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

VITOR AUGUSTO MACHADO JORGE

Enabling Loop-Closures and Revisits in Active SLAM Techniques by using Dynamic Boundary Conditions and Local Potential Distortions

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

Advisor: Prof. Dr. Edson Prestes

Porto Alegre
May 2017

CIP — CATALOGING-IN-PUBLICATION

Jorge, Vitor Augusto Machado

Enabling Loop-Closures and Revisits in Active SLAM Techniques by using Dynamic Boundary Conditions and Local Potential Distortions / Vitor Augusto Machado Jorge. – Porto Alegre: PPGC da UFRGS, 2017.

160 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2017. Advisor: Edson Prestes.

1. Active SLAM. 2. BVP. 3. Integrated Exploration. 4. Potential Rails. 5. Ouroboros. 6. SLAM. I. Prestes, Edson. II. Título.

“If I have seen farther than others,
it is because I stood on the shoulders of giants.”
— Sir Isaac Newton

AGRADECIMENTOS

My life in Porto Alegre, was not easy. In particular, the idea of starting a Ph.D. study terrified me from the beginning. I had many reasons, which I am not going to detail here, to quit even before I started. I could never ever even start such a monumental enterprise without three fundamental people: my parents, Jorge and Tânia, and my advisor, Edson Prestes. They were the only people who did not forsaken me during these hard times. Finally, this and other works, published in international conferences and journals, were executed in tight cooperation with my friend Renan Maffei, to whom I hold strong affinity and exchange many ideas. It may be terrible to find yourself in an environment where those surrounding you are competitors instead of colleagues. It is my pleasure to say that the Phi Robotics and Research Group is nothing like that. I also cannot forget to thank all the people in the group who also worked with me, teaching me and also making this burden less painful.

ABSTRACT

Truly autonomous robots must know the environment in order to execute complex tasks. In unknown environments, the robot must construct a map and localize itself using noisy proprioceptive and exteroceptive sensors. This is problematic, since the partial and possibly inaccurate map of the environment will be used to correct localization errors. This important problem of mobile robotics is known as Simultaneous Localization and Mapping (SLAM). When a robot autonomously execute a SLAM algorithm concurrently with an exploration strategy, this problem is called Active SLAM or Integrated Exploration. One of the main challenges behind both these problems is the treatment of loop closures. While the robot traverses unknown regions or sparse environments, the robot pose and the map may not be properly corrected due to lack of information. When this happens, the uncertainties about the map and the robot pose increase, which may lead to unrecoverable SLAM errors. On the other hand, when a loop is closed successfully, these uncertainties drastically decrease. Therefore, path chosen to explore the environment can considerably improve or degrade the quality of both localization and mapping. One well known way to explore the environment is the adaptation of the Boundary Value Problem (BVP) for the Laplace Equation and Dirichlet boundary conditions. Even though it is easy to implement, resulting in smooth exploration trajectories, it does not carefully address SLAM errors, since it follows a gradient decent which not always allows revisits, a crucial limitation for Active SLAM. Despite being a greedy frontier driven exploration strategy, we consider the flexibility of the BVP and Dirichlet boundary conditions still under-explored for Active SLAM. Our proposal is to modify the BVP Exploration algorithm to execute complex exploration behaviors, such as revisits and, in particular, loop-closures. We present two new approaches: the first makes use of a time driven boundary value condition together with potential distortions to generate loop closing behaviors and a potential field that never ceases to exist, even after the exploration ends; the second enables loop closure behaviors with BVP by taking advantage of potential propagation in unknown space generated by a pair of dynamic boundary conditions functioning as virtual walls and goals. Both approaches take advantage of a local optimization that uses the Voronoi Skeleton to reduce the computational cost of the algorithm. Tests in real and simulated environments using a Pioneer 3DX show that the proposed approaches present better results when compared with competing approaches.

Keywords: Active SLAM. BVP. Integrated Exploration. Potential Rails. Ouroboros. SLAM.

Viabilizando Fechamento de Ciclos e Revisitas em Técnicas de SLAM Ativo usando Condições de Contorno Dinâmicas e Distorções de Potencial Locais

RESUMO

Robôs verdadeiramente autônomos devem conhecer o ambiente para executar tarefas complexas. Em ambientes desconhecidos o robô deve concorrentemente construir o mapa do ambiente e se localizar usando sensores proprioceptivos e exteroceptivos imprecisos. Isto é problemático, uma vez que o mapa parcial e possivelmente incorreto do ambiente será usado para corrigir erros de localização. Este problema importante da robótica móvel é conhecido como Localização e Mapeamento Simultâneos (SLAM). Quando um robô autonomamente executa o algoritmo de SLAM concorrentemente com uma estratégia de exploração, o problema passa a se chamar SLAM Ativo ou Exploração Integrada. Um dos principais desafios por trás destes problemas é o tratamento de fechamento de ciclos. Ao atravessar regiões desconhecidas ou ambientes esparsos, a pose do robô e o mapa podem não ser propriamente corrigidos por falta de informação. Quando isto acontece, as incertezas da posição do robô e do mapa aumentam, podendo levar a erros irrecuperáveis. Por outro lado, quando o ciclo é fechado corretamente, estas incertezas diminuem consideravelmente. Portanto, a escolha do caminho para explorar o ambiente pode drasticamente melhorar ou degradar a qualidade do mapeamento e da localização. Uma técnica bem conhecida de exploração de ambientes é a adaptação do problema de valor de contorno (BVP) para a equação de Laplace e condições de contorno de Dirichlet. Apesar de ser fácil de implementar, resultando em trajetórias de exploração suaves, esta técnica não endereça cuidadosamente erros de SLAM, uma vez que ela segue a descida do gradiente, o que pode não possibilitar revisitas, uma limitação crucial para o SLAM Ativo. Mesmo sendo uma técnica de exploração gulosa e direcionada a fronteiras, consideramos que a flexibilidade do BVP e condições de contorno de Dirichlet ainda são pouco exploradas. Nossa proposta é modificar o algoritmo de Exploração por BVP para executar comportamentos complexos, tais como revisitas e, em particular, fechamentos de ciclo. Apresentamos duas novas abordagens: a primeira faz uso de uma condição de contorno direcionada pelo tempo combinada a distorções de potencial para gerar comportamentos de fechamento de ciclo, além de um potencial que nunca cessa de existir, mesmo após o ambiente ter sido completamente explorado; a segunda, propicia o fechamento de ciclos aproveitando a propagação do potencial em regiões desconhecidas, através de um par dinâmico de condições de contorno que funcionam como obstáculos e objetivos virtuais. Ambas abordagens aproveitam o Esqueleto de Voronoi do ambiente para reduzir o custo computacional do algoritmo. Testes em ambientes reais e simulados usando o robô Pioneer 3DX mostram que as técnicas apresentadas apresetam melhores resultados quando comparadas a técnicas concorrentes.

Palavras-chave: SLAM Ativo; BVP; Exploração Integrada; Potential Rails; Ouroboros; SLAM.

LIST OF ABBREVIATIONS AND ACRONYMS

1D	One-dimensional
2D	Two-dimensional
ND	N-dimensional
BVP	Boundary Value Problem
BLUE	Best Unbiased Linear Estimator
DBN	Dynamic Bayes Network
DP	Distributed Particle
DP-Map	Distributed Particle Map
DP-Mapping	Distributed Particle Mapping
DP-SLAM	Distributed Particle SLAM
EKF	Extended Kalman Filter
EM	Extended Map
KL	Kullback-Leibler (distance)
PDE	Partial Differencial Equation
RBPF	Rao-Blackwellized Particle Filter
RGB-D	Red, Green, Blue, and Depth
SIR	Sampling Importance Resampling
SDP-SLAM	Segmented Distributed Particle SLAM
SLAM	Simultaneous Localization and Mapping
SPLAM	Simultaneous Planning, Localization, and Mapping
TDBVP	Time-Driven Boundary Value Problem

