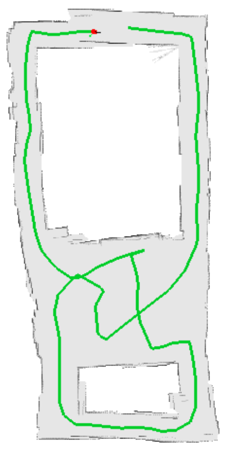
Fig. 4.3 presents the results of experiments using 400, 750 and 1000 particles. Maps (a), (d) and (g) were created using 400 particles. In (a), we observe that the greedy exploration associated with DP-SLAM did not produce good results. This happened because the particle filter converged quickly to hypotheses that were not globally good. In (d), we can see that with MDP-SLAM the result is better than the previous one, however, the map is not good yet. In (e), using our integrated exploration strategy we finally obtained a good result, because the exploration guided the robot through areas that reduced the uncertainty, facilitating the SLAM process. When we increase the number of particles, all maps improve as we can see in (b), (e) and (h), using 750 particles, and in (c), (f) and (i), using 1000 particles. Despite of the general improvement, the same observations made on the 400 particles experiment apply to the others.

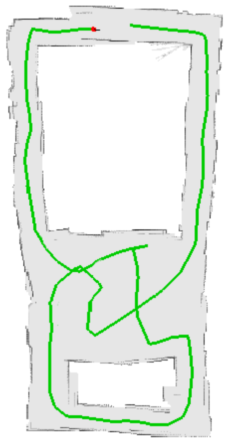
Analyzing the results, we observe that the modified resampling indeed improves DP-SLAM. Nonetheless, with only this modification, the number of particles required to obtain good solutions continues high. For example, even using 1000 particles the map is not good enough, as shown in (f). The results are better when using a good navigation strategy, like in BVP-MDP. Yet, even 400 particles is still a high number, and this number augment in large environments. Hence, this served as motivation for adopting a different approach, such as SDP-SLAM, which will be evaluated next.

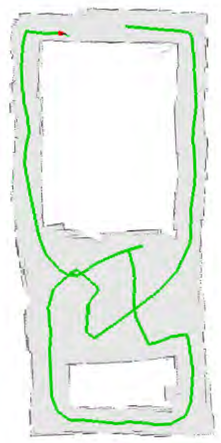

(a) G-DP 400p

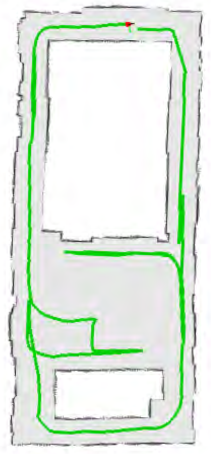

(b) G-DP 750p

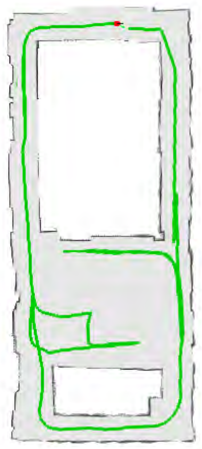

(c) G-DP 1000p


(d) G-MDP 400p


(e) G-MDP 750p


(f) G-MDP 1000p


(g) BVP-MDP 400p


(h) BVP-MDP 750p

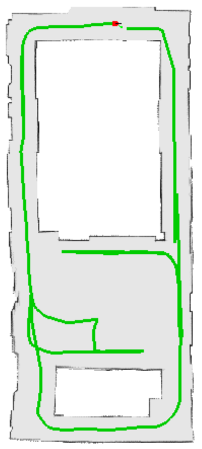
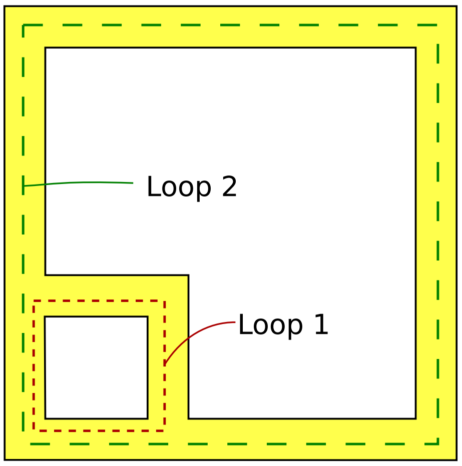

(i) BVP-MDP 1000p

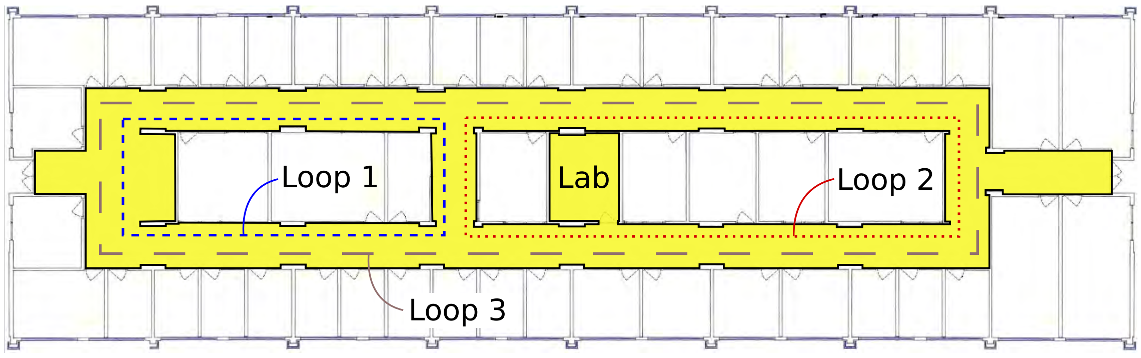
Figure 4.3: Experiments results of our first approach, using 400, 750 and 1000 particles. (a-c) Greedy exploration using DP-SLAM, (d-f) Greedy exploration using MDP-SLAM, (g-i) Our integrated exploration strategy.

4.2 Evaluation of SDP-SLAM

The evaluation of SDP-SLAM was made through experiments in simulated and real environments, that are illustrated in Figure 4.4. A simulated environment containing two loops is shown in (a). The inner loop (loop 1) and the outer loop (loop 2) have lenghts of $28m$ and $80m$, respectively. Figure 4.4-(b) shows a real environment containing three loops, corresponding to corridors of a building from the UFRGS Institute of Informatics. The two inner loops have lenghts of $43m$ (loop 1) and $57m$ (loop 2), and together form a larger loop of $88m$ (loop 3).



(a) Simulated environment



(b) Real environment

Figure 4.4: Ground truth of the environments mapped in the experiments with SDP-SLAM.

We chose these environments because they contain nested loops, that aggravate the particle depletion problem. For instance, during the mapping of an inner loop, a RBPF strategy discard particles that do not have the highest weights, but that can be needed later to map an outer loop.

Since the interest of this work is in SLAM, all experiments performed in the same environment were tested with the same robot trajectory. In both simulated and real environments the trajectories were manually obtained.

The robot trajectory in the simulated environment is presented in Figure 4.5. The robot starts in the lower left corner (position 1) and contours the inner loop (loop 1) in a counterclockwise direction (positions 1-5). Then, the robot contours the outer loop (loop 2) in the same direction (positions 5-10), until return to its initial position.

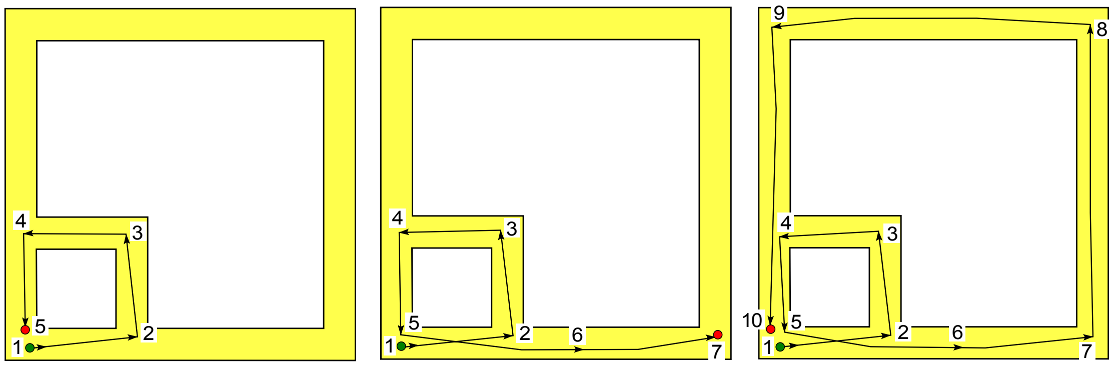


Figure 4.5: Robot trajectory in the simulated environment.

The robot trajectory in the real environment is presented in Figure 4.6. The robot leaves the Lab (position 1), and contours the larger inner loop (loop 2) in counterclockwise direction (positions 1-13). During its journey, the robot takes a few detours to visit unknown regions (positions 5-6 and 11-12). Then, the robot contours the outer loop (loop 3), in the same direction (positions 13-22), until return to the Lab (position 23).

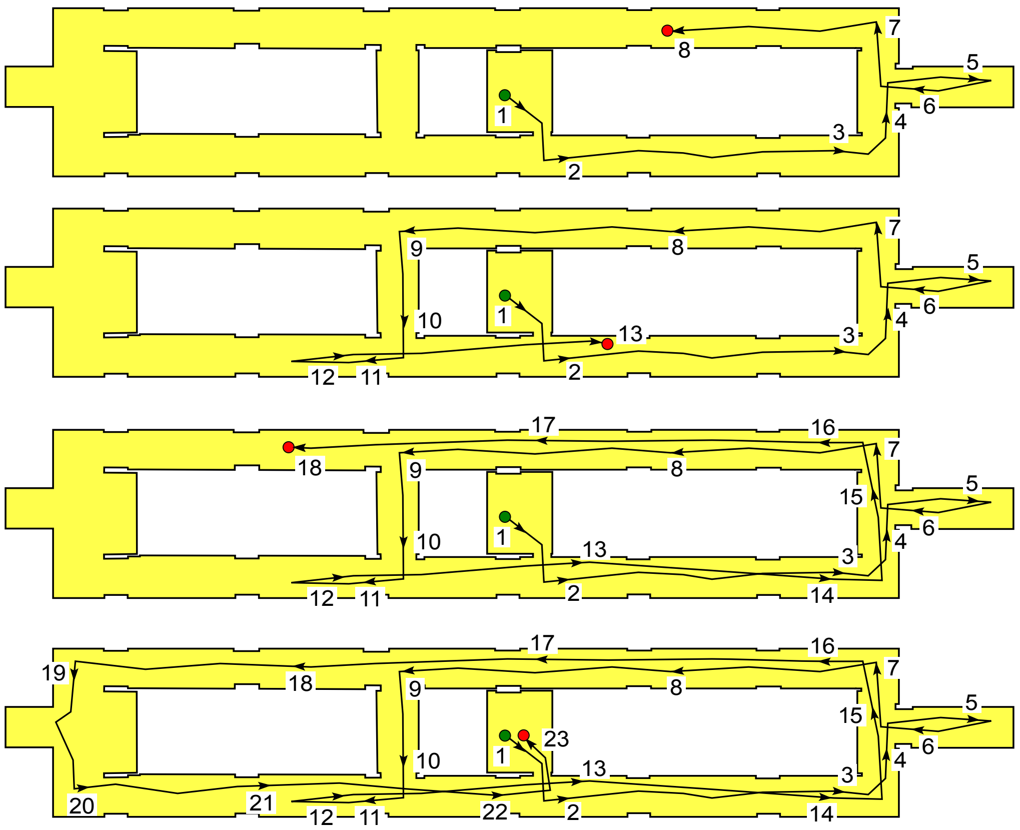


Figure 4.6: Robot trajectory in the real environment.

SDP-SLAM was evaluated through comparisons with other methods and the separated analysis of the three fundamental parts of the algorithm: the segmentation, the matching and the topology estimation.

First, we compare our proposal to DP-SLAM, presented in Section 2.4.4, and to the modified DP-SLAM, presented in Section 3.1. This is the algorithm that runs in the lower level of SDP-SLAM. In fact, while a segmentation does not occur, our method behaves as the modified DP-SLAM. Our motivation in these tests is to check if the segmentation process leads to a better performance of the method.

Later, we compare SDP-SLAM to SegSLAM, which introduced the idea of segments combinations. However, in our comparisons we did not use the exact SegSLAM implementation because, while the SDP-SLAM uses modified 2D occupancy grids, SegSLAM represents 3D environments using octrees, whose implementation would be an apart issue. Therefore, we compared the results obtained maintaining the same environment representation, but varying characteristics of the algorithm. For example, we compared our matching process to the SegSLAM matching process, and our topology estimation to the SegSLAM topology estimation. But we do not compare our segmentation process to the SegSLAM segmentation process. In SegSLAM, the segmentation is defined by a predictive score of how well the current map matches with the future observations. Since these future observations are, in fact, present observations, the method must run a few seconds in the past. Thus, we prefer to experiment SDP-SLAM with two simpler types of segmentation, one based on the number of steps of the filter and other based on the measure of particles dispersion.

4.2.1 Comparisons with a traditional RBPF SLAM

Before comparing our method to the DP-SLAM algorithm, we need to define how this comparison can be made. It is not fair to compare the SDP-SLAM to DP-SLAM using the same number of particles in both methods. DP-SLAM requires hundreds of particles to obtain good results, while our method does not. Moreover, it is impractical to use so many particles in SDP-SLAM, because the method is far more complex than DP-SLAM, and therefore, much slower.

To define which configurations of SDP-SLAM and DP-SLAM are equivalent in terms of time cost, we performed multiple instances of both methods varying the number of particles used. Then, we computed the average iteration runtime for each configuration, as shown in Figure 4.7. The runtime measurements were made at three distinct stages: upon the completion of the traditional SLAM steps (sampling, weighting, and resampling), after the estimation of the best topology to be used (step present only in SDP-SLAM), and after the map reconstruction, consulting the ancestry tree.

DP-SLAM was tested using 5, 10, 20, 100, 200 and 400 particles. SDP-SLAM was tested using 5 and 10 particles at the bottom level, and 5, 30 and 60 particles at the top level. The number of particles of the SDP-SLAM bottom level must be small, because it indicates how many submaps exist in each segment. If this number is large, the amount of possible submaps combinations becomes excessively high. There are no such restrictions regarding the SDP-SLAM top level, the only problem is that the method slows down using a high number of particles.

All the results in Figure 4.7 were obtained through experiments in the real environment. The solid bars represent the runtime mean between each iteration of the method, while the error bars represent the runtime standard deviation. In SDP-SLAM, the standard deviation is much larger than in DP-SLAM, because the processing of the method

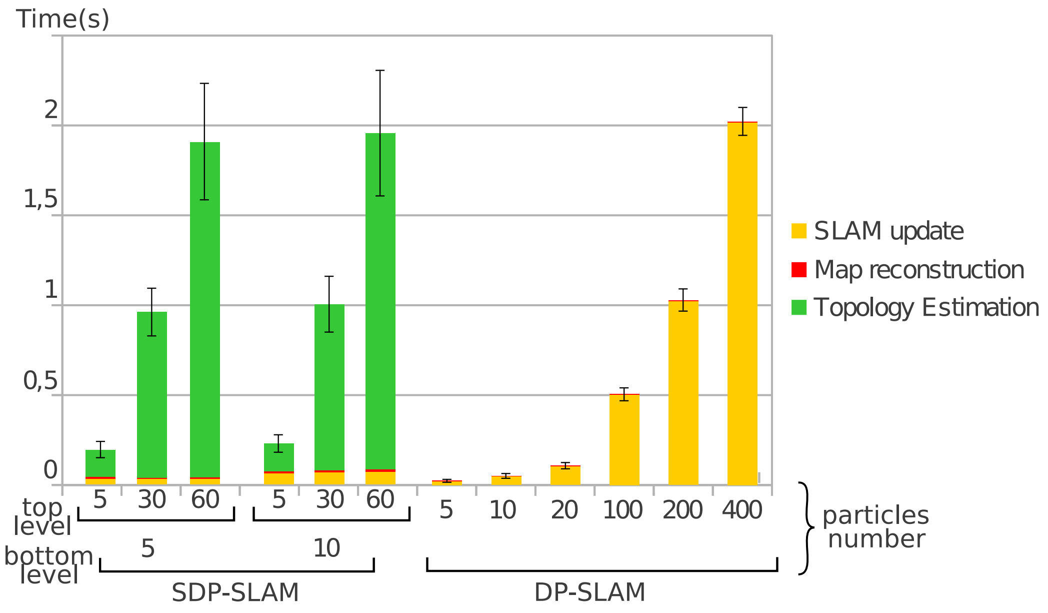


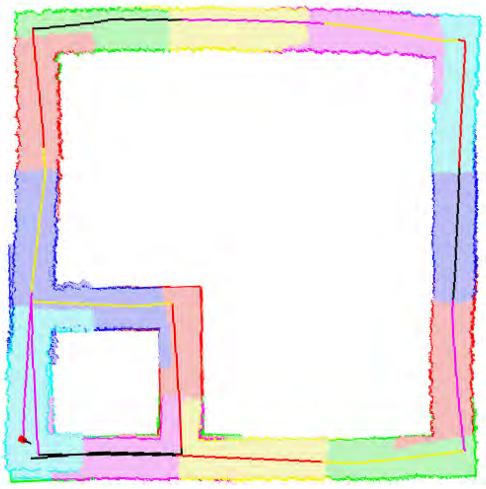
Figure 4.7: Runtime comparisons between SDP-SLAM and DP-SLAM, varying the number of particles used.

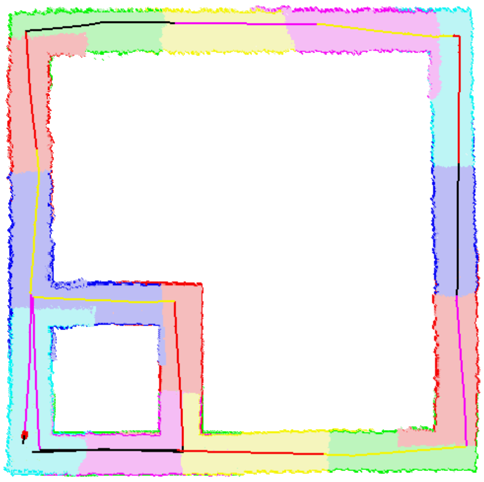
varies widely according to the robot position. For instance, the submaps matching technique, that is very costly, is performed only when submaps are overlapping. In contrast, the processing of DP-SLAM never changes.