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUÇÃO

Robôs são máquinas de propósito geral que podem agir por conta própria no ambiente autonomamente ou sob controle de outros agentes (PRESTES et al., 2013). Nos últimos anos seu uso se espalhou por diferentes domínios incluindo aplicações militares, humanitárias, civis e industriais. O uso de robôs e sistemas não tripulados com propósitos militares já são comuns e continua a evoluir (YAMAUCHI, 2004; LEE et al., 2010; NASKAR et al., 2011; MAXWELL; LARKIN; LOWRANCE, 2013; BHAT; MEENAKSHI, 2014) em guerras, exoesqueletos e próteses artificiais (VOTH, 2004). Em esforços humanitários e de resposta à emergências (CASPER; MURPHY, 2003; MURPHY; PRATT; BURKE, 2008; OKEREAFOR et al., 2013; MADHAVAN et al., 2014; MADHAVAN et al., 2015), robôs podem potencialmente poupar vidas humanas de situações perigosas, atuando em condições extremas que humanos normais não suportariam. Considerando aplicações civis, podemos salientar o uso de robôs em ambientes domésticos, que evoluiu de uma simples automação residencial (THRING, 1968) para robôs aspiradores de pó (LIU et al., 2004; HASAN; ABDULLAH-AL-NAHID; REZA, 2014) que podem agora limpar sua casa enquanto você estiver fora. Atualmente, uma importante conquista é o advento de carros autônomos (JO et al., 2014; JO et al., 2015; PETTERSSON; KARLSSON, 2015; SHIM et al., 2015), que vêm apresentando rápido desenvolvimento e poderão ser empregados em larga escala em poucos anos (ROSS, 2014). Na indústria, vemos mundialmente que robôs não são apenas um dos fatores chave de desempenho, mas também o símbolo de superioridade tecnológica. Não é surpresa que a pesquisa relacionada a robôs industriais esteja crescendo ao longo dos anos, juntamente com um crescimento expressivo do mercado[1] de em torno de 15% por ano. Da mesma forma, o uso crescente de robôs em aplicações civis está revelando a importância da navegação autônoma (DU et al., 2011). Atualmente, robôs estão em todo lugar e a maioria dos campos científicos estão de alguma forma inter-relacionados com a robótica.

Escopo

É o alvorecer de robôs autônomos. Contudo, o seu emprego permanece um desafio e o ambiente onde eles operam tem papel fundamental nisso. Sem uma boa representação do ambiente, um robô não tem como conhecer sua posição, ou àquelas de outros objetos presentes no ambiente. Objetivamente, isso significa que um robô não saberá onde ir para exe-

[1] <www.therobotreport.com>

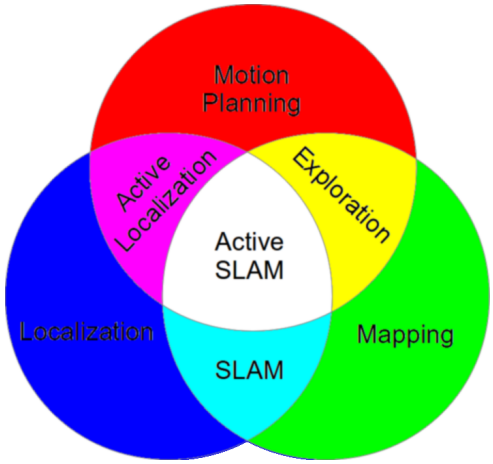
cutar mesmo a tarefa mais simples de interação com o ambiente. E mais, como a complexidade do ambiente pode limitar drasticamente o nível de autonomia de um sistema autônomo, esta é até mesmo considerada uma das dimensões usadas para medir o nível de autonomia de tais sistemas (HUANG et al., 2005). Não surpreendentemente, qualquer robô verdadeiramente autônomo depende de uma representação fiel do ambiente (em outras palavras, um mapa), para executar tarefas complexas (AMIGONI, 2008), seja ele um carro autônomo, mantendo-se na faixa de trânsito correta e escolhendo para onde ir; um aspirador de pó robô, decidindo quais lugares devem ser limpos; um robô de patrulha, selecionando a próxima área a patrulhar; ou um robô industrial, cooperando com humanos para atingir determinado objetivo.

Apesar disso, em muitos casos, o mapa do ambiente não está disponível, e robôs devem ser capazes de construir um de forma autônoma. Isso, por si só, é um desafio devido às incertezas presentes nas leituras dos sensores usados para detectar objetos ou características presentes no ambiente. Além disso, há erros cumulativos em sensores de estimativa de movimento causados pela falta de precisão nos mesmos — erros de odometria e telemetria. Erros de estimativa de movimento podem complicar o mapeamento uma vez que eles levam a erros de localização. Como um robô pode construir um mapa se ele não tem certeza de onde está? Uma forma de endereçar este problema é associar sua posição com a posição de assinaturas específicas presentes no ambiente e gravadas em visitas anteriores ao mesmo local. Quando um robô detecta tal assinatura, é possível corrigir seu erro de localização. Infelizmente, em muitas situações, o robô deve fazer isso enquanto está construindo o mapa, um problema conhecido como Mapeamento e Localização Simultâneos (SLAM) (THRUN; BURGARD; FOX, 2005).

Há abordagens online para o problema de SLAM (THRUN; BURGARD; FOX, 2005), tais como o Filtro de Partícula Rao-Blackwellizado (RBPF) (DOUCET et al., 2000; HAHNEL et al., 2003; GRISETTI; STACHNISS; BURGARD, 2005; GRISETTI; STACHNISS; BURGARD, 2007) o filtro de Kalman Estendido (JULIER; UHLMANN, 1997; HUANG; DISSANAYAKE, 2007; THRUN; BURGARD; FOX, 2005), assim como abordagens de SLAM completas, tais como o algoritmo GraphSLAM (GRISETTI et al., 2010). A diferença entre estas duas abordagens é que o SLAM online considera apenas o estado corrente do robô (e o mapa) para calcular a próxima estimativa de localização e mapeamento, enquanto que no SLAM completo, todos os estados (do robô e do mapa) são usados o tempo todo (veja o Capítulo 4).

Abordagens podem ainda ser classificadas de acordo com o uso ativo de estratégias de exploração para melhorar os resultados do SLAM (SICILIANO; KHATIB, 2008; THRUN; BURGARD; FOX, 2005; AMIGONI, 2008). Isto é, temos o SLAM simples (tradicional, ou apenas SLAM) quando o método de SLAM não possui conexão com o planejamento de movi-

Figure 1.1: Distinção entre SLAM e Exploração Integrada (SLAM Ativo)(MAKARENKO et al., 2002).



mento do robô. Por outro lado, quando os resultados combinados do mapeamento e da localização de alguma forma influenciam o planejamento de movimento, temos o SLAM Ativo — que é também chamado de exploração integrada[2] (ver Figura 1.1). Esta distinção é importante pois algoritmos de SLAM podem apenas corrigir erros de localização usando informação dos sensores exteroceptivos disponíveis e, dependendo de onde o robô decidir ir, a qualidade do mapa construído pode melhorar ou piorar. Assim, conduzir o robô a locais apropriados pode melhorar o resultado do algoritmo de SLAM (ver Capítulo 5). O SLAM Ativo tem o objetivo de explorar o ambiente eficientemente, mas escolhendo ações que podem potencialmente melhorar o resultado do mapeamento. Infelizmente, durante o processo de exploração, é difícil determinar onde o robô deve ir, pois pode não ser possível saber com antecedência onde cada caminho diferente levará, como um longo corredor, um ambiente amplo e vazio, um beco sem saída, ou um ciclo. O ciclo é um exemplo de ponto de interesse distinto para algoritmos de SLAM (STACHNISS; HAHNEL; BURGARD, 2004; STACHNISS; GRISETTI; BURGARD, 2005a; STACHNISS; GRISETTI; BURGARD, 2005b). Se o robô revisita um ciclo, uma revisita a uma região previamente conhecida é realizada e assim erros de localização e mapeamento podem ser corrigidos. Por outro lado, uma falha ao tentar realizar tal procedimento pode resultar em erros de odometria cada vez maiores até um ponto onde algoritmos de SLAM isoladamente não sejam mais capazes de corrigir a localização e o mapa.

[2]Durante todo o texto, ambos os termos, SLAM Ativo e Exploração Integrada, serão usados de forma intercambiável. Contudo, alguns autores também usam o termo Planejamento, Localização e Mapeamento Simultâneos (SPLAM).

Entre as diferentes estratégias encontradas na literatura de algorítmos puramente exploratórios, ressaltamos a adaptação do problema de valor de contorno (BVP) envolvendo a equação de Laplace para o processo de exploração (PRESTES et al., 2002; PRESTES et al., 2003; PRESTES; ENGEL, 2011). O BVP tradicional possui apenas dois tipos de condições de contorno: obstáculos e objetivos (regiões inexploradas). O valor de potencial para cada uma das condições de contorno é definido *a priori* por uma função específica de valor constante. A implementação do algoritmo é direta; não existe mínimo local no campo potencial; e o método resulta em um caminho suave que guia o robô até a região inexplorada mais atraente (maior e/ou mais próxima). Contudo, um robô usando BVP nunca volta a uma região previamente explorada e, portanto, nunca fecha ciclos intencionalmente. Um robô usando exploração por BVP só faz revisitas se tais regiões estão no caminho da região inexplorada que está atraindo o robô. Prestes and Engel (2011) ainda desenvolveram uma maneira de distorcer o campo potencial gerado pela exploração por BVP. Apesar disso, a distorção do potencial não pode intencionalmente executar revisitas, uma vez que o movimento do robô segue a decida do gradiente gerado pelo algoritmo.

Objetivo

Esta Tese busca desenvolver novos comportamentos exploratórios modificando as características do algoritmo de Exploração por BVP e trabalhos relacionados para endereçar o problema de SLAM Ativo.

Consideramos a exploração por BVP uma técnica compreensiva e genérica que pode ser adaptada de formas diferentes para que um robô mapeie um dado ambiente. Atualmente, a técnica é apenas uma abordagem gulosa que não pode oferecer benefícios ao processo de exploração integrada. Os dois pilares da exploração por BVP são as condições de contorno e as distorções de potencial. A hipótese dessa tese é a de que diferentes construtos de condições de contorno combinados com distorções de potencial locais podem melhorar o resultado do processo de Exploração Integrada. Em particular, essa Tese busca comportamentos que tornem possíveis revisitas assim como fechamentos de ciclos no processo de Exploração Integrada.

Nesta Tese focamos no uso inovador de distorções de potencial dinâmicas e a criação de novas condições de contorno, não associadas a valores constantes intrinsicamente ligadas às fronteiras entre o espaço conhecido e desconhecido, para gerar comportamentos complexos. Estas funções devem ser capazes de ligar e desligar as condições de contorno assim como controlar seu valor, formato e posição. Da mesma maneira, essa Tese busca uma estratégia para

o controle autônomo de distorções de potencial locais que possam assistir o robô na escolha das trajetórias de exploração mais apropriadas em relação ao algoritmo de exploração por BVP tradicional. Especificamente, os parâmetros de distorção e condições de contorno devem ser manipulados apropriadamente para mover o robô em direção a regiões de interesse. O resultado prático destas modificações são condições de contorno artificiais que possibilitam a criação de uma variedade de comportamentos, quando comparados àqueles dos algoritmos de exploração atuais.

Contribuições desta Tese

Esta Tese apresenta resultados (JORGE et al., 2015; MAFFEI et al., 2014) que mostram que a exploração por BVP pode ser modificada para melhorar o resultado de abordagens de SLAM Ativo. Em particular, essa Tese mostra que o uso e a manipulação correta de condições de contorno e distorções de potencial locais podem gerar novos e interessantes comportamentos exploratórios, tais como revisitas e fechamentos de ciclos. Além disso, essa Tese mostra que a combinação destas modificações com um esqueleto de Voronoi do ambiente pode reduzir consideravelmente o custo computacional do algoritmo. Tal estratégia elimina ainda o risco de achatamento[3] do campo potencial (ZHANG et al., 2010; SILVEIRA, 2015), uma vez que o algoritmo pode ser aplicado em uma janela local.

A primeira contribuição dessa Tese é o algoritmo *Potential Rails* (MAFFEI et al., 2014), que executa o fechamento de ciclos usando distorções de potencial locais (PRESTES; ENGEL, 2011). O algoritmo faz uso de uma função dependente do tempo aplicada sobre um esqueleto de Voronoi do espaço livre construído a cada iteração do algoritmo, resultando em campo potencial que varia com o tempo, na forma de uma condição de contorno complementar, onde a grade e as células pertencentes ao esqueleto de Voronoi contém informação sobre o tempo da última visita do robô a elas. Então, usando esta nova condição de contorno, o campo potencial gerado não apenas guia o robô em direção a regiões desconhecidas, mas também àquelas não visitadas há mais tempo. E ainda, o método nunca converge para um campo potencial achatado, mesmo que não existam mais regiões inexploradas fazendo fronteira com a região livre, já que sempre

[3]Em teoria, uma função harmônica, tal como a equação de Laplace, não possui mínimos locais, Ainda sim, a solução para tais equações é computada numericamente, o resultado do campo escalar pode conter erros de ponto flutuante. No escopo de movimentação de robôs, esta situação ocorre quando há passagens estreitas subsequentes, ou longos corredores estreitos no caminho entre o robô e o objetivo. Neste caso, quando o campo harmônico é computado sobre uma grade regular, podem haver muitas células com o mesmo valor escalar truncado, o achatamento, onde deveria haver uma descida suave em direção ao menor valor (objetivo) de potencial. Nestes casos, isso levará a problemas para determinar a direção que o robô deve ir – ou seja, descer o gradiente.

haverá uma célula mais antiga para revisitar. Quando o processo de exploração termina, o robô é imediatamente guiado para aquelas regiões que não foram visitadas a mais tempo, em um comportamento de ciclo supostamente perpétuo – que pode ser interessante para robôs de patrulha e exploração.

Outra contribuição é o algoritmo *Ouroboros* (JORGE et al., 2015), uma técnica de exploração integrada que usa condições de contorno virtuais e dinâmicas, funcionando como pontos de atração e barreiras dinâmicas (um muro virtual), que seguem o robô para possibilitar o controle do processo de fechamento de ciclos usando como base o algoritmo de problema de valor de contorno. *Ouroboros* faz uso de um esqueleto de Voronoi construído a cada instante sobre o espaço livre e desconhecido para codificar a informação global do mapa e também auxiliar no processo de fechamento de ciclos. Experimentos mostram que a técnica proposta é capaz de efetivamente controlar todo o processo de fechamento de ciclos de forma robusta, mesmo em ambientes esparsos quando comparada a outras técnicas de fechamento de ciclos. Por fim, durante o fechamento de um ciclo, nós associamos a incerteza do processo de SLAM a condições de contorno específicas que fazem o robô se manter ou deixar o loop, o que gerou uma versão melhorada da técnica, *Ouroboros* 2.0, que foi comparada com o Ouroboros tradicional e uma técnica de Stachniss, Hahnel and Burgard (2004) onde mostramos que o *Ouroboros* 2.0 é mais robusto que ambas em termos de mapeamento e localização.

Essa Tese foi escrita em inglês e é apresentada da seguinte forma. O Capítulo 2 apresenta a introdução, definição do problema, objetivos e contribuições. Os Capítulos 3 e 4 apresentam o background teórico que suporta esta Tese. Os trabalhos relacionados de exploração integrada são apresentados no Capítulo 5. Os Capítulos 6 and 7 apresentam as estratégias desenvolvidas, Potential Rails e *Ouroboros*, juntamente com resultados experimentais associados a ambas as estratégias. No Capítulo 8 concluímos o trabalho discutindo suas implicações e direções futuras.

2 INTRODUCTION

Robots are general purpose machines that can act on their own in the environment autonomously or under the control of other agents (PRESTES et al., 2013). In the recent years their use is spreading across different domains, including but not limited to military, humanitarian, civil, and industrial applications. The use of robotics and unmanned systems for military purposes is now common place and continues to evolve (YAMAUCHI, 2004; LEE et al., 2010; NASKAR et al., 2011; MAXWELL; LARKIN; LOWRANCE, 2013; BHAT; MEENAKSHI, 2014) in areas such as modern warfare, exoskeletons, and artificial prosthetics (VOTH, 2004). In many humanitarian and emergency response and recovery efforts (CASPER; MURPHY, 2003; MURPHY; PRATT; BURKE, 2008; OKEREAFOR et al., 2013; MADHAVAN et al., 2014; MADHAVAN et al., 2015), robots can potentially spare human lives of dangerous situations, acting in extreme conditions that humans cannot stand. Considering civilian applications, we can highlight the use of robots in home use, which evolved from simple house automation (THRING, 1968) to autonomous cleaning robots (LIU et al., 2004; HASAN; ABDULLAH-AL-NAHID; REZA, 2014) that can now clean the house for you while you are away. Currently, one important achievement is the advent of autonomous cars (JO et al., 2014; JO et al., 2015; PETTERSSON; KARLSSON, 2015; SHIM et al., 2015), which are undergoing rapid development and may be deployed in large scale in a few years (ROSS, 2014). In the industry, we see worldwide that robots are not only a key performance factor but also a symbol of technological superiority. It is no surprise that research related to industrial robots is growing throughout the years together with an expressive market growth[1] of about 15% a year. In the same way, the increase in the use of robots in service and domestic applications is drawing attention to the importance of autonomous navigation (DU et al., 2011). Robots are getting everywhere and the majority of scientific fields are somehow interrelated to robotics.