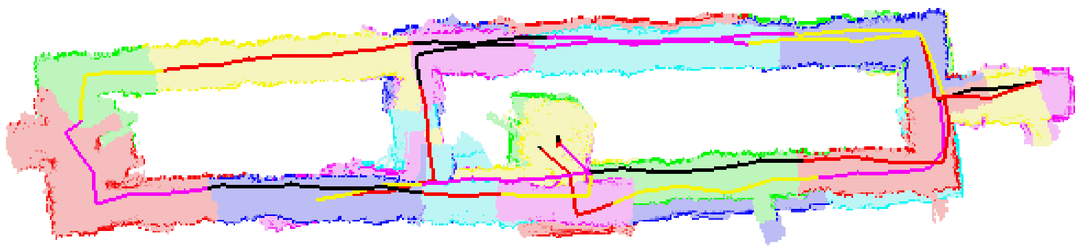
We observe that the SLAM update time is linear in the number of particles used by DP-SLAM or by the lower level of SDP-SLAM. In our tests, this time was approximately 1 second using 200 particles, and 2 seconds using 400 particles. The map reconstruction time is practically negligible and independent of the number of particles. In SDP-SLAM, the map reconstruction time is slightly larger than in DP-SLAM, because, besides the queries on the ancestry tree, the graph of segments has to be consulted and the chains of transformations must be applied to each cell of the map. The time to estimate the combination of segments is the main difference between the two methods, since it depends on the number of particles used at the top level, that only exists in SDP-SLAM. Using 30 particles costs approximately 1 second, whereas using 60 particles costs 2 seconds. This linearity is expected, because using the twice of particles in the top level, the tendency is to occur the twice of the submaps matching, which is the slowest part of the SDP-SLAM.

Now, we present some comparisons between resulting maps of the experiments with SDP-SLAM and DP-SLAM. We selected configurations that run in similar times (some in nearly one second, and others in nearly two seconds). For DP-SLAM, the parameters chosen were 200 and 400 particles, and, for SDP-SLAM, the parameters chosen were 30/5 particles (30 at the top level and 5 at the bottom level) and 60/10 particles (60 at the top level and 10 at the bottom level). We also decided to compare SDP-SLAM to MDP-SLAM. The configurations selected for MDP-SLAM were the same configurations of DP-SLAM, since there are just a few differences between both algorithms.

The resulting maps obtained using SDP-SLAM are shown in Figure 4.8. We prefer to separately show the maps built by SDP-SLAM to analyze the division of submaps. Later, these same maps will be compared in grayscale with the DP-SLAM results, where we can perceive more details. The different colored regions in each map represent different submaps, while the colored lines represent the robot path. In the simulated environment, the distance traversed by the robot has approximately $110m$, and using a periodic segmentation performed at every 30 iterations of the process, we generated maps with 16 segments, shown in (a) and (b). In turn, the distance traversed in the real environment has approximately $160m$, and, using the same segmentation process, we generated maps with 21 segments, shown in (c) and (d). Visually, the number of segments may seem smaller, but this happens because some of them are overlapping.


(a) Simulated - 30/5 particles


(b) Simulated - 60/10 particles


(c) Real - 30/5 particles

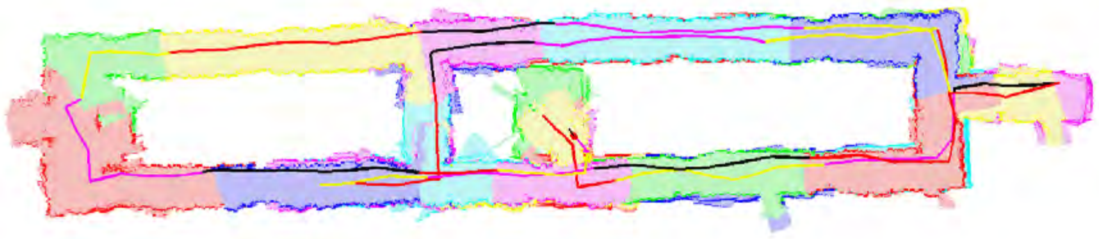
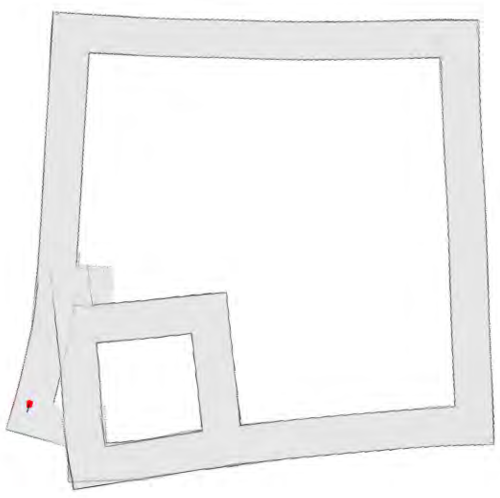
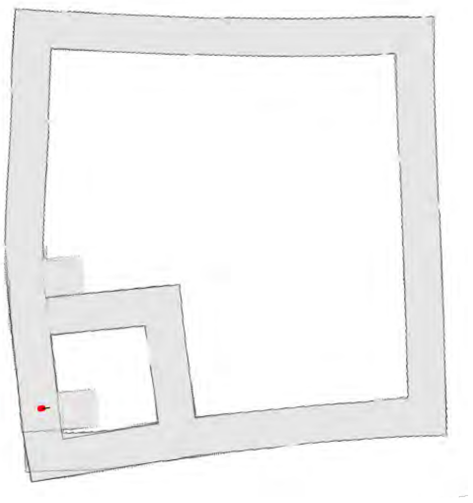

(d) Real - 60/10 particles

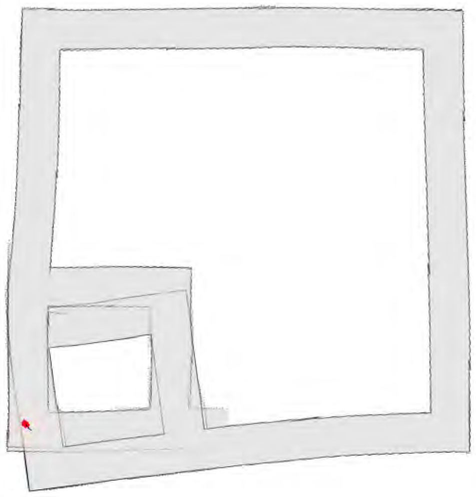
Figure 4.8: Results of the experiments with SDP-SLAM, showing the paths traversed by the robot and the segmented maps.

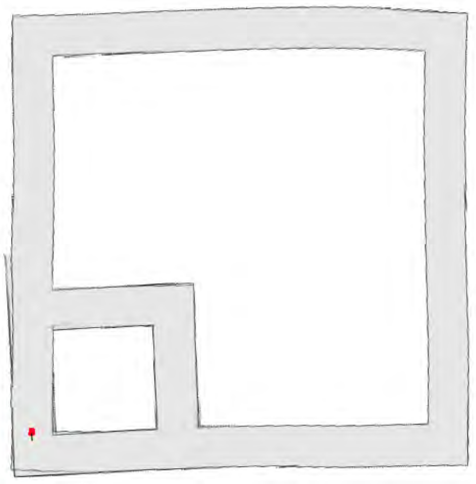
The comparisons between SDP-SLAM, DP-SLAM and MDP-SLAM are presented in two steps. First, we analyze the experiments in simulated environment, and later, the experiments in real environment.

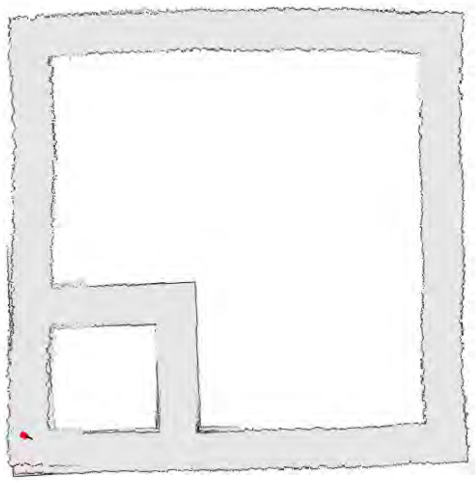
Figure 4.9 shows the resulting maps of the experiments in simulated environment, where the robot is the red point, the obstacles are black and the free-space is gray.


(a) DP-SLAM - 200 particles


(b) DP-SLAM - 400 particles


(c) MDP-SLAM - 200 particles


(d) MDP-SLAM - 400 particles


(e) SDP-SLAM - 30/5 particles

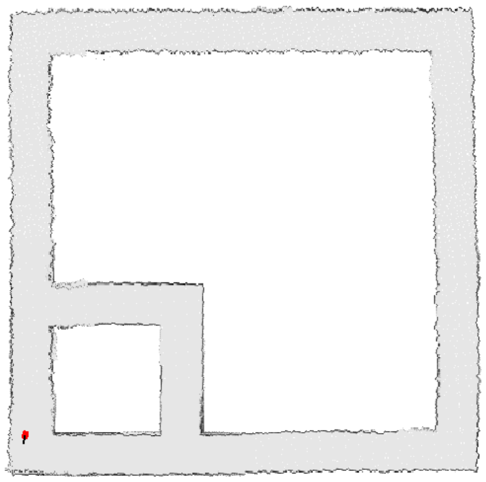

(f) SDP-SLAM - 60/10 particles

Figure 4.9: Comparison between DP-SLAM, MDP-SLAM and SDP-SLAM through experiments in the simulated environment.

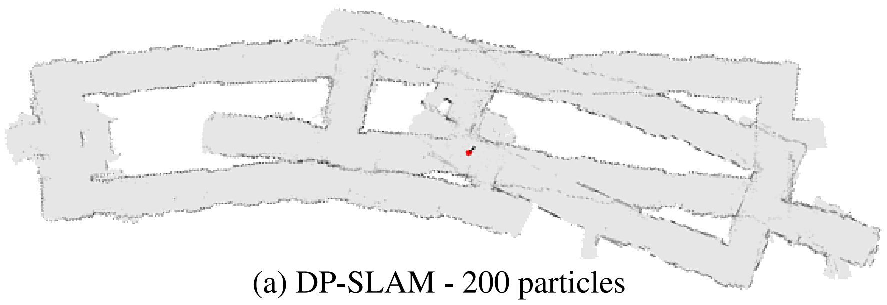
In Figure 4.9-(a-b), we present the maps built with DP-SLAM using 200 and 400 particles, respectively. We can see that, with these number of particles, the method did not properly close the larger loop. In (c-d), we present the maps built with MDP-SLAM using 200 and 400 particles. With 400 particles, MDP-SLAM almost succeeded to properly close the loop, but the resulting map is slightly tilted and there are some wrong overlaps at the end of the path. In (e), we show the map constructed with our method, using 5 particles on the bottom level to built submaps and 30 particles on the top level to estimate combinations of submaps. The resulting map is very similar to the map built by MDP-SLAM using 400 particles, since it presents some inconsistencies. A possible explanation is that, by using only 5 particles, some sections of the map did not have good local solutions. Perhaps, with this number of particles at the bottom level, the size of the submaps should be smaller. Finally, in (d), it is used the twice of particles at both lower and upper level, and the resulting map was better. The alignment error is smaller and the corridors of the environment are visually more straighter.

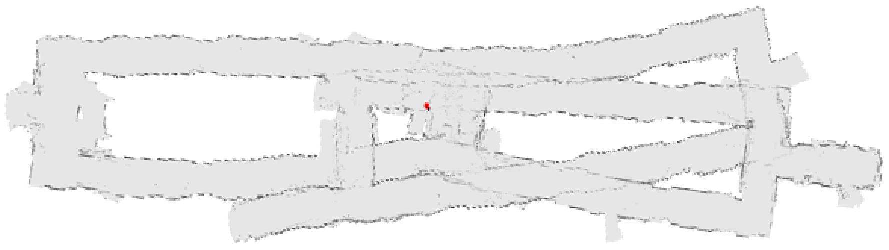
The results in the real environment are shown in Figure 4.10. In this environment the path traversed by the robot was larger than in the simulated one, and the methods ended up having more difficulties. The maps created by DP-SLAM using 200 and 400 particles are shown in (a-b). The results are very poor, since DP-SLAM was not able to close any loop. The same happens with MDP-SLAM using 200 particles, as shown in (c). Using 400 particles, as shown in (d), MDP-SLAM was barely able to close the minor loop (loop 1), but not the larger loop (loop 2). In (e), we show the result of the SDP-SLAM running with 5 particles at the bottom level and 30, at the top level. The resulting map is better than the map built by MDP-SLAM using 400 particles, since there was no alignment error visually as large as in that map. Again, the best result was with SDP-SLAM using 10 particles at the lower level and 60 at the upper level. The final map, shown in (d), still has some misalignments, especially in the right corner of the larger loop, but the loop was closed.

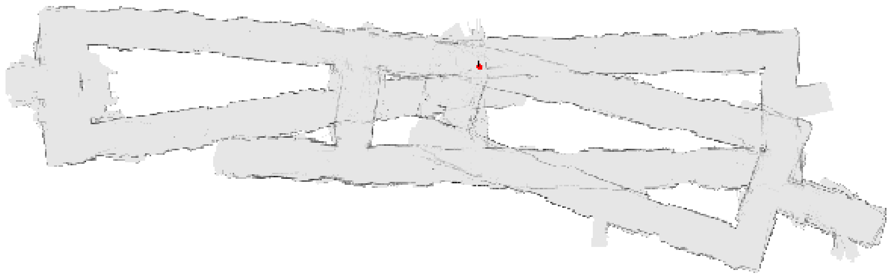
4.2.2 Analyzing the topology estimation

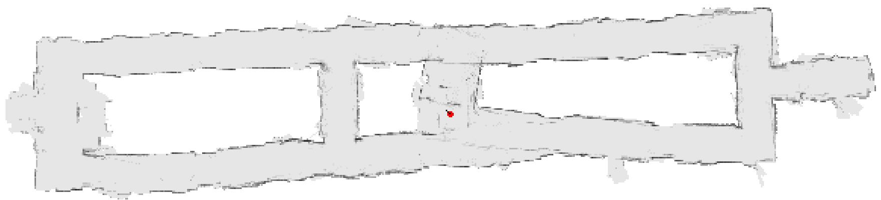
The evaluation of the SDP-SLAM topology estimation process was made through the comparison with the process used by the SegSLAM algorithm. In SegSLAM, the estimation of the topological map is made in a simplistic form: samples of possible submaps combinations are randomly generated, evaluated, and inserted into a priority queue, from where the particles select the best combinations. According to Fairfield et al. (FAIRFIELD; WETTERGREEN; KANTOR, 2010), this sampling process is used to generate combinations just from the last few segments visited by the robot. Therefore, only local consistency is guaranteed, because the alignment of submaps is made on a small portion of the environment. However, we are interested in obtaining globally consistent solutions, so, we need to sample combinations of submaps from all segments. This global consistency aims to building trustful environment representations that contain loops.

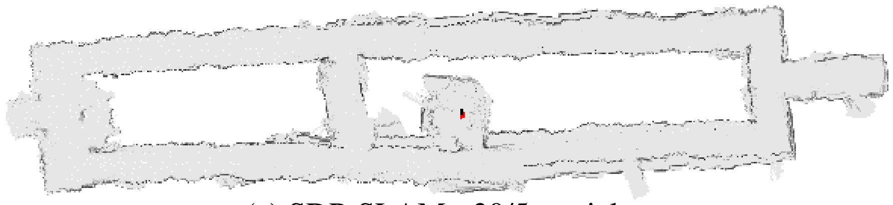
To perform the comparisons between the two strategies, we applied the same tests conducted earlier to SDP-SLAM to the method using the SegSLAM topology estimation. The environment representation, and the processes of segmentation and submaps matching were kept unchanged, thus the results are associated only to the differences between the topology estimations. The two methods were tested with the same configurations, for instance, if SDP-SLAM uses 30/5 particles (30 particles at the top level and 5 at the bottom level), SegSLAM will use 5 particles to build submaps and 30 samples to generate submaps combinations. Thus, both methods perform the same number of submaps


(a) DP-SLAM - 200 particles


(b) DP-SLAM - 400 particles


(c) MDP-SLAM - 200 particles


(d) MDP-SLAM - 400 particles


(e) SDP-SLAM - 30/5 particles

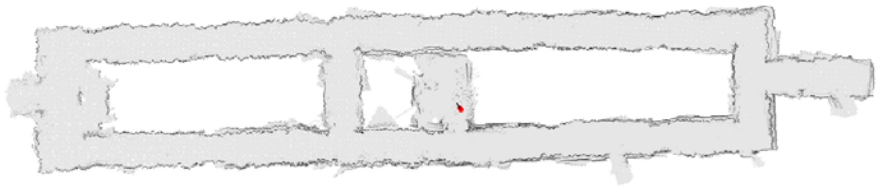
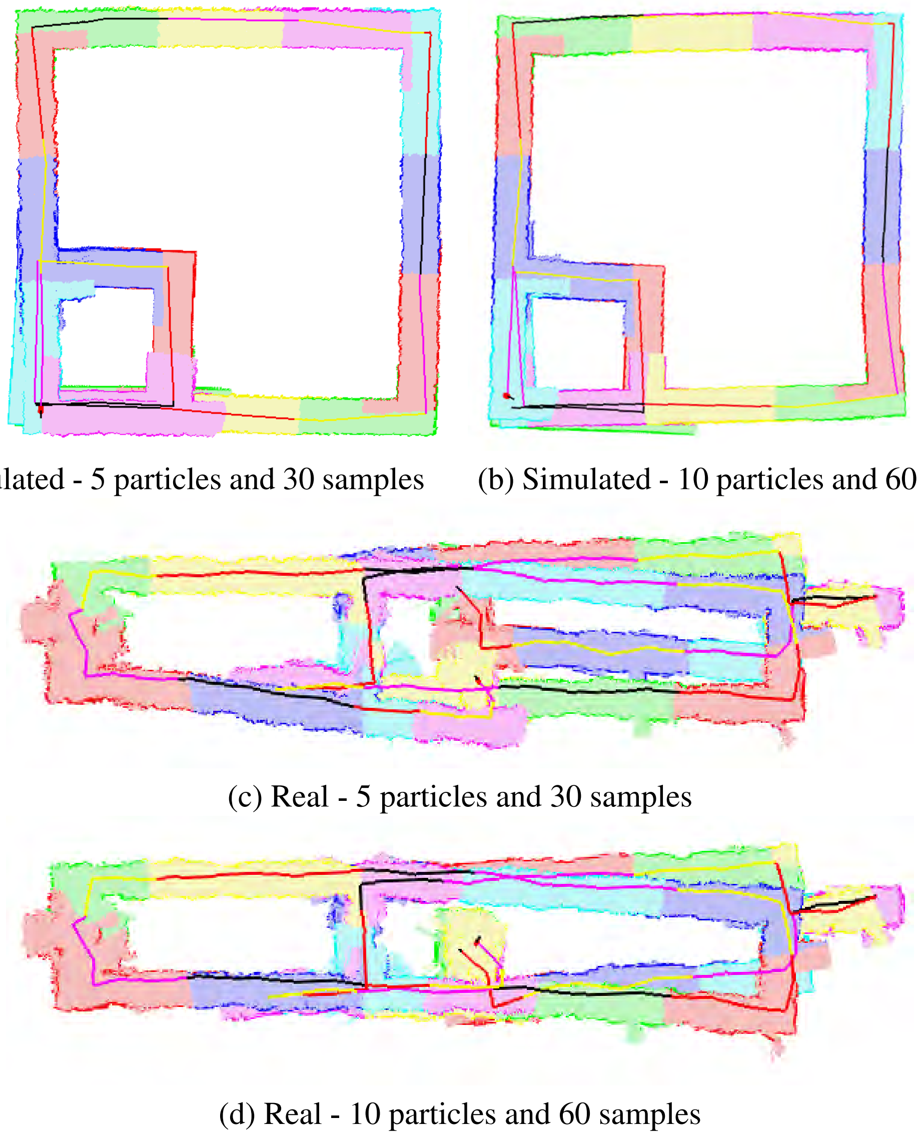

(f) SDP-SLAM - 60/10 particles

Figure 4.10: Comparison between DP-SLAM, MDP-SLAM and SDP-SLAM through experiments in the real environment.

matching operations.

Figure 4.11 shows the resulting maps of the experiments in simulated and real environments using the SegSLAM topology estimation. We tested configurations with 5 and 10 particles, generating 30 and 60 samples of submaps combinations. Since the segmentation was based on the number of steps of the process, the same numbers of segments of the SDP-SLAM experiments were generated: 16 segments in the simulated experiment, as shown in (a) and (b), and 21 segments in the real experiment, as shown in (c) and (d).



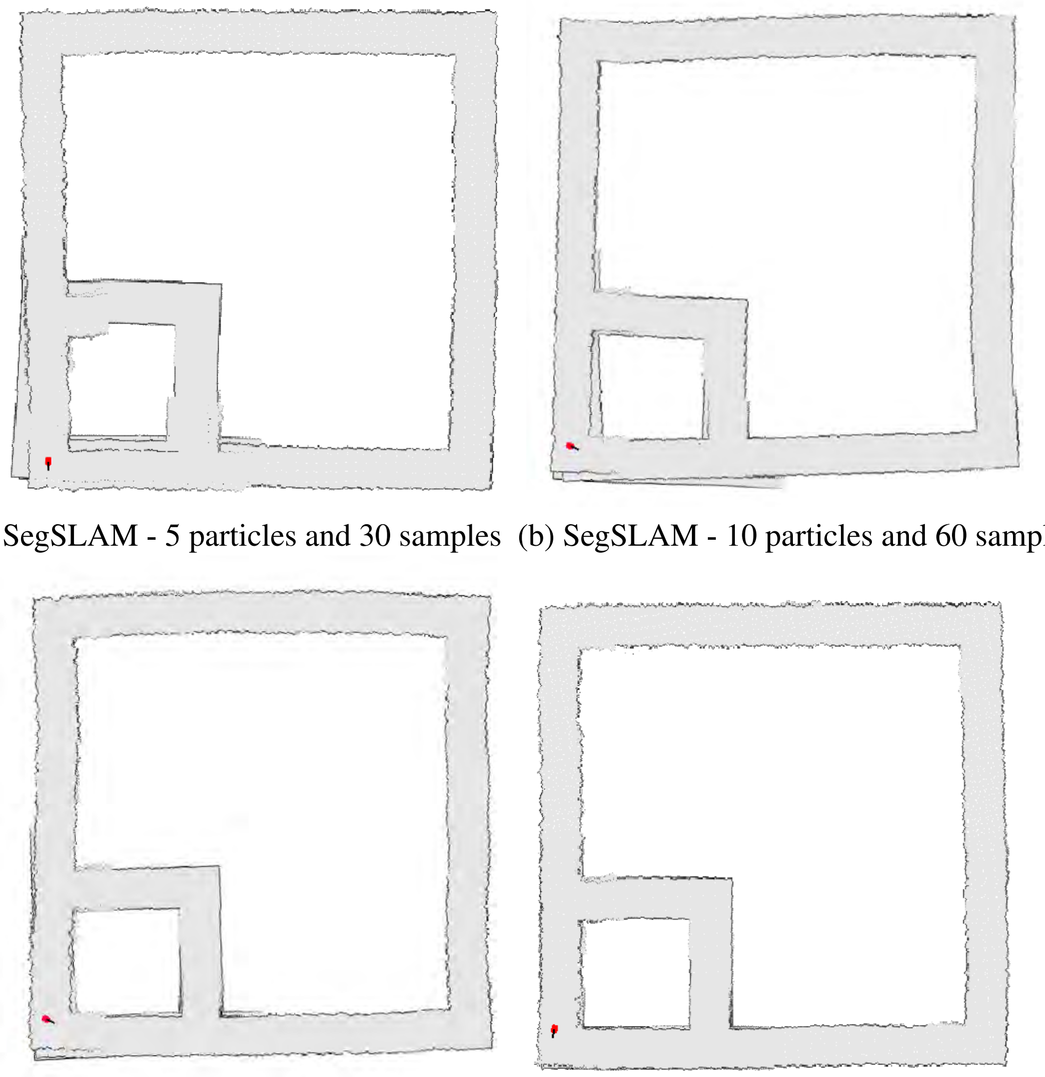
(a) Simulated - 5 particles and 30 samples (b) Simulated - 10 particles and 60 samples

(c) Real - 5 particles and 30 samples

(d) Real - 10 particles and 60 samples

Figure 4.11: Results of the experiments using the SegSLAM topology estimation, showing the paths and the segmented maps.

Comparing these maps with the results of the previous experiments, we observe the advantages of the SDP-SLAM estimation process. Figure 4.12 shows the resulting maps of the experiments in simulated environment. The map built using the SegSLAM topology estimation with 30 samples and 5 particles, shown in (a), is more misaligned than the map built by SDP-SLAM using 30/5 particles, shown in (c). Using 60 samples and 10 particles the result of SegSLAM do not present much improvement, as shown in (b), because the map is still misaligned. On the other hand, as shown in (d), the map built by SDP-SLAM using 60/10 particles is the closest to the environment ground truth.

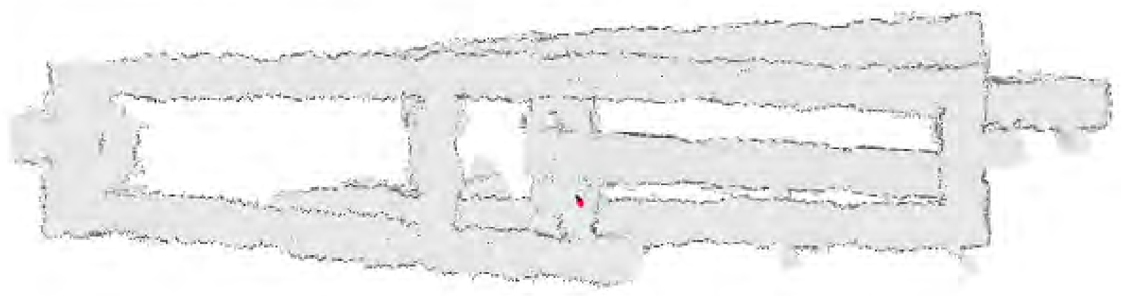


(a) SegSLAM - 5 particles and 30 samples (b) SegSLAM - 10 particles and 60 samples

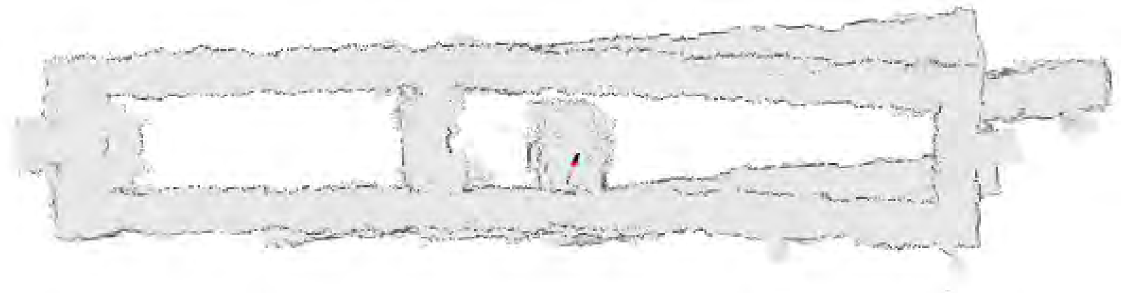
(c) SDP-SLAM - 30/5 particles (d) SDP-SLAM - 60/10 particles

Figure 4.12: Comparison between the topology estimation of SDP-SLAM and SegSLAM, through experiments in the simulated environment.

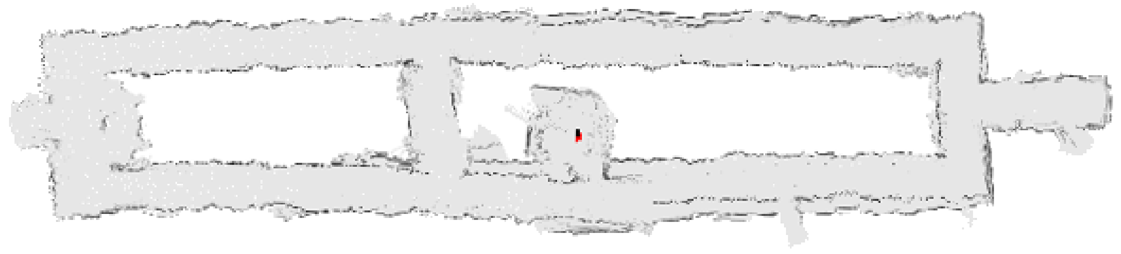
The resulting maps of the experiments in the real environment are presented in Figure 4.13. Here, since the environment is larger than the previous one, the difference between the results of the two methods is more visible. With the SegSLAM topology estimation, it is not possible to close the loop, as shown in (a) and (b). The resulting maps are not good, both using 5 particles with 30 samples and using 10 particles with 60 samples, despite the better result obtained with the second configuration. Comparatively, the map presented in (c), obtained by SDP-SLAM using 30/5 particles, is better than the maps produced by SegSLAM, even presenting alignment errors. The result improves when the particles number increases, as we can see in (d), where SDP-SLAM uses 60/10 particles.



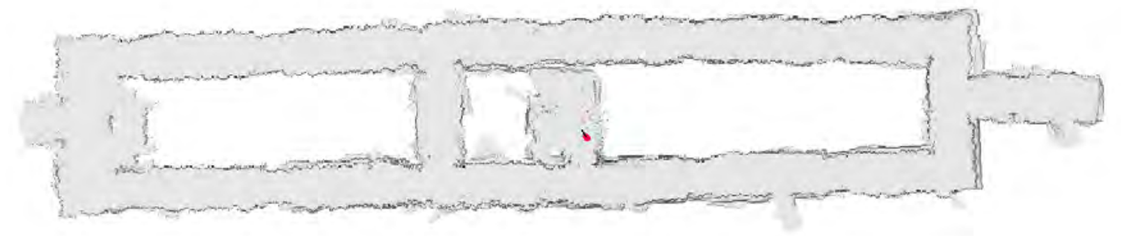
(a) SegSLAM - 5 particles and 30 samples



(b) SegSLAM - 10 particles and 60 samples



(c) SDP-SLAM - 30/5 particles



(d) SDP-SLAM - 60/10 particles

Figure 4.13: Comparison between the topology estimation of SDP-SLAM and SegSLAM, through experiments in the real environment.

We also made the evaluation of the topology estimations by measuring the mean alignment error of the solutions, in meters, during the SLAM process. This measure is given by the mean ICP nearest neighbor error, defined in Equation 3.4.2 of the Section 2.5.2, that is computed in the submaps matching process. Figures 4.14 and 4.15 show the mean error variation from the solutions of the experiments in simulated and real environment, respectively. All lines in these figures have straight line segments, because the submaps matching process is only performed when there are overlapping submaps, that is, during periods in which the robot is closing loops. In the rest of the process the weights of the samples were not changed.

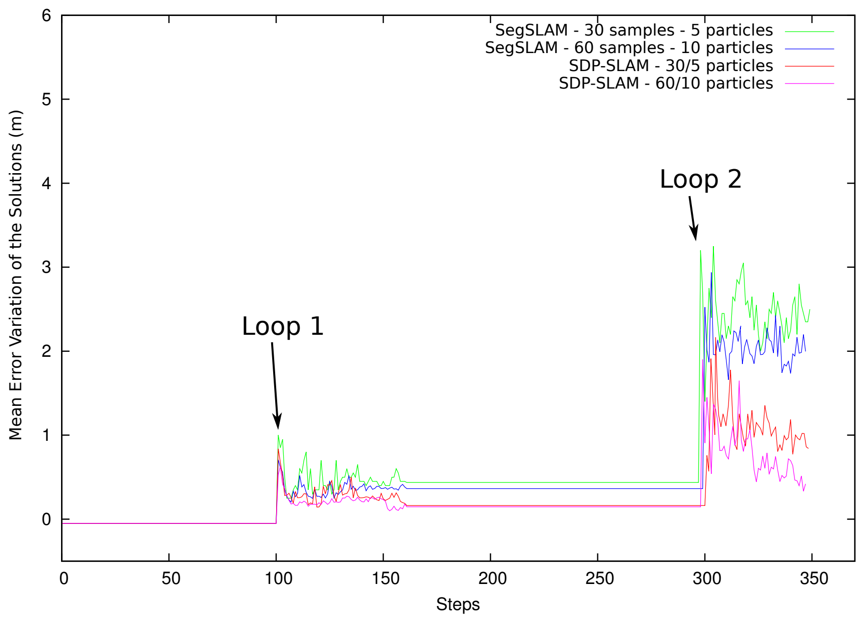


Figure 4.14: Mean error variation from the solutions of the experiments in simulated environment.

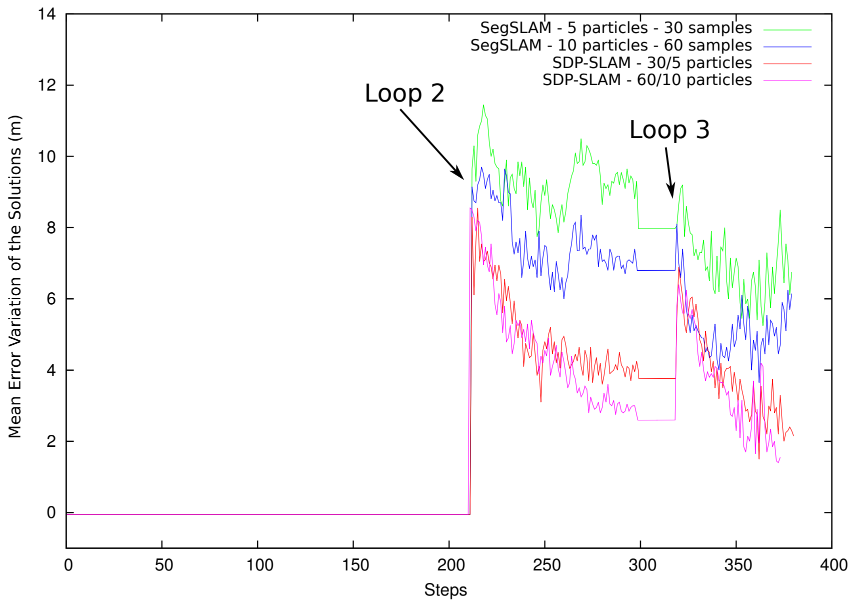


Figure 4.15: Mean error variation from the solutions of the experiments in real environment.

Figure 4.14 shows that, at the closure of the small loop (loop 1), the mean error is low ($< 0.5m$) for any of the settings used. The big problem happens when the robot returns to its initial position, after covering the major loop (loop 2). At first, the error is large, because the overlap among submaps is not good in the beginning of loop closures. After this, using the strategy adopted by SegSLAM, the error oscillates around 2, while with SDP-SLAM the error continues to decrease below $1m$.

The mean and standard deviation of the ICP error during the experiments in simulated environment are shown in Table 4.1. The error is smaller using the SDP-SLAM topology estimation than using the SegSLAM topology estimation. The best result is obtained with the configuration of 60/10 particles in SDP-SLAM.

	SegSLAM		SDP-SLAM	
Particles	μ	σ	μ	σ
30/5	1.43	0.84	0.66	0.34
30/10	1.26	1.04	0.58	0.36
60/5	1.23	1.18	0.61	0.40
60/10	1.15	1.22	0.48	0.45

Table 4.1: Mean and standard deviation of the ICP error in SegSLAM and SDP-SLAM, during the experiments in simulated environment.

In Figure 4.15, we observe that the errors reach values nearly three times higher than in the simulated environment. This can be explained by a couple of factors. First, the robot motion model is more inaccurate in the real environment, and second, the environment is larger, and therefore more difficult to map. Also, the robot started its path traveling around the larger inner loop (loop 2), instead of the minor loop (loop 1). This situation led to a mean error bigger in the beginning of the experiment than in the rest.

The mean and standard deviation of the ICP error during the experiments in real environment are shown in Table 4.2. The error values are bigger than in Table 4.1, but again, the error is smaller using the SDP-SLAM than using the SegSLAM topology estimation.