Scope

It is the dawn of autonomous robots. However, the deployment of autonomous robots remains a challenge and the environment where they operate plays a fundamental role in that. Without a good representation of the environment, a robot cannot know its position, or those of the other objects present in the environment. Ultimately, it means that the robot will not know where to go to execute a given task. Moreover, since the environment complexity can

[1] <www.therobotreport.com>

drastically limit the autonomy level of unmanned systems, it is sometimes considered as one of the dimensions used to measure the autonomy level of such systems (HUANG et al., 2005). Not surprisingly, any truly autonomous robot relies on a faithful representation of the environment, i.e., a map, to accomplish complex tasks (AMIGONI, 2008), be it a self-driving car, keeping itself in the appropriate road lane and selecting where to go; an autonomous vacuum cleaner, deciding what places must be cleaned; a surveillance robot, selecting the next area to patrol; or an industrial robot, cooperating with humans to achieve a certain goal.

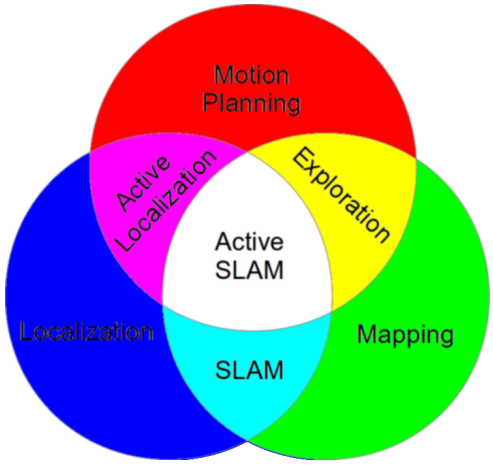
Still, in many cases, a map of the environment is not available, and robots must be able to construct one autonomously. This, by itself, is a challenge due to uncertainties present in the sensor readings, used to detect objects or features in the environment, and to cumulative errors caused by inaccurate motion estimation sensors — e.g., odometry and telemetry errors. Motion estimation errors can further complicate mapping, since they lead to localization errors. How does a robot construct a map if it cannot be sure where it is? One way to address this problem is to associate the position of the robot with specific environment signatures recorded from previously visited regions. When the robot detects such signatures in the environment, it is possible to correct its position. Unfortunately, in many situations, the robot must do so while the map is still under construction, a problem known as Simultaneous Localization and Mapping (SLAM) (THRUN; BURGARD; FOX, 2005).

There are on-line approaches to the SLAM problem (THRUN; BURGARD; FOX, 2005), such as the Rao-Blackwellized Particle Filter (RBPF) (DOUCET et al., 2000; HAHNEL et al., 2003; GRISETTI; STACHNISS; BURGARD, 2005; GRISETTI; STACHNISS; BURGARD, 2007) and the Extended Kalman Filter (JULIER; UHLMANN, 1997; HUANG; DISSANAYAKE, 2007; THRUN; BURGARD; FOX, 2005), as well as full SLAM approaches, such as the Graph-SLAM (GRISETTI et al., 2010) algorithm. The main difference between these approaches is that online SLAM considers the current state of the robot (and the map) to estimate the SLAM posterior, while in full SLAM all previous states are used (see Chapter 4).

Approaches can also be further classified according to the active use of exploration strategies to improve SLAM results (SICILIANO; KHATIB, 2008; THRUN; BURGARD; FOX, 2005; AMIGONI, 2008). That is, when SLAM has no connection with motion planning, we have plain SLAM. On the other hand, when the combined results of mapping and the localization processes somehow drive the robot exploration, we have Active SLAM — also called integrated exploration[2] (see Figure 2.1). This distinction is important because SLAM algorithms can only correct localization errors using information from the available exteroceptive sensors

[2]Throughout this work, both terms, Active SLAM and Integrated Exploration, will be used interchangeably. However, some authors also use the term Simultaneous Planning, Localization, and Mapping (SPLAM).

Figure 2.1: Distinction between SLAM and Integrated Exploration (Active SLAM) (MAKARENKO et al., 2002).



and depending on where the robot decides to go, map quality may decrease or increase. Thus, driving the robot to proper locations can improve the result of the SLAM algorithm (see Chapter 5). Active SLAM aims to explore the environment efficiently, but choosing actions which can potentially improve mapping accuracy. Unfortunately, during the exploration process, it is difficult to choose where to go, since it may not be possible to know in advance where a path will lead, such as long corridors, a vast empty space, dead-ends or loops. Among the available places some are more interesting than others. A loop is one example of distinct point of interest (STACHNISS; HAHNEL; BURGARD, 2004; STACHNISS; GRISETTI; BURGARD, 2005a; STACHNISS; GRISETTI; BURGARD, 2005b). If the robot timely closes a loop, a revisit to a previously known region is performed and localization and mapping errors can be corrected. On the other hand, failure to do so might result in increasing cumulative odometry errors which may not be recoverable using SLAM techniques alone.

Among different strategies found in the literature for exploratory tasks, we highlight the adaptation of the boundary-value problem (BVP) for the Laplace equation to the exploration process (PRESTES et al., 2002; PRESTES et al., 2003; PRESTES; ENGEL, 2011). Traditional BVP exploration has only two types of boundary conditions, obstacles and goals (unexplored regions). The potential value of each of the boundary conditions is defined *a priori* by specific constant value functions. The implementation of the algorithm is straightforward; there is no local minima in the potential field; and the method results in a smooth path that guides the robot toward the closest/largest unexplored regions. However, a robot using BVP never goes back to a previously known region and, therefore, it never intentionally closes loops. A robot using BVP exploration only performs revisits if such regions are in the path of an unknown region

that is dragging the robot. Prestes and Engel (2011) devised a way to distort the potential field generated by BVP exploration. Nevertheless, a potential distortion cannot be used to intentionally perform revisits, since the robot motion will still follow the gradient descent generated by the algorithm.

Objective

This thesis aims to devise new exploratory behaviors modifying the characteristics of BVP exploration and related works to address the problem of Active SLAM.

We regard BVP exploration as a comprehensive and generic technique that could be adapted in many different ways to help a robot to map a given environment. As is, the technique is only a greedy approach which cannot offer much benefits to an integrated exploration process. The two pillars of BVP exploration are the boundary conditions and the potential distortions. We hypothesize that the use of different constructs of boundary conditions combined with local potential distortions can improve the results of the integrated exploration process. In particular, we are interested in behaviors that could enable revisits as well as active loop-closures during the exploration process.

In this thesis we focus on the innovative use of dynamic local potential distortions and creation of new boundary conditions, not associated to constant functions or bounded to the frontiers between known and unknown space, to generate complex behaviors. Those functions must be able to turn the boundary condition *on* and *off*, and also control their value, shape, and position. Similarly, we must devise a strategy for the autonomous control of local potential distortions that could assist the robot to elect more appropriate exploration trajectories when compared to the traditional BVP algorithm. Specifically, the distortion parameters must be chosen properly to move the robot toward the most interesting directions. The practical result of these modifications are artificial boundary conditions which enable the creation of vastly different behaviors, when compared to the current BVP exploration algorithms.

Contributions of this Thesis

This thesis presents results (JORGE et al., 2015; MAFFEI et al., 2014) showing that BVP Exploration can be modified to improve the results of Active SLAM approaches. In particular, we show that the proper use and manipulation of dynamic boundary conditions and

local potential distortions can generate new interesting exploration behaviors, such as revisits and loop-closures. Additionally, we show that the combination of these modifications with a Voronoi diagram of the environment can considerably reduce the computational cost of BVP, accelerating the convergence of the algorithm. Furthermore, it eliminates the risk of flattening[3] of the potential field (ZHANG et al., 2010; SILVEIRA, 2015), since the algorithm can be applied in a local window.