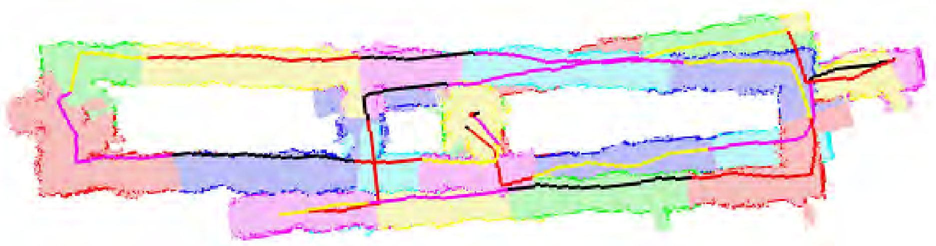
	SegSLAM		SDP-SLAM	
Particles	μ	σ	μ	σ
30/5	8.41	2.20	4.50	1.37
30/10	7.55	2.24	4.28	1.48
60/5	7.04	2.48	4.12	1.53
60/10	6.63	2.56	3.86	1.62

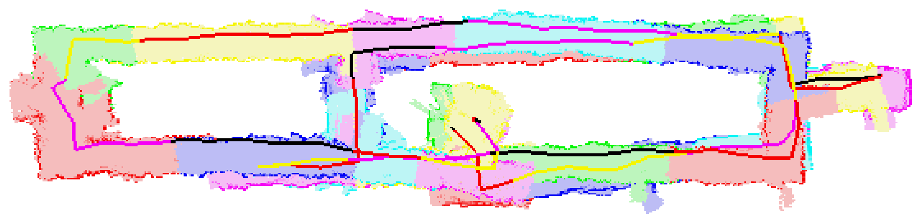
Table 4.2: Mean and standard deviation of the ICP error in SegSLAM and SDP-SLAM, during the experiments in real environment.

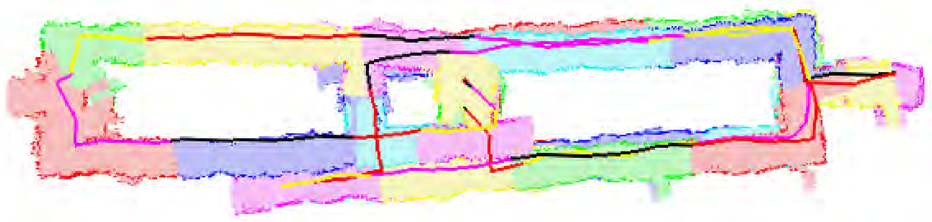
4.2.3 Analyzing the matching process

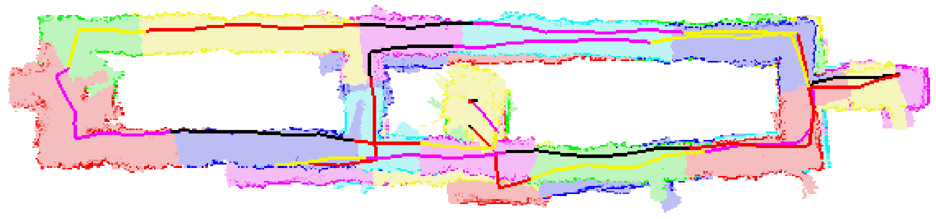
The evaluation of the matching process was made comparing the SDP-SLAM matching to the matching performed by SegSLAM. Both methods use the Iterative Closest Point algorithm (ICP) to associate points in different submaps. However, SegSLAM extract points from the obstacles, while our method extract points from the free-space.

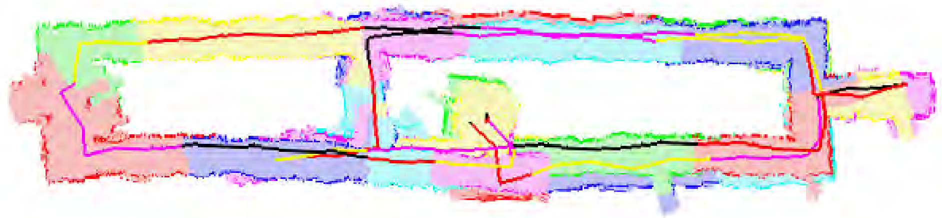
We compare the two matching strategies through experiments in the real environment, using 30/5, 30/10, 60/5 and 60/10 particles. The environment segmentation was made periodically at each 30 steps, thus the size of the submaps is the same in all experiments. The mean and standard deviation of the ICP error during the experiments are shown in Table 4.3, while some of the resulting maps are shown in Figure 4.16.


(a) Obstacles - 30/5 particles


(b) Obstacles - 60/5 particles


(c) Obstacles - 30/10 particles


(d) Obstacles - 60/10 particles


(e) Free-space - 30/5 particles

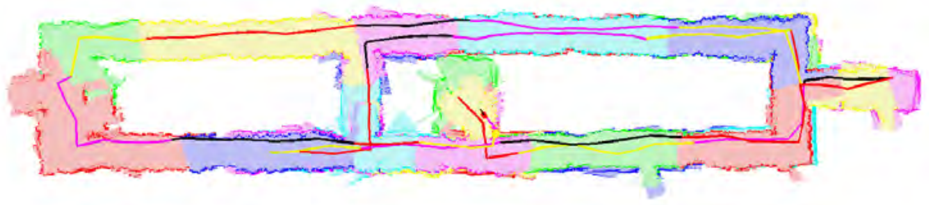

(f) Free-space - 60/10 particles

Figure 4.16: Comparison between the matching extracting points from obstacles and from free-space, through experiments in the real environment.

	Points extracted from			
	Obstacles		Free-space	
Particles	μ	σ	μ	σ
30/5	6.27	1.76	4.50	1.37
30/10	5.04	1.95	4.28	1.48
60/5	5.83	2.20	4.12	1.53
60/10	4.22	2.39	3.86	1.62

Table 4.3: Mean and standard deviation of the ICP error in experiments in the real environment, extracting points from obstacles and from free-space.

The error is smaller extracting points from the free-space than extracting from the obstacles. Since SDP-SLAM uses relatively few points to perform the matching (between 100 and 200 points depending on the size of the submaps, using this grid resolution), the initial association made between the points is of great importance. As presented in Section 3.4.2, this is an advantage of extracting points from the free-space.

In these experiments in real environment, the average time of a matching operation increased by 40 percent with the inclusion of our proposed mechanism of point selection. However, without such pre-processing phase, the matching results are far worse, as shown in Figure 4.16. In (a-d), we present the maps obtained with the matching strategy oriented to obstacles, using 30/5, 30/10, 60/5 and 60/10 particles. None of these maps is better than the maps built with the matching strategy oriented to free-space, shown in (e-f). The best maps obtained with the matching strategy using points from obstacles, shown in (b) and (d), are similar to the map presented in (e). Nevertheless, both maps are much worse than the map shown in (f), obtained with our proposed matching using 60/10 particles.

4.2.4 Analyzing the segmentation process

We conclude this chapter with the last main part of the algorithm that remains to be analyzed: the segmentation process. In all the experiments performed so far, we used a periodic segmentation made at every 30 robot steps (iterations of the SLAM process). Now, we compare the results obtained by SDP-SLAM varying this segmentation parameter, in experiments performed in the real environment.

An important consideration to be made is that varying the size of the map segments, the time spent in each iteration of the method also varies. This happens because the size of the segments implies the size of the sets of points used by the matching process. Figure 4.17 shows a comparison between the average iteration runtimes of the experiments, where the segmentations were made at every 15 steps, 30 steps (configuration used in all SDP-SLAM experiments analyzed until now) and 60 steps of the robot journey.

Performing the segmentation at every 60 steps, the runtime increases more than 100%, compared to the results of 30 steps. Thus, for 60 steps, we also tested configurations using only 15 particles in the top level, that have shorter runtimes. On the other hand, when segmenting at every 15 steps, the runtime decreases by about half. So, for 15 steps, we tested configurations with until 120 particles in the top level.

The mean and standard deviation of the matching error during the experiments in real environment are shown in Table 4.4, while some resulting maps are shown in Figure 4.18. When segmenting at every 15 steps, Figure 4.18-(a-b), the submaps quality is high, yet the number of possible solutions increases. Therefore, more particles are required at the top level to estimate good submaps combinations and reduce the alignment errors. This can

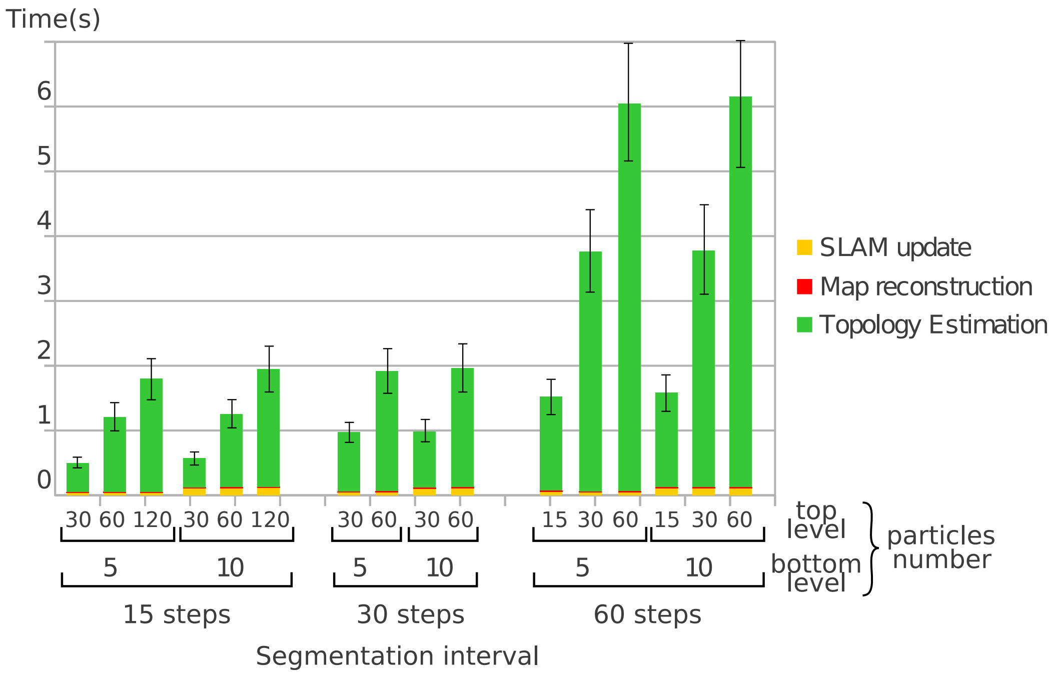
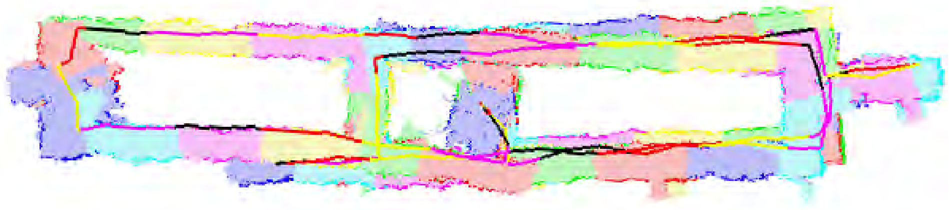


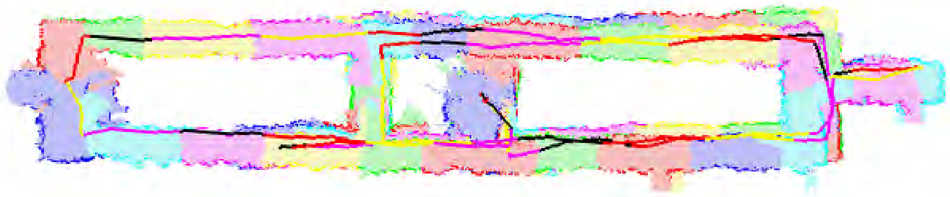
Figure 4.17: Runtime comparisons between experiments of SDP-SLAM in real environment, segmenting at every 15, 30 and 60 steps of the robot.

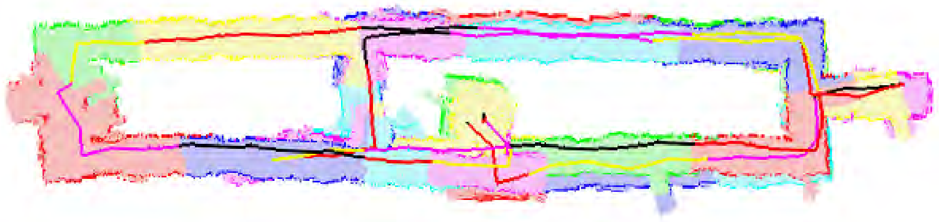
be observed, specially, in the experiment with 120/10 particles, shown in (b), whose result was similar to that obtained with 60/10 particles segmenting at every 30 steps, shown in (d). Conversely, we observe that the mean error is the largest when the interval between segmentations is too large, such as in the segmentation after 60 steps, shown in Figure 4.18-(e-f). This occurs because the submaps are too big to be properly built using a very low number of particles, like 5, at the bottom level. In fact, just doubling the number of particles at the bottom level (from 5 to 10) the mean errors drop significantly.

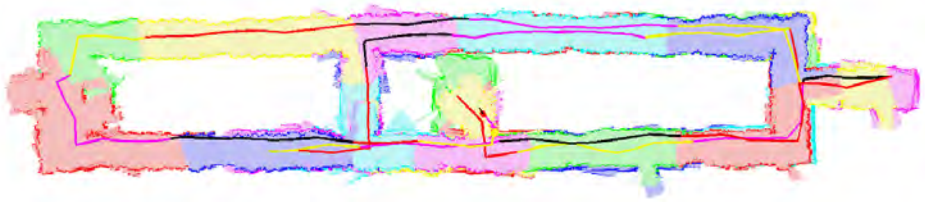
In addition to these experiments, we evaluate the strategy of segmentation based on the dispersion of the bottom level particles. Experiments were performed, in the real environment, comparing a segmentation made when the dispersion reaches $1m$ with a segmentation made when the dispersion reaches $1.5m$. The configurations tested were 30/5, 60/5, 30/10 and 60/10 particles. The mean and standard deviation of the matching error during these experiments are shown in Table 4.5, and the resulting maps are shown in Figure 4.19.

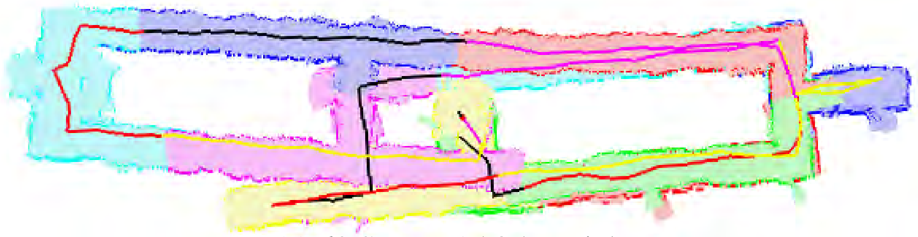
Considering the same threshold of particles dispersion and the same number of particles at the top level, the errors are smaller when more particles are used at the bottom level. This can be perceived, in Figure 4.19, by comparing the maps (b) and (c), and the maps (d) and (f). The reason for this is that using more particles at the bottom level, the dispersion among them grows faster, causing more frequent segmentations. It is possible to notice that the submaps created using 5 particles at the bottom level are much greater than the submaps created using 10 particles.


(a) 15 Steps - 60 / 5 particles


(b) 15 Steps - 120 / 10 particles


(c) 30 Steps - 30 / 5 particles


(d) 30 Steps - 60 / 10 particles


(e) 60 Steps - 15 / 5 particles

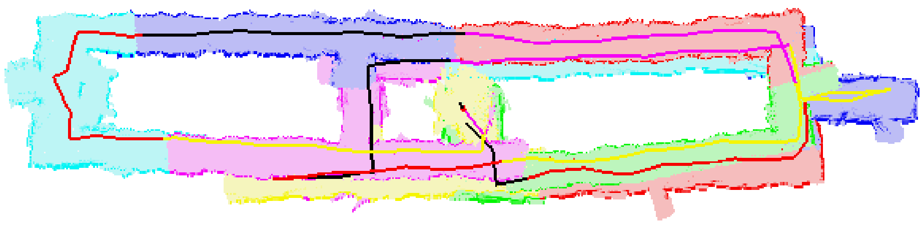

(f) 60 Steps - 15 / 10 particles

Figure 4.18: Comparisons between different segmentations configurations, through experiments in the real environment.

Segmentation	Particles	μ	σ
at every 15 steps	30/5	5.04	1.25
	60/5	4.72	1.42
	120/5	4.31	1.74
	30/10	5.16	1.39
	60/10	4.78	1.66
	120/10	4.23	1.81
at every 30 steps	30/5	4.50	1.37
	60/5	4.12	1.53
	30/10	4.28	1.48
	60/10	3.86	1.62
at every 60 steps	15/5	9.18	1.83
	30/5	8.62	2.22
	60/5	8.01	2.69
	15/10	5.85	2.03
	30/10	5.31	2.36
	60/10	4.42	2.82

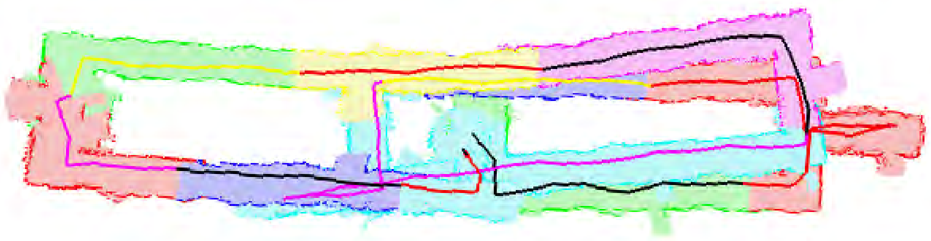
Table 4.4: Mean and standard deviation of the ICP error in experiments in the real environment, using different periodic segmentations parameters.