The first contribution of this thesis is the Potential Rails algorithm (MAFFEI et al., 2014), which performs loop-closure using local potential distortions (PRESTES; ENGEL, 2011). Potential Rails makes use of a time-dependent function applied to an on-line constructed Voronoi skeleton of the **free space**, resulting in a time-varying potential field that works as a complimentary boundary condition, where both the grid and skeleton cells encode the visiting time information. Then, using this new boundary condition, we generate a potential field that not only guides the robot to unknown regions, but also to those not visited for the longest time. Moreover, our method never converges to a flattened potential field[4], even if there are no more unexplored boundary cells, since there will always be an "oldest cell". When the integrated exploration is finished, the robot is immediately guided to those regions not visited for the longest time in an allegedly perpetual cycle – what can be interesting for patrol and exposition robots.

Another contribution is the *Ouroboros* algorithm (JORGE et al., 2015), an integrated exploration strategy which uses virtual dynamic boundaries, functioning as tractor points and dynamic shields (a virtual wall), that follow the robot to enable and control the loop-closure process using the BVP algorithm. *Ouroboros* makes use of an on-line constructed Voronoi skeleton of **free** and **unknown space** to keep track of global map information and also to assist in the loop-closure. Experiments show that the proposed method is able to effectively control the entire loop-closure process in a robust way, even in sparse maps when compared to BVP. Finally, during a loop closure, we associate the uncertainty about the SLAM posterior to specific boundary conditions that make the robot follow or leave a loop – i.e., leaving only when the uncertainty is reduced. This generated an improved version of our technique, *Ouroboros* 2.0, that together with Ouroboros, were compared with the technique from Stachniss, Hahnel and Burgard (2004), which is one of the state of the art techniques on loop closure. This Thesis

[3]In theory, an harmonic function, such as the Laplace equation, does not have local minima. Still, when its solutions is computed numerically, the resulting scalar field may contain floating-point errors. In the scope of robot motion, this situation typically happens when there are subsequent narrow passages, or a long narrow corridor, in the path between the robot and the goal. In this case, when the harmonic field is computed over a discrete regular grid, there may be many cells of the grid having the same truncated scalar value, flattening, where there should be a smooth decrease toward the lowest (goal) value. In such cells, it will lead to problems to compute the direction the robot should go – i.e., the gradient descent.

[4]This problem occurs only using Dirichlet boundary conditions in a global map.

shows that *Ouroboros* 2.0 is the most robust technique in terms of SLAM results – i.e. best localization and mapping.

This thesis is presented as follows. Chapters 3 and 4 present the theoretical background supporting this Thesis. The related work on integrated exploration is presented in Chapter 5. Chapters 6 and 7 present both strategies, Potential Rails and *Ouroboros*, along with experimental results for both strategies. In Chapter 8 we conclude this work discussing its implications and future directions.

3 THEORETICAL FOUNDATION

Mathematical Nomenclature

In this section, in Table 3.1, we define the required nomenclatures for this thesis.

Terms	Meaning
$\mathcal{R}$	A robot
$\mathbb{R}^n$	The n-dimensional Real space.
$\mathbb{V}^n$	The n-dimensional vector space.
$p,\ q,\ s$	Points in $\mathbb{R}^n$.
A, B, D, T	Constants.
$\alpha, \beta, \gamma, \theta$	Angles.
$\overrightarrow{\mathbf{u}},\ \overrightarrow{\mathbf{v}},\ \overrightarrow{\mathbf{h}},\ \overrightarrow{\mathbf{w}}$	Vectors in $\mathbb{V}^n$.
$\mathcal{W}$	The environment $\mathcal{W} \subseteq \mathbb{R}^n$ where the robot is.
$\mathcal{O}$	The subset of the environment, $\mathcal{O} \subseteq \mathcal{W}$, occupied by obstacles.
$\mathcal{F}$	The subset of the environment, $\mathcal{F} \subseteq \mathcal{W}$, which is known and not occupied by obstacles.
$\mathcal{U}$	The subset of the environment, $\mathcal{U} \subseteq \mathcal{W}$, which is unknown to the robot.
$SE(n)$	The special Euclidean group (SPONG; HUTCHINSON; VIDYASAGAR, 2006), where n is the number of orthonormal vectors which define rotations.
$\mathcal{C}$	The configuration space $\mathcal{C} \subseteq \{\mathbb{R}^n \times SE(n)\}$.
$\mathcal{C}_{obs}$	The subspace, $\mathcal{C}_{obs} \subseteq \mathcal{C}$, which belongs to the obstacle configuration space (**C-obstacle**).
$\mathcal{C}_{free}$	The subspace, $\mathcal{C}_{free} \subseteq \mathcal{C}$, where the robot can operate without collisions.
$\mathcal{C}_{unk}$	The unknown portion of the configuration space.
$\mathbf{x}$	A robot pose, $\mathbf{x} \in \mathcal{C}$.
$\mathbf{x}_{1:t}$	A trajectory corresponding to a sequence of poses at discrete time steps from 1 to t.
$\mathbf{u}$	The state of proprioceptive sensors.
$\mathbf{z}$	The state of exteroceptive sensors.

$\mathbf{z}_{1:t}$	The sequence of discrete exteroceptive sensor states of a robot over a period of time t.
$\mathcal{A}(\mathbf{x})$	The portion of the environment occupied by the robot at pose $\mathbf{x}$.
$\mathcal{M}$	The region that the robot must explore, it may include known mapped regions and unknown regions.
$\mathbf{m}$	The discrete map, usually a regular grid with a given resolution.
$\mathcal{T}$	A transformation which moves the robot from one pose to another in the environment.
$\mathbf{H}, \mathbf{P}, \mathbf{X}$	Definition of matrices.
$\mathcal{Z}(\mathbf{x})$	The area $\mathcal{Z} \subseteq \mathcal{W}$ covered by a range finder scan $\mathbf{z}$ at a pose $\mathbf{x}$.
$\partial_{\mathcal{D}}\mathcal{X}$	The subset encompassing the boundaries of the set $\mathcal{X}$ which make frontier with set $\mathcal{D}$.
$\mathcal{D}_{\mathcal{N}}$	Whenever a set $\mathcal{D}$ has the subscript $\mathcal{N}$, it represents a *new* set which was changed due to operations applied on it, e.g., it can be a subset or superset of $\mathcal{D}$.
$\boldsymbol{\nu}(p)$	The potential field at a point $p \in \mathbb{R}^2$.
$\nabla\, ^2\boldsymbol{\nu}(p)$	The Laplace operator applied to the potential field at point $p \in \mathbb{R}^2$.
$\nabla\, \boldsymbol{\nu}(p)$	The gradient vector of the potential field at point $p \in \mathbb{R}^2$, i.e., $\nabla\, \boldsymbol{\nu}(p) = \frac{\partial \boldsymbol{\nu}(p)}{\partial x} + \frac{\partial \boldsymbol{\nu}(p)}{\partial y}$.
$\boldsymbol{H}(.)$, $\boldsymbol{EM}(.)$	Whenever a bold italic symbol appears, it represents a function. It may contain one or more letters, e.g., $\boldsymbol{log}$, $\boldsymbol{sin}$, $\boldsymbol{EM}$, $\boldsymbol{H}$, and so on.
$\boldsymbol{cl}(\mathcal{X})$	The minimum closed set containing a set $\mathcal{X}$. If $\mathcal{X}$ is closed $\boldsymbol{cl}(\mathcal{X}) = \mathcal{X}$.
$\boldsymbol{int}(\mathcal{X})$	The interior of a given set $\mathcal{X}$. If $\mathcal{X}$ is an open set, $\boldsymbol{int}(\mathcal{X}) = \mathcal{X}$
$\boldsymbol{p}(x)$, $\boldsymbol{q}(x)$	Two possible representations of the probability of x.

In the remainder of this chapter we formalize the required background information for this thesis. We present basic concepts required for the problem of autonomous robot exploration, extending some of the basic notations from Lavalle's book (LAVALLE, 2006) regarding the definitions associated to configuration space. We also make extensive use of notations from Thrun et al. (THRUN; BURGARD; FOX, 2005) for the robot pose and sensors states.

The Robot and its Configuration Space

As mentioned in Chapter 2, the environment itself imposes a challenge to the autonomy of a robot. In this section, we formally introduce the configuration space, which depends among other things: on the set of obstacles present in the environment; on the robot kinematic constraints; and, on the space the robot occupies in the environment. The configuration space allows the description of the useful workspace available to the robot along with the explicit definition of motion constraints. Here, we are interested in the description of rigid and unarticulated robots[1].