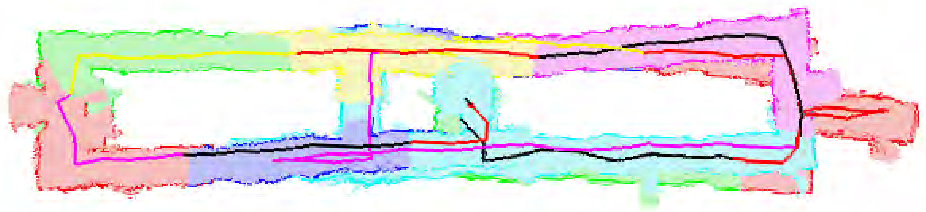
Segmentation	Particles	μ	σ
when dispersion reaches $1m$	30/5	7.22	1.77
	60/5	5.37	1.93
	30/10	5.83	1.30
	60/10	4.21	1.57
when dispersion reaches $1.5m$	30/5	8.08	1.91
	60/5	5.54	2.16
	30/10	6.48	1.63
	60/10	4.81	1.96

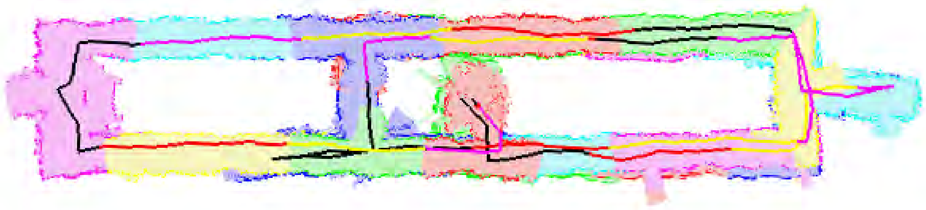
Table 4.5: Mean and standard deviation of the ICP error in experiments in the real environment, performing the segmentation based on the particles dispersion.

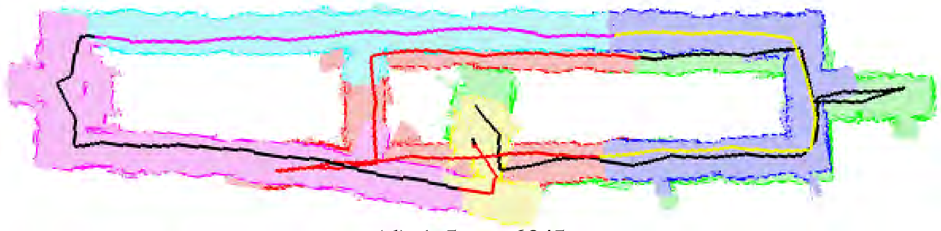
In these experiments, we observe that the segmentation based on the particles dispersion have an almost periodic behavior. This issue is directly linked to the construction of our algorithm. SDP-SLAM creates submaps totally independent of each other, so the particles dispersion associated to a segment is not affected by revisits to regions mapped by other segments. In SDP-SLAM, the update of a submap does not take into account informations of other maps, aiming to enable the combination of any submaps. Such information could be used to improve the segmentation process and the quality of the submap being constructed. However, this would definitely associate the submap to a specific submaps combination, which is contrary to the basic principle of our method.

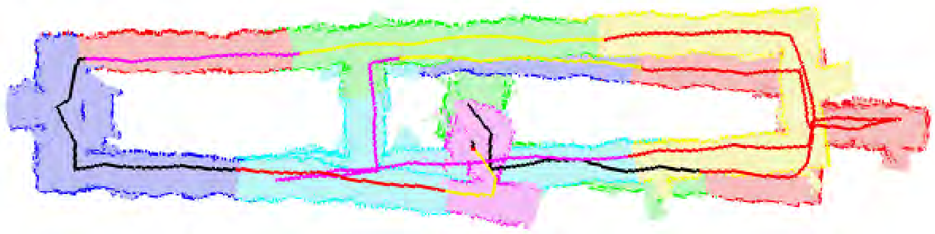
Nevertheless, the periodic segmentation served very well in the kind of environment tested in this work.


(a) 1m - 30/5


(b) 1m - 60/5


(c) 1m - 60/10


(d) 1.5m - 60/5


(e) 1.5m - 30/10

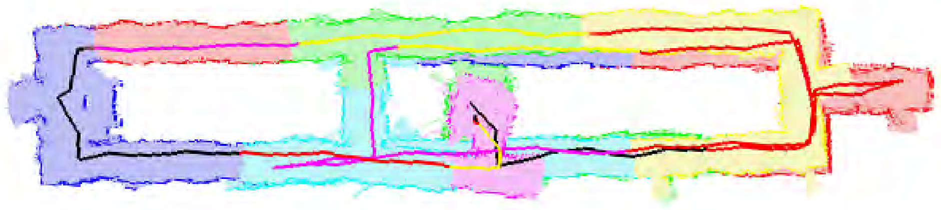

(f) 1.5m - 60/10

Figure 4.19: Comparisons between different segmentations configurations based on the particle dispersion.

5 CONCLUSION

In this work, we presented a submap-based SLAM algorithm called Segmented Distributed Particle SLAM (SDP-SLAM). Our method combines the Distributed-Particle SLAM (DP-SLAM) (ELIAZAR; PARR, 2003) with the Segmented SLAM (SegSLAM) (FAIRFIELD; WETTERGREEN; KANTOR, 2010). DP-SLAM is a Rao-Blackwellized particle filter that has optimized structures to store the particles maps, but requires a large number of particles to obtain good results. However, it also presents features that favors the implementation of a submap-based strategy. SegSLAM is a submap-based SLAM approach that partitions the environment in segments, composed by multiple submaps, and builds global solutions by combining submaps of different segments.

SDP-SLAM, like SegSLAM, is an algorithm that segments and combines different hypotheses of robot trajectories to reconstruct the environment map. These segmented trajectories, and their associated submaps, are built by a particle filter that uses the optimized structures of DP-SLAM. The composition of global solutions is performed by consulting these structures. Furthermore, a second particle filter is used by SDP-SLAM to estimate good combinations of submaps, that are evaluated using map matching techniques. The main contributions of SDP-SLAM are the development of novel strategies for the matching of submaps and for the estimation of submaps combinations.

We started this dissertation discussing in details the SLAM problem, in Chapter 2. We presented the formal derivation of the problem and discussed the main approaches to solve it, such as particle filters. We described the Rao-Blackwellized particle filter (RBPF) approach, focusing on DP-SLAM. We observed the problems of this type of strategy and discussed improvements present in the literature. We focussed on the submap-based SLAM approaches, such as SegSLAM.

Next, in Chapter 3, we presented our proposal. First, we described our initial approach of improvement over DP-SLAM, used in an integrated exploration strategy. Then, SDP-SLAM was described, starting with the overview of the algorithm, followed by its formal derivation. The modification of DP-SLAM proposed in our initial approach is an important component of SDP-SLAM. Another components, that were described in details, are the segmentation process, the map matching process and the topology estimation process, using the probabilistic graph of segments (PGS).

In Chapter 4 we performed a series of experiments, in both simulated and real environments, to evaluate our method. We focused on large nested loops configurations, that are very challenging in the SLAM. Our method was compared with traditional RBPF aproaches, such as DP-SLAM and our modified DP-SLAM, and with submap-based strategies, such as SegSLAM.

The results obtained in the experiments showed that SDP-SLAM generates better solutions than the original DP-SLAM and the modified DP-SLAM, using a much smaller number of particles. Is noteworthy that the complexity of SDP-SLAM is greater than DP-SLAM, thus if we use the same number of particles in both methods, the SDP-SLAM clearly will operate more slowly.

Regarding the comparisons with submap-based strategies, we performed experiments analyzing each individual step of SDP-SLAM: the topology estimation, the matching process and the segmentation.

The evaluation of the topology estimation process showed that our method indeed searches for solutions with low alignment errors. As measured in the experiments, the error associated to the samples of submaps combinations tends to decrease over time.

In the matching evaluation, we compared the traditional technique performed among points extracted from obstacles with our proposed technique that extracts points from free-space. Our approach improves the initial association among points of the submaps to be matched. Since we perform the matching using small sets of points, this initial association is very important, so our improvement reduced the errors on the matching.

Finally, in the segmentation evaluation, several configurations of periodic segmentation were tested. We observed that varying the size of the segments, implies considerably in the quality of the maps obtained. We also tested a segmentation strategy based on the dispersion of the particles used to built the submaps. This strategy presented a periodic behavior, because all the submaps are built independently in SDP-SLAM. Thus, revisiting and closing loops in the global scope, do not imply reductions in the particles dispersion at local levels.

5.1 Future Work

As observed in the last part of Chapter 4, our method could be improved if the update of a submap considered the information associated with other submaps. For example, when the robot revisited a known region, the submap of that region could be used to enhance the quality of the new submap. This approach improves the quality of the submaps, but reduces the possibilities of submaps combinations. An idea is to apply this strategy only when a set of submaps are well established (eg. after closing a perfect loop). Thus, such submaps would be permanent components of the global solutions.

Another future work is related to the matching process. In fact, this is an idea that emerged in the closing stages of this work, and hence not had the chance to be implemented. As explained in Chapter 3, our method extract points from the free-space to perform the matching, instead of extracting points from the obstacles. That said, we probably can improve the quality of the matching if both sets of points are used. For example, matching free-space points to other free-space points, and obstacle points to other obstacle points.

Finally, it is important to mention that is being developed a Special Interest Group on Humanitarian Technologies (RAS-SIGHT), in which the advisor of this work, prof Dr. Edson Prestes, and other members of the IEEE Robotics and Automation Society (RAS) are inserted. This group focus on how robotics and automation can help man in his activities. With that in mind, this work can be inserted in the context of humanitarian robots, for example, performing the SLAM in search and rescue tasks or domestic activities in large environments, etc.

REFERENCES

BAILEY, T.; NIETO, J.; GUIVANT, J.; STEVENS, M.; NEBOT, E. Consistency of the EKF-SLAM Algorithm. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2006., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2006. p.3562–3568.

BENTLEY, J. L. Multidimensional binary search trees used for associative searching. **Communications of the ACM**, New York, NY, USA, v.18, n.9, p.509–517, 1975.

BESL, P. J.; MCKAY, N. D. A method for registration of 3-D shapes. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Washington, DC, USA, v.14, n.2, p.239–256, 1992.

BLANCO, J. L.; FERNANDEZ-MADRIGAL, J. A.; GONZALEZ, J. Toward a Unified Bayesian Approach to Hybrid Metric–Topological SLAM. **IEEE Transactions on Robotics**, Piscataway, NJ, USA, v.24, n.2, p.259–270, 2008.

BOSSE, M.; NEWMAN, P.; LEONARD, J.; SOIKA, M.; FEITEN, W.; TELLER, S. An Atlas framework for scalable mapping. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2003., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2003. v.2, p.1899–1906.

CHEN, Y.; MEDIONI, G. Object modelling by registration of multiple range images. **Image and Vision Computing**, Newton, MA, USA, v.10, n.3, p.145–155, 1992.

CHOSET, H.; LYNCH, K. M.; HUTCHINSON, S.; KANTOR, G. A.; BURGARD, W.; KAVRAKI, L. E.; THRUN, S. **Principles of Robot Motion**: theory, algorithms, and implementations. Cambridge, MA: MIT Press, 2005.

DELLAERT, F.; FOX, D.; BURGARD, W.; THRUN, S. Monte Carlo Localization for Mobile Robots. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 1999., Piscataway, NJ, USA. **Proceedings** IEEE Press, 1999.

DOUCET, A.; FREITAS, N. d.; MURPHY, K. P.; RUSSELL, S. J. Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In: CONFERENCE ON UNCERTAINTY IN ARTIFICIAL INTELLIGENCE (UAI), 16., San Francisco, CA, USA. **Proceedings** Morgan Kaufmann Publishers Inc., 2000. p.176–183.

ELIAZAR, A. I.; PARR, R. DP-SLAM 2.0. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2004., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2004. v.2, p.1314–1320.

ELIAZAR, A. I.; PARR, R. Hierarchical linear/constant time slam using particle filters for dense maps. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NIPS), 18., Cambridge, MA, USA. **Proceedings** MIT Press, 2006. p.339–346.

ELIAZAR, A.; PARR, R. DP-SLAM: fast, robust simultaneous localization and mapping without predetermined landmarks. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI), 18., San Francisco, CA, USA. **Proceedings** Morgan Kaufmann Publishers Inc., 2003. p.1135–1142.

ESTRADA, C.; NEIRA, J.; TARDOS, J. D. Hierarchical SLAM: real-time accurate mapping of large environments. **IEEE Transactions on Robotics**, Piscataway, NJ, USA, v.21, n.4, p.588–596, 2005.

FAIRFIELD, N.; WETTERGREEN, D. Evidence grid-based methods for 3D map matching. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2009., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2009. p.1637–1642.

FAIRFIELD, N.; WETTERGREEN, D.; KANTOR, G. Segmented SLAM in three-dimensional environments. **Journal of Field Robotics**, Chichester, UK, v.27, n.1, p.85–103, 2010.

FRESE, U. Treemap: an o(log n) algorithm for indoor simultaneous localization and mapping. **Autonomous Robots**, Hingham, MA, USA, v.21, n.2, p.103–122, 2006.

GRANSTROM, K.; CALLMER, J.; RAMOS, F.; NIETO, J. Learning to detect loop closure from range data. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2009., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2009. p.15–22.

GRISETTI, G.; KUMMERLE, R.; STACHNISS, C.; BURGARD, W. A Tutorial on Graph-Based SLAM. **IEEE Intelligent Transportation Systems Magazine**, Piscataway, NJ, USA, v.2, n.4, p.31–43, 2010.

GRISETTI, G.; STACHNISS, C.; BURGARD, W. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2005., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2005. p.2443–2448.

GUIVANT, J. E.; NEBOT, E. M. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. **IEEE Transactions on Robotics and Automation**, Piscataway, NJ, USA, v.17, n.3, p.242–257, 2001.

HAHNEL, D.; BURGARD, W.; FOX, D.; THRUN, S. An efficient fastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2003., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2003. v.1, p.206–211.

HUANG, G. P.; MOURIKIS, A. I.; ROUMELIOTIS, S. I. Analysis and improvement of the consistency of extended Kalman filter based SLAM. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2008., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2008. p.473–479.

HUANG, G. P.; MOURIKIS, A. I.; ROUMELIOTIS, S. I. Observability-based Rules for Designing Consistent EKF SLAM Estimators. **International Journal of Robotics Research**, Thousand Oaks, CA, USA, v.29, n.5, p.502–528, 2010.

JULIER, S. J.; UHLMANN, J. K. A new extension of the Kalman filter to nonlinear systems. In: INTERNATIONAL SYMPOSIUM ON AEROSPACE/DEFENSE SENSING, SIMULATION AND CONTROLS, 11., Bellingham, WA, USA. **Proceedings** SPIE, 1997.

KANG, J.-G.; CHOI, W.-S.; AN, S.-Y.; OH, S.-Y. Augmented EKF based SLAM method for improving the accuracy of the feature map. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2010., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2010. p.3725–3731.

KIM, B.; KAESS, M.; FLETCHER, L.; LEONARD, J.; BACHRACH, A.; ROY, N.; TELLER, S. Multiple relative pose graphs for robust cooperative mapping. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2010., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2010. p.3185–3192.

KIM, I.; KWAK, N.; LEE, H.; LEE, B. Improved particle fusing geometric relation between particles in fastslam. **Robotica**, New York, NY, USA, v.27, p.853–859, 2009.

KNIGHT, J.; DAVISON, A.; REID, I. Towards constant time SLAM using postponement. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2001., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2001. v.1.

LEE, S.-J.; SONG, J.-B. A new sonar salient feature structure for EKF-based SLAM. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2010., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2010. p.5966–5971.

LEE, T.-k.; LEE, S.; OH, S.-y. A hierarchical RBPF SLAM for mobile robot coverage in indoor environments. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2011., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2011. p.841–846.

LIU, J. S. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. **Statistics and Computing Journal**, Dordrecht, Netherlands, v.6, n.2, p.113–119, 1996.

MAKARENKO, A. A.; WILLIAMS, S. B.; BOURGAULT, F.; DURRANT-WHYTE, H. F. An experiment in integrated exploration. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2002., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2002. v.1, p.534–539.

MCDONALD, J. B.; KAESS, M.; CADENA, C.; NEIRA, J.; LEONARD, J. J. 6-DOF Multi-session Visual SLAM using Anchor Nodes. In: EUROPEAN CONFERENCE ON MOBILE ROBOTS (ECMR), 5., Orebro, Sweden. **Proceedings** AASS, 2011. p.69–76.

MONTEMERLO, M.; THRUN, S. **FastSLAM**: a scalable method for the simultaneous localization and mapping problem in robotics. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007. (Springer Tracts in Advanced Robotics).

MONTEMERLO, M.; THRUN, S.; KOLLER, D.; WEGBREIT, B. FastSLAM: a factored solution to the simultaneous localization and mapping problem. In: AAAI NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, Edmonton, Canada. **Proceedings** AAAI, 2002.

MONTEMERLO, M.; THRUN, S.; KOLLER, D.; WEGBREIT, B. FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (IJCAI), 16., San Francisco, CA, USA. **Proceedings** Morgan Kaufmann Publishers Inc., 2003.

MOUNT, D.; ARYA, S. ANN: a library for approximate nearest neighbor searching. In: CGC WORKSHOP ON COMPUTATIONAL GEOMETRY, 2., Durham, NC, USA. **Proceedings** Duke University, 1997.

MURPHY, K. P. Bayesian Map Learning in Dynamic Environments. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS (NIPS), 12., Cambridge, MA, USA. **Proceedings** MIT Press, 1999. p.1015–1021.

NI, K.; STEEDLY, D.; DELLAERT, F. Tectonic SAM: exact, out-of-core, submap-based slam. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2007., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2007. p.1678–1685.

NIETO, J. I.; BAILEY, T.; NEBOT, E. M. Scan-SLAM: combining ekf-slam and scan correlation. In: INTERNATIONAL CONFERENCE OF FIELD AND SERVICE ROBOTICS, 5., Secaucus, NJ, USA. **Proceedings** Springer-Verlag New York: Inc., 2006. p.167–178. (Springer Tracts in Advanced Robotics, v.25).

PAZ, L. M.; TARDOS, J. D.; NEIRA, J. Divide and Conquer: ekf slam in. **IEEE Transactions on Robotics**, Piscataway, NJ, USA, v.24, n.5, p.1107–1120, 2008.

PRESTES, E.; ENGEL, P. M. Exploration driven by local potential distortions. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2011., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2011. p.1122–1127.

RANGANATHAN, A.; MENEGATTI, E.; DELLAERT, F. Bayesian inference in the space of topological maps. **IEEE Transactions on Robotics**, Piscataway, NJ, USA, v.22, n.1, p.92–107, 2006.

SAEEDI, S.; PAULL, L.; TRENTINI, M.; SETO, M.; LI, H. Efficient map merging using a probabilistic generalized Voronoi diagram. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2012., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2012. p.4419–4424.