Consider an holonomic[2] autonomous ground robot, $\mathcal{R}$, in a given environment, $\mathcal{W} \subseteq \mathbb{R}^2$ (see Fig. 3.1 (a)). $\mathcal{R}$ is in a given position, $p_R \in \mathcal{W}$, and orientation given by an orthonormal base $\mathbf{H}_R \in SE(2)$ in relation to the reference frame of the world. In practice, the heading vector is a unit vector, $\overrightarrow{\mathbf{h}} \in \mathbb{V}^2$, where $\overrightarrow{\mathbf{h}} = (\boldsymbol{cos}(\theta), \mathbf{sin}(\theta))$, where θ is the yaw angle, defined by $\mathbf{H}_R$, of the robot in the plane. Thus, we can define a robot pose as the pair $\mathbf{x} = (p_R, \mathbf{H}_R)$ (see Fig. 3.1 (b)). Note that $\mathcal{R}$ occupies a region in the environment, given a pose $\mathbf{x}$, which we define as the subspace $\mathcal{A}(\mathbf{x}) \subseteq \mathcal{W}$. Furthermore, a partition of $\mathcal{W}$ is occupied by physical objects — such as doors, walls, and so on — that work as obstacles for $\mathcal{R}$. The partition of the environment occupied by obstacles defines the obstacle subspace, $\mathcal{O} \subseteq \mathcal{W}$. By definition, a robot cannot go to a place where there are obstacles, independently of its size. Unfortunately, $\mathcal{R}$ is never a point in the environment and depending on its position and orientation, it may collide with obstacles even if $p_R \notin \mathcal{O}$. We highlight that $\mathcal{O}$ and the space occupied by the robot, $\mathcal{A}(\mathbf{x})$, are closed sets[3]. Thus, the free space, $\mathcal{F} \subseteq \mathcal{W}$ with $\mathcal{F} \cap \mathcal{O} = \emptyset$, is an open set — since its limit points belong to obstacles. These definitions have important implications in the context of this work[4]. For instance, we can mathematically define when a robot hits an obstacle as

$$\boldsymbol{int}(\mathcal{O}) \cap \boldsymbol{int}(\mathcal{A}(\mathbf{x})) = \emptyset \text{ and } \mathcal{O} \cap \mathcal{A}(\mathbf{x}) \neq \emptyset, \tag{3.1}$$

[1] Note that the configuration spaces for non-rigid and articulated robots — e.g., non-rigid robots (GAYLE; LIN; MANOCHA, 2005), snake robots (ROLLINSON; BUCHAN; CHOSET, 2011), robotic arms (TSAI; HUANG, 2009), and so on — are usually more complex to handle than those of single rigid bodies (LATOMBE, 2008).

[2] An holonomic robot is a robot which has zero kinematic constraints (SIEGWART; NOURBAKHSH, 2004).

[3] Open and closed sets are similar to open and closed intervals. Examples of open and closed intervals are $(0, 1) \subset \mathbb{R}$ and $[1, 2] \subset \mathbb{R}$ respectively.

[4] For example, we can describe the environment, $\mathcal{W}$, as a topological space. For that purpose, we define the subset of the environment corresponding to the partition of free space of the environment, $\mathcal{F} \subseteq \mathcal{W}$, as an open set. This subspace satisfies the following properties (HOCKING; YOUNG, 2012): the union of any number of partitions of free space is an open set; the intersection of finite partitions of free space is an open set; and, $\mathcal{F}$ and the empty set, $\emptyset$, are open. In contrast, we define the partition of space occupied by obstacles, $\mathcal{O} \subseteq \mathcal{W}$, as a closed set where the following properties hold (HOCKING; YOUNG, 2012): the intersection of any number of obstacles is closed; the union of any number of obstacles is closed; and, $\mathcal{O}$ and the empty set, $\emptyset$, are closed. Note that the empty set is an open and closed set. Thus, we can describe the environment as a topological space.

where $\boldsymbol{int}()$ returns the interior of a space. Hence, when the boundaries of the robot and the obstacle intersect, a collision occurs.

Suppose that we need to know if a planned path, a sequence of poses $\mathbf{x}_{1:t}$, is collision free[5] (see Fig. 3.1 (a)). All poses of $\mathcal{R}$ in the environment, i.e., all the pairs combining a position and orientation, define the configuration space, $\mathcal{C} \subseteq \mathbb{R}^2 \times SE(2)$. The first step to find collision free paths is to identify two disjoint sets of poses: the set of poses that result in collisions, i.e., the obstacle configuration space, $\mathcal{C}_{obs} \subset \mathcal{C}$; and the free configuration space, $\mathcal{C}_{free} \subseteq \mathcal{C}$, which corresponds to the workspace of the robot[6]. Then, $\mathcal{C}_{obs}$ is defined as

$$\mathcal{C}_{obs} = \{\mathbf{x} \in \mathcal{C} \mid \mathcal{A}(\mathbf{x}) \cap \mathcal{O} \neq \emptyset\}. \tag{3.2}$$

Eq. 3.2 formalizes the notion that invalid poses happen when the robot tries to occupy the space of the obstacle — e.g., the space occupied by $\mathcal{R}$ at pose $\mathbf{x}_2$, $\mathcal{A}(\mathbf{x_2})$, in Fig. 3.1 (b). Therefore, any pose where the space, $\mathcal{A}(\mathbf{x})$, occupied by the robot intersects the space of an obstacle belongs to $\mathcal{C}_{obs}$ (see Fig. 3.1 (c)). Hence, we can define the free configuration space as

$$\mathcal{C}_{free} = \mathcal{C} \backslash \mathcal{C}_{obs}, \tag{3.3}$$

where, $\mathcal{C}_{free}$ is an open set, given that $\mathcal{C}_{obs}$ is a closed set and $\mathcal{C}_{obs} \cap \mathcal{C}_{free} = \emptyset$. Note that the navigation considering the configuration space may involve a search in a high dimensional space. A non-isotropic robot could handle the constraints imposed by $\mathcal{C}$ case by case, analyzing all the available configurations, or handling a specific subset of cases, such as the worst case scenario[7]. The latter approach is equivalent to the projection of the configuration space, $\mathcal{C}$, into a representation of $\mathcal{W}$, or $\mathcal{N} \subseteq \mathbb{R}^2$, containing consolidated information about the obstacle configuration space, or

$$\zeta : \mathcal{C} \rightarrow \mathbb{R}^2. \tag{3.4}$$

Note that ζ is a surjective function, since $\forall \mathbf{x} \in \mathcal{C},\ \exists p \in \mathcal{W},\ \zeta(\mathbf{x}) = p$, whose purpose is to project a point in the configuration space into the environment. Then, we can define a new

[5]When we want to know if a pose of a robot results in a collision free situation, we have a **Findplace** (LOZANO-PEREZ, 1983) problem. When we want to do that for an entire sequence of poses, we call it a **Findpath** problem.

[6]Even though robots, in general, are not isotropic, those with such characteristic are usually easier to handle and to exemplify . In this thesis, the characteristics of the robot used in the experiments, Mobile Robots Pioneer 3DX, can be approximated to the isotropic 2D case — a circle with minimal radius r containing the robot.

[7]Handling only the worst case scenario may lead to limitations in the capacity the robot has to transport itself to other locations. For instance, an elongated robot may only fit in a narrow corridor in a specific set of poses, and the worst case scenario would forbid the use of such corridor by the robot.

obstacle region[8], $\mathcal{O}_\mathcal{N}$, as

$$\mathcal{O}_\mathcal{N} = \{\zeta(\mathbf{x}) \mid \mathbf{x} \in \mathcal{C}_{obs}\}, \tag{3.5}$$

which is used to modify the free space, eliminating forbidden positions as well as obstacles,

$$\mathcal{F}_\mathcal{N} = \mathbb{R}^2 \backslash \mathcal{O}_\mathcal{N}. \tag{3.6}$$

Eqs; 3.5 and 3.6 together compose a navigable representation of the environment, $\mathcal{N} = \mathcal{O}_\mathcal{N} \cup \mathcal{F}_\mathcal{N}$.

The practical result of such approach is the dilation of the thickness of obstacles, whose value depends on the dimensions of the robot, in the direction of free space[9]. This way, we can treat the robot as a point-wise object inside $\mathcal{F}_\mathcal{N}$. Fig. 3.1 (c) presents an example of $\mathcal{C}_{obs}$ (hatched lines) and $\mathcal{C}_{free}$ (white) for a circular robot, where the configuration space does not depend on the robot orientation. Observe that a collision will never occur as long as the robot is inside the free space at a position $p_R \in \mathcal{F}_\mathcal{N}$ at a distance $d > r$ from any obstacle (see Fig. 3.1 (d)).