SMITH, R.; SELF, M.; CHEESEMAN, P. Estimating uncertain spatial relationships in robotics. In: COX, I. J.; WILFONG, G. T. (Ed.). **Autonomous Robot Vehicles**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1990. v.8, p.167–193.

STACHNISS, C. **Robotic Mapping and Exploration**. 1st.ed. Berlin Heidelberg: Springer Publishing Company, Incorporated, 2009. (Springer Tracts in Advanced Robotics).

STACHNISS, C.; GRISETTI, G.; BURGARD, W. Recovering Particle Diversity in a Rao-Blackwellized Particle Filter for SLAM After Actively Closing Loops. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2005., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2005. p.655–660.

STACHNISS, C.; HAHNEL, D.; BURGARD, W. Exploration with active loop-closing for FastSLAM. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2004., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2004. v.2, p.1505–1510.

TARDOS, J. D.; NEIRA, J.; NEWMAN, P. M.; LEONARD, J. J. Robust Mapping and Localization in Indoor Environments Using Sonar Data. **The International Journal of Robotics Research**, Thousand Oaks, CA, USA, v.21, n.4, p.311–330, 2002.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)**. Cambridge, MA, USA: MIT Press, 2005. (Intelligent robotics and autonomous agents).

THRUN, S.; MONTEMERLO, M. The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. **The International Journal of Robotics Research**, Thousand Oaks, CA, USA, v.25, n.5-6, p.403–429, 2006.

TUNGADI, F.; KLEEMAN, L. Loop exploration for SLAM with fusion of advanced sonar features and laser polar scan matching. In: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2009., Piscataway, NJ, USA. **Proceedings** IEEE Press, 2009. p.388–394.

APPENDIX SDP-SLAM - EM PORTUGUÊS

Introdução

Localização e Mapeamento Simultâneos (SLAM - *Simultaneous Localization and Mapping*) é o problema de construir um mapa do ambiente onde um robô está situado, ao mesmo tempo em que a localização deste robô é estimada. Resolver o problema de SLAM é um dos principais requisitos para a construção de robôs verdadeiramente autônomos e, de fato, tem sido o foco de muito estudo durante as últimas décadas (SMITH; SELF; CHEESEMAN, 1990; MONTEMERLO et al., 2002; ELIAZAR; PARR, 2003; GRISETTI et al., 2010).

Embora muitos algoritmos de SLAM tenham sido apresentados com sucesso, o SLAM permanece particularmente difícil quando trabalha-se em ambientes de grande escala. Ao longo dos últimos anos, o tamanho dos ambientes a serem testados tem aumentado muito, e por conseguinte, o tempo necessário para que os métodos obtenham soluções adequadas também aumentou. Para contornar a alta complexidade associada a um ambiente de grande escala, muitos métodos reduzem o problema em múltiplas instâncias simples de SLAM. Estes métodos, chamados de SLAM baseados em submapas, processam regiões limitadas do ambiente independentemente, e depois ajustam os resultados individuais para compor uma solução.

Uma abordagem popular de SLAM baseado em submapas é o uso de mapas híbridos, ou seja, combinar um mapa topológico com o mapa métrico do ambiente (BOSSE et al., 2003) (LEE; LEE; OH, 2011) (ESTRADA; NEIRA; TARDOS, 2005) (BLANCO; FERNANDEZ-MADRIGAL; GONZALEZ, 2008). Em tais estratégias, os nós do grafo representando o mapa topológico são associados a submapas métricos do ambiente, enquanto as arestas representam as ligações entre os submapas. Um dos primeiros trabalhos utilizando mapas híbridos foi o método Atlas (BOSSE et al., 2003). No Atlas, assim como na maior parte dos algoritmos baseados em submapas, as incertezas de cada submapa são modeladas de acordo com o seu próprio referencial. Conexões entre submapas são detectadas por um processo de casamento de mapas, e refinadas sempre que as incertezas dos submapas diminuem. Mapas globais são reconstruídos através de um processo *off-line* de alinhamento baseado em otimização de mínimos quadrados.

Outro exemplo é o SLAM Hierárquico (ESTRADA; NEIRA; TARDOS, 2005), que apesar de muito semelhante ao método Atlas, apresenta uma técnica de fechamento de ciclos que impõe consistência a nível global, e, portanto, aumenta a precisão do mapa resultante. Quando um ciclo é detectado no mapa topológico, o algoritmo faz com que a soma das transformações dos sistema de coordenadas dos submapas do ciclo sejam iguais a zero. Então, uma técnica não-linear de otimização de mínimos quadrados é aplicada para estimar o alinhamento correto dos submapas. Blanco et al. propôs o HMT-SLAM

(BLANCO; FERNANDEZ-MADRIGAL; GONZALEZ, 2008), que realiza uma estimativa unificada dos mapas métrico e topológico do robô. O HMT-SLAM introduz uma distribuição posterior híbrida sobre tanto a parte métrica quanto a parte topológica do caminho do robô. Portanto, a propagação das partículas é baseada não somente nos modelos de movimentação e de observação (como é feito nas estratégias tradicionais de filtro de partículas), mas também sobre um modelo de transição associado ao mapa topológico.

Outro algoritmo é o SegSLAM (FAIRFIELD; WETTERGREEN; KANTOR, 2010), que introduz a idéia do uso de segmentos, ao invés de submapas individuais. Um segmento representa uma região limitada do meio ambiente que pode ser descrito por vários submapas diferentes. Assim, diferentes combinações de submapas (um para cada segmento) produzem soluções diferentes, isto é, mapas globais diferentes.

Neste trabalho, propomos uma nova estratégia de SLAM, que combina a ideia de segmentos do ambiente contendo múltiplos submapas, introduzidas pelo algoritmo SegSLAM, com uma estrutura de dados otimizada para armazenar os mapas das partículas, introduzida pelo método chamado DP-SLAM (ELIAZAR; PARR, 2003). O aspecto distribuído de tal estrutura permite a segmentação do ambiente em múltiplos submapas. A principal contribuição deste trabalho é um novo algoritmo de SLAM baseado em submapas para ambientes estruturados, chamado SDP-SLAM (*Segmented Distributed-Particle SLAM* / SLAM Segmentado com partículas distribuídas). Outras contribuições são novas abordagens para estimar boas combinações de diferentes segmentos do mapa e para realizar o casamento dos mapas.

Fundamentação Teórica

DP-SLAM

DP-SLAM (*Distributed-Particle SLAM* / SLAM com partículas distribuídas) (ELIAZAR; PARR, 2003) é um algoritmo de filtro de partículas Rao-Blackwellized (RBPF), que utiliza uma estrutura otimizada para armazenar os mapas das partículas. Em estratégias baseadas em RBPF, o processo de reamostragem das partículas exige a cópia de várias instâncias do mapa. Considerando que, em cada instante, o robô varre uma área muito menor do que o mapa completo, a variação dos mapas das partícula entre instantes consecutivos ocorre apenas numa região pequena. O DP-SLAM aproveita-se desse fato para introduzir uma representação melhorada do mapa. Basicamente, ele mescla todos os mapas das partículas em um único mapa contendo as diferenças observadas por cada partícula, por meio de um processo chamado DP-Mapping.

O DP-Mapping utiliza duas estruturas de dados eficientes: uma árvore de ascendência e uma grade de ocupação modificada. A árvore de ascendência descreve a genealogia de todas as partículas ativas do filtro. As folhas da árvore representam as partículas ativas, enquanto que os nós internos são os ancestrais destas partículas, i.e., as partículas das quais derivam. A segunda estrutura é um mapa de grade que mantém uma árvore de observação, para cada célula. Se, para uma dada célula, uma partícula realiza uma observação diferente das feitas por seus ancestrais, então a árvore associada a tal célula é atualizada com as observações desta partícula. Logo, para obter o mapa completo de uma partícula é necessário consultar as observações feitas tanto pela partícula quanto por seus ancestrais.

Para garantir que a árvore de ascendência não cresça indefinidamente, um processo de poda é realizado. Quando uma partícula não gera uma partícula filha (i.e., quando é descartada pelo processo de reamostragem do filtro), ela é removida da árvore de ascendência visto que sua informação não será herdada por ninguém. Além disso, uma partícula que gera uma única filha é fundida com sua filha, para impedir a criação de ramos sem ramificações.

Apesar de toda a otimização do espaço, o DP-SLAM ainda exige um grande número de partículas para obter bons resultados.

SegSLAM

SegSLAM (FAIRFIELD; WETTERGREEN; KANTOR, 2010) é uma estratégia de SLAM baseado em submapas que estende o passo de predição do filtro de partículas para decidir quando uma partícula deve permanecer no submapa atual, retornar a um submapa existente ou criar um novo submapa. Diferentemente do SLAM tradicional baseado em RBPF, no SegSLAM as partículas s_t são amostradas a partir da distribuição não apenas da postura do robô, mas também dos submapas ($s_t = \{x_t, \theta_t\}$). Em cada partícula, a postura do robô é descrita de acordo com o sistema de coordenadas local do submapa em que o robô está inserido.

Outra distinção importante para o SLAM baseado em RBPF é que, enquanto tradicionalmente as partículas descrevem hipóteses da trajetória completa do robô, no SegSLAM, as partículas são apenas responsáveis por gerar trechos da trajetória do robô e os submapas associados, os quais são armazenados em uma estrutura chamada SegMap (mapa de segmentos). Esta estrutura mantém as conexões entre os submapas utilizando um grafo. Portanto, para reconstruir uma possível trajetória do robô é necessário concatenar segmentos consecutivos compatíveis. Isto é feito pela amostragem de submapas a partir do SegMap. As combinações possíveis de submapas são ponderadas por uma técnica de casamento de mapas. Depois, uma lista de segmentos potencialmente associáveis é gerada. No final, as partículas escolhem associações de submapas desta lista, ou criam novos segmentos, caso não nenhuma boa associação for encontrada.

Embora o aumento na diversidade de soluções (resultante das inúmeras possibilidades de combinações de submapas) seja benéfica para contornar o problema da destruição de partículas, ele aumenta consideravelmente o espaço de busca. No SegSLAM, a etapa de amostragem é feita considerando apenas uma análise local, isto é, o algoritmo seleciona os melhores exemplos de cadeias de alguns submapas, não amostras de todo o mapa. É difícil encontrar uma boa combinação de submapas dentro do espaço total de soluções sem uma pesquisa bem dirigida, portanto inúmeras amostras devem ser geradas, o que se torna inviável conforme o número de segmentos cresce.

SDP-SLAM

O método proposto neste trabalho - SDP-SLAM - combina mapas baseados em segmentos, introduzidos pelo SegSLAM, com a árvore de ascendência das partículas, introduzida pelo DP-Slam. A ideia desta abordagem é capturar a alta diversidade de soluções, ou seja, as inúmeras possibilidades de combinações de submapas.

O SDP-SLAM baseia-se no fato de que em ambientes grandes, observações de regiões próximas são altamente relacionadas, enquanto que observações de regiões distantes não são, por conseguinte, a segmentação do ambiente em submapas é viável. Seguindo as formalizações de HMT-SLAM (BLANCO; FERNANDEZ-MADRIGAL; GONZALEZ, 2008) e SegSLAM (FAIRFIELD; WETTERGREEN; KANTOR, 2010), definimos o mapa global do ambiente como

$$\Theta = \langle \{\theta_i\}_{i \in V(\Upsilon_t)} \, , \, \{T_{a,b}\}_{a,b \in V(\Upsilon_t)} \rangle \qquad (.1.1)$$

onde $\theta_i = \{\theta_i^{(1)}, \theta_i^{(2)}, \cdots, \theta_i^{(p)}\}$ representa o conjunto de p submapas métricos associados ao segmento i pertencente ao conjunto de segmentos Υ_t. $T_{a,b}$ representa as transformações de referencial entre submapas de dois segmentos adjacentes a e b, pertencente a Υ_t. De fato, para cada par de segmentos consecutivos, há um conjunto de transformações $T_{a,b} = \{T_{a,b}^{(1,1)}, T_{a,b}^{(1,2)}, \cdots, T_{a,b}^{(1,p)}, T_{a,b}^{(2,1)}, \cdots, T_{a,b}^{(p,p)}\}$ associando os submapas gerados por todas as partículas.

O estado $\mathbf{s}_t$ do robô no instante t é dado por

$$\mathbf{s}_t = \langle \mathbf{x}_t \, , \, \gamma_t \rangle \qquad (.1.2)$$

onde $\mathbf{x}_t$ representa a posição métrica do robô, enquanto γ_t indica a qual submapa o robô está associado.

Conhecendo tais definições, a distribuição posterior do SLAM considerando $\mathbf{s}_{1:t}$ e Θ é definida como

$$p(\mathbf{s}_{1:t}, \Theta \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \qquad (.1.3)$$

onde $\mathbf{z}_{1:t}$ e $\mathbf{u}_{1:t}$ são, respectivamente, os conjuntos de observações e ações do robô.

A estrutura do SDP-SLAM é apresentada na Figura 1. O SDP-SLAM é composto por dois níveis de filtros de partículas. O processo de nível inferior é responsável por estimar os segmentos do caminho robô $\mathbf{x}_{1:t}$, os mapas locais associados $\theta_{1:t}$ e as transformações entre submapas $T_{1,2:t-1,t}$. O processo de nível superior é responsável por estimar a topologia do mapa global, em outras palavras, encontrar as melhores combinações de submapas $\gamma_{1:t}$.

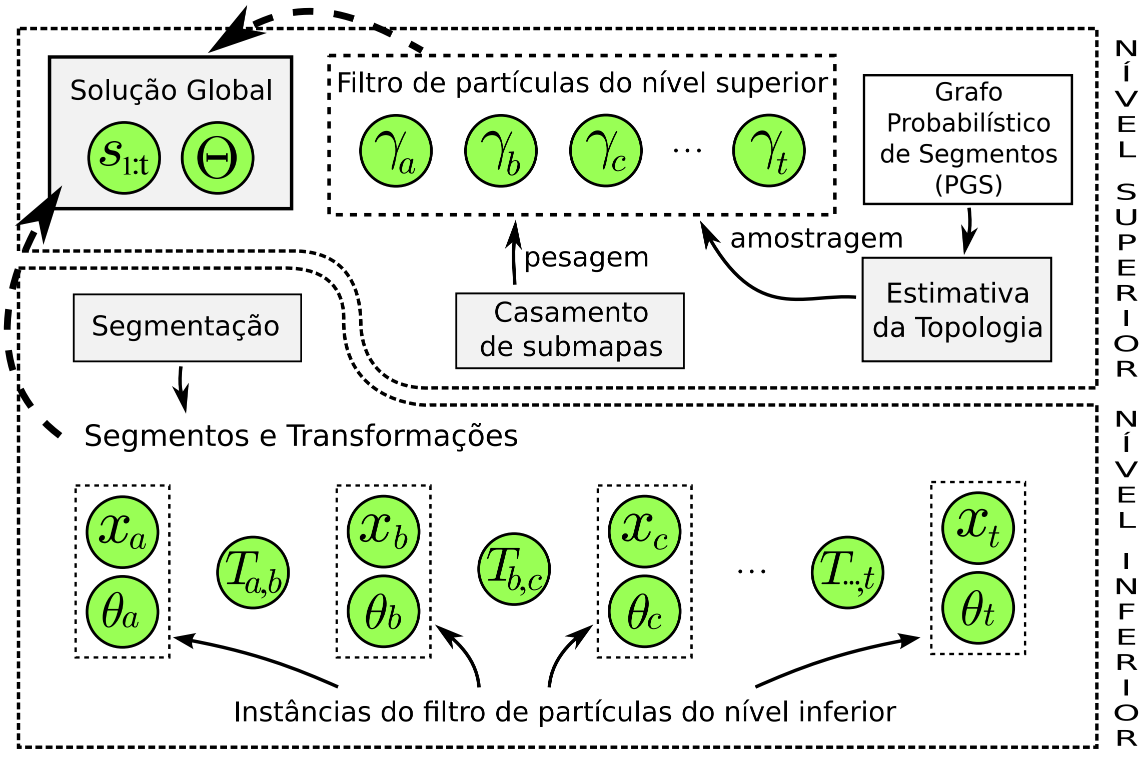

Solução Global
$s_{1:t}$
Θ
Filtro de partículas do nível superior
γ_a γ_b γ_c ⋯ γ_t
Grafo Probabilístico de Segmentos (PGS)
pesagem
amostragem
Casamento de submapas
Estimativa da Topologia
NÍVEL SUPERIOR
Segmentação
Segmentos e Transformações
x_a θ_a $T_{a,b}$ x_b θ_b $T_{b,c}$ x_c θ_c ⋯ $T_{..,t}$ x_t θ_t
Instâncias do filtro de partículas do nível inferior
NÍVEL INFERIOR

Figura 1: Estrutura do SDP-SLAM.

No nível inferior, a operação do filtro de partículas é similar ao SegSLAM, onde as partículas são apenas responsáveis por gerar submapas localmente precisos. A principal diferença está na forma como os mapas e trajetórias são construídas, utilizando o processo de DP-Mapping.

No nível superior, um filtro de partículas é utilizado para estimar boas combinações de submapas do ambiente. A propagação das partículas é feita pelo modelo de transição das partículas, que é uma função do conjunto de partículas no instante interior e do conjunto de todas as ações e observações feitas pelo robô:

$$\gamma_{1:t} = q(\gamma_{1:t} \,|\, \gamma_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

Este modelo de transição usa uma estrutura chamada Grafo Probabilístico de Segmentos (*Probabilistic Graph of Segments* - PGS), que mantém as conexões ponderadas entre submapas de segmentos adjacentes. O PGS é uma aproximação de Monte Carlo da distribuição de probabilidade de todos as possíveis topologias do ambiente. Como dito anteriorment, um segmento do ambiente contém múltiplos submapas, cada um construído por uma partícula diferente. Esses submapas são representados por nós do PGS, que se agrupam em níveis representando os segmentos. As ligações entre submapas de segmentos adjacentes são as arestas do PGS.

O algoritmo do SDP-SLAM é apresentado no Algoritmo 1. O primeiro passo (linha 1) corresponde à inicialização dos filtros de partículas. No nível inferior, todas as partículas começam na mesma posição com submapas vazios. No nível superior, todas as partículas começam com combinações de submapas vazias. O PGS começa com apenas um segmento, uma vez que o ambiente ainda não está segmentado.

No laço principal, o primeiro passo (linha 2) é a aquisição da odometria e das medições dos sensores de alcance. Em seguida, o filtro do nível inferior é atualizado para construir hipóteses de mapas locais. Os passos iniciais são os mesmos realizados pelo DP-SLAM (linha 3). Primeiro, as partículas são propagadas no interior do segmento atual e avaliadas. A seguir, a reamostragem é feita, a árvore de ascendências é atualizada, e as observações das partículas são atualizadas no mapa de partículas distribuídas.

Na sequência, ocorre a decisão de segmentação (linha 4). A maioria das estratégias de SLAM baseadas em submapas realiza a segmentação num intervalo de tempo praticamente uniforme, ou de acordo com medidas de erro (BOSSE et al., 2003) (LEE; LEE; OH, 2011). Nosso método foi testado com uma segmentação periódica em tempo fixo e uma segmentação baseada na dispersão das partículas.

Sempre que uma segmentação ocorre, o filtro de partículas do nível inferior é interrompido e a parte da árvore de ascendências associada ao último segmento é fixada, pois não poderá ser modificada posteriormente (linha 5). O conjunto atual de partículas é reiniciado para permitir a construção de novos submapas independentes (linha 6). No nível superior, um novo conjunto de nós representando os submapas do novo segmento é inserido no PGS. Entre as informações armazenadas nos nós estão a identificação das partículas, necessárias para consultas na árvore ascendência; e as transformações iniciais e finais dos submapas (i.e. a primeiro e a última postura do robô dentro do submapa), usadas para combinar submapas em um mesmo sistema de coordenadas.

O próximo passo é a atualização das partículas do nível superior (linha 7), responsáveis por estimar hipóteses de combinações de submapas. Adotamos uma estratégia de elitismo, logo, somente um pequeno conjunto das piores partículas são eliminados, enquanto as melhores partículas são mantidas. A amostragem de novas partículas é feita

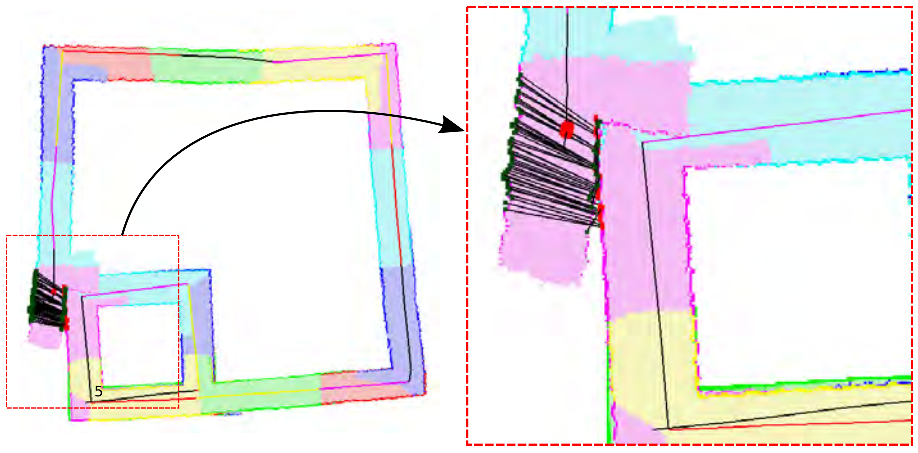
Algoritmo 1 - SDP-SLAM

```
1  Inicialização
   enquanto o robô está navegando faça
2      Leia as medidas de odometria e dos sensores de alcance.
       Processo do nível inferior:
       início
3          Atualização do DP-SLAM
4          Decisão de segmentação
           se ocorrer uma segmentação então
5              Fixagem da árvore de ascendências
6              Reinício das partículas
           fim
       fim
       Processo do nível superior:
       início
           se ocorrer uma segmentação então
7              Inserção de um novo nível de nós no PGS
           fim
8          Estimativa de combinações de submapas
9          Pesagem de combinações de submapas
10         Atualização do PGS
       fim
   fim
```

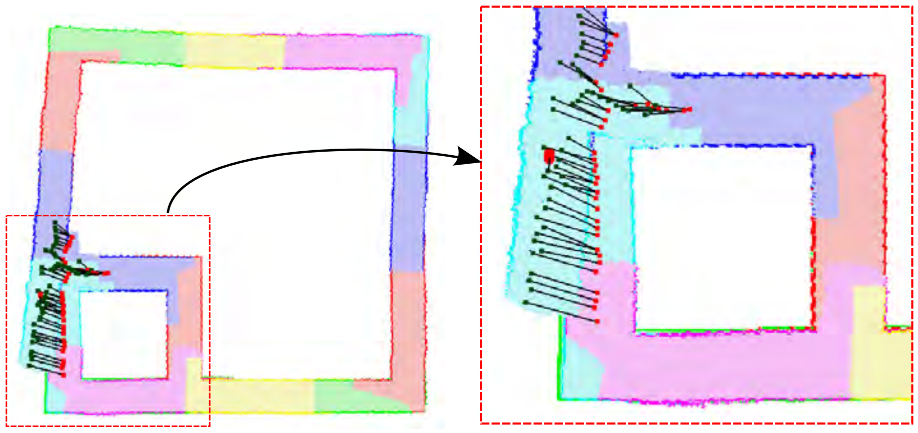
consultando o PGS. (Na última etapa deste algoritmo serão explicadas como as probabilidades do PGS são definidos.)

A avaliação das partículas do nível superior é feita através do casamento de submapas que se sobrepoem em cada amostra (linha 8). Assim como no SegSLAM, o processo de casamento é realizado com o algoritmo ICP (*Iterative Closest Point* / Ponto Mais Próximo Iterativo) (FAIRFIELD; WETTERGREEN, 2009). O ICP é um método muito simples e rápido, mas exige que os dois conjuntos de pontos a serem comparados tenham uma associação forte, caso contrário, o método pode convergir para mínimos locais ou até mesmo não convergir. Em geral, o ICP utiliza as informações sobre os obstáculos, desprezando informações sobre os espaços vazios do ambiente adquiríveis pelo uso de sensores de distância. No entanto, quando o erro de alinhamento dos segmentos é muito grande, a associação dos pontos pode ser incorreta, como mostrado na Figura 2(a), onde os pontos de uma parede foram associados erroneamente a pontos de uma outra parede. No nosso caso, podemos selecionar pontos a partir do centro das regiões de espaço livre, como mostrado em (b). Usando esses pontos nós conseguimos reduzir ambiguidades como a ilustrada em (a).

O último passo do nosso método é a atualização do PGS (linha 9). Sistematicamente, amostras de combinações de submapas são geradas e avaliadas pelo processo de casamento de mapas usando o ICP. O resultado da avaliação é uma medida de erro do ICP. O erro de cada uma das amostras é adicionado aos erros acumulados das ligações entre submapas que compõem tais amostras. Por exemplo, o erro acumulado E_{1b2a} da ligação entre γ_{1b} e γ_{2a} é a soma dos erros de todas as amostras contendo o par γ_{1b} e γ_{2a}. A ideia é que, ao longo do tempo, as conexões com baixos erros acumulados possuam grande



(a) Extraindo pontos dos obstáculos



(b) Extraindo pontos do centro do espaço livre

Figura 2: Comparação entre o casamento de submapas extraindo pontos dos obstáculos e extraindo pontos do centro do espaço livre.

chance de formar boas soluções. Assim, a probabilidade de selecionar um dado par de submapas é inversamente proporcional ao erro acumulado da associação deste par. Calculamos o inverso do erro acumulado e normalizamos os valores para obter probabilidades.

$$p(\gamma_{1b}, \gamma_{2a}) = \frac{\sum_{i,j=1}^{p} 1/E_{1i2j}}{1/E_{1b2a}} \tag{.1.4}$$

A Figura 3 mostra um exemplo do funcionamento do SDP-SLAM. Em (a), três submapas de um mesmo segmento são representados no mapa. Neste ponto, a árvore de ascendências contém apenas as partículas do primeiro segmento, como mostrado em (b), enquanto que o PGS possui os três nós (submapas 1A, 1B, 1C) do primeiro segmento, como mostrado em (c). Quando o método processa o segundo segmento, em (d), a árvore de ascendências contém as partículas do primeiro e do segundo segmento, como mostrado em (e). Em (f), o gráfico de segmentos contém dois níveis de nós, representando ambos os segmentos. Então, como mostrado em (g), é possível combinar os submapas dos dois segmentos e avaliar as combinações da amostra. In (h), abrimos um parêntese para mostrar que a árvore de ascendências continua a ser podada, assim como no DP-SLAM. Em (i), o peso de cada ligação entre submapas é atualizado no gráfico de segmentos. Tais pesos serão utilizados durante a etapa de amostragem do filtro de partículas do nível superior. Finalmente, em (j), uma possível trajetória do robô é reconstruída através da combinação de dois submapas ($1A$ e $2C$). O mapa global é obtido através da consulta das observações feitas por cada partícula dos ramos selecionados da árvore de ascendência, destacados em (k). Uma transformação T deve ser aplicada a ambos os submapas para colocá-los no mesmo sistema de coordenadas, como mostrado em (l). Esta transformação é uma composição da postura final do primeiro submapa com a postura inicial do segundo submapa.

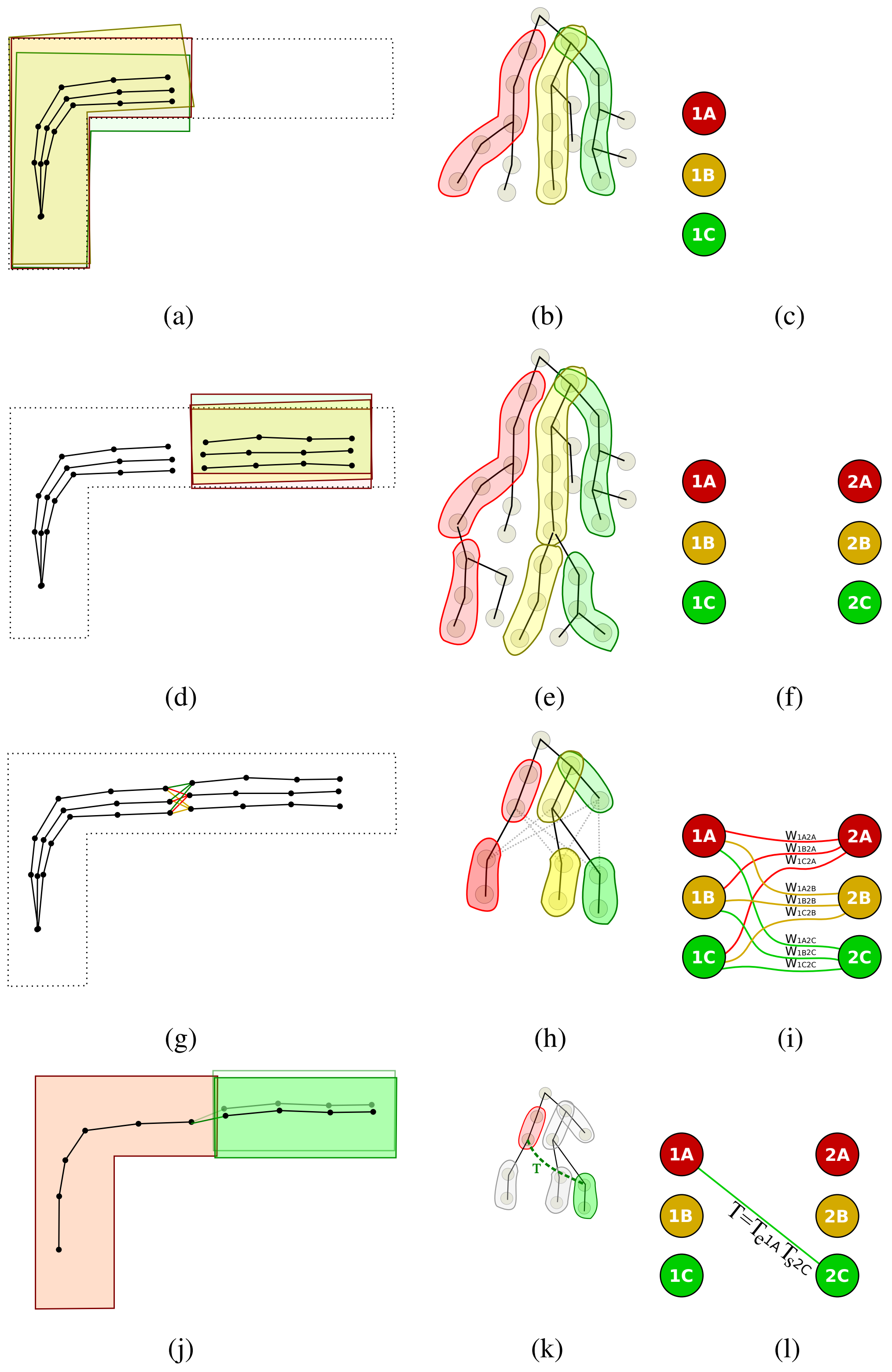
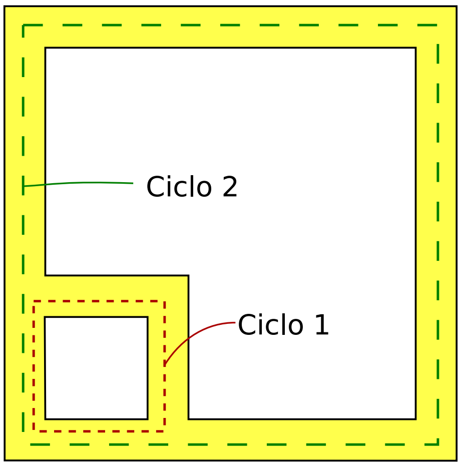


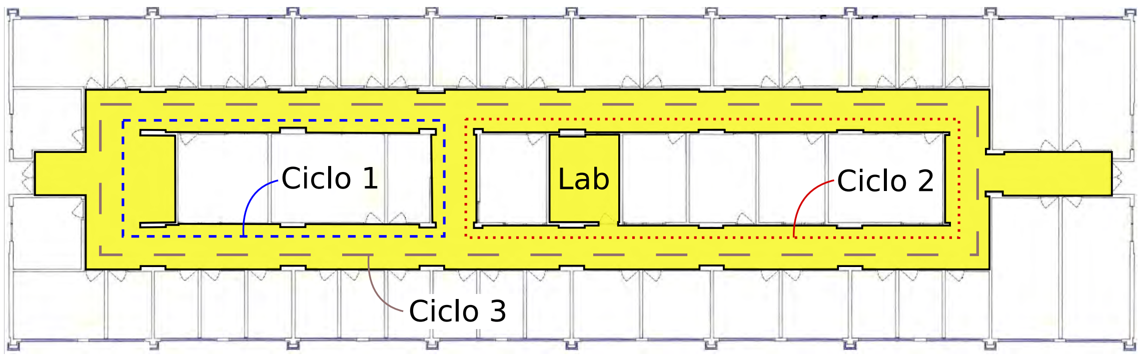
Figura 3: Exemplo de funcionamento do SDP-SLAM.

Experimentos

A avaliação do SDP-SLAM foi feita através de experimentos em ambientes reais e simulados, que estão ilustrados na Figura 4(a) e (b) respectivamente. O ambiente simulado contém um ciclo interno (ciclo 1) e um ciclo externo (ciclo 2), com comprimentos de $28m$ e $80m$, respectivamente. O ambiente real contém três ciclos, que correspondem a corredores de um prédio do Instituto de Informática da UFRGS. Os dois ciclos internos têm comprimentos de $43m$ (ciclo 1) e $57m$ (ciclo 2) e, em conjunto, formam um ciclo maior de $88m$ (ciclo 3).



(a) Ambiente simulado



(b) Ambiente real

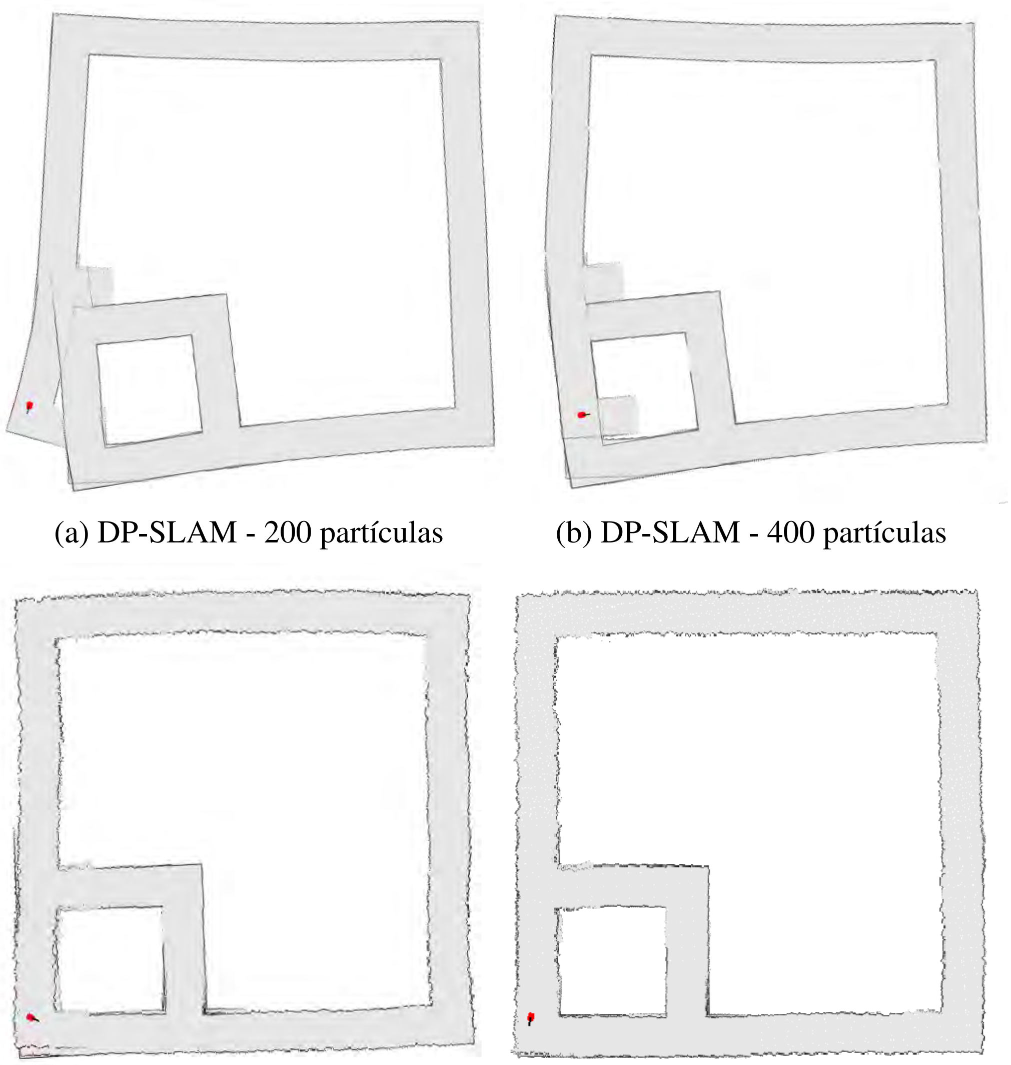
Figura 4: Ambientes mapeados nos experimentos com o SDP-SLAM.