These definitions are important for the formalization of the exploration problem in the next section.

Exploration and the Configuration Space

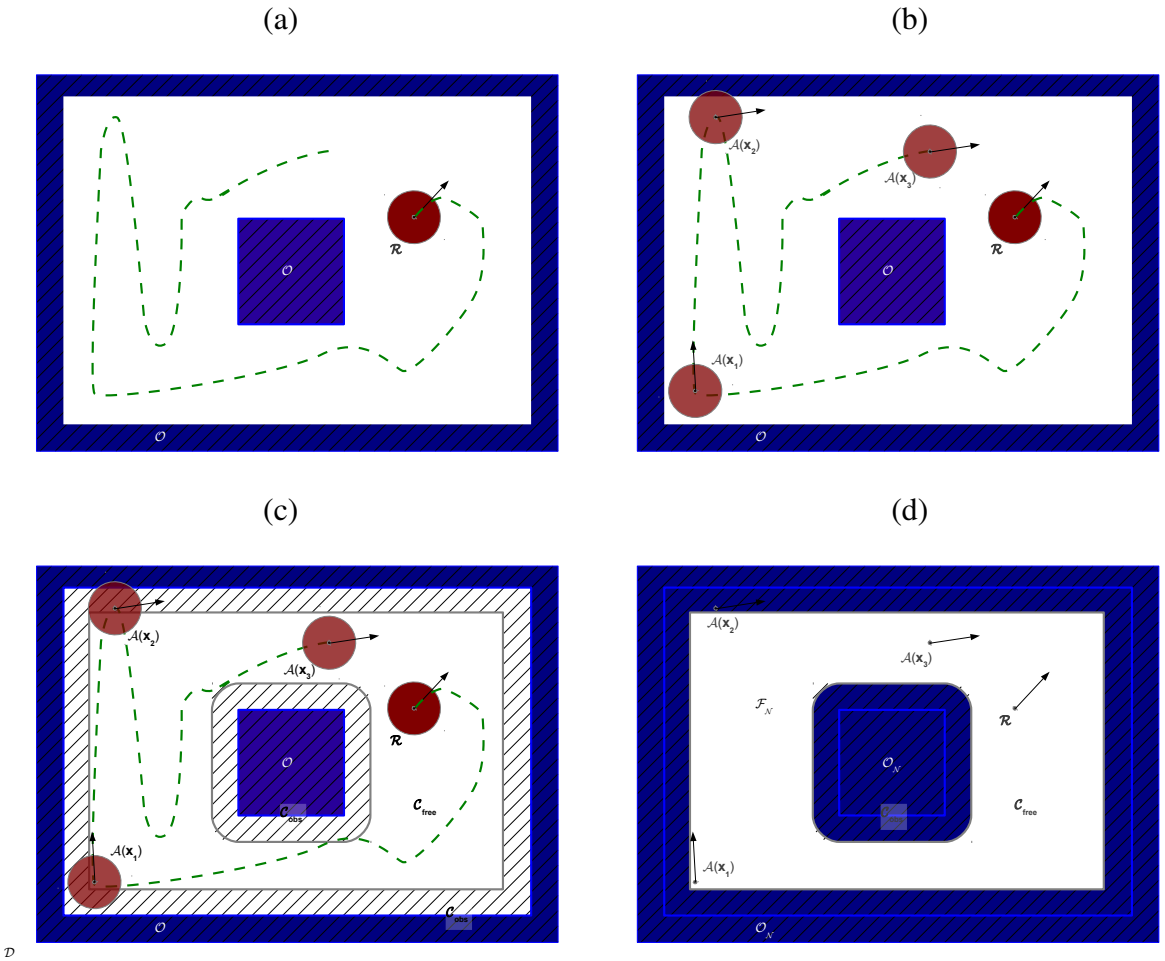
In this section we formally define the exploration problem. We extend the notions presented in the previous section created for path planning (LOZANO-PEREZ, 1983; LAVALLE, 2006) to the problem of exploration which considers unknown and partially unknown environments.

In common language, exploration is the process of discovering new places. This implies the explorer is willing to collect information about unknown environments. Once the required information is collected, the environment is considered known or explored. This definition is not so different from that of autonomous robot exploration. Recall that exploration is the intersection between robot planning and mapping (MAKARENKO et al., 2002) (see Fig. 2.1),

[8]Note that this representation aims to compact the information of a possibly high dimensional configuration space into a representation of the environment in $\mathcal{R}^2$. However, a circular robot without kinematic constraints has a configuration space which is orientation independent. Therefore, in this special case, $\mathcal{C}$ and $\mathcal{W}$ are in $\mathcal{R}^2$.

[9]The dilation of the obstacle subspace can be seen as a Minkowski sum(CARLONE; LYONS, 2014), $\mathcal{O} \oplus \mathcal{A}(r)$. This results in the enlargement of all obstacles considering a circle of radius r encompassing the robot. Similarly, the free space is contracted considering a Minkowski subtraction, i.e., $\mathcal{F} \ominus \mathcal{A}(r)$.

Figure 3.1: Figure (a) shows an environment, $\mathcal{W}$, consisting of: obstacles, $\mathcal{O}$ (hatched blue regions); the free space (white); with a circular robot, $\mathcal{R}$, and the region occupied by it (red circle); along with its heading (black arrow). The problem is to determine if a planned path (green dashed line) is feasible or results in collisions. In (b), we can see the space $\mathcal{A}$ occupied by the robot in three different poses: $\mathbf{x_1}$; $\mathbf{x_2}$; and $\mathbf{x_3}$. Observe that $\mathbf{x_2}$ results in a collision, making the path (green) unfeasible for the robot. $\mathcal{C}_{free}$ (white only region) and $\mathcal{C}_{obs}$ (hatched region) for $\mathcal{R}$ are shown in (c). Finally, we display the wall thickening approach (CONNOLLY; GRUPEN, 1993; ZELEK, 1998) which transfer the dimensions of the robot to the walls, enabling the adoption of a point-wise robot.



considering that there is no localization errors in the robot pose. Thus, we can say that autonomous exploration is the process of moving around until a desired area is fully covered. Observe that the robot does not know the environment and must choose by itself where to go. Furthermore, exploration implies a memory to store information about the environment. In other words, exploration requires the construction of a map, $\mathbf{m}$, representing the known regions of the environment. In the context of this thesis, the map captures information about obstacles and the free space available for the autonomous robot. This implies we have to model the unknown partition of the environment in our map also considering the configuration space $\mathcal{C}$.

Consider an omni-directional robot, $\mathcal{R}$, equipped with a set of sensors $S = \{S_p, S_e\}$, where S_p is a set of proprioceptive sensors, while S_e is an exteroceptive sensor, such that $S_e \cap S_p = \emptyset$. This robot is given the task of exploring the environment $\mathcal{W} \in \mathbb{R}^2$. Observe that $\mathcal{W}$ is a surface (a topological space) where the robot is placed to perform the exploration.

The autonomous exploration process requires a motion planning strategy, i.e., the robot must perform a set of ordered movements that make it go from a given pose to the next. The idea is to move in the environment exploring unknown regions until there is none left which is reachable by the robot. However, after a given command is issued, robot motion is influenced by internal and external factors that can lead to errors between the expected and resulting poses of the robot. Thus, an autonomous robot requires a set of proprioceptive sensors, S_p, to measure its own movement, including pose, velocity, acceleration. Examples of motion estimation sensors include, but are not limited to, odometers, accelerometers, gyroscope, and so on[10]. In particular, we want to use S_p to estimate the robot pose over time. In this Thesis we are interested in proprioceptive sensors used to measure small transitions between one pose and the next. That is, at each time-step, t, the robot executes a control command(or performs an odometry reading using S_p) $\mathbf{u}_t$ corresponding to a transformation, $\mathcal{T}$ which takes $\mathcal{R}$ from a pose $\mathbf{x}_{t-1}$ to the next, $\mathbf{x}_t$, with frequency f_p. That is,

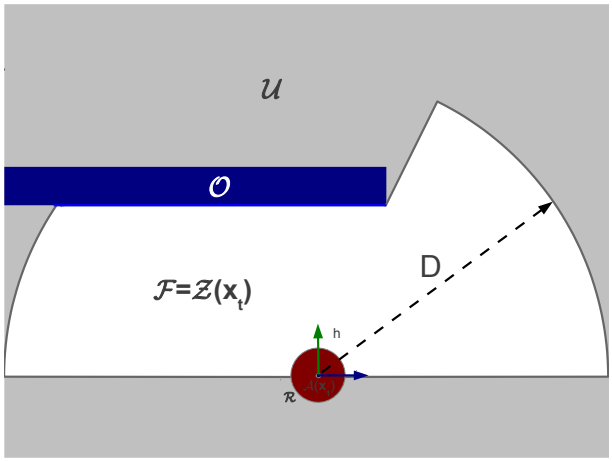
$$\mathbf{x}_t = \mathcal{T}(\mathbf{u}_t, \mathbf{x}_{t-1}), \tag{3.7}$$

Note that this is only true if $\mathbf{u}_t$ is precise. In practice, there is an uncertainty in the robot motion which must be taken into consideration. This is the primary cause of localization errors in an integrated exploration process. As a result, we cannot ensure that the poses encompassing a given trajectory, $\mathbf{x}_{1:t}$, are correct, which may lead to map construction errors. A key aspect of the integrated exploration process is the use of a motion model (THRUN; BURGARD; FOX, 2005; ELIAZAR; PARR, 2004b) to compensate systematic errors and to provide an uncertainty regarding the robot motion.

Another factor, which drastically influences the exploration process is how the robot perceives the environment and performs mapping. Such process involves the use of one or more exteroceptive sensors, S_e, which are used to detect features and obstacles in the environment. Exteroceptive sensors include cameras, sonar and laser range finders, bumpers, and so on. Here, S_e corresponds to a laser range finder which returns a set of distance readings from the sensor to eventual obstacles in the surroundings of $\mathcal{R}$, forming a scan, $\mathbf{z}$, which covers an area (see Fig 3.2), $\mathcal{Z}(\mathbf{x_e}) \subseteq \mathcal{W}$, where $\mathbf{x_e}$ is the pose of the exteroceptive sensor mounted on the robot.

[10] We consider that there is no sensor available to return the absolute localization of the robot, e.g., GPS.

Figure 3.2: An example of subregion covered by a robot using an exteroceptive sensor. The sensor enables the division of the environment, $\mathcal{W}$, into free, $\mathcal{F}$, and obstacle, $\mathcal{O}$, and the unknown configuration space, $\mathcal{U}$. Boundaries connecting obstacles or delimiting unknown regions separate the free subspace from the other two. Also, note that the region covered by the sensor reading is a closed set where the boundaries in lighter blue define obstacles while the boundaries in dark gray define the frontiers with unknown regions. D is the maximum distance covered by the exteroceptive sensor.



The covered area, $\mathcal{Z}(\mathbf{x_e})$, is usually larger than $\mathcal{A}(\mathbf{x})$ — we henceforth assume that the pose of the robot is the same of the exteroceptive sensor, i.e. $\mathbf{x_e} = \mathbf{x}$. Further, S_e has a given maximum range, D, which defines the maximum distance, $z \in \mathbf{z}$, with which a range finder can detect an obstacle[11].

Given an ideal range finder, S_e, every measurement, $z_{t,\beta}$, from a scan, $\mathbf{z}$, depends on the possible range finder orientation angles. A measurement, $z_{t,\beta}$, corresponds to a distance from the position, $p_s \in \mathbb{R}^2$, of S_e to the position, $p_o \in \mathbb{R}^2$, of the nearest obstacle, in an orientation β. Here, $z_{t,\beta} \in \mathbb{R}_{\geq 0}$ and $0 \leq \beta \leq \mathrm{B}$, where B is the maximum sensor aperture angle. Measurements are performed over a discrete time-step, t, with constant frequency, f_e. In addition, $z_{t,\beta}$ is associated to a measurement vector,

$$\overrightarrow{\mathbf{v}}_{t,\beta} = \frac{p_o - p_s}{||p_o - p_s||},$$

where $\overrightarrow{\mathbf{v}}_{t,\beta}$, is usually parallel to '$\mathcal{W}$ — recall that $\mathcal{W}$ is a plane. In our case, the central sensor reading, $\overrightarrow{\mathbf{v}}_{t,\frac{\mathrm{B}}{2}}$, is co-directional with the robot heading[12].

[11] For an autonomous robot, the characteristics of a range finder are equivalent to the vision of a diver exploring the ocean. The farther the diver can see, the easier it is to collect ocean information. In the same way, the farther the range of the sensor, the larger the exploration coverage for a robot.

[12] The robot has only one sensor, S_e, centered and fixed considering the robot heading. However, if the sensor was mounted on a moving base, this condition would not hold.

Measurement information is used to define a free region $\mathcal{Z}_{free}(t,\beta) \subseteq \mathcal{Z}(\mathbf{x}_t)$ given by

$$\mathcal{Z}_{free}(t,\beta) = \{p \in \mathcal{W} \mid p = p_e + b\overrightarrow{\mathbf{v}}_{t,\beta} \quad \text{and} \quad 0 < b < z_{t,\beta}\}, \tag{3.8}$$

Similarly, measurement information is used to define obstacle regions $Z_{obs} \subset \mathcal{Z}(\mathbf{x}_t)$ as

$$\mathcal{Z}_{obs}(t,\beta) = \{p \in \mathcal{W} \mid p = p_e + z_{t,\beta}\overrightarrow{\mathbf{v}}(t,\beta)\}. \tag{3.9}$$

Finally, if $z_{t,\beta} = D$, the maximum range, it means S_e did not detect obstacles in the supported range of the sensor. Due to its capability to detect obstacles from a distance, such sensors enable a wide coverage of the environment and the minimization of the risk of collisions, since obstacles are detected long before the robot reaches them. We define $\mathcal{Z}(\mathbf{x})$ as a closed set, so that we can extend Eq. 3.1 to define when the range finder detects a region belonging to an obstacle:

$$\boldsymbol{int}(\mathcal{O}) \cap \boldsymbol{int}(\mathcal{Z}(\mathbf{x})) = \emptyset \text{ and } \mathcal{O} \cap \mathcal{Z}(\mathbf{x}) \neq \emptyset. \tag{3.10}$$

Note that the range finder usually detects only the borders of the obstacles. With such sensors, we can create a map through the exploration process. Ideally, after the exploration process, the resulting map is equivalent to

$$\mathcal{M}_{1:t} = \bigcup_{t=0}^{T} \mathcal{Z}(\mathbf{x_t}), \tag{3.11}$$

where T is total exploration time, i.e., the mapped region is equivalent to the union of all exteroceptive sensor scans performed during the exploration period, T, in the corresponding poses at each time-step, t.

When the map of the environment is available, we have seen that a robot can be in two situations: either $\mathcal{R}$ is completely inside the free space, $\mathcal{F}$; or, according to Eq. 3.1, with its boundaries touching those of an obstacle. For the exploration activity, where the map is not available, we need to take into account the unknown region of the environment, $\mathcal{U} \subseteq \mathcal{W}$. Recall from Section 3.2 that obstacles, $\mathcal{O}$, and the space occupied by the robot, $\mathcal{A}(\mathbf{x}) \subset \mathcal{W}$, are closed sets, while the free space, $\mathcal{F}$, is an open set. As mentioned before, these characteristics are important for topological spaces. Hence, in order to maintain such characteristics for partial maps we must model the four possible types of boundaries which could occur (see Fig. 3.3):

(i) $\mathcal{O}$ and $\mathcal{F}$;

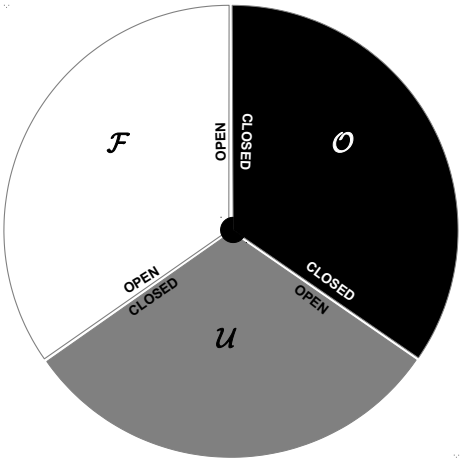
(ii) $\mathcal{O}$ and $\mathcal{U}$;

(iii) $\mathcal{F}$ and $\mathcal{U}$;

(iv) $\mathcal{O}$, $\mathcal{F}$, and $\mathcal{U}$.

In the first case, as expected, $\mathcal{O}$ is a closed set while $\mathcal{F}$ is an open set. However, in the second case, when in contact with $\mathcal{O}$, the boundary of $\mathcal