Nós escolhemos tais ambientes, pois eles contêm ciclos aninhados, que agravam o problema de empobrecimento das partículas. Por exemplo, durante o mapeamento de um ciclo interno, uma estratégia baseada em RBPF descarta partículas que não têm os pesos mais altos, mas que podem ser necessárias futuramente para mapear um ciclo externo.

Comparação com o DP-SLAM

A Figura 5 mostra os mapas resultantes dos experimentos em ambiente simulado, onde o robô é representado pelo ponto vermelho, os obstáculos pelas linhas pretas e o espaço livre pelas áreas cinzas.

Na Figura 5(a) e (b), apresentamos os mapas construídos com o DP-SLAM usando 200 e 400 partículas, respectivamente. Com estas configurações, o método não conseguiu

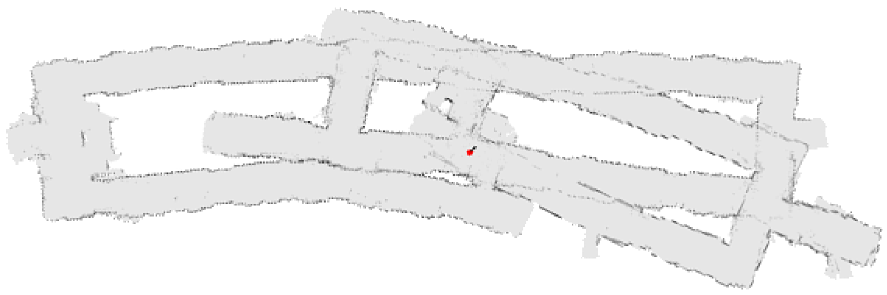

(a) DP-SLAM - 200 partículas (b) DP-SLAM - 400 partículas

(c) SDP-SLAM - 30/5 partículas (d) SDP-SLAM - 60/10 partículas

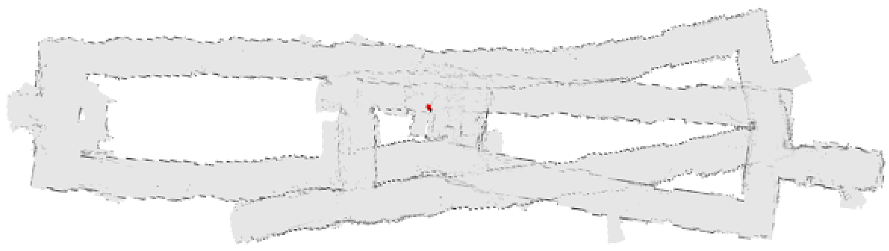
Figura 5: Comparação entre o DP-SLAM e o SDP-SLAM através de experimentos no ambiente simulado.

fechar corretamente o ciclo maior. Em (c), mostramos o mapa construído com o nosso método, usando 30/5 partículas (5 partículas no nível inferior e 30 partículas no nível superior). O mapa resultante é bom, mas apresenta algumas inconsistências. Por fim, em (d), é utilizado o dobro das partículas, tanto no nível inferior quanto superior, e o mapa resultante é visualmente melhor.

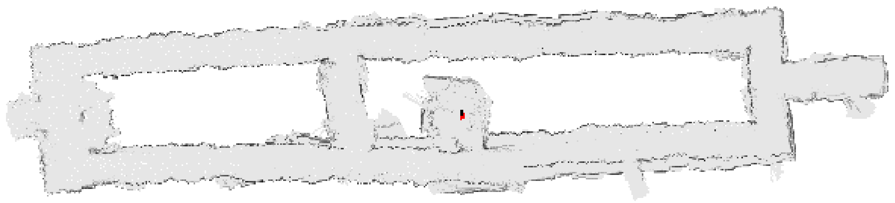
Os resultados no ambiente real, são mostrados na Figura 6. Neste ambiente, o caminho percorrido pelo robô é maior do que no ambiente simulado, e os métodos encontraram mais dificuldades. Os mapas criados pelo DP-SLAM com 200 e 400 partículas são mostrados em (a) e (b). Os resultados são muito ruins, uma vez que o DP-SLAM sequer foi capaz de fechar um único ciclo. Em (c), mostramos o resultado do SDP-SLAM com 5 partículas no nível inferior e 30 no nível superior. O mapa resultante é melhor do que o mapa construído pelo DP-SLAM com 400 partículas, uma vez que os erros de alinhamento são visualmente menores. Uma vez mais, o melhor resultado foi obtido com o SDP-SLAM utilizando 10 partículas no nível inferior e 60 no nível superior.



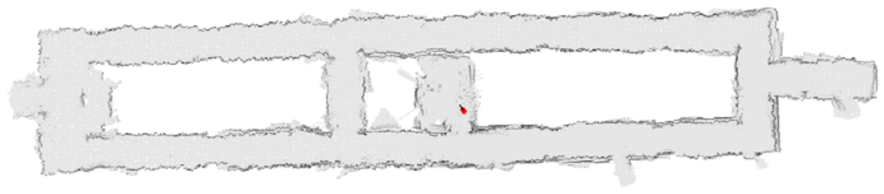
(a) DP-SLAM - 200 part.



(b) DP-SLAM - 400 part.



(c) SDP-SLAM - 30/5 part.



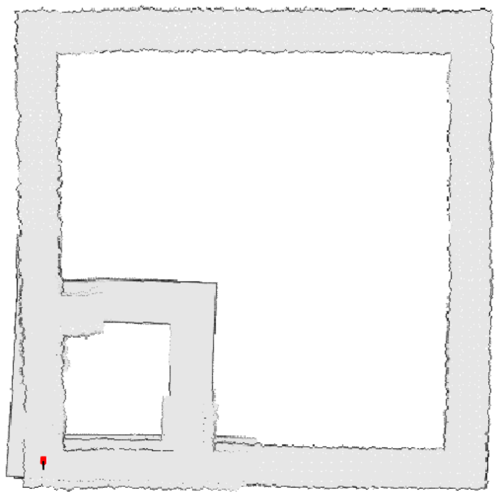
(d) SDP-SLAM - 60/10 part.

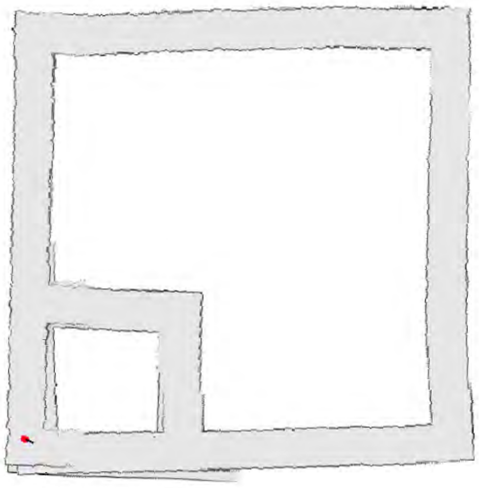
Figura 6: Comparação entre o DP-SLAM e o SDP-SLAM através de experimentos no ambiente real.

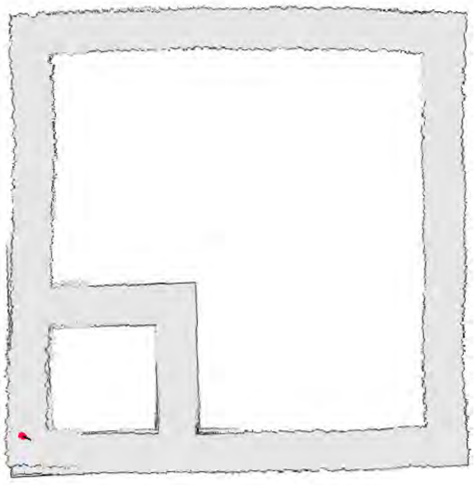
Comparação com o SegSLAM

Além da comparação com o DP-SLAM, também comparamos o SDP-SLAM com o SegSLAM. No entanto, nós não utilizamos a implementação exata do SegSLAM. Na verdade, o que comparamos foi a estimativa da topologia nos dois métodos. O restante dos dois processos, tais como o casamento de submapas e a segmentação do ambiente, foram mantidos iguais nestes experimentos.

A Figura 7 mostra os mapas resultantes dos experimentos em ambiente simulado. O mapa construído utilizando a estimativa de topologia do SegSLAM com 30 amostras de combinações e 5 partículas, mostrado em (a), é mais desalinhado do que o mapa construído pelo SDP-SLAM com 30/5 partículas, mostrado em (c). Utilizando 60 amostras e 10 partículas, o resultado do SegSLAM não apresenta muita melhora, conforme mostrado em (b), visto que o mapa continua desalinhado. Por sua vez, como mostrado em (d), o mapa construído pelo SDP-SLAM com 60/10 partículas é o mais próximo do ambiente real.


(a) SegSLAM - 30/5 partículas


(b) SegSLAM - 60/10 partículas


(c) SDP-SLAM - 30/5 partículas

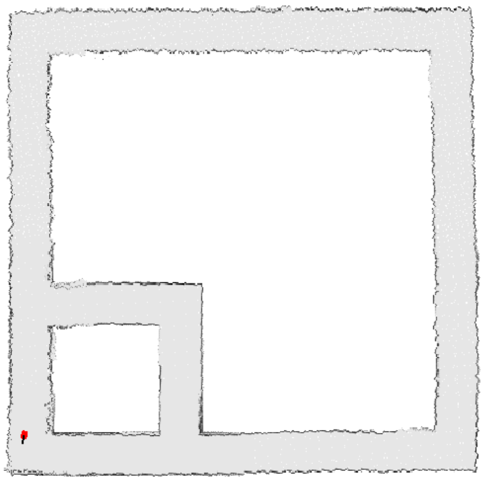
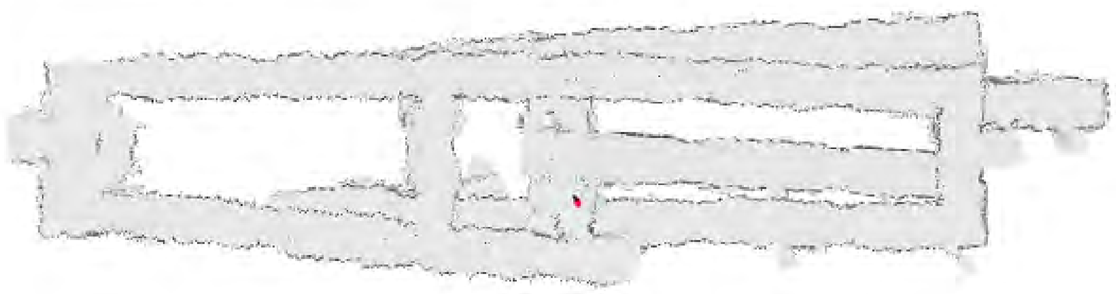
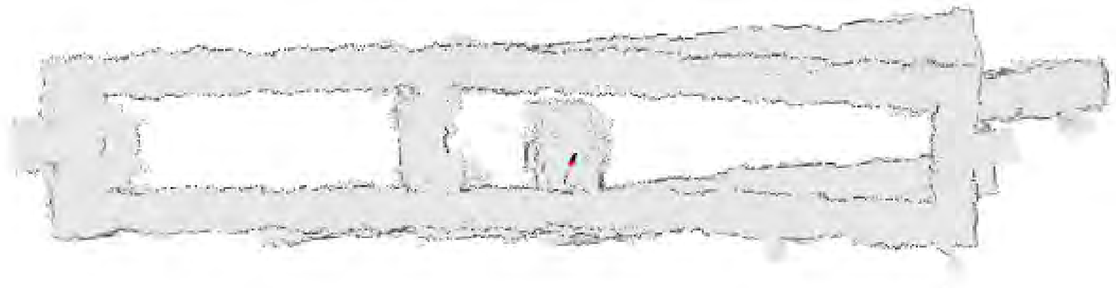

(d) SDP-SLAM - 60/10 partículas

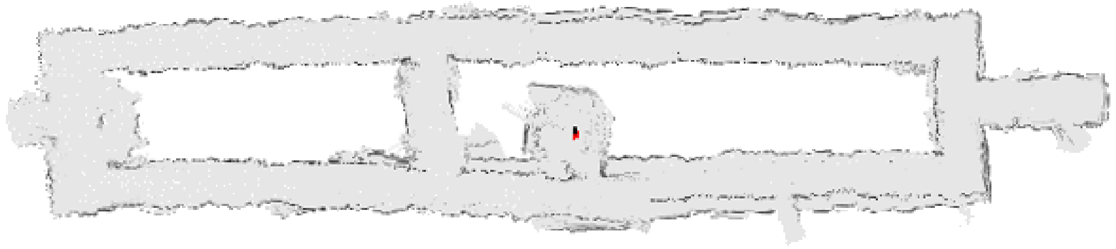
Figura 7: Comparação entre a estimativa de topologia do SDP-SLAM e do SegSLAM através de experimentos no ambiente simulado.

Os mapas resultantes das experiências no ambiente real, são apresentados na Figura 8. Aqui, visto que o ambiente é maior do que o testado anteriormente, a diferença entre os resultados dos dois métodos é mais visível. Com a estimativa de topologia do SegSLAM, não é possível fechar o ciclo, como mostrado em (a) e (b). Os mapas resultantes não são bons, tanto utilizando 5 partículas com 30 amostras e utilizando 10 partículas com 60 amostras, apesar do melhor resultado obtido com a segunda configuração. Comparativamente, o mapa apresentado em (c), obtido pelo SDP-SLAM com 30/5 partículas, é melhor do que os mapas produzidos pelo SegSLAM, apesar de conter erros de alinhamento. O resultado melhora quando o número de partículas aumenta, conform podemos ver em (d), onde o SDP-SLAM usa 60/10 partículas.

A média e o desvio padrão do erro do ICP durante os experimentos em ambiente simulado são mostrados na Tabela 1(a). O erro é menor utilizando a estimativa de topologia


(a) SegSLAM - 30/5 part.


(b) SegSLAM - 60/10 part.


(c) SDP-SLAM - 30/5 part.

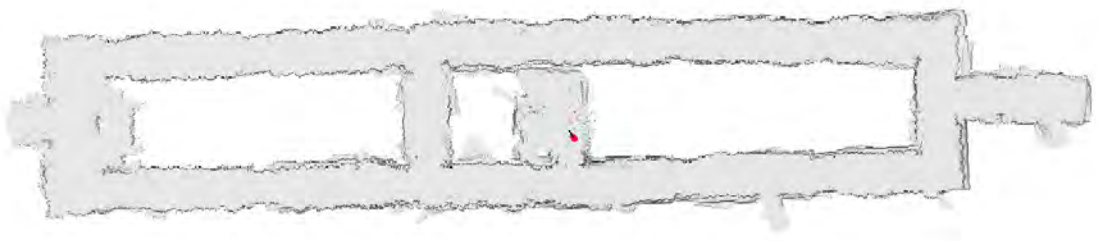

(d) SDP-SLAM - 60/10 part.

Figura 8: Comparação entre a estimativa de topologia do SDP-SLAM e do SegSLAM através de experimentos no ambiente real.

do SDP-SLAM do que com a estimativa de topologia do SegSLAM. O melhor resultado é obtido com a configuração de 60/10 partículas no SDP-SLAM. O erro durante os experimentos em ambiente real são mostrados na Tabela 1(b). Os valores são maiores do que na Tabela 1(a), mas novamente, o erro é menor quando escolhe-se o SDP-SLAM.

	SegSLAM		SDP-SLAM	
Partículas	μ	σ	μ	σ
30/5	1.43	0.84	0.66	0.34
30/10	1.26	1.04	0.58	0.36
60/5	1.23	1.18	0.61	0.40
60/10	1.15	1.22	0.48	0.45

(a) Ambiente simulado

	SegSLAM		SDP-SLAM	
Partículas	μ	σ	μ	σ
30/5	8.41	2.20	4.50	1.37
30/10	7.55	2.24	4.28	1.48
60/5	7.04	2.48	4.12	1.53
60/10	6.63	2.56	3.86	1.62

(b) Ambiente real

Tabela 1: Média e desvio padrão do erro do ICP no SegSLAM e no SDP-SLAM durante os experimentos.

Conclusão

Neste trabalho, apresentamos um algoritmo de SLAM baseado em submapas chamado SDP-SLAM (*Segmented Distributed Particle SLAM* / SLAM Segmentado com Partículas Distribuídas). Os resultados obtidos nos experimentos mostraram que o SDP-SLAM gera soluções melhores do que o DP-SLAM original, utilizando-se um número muito menor de partículas. Também foram realizados experimentos comparando o SDP-SLAM com o SegSLAM. A avaliação do processo de estimativa de topologia mostrou que o método de fato busca soluções com baixos erros de alinhamento. Conforme medido nos experimentos, o erro associado às amostras de combinações de submapas tende a diminuir ao longo do tempo.

Como trabalho futuro, pretende-se melhorar o SDP-SLAM alterando a atualização de submapas para considerar as informações associadas a outros submapas. Isto provavelmente irá melhorar a qualidade dos submapas, mas irá reduzir as possibilidades de combinações de submapas. A ideia é a de aplicar tal estratégia somente quando um conjunto de submapas estiver bem estabelecido (por exemplo, depois de fechar um ciclo perfeito no ambiente). Assim, estes submapas seriam componentes permanentes das soluções globais.

Outro trabalho futuro está relacionado ao processo de casamento de submapas. O nosso método extrai pontos a partir do espaço livre, ao invés de extrair pontos a partir dos obstáculos. Pretendemos analisar se o uso dos dois conjuntos de pontos pode melhorar a qualidade do processo de casamento de submapas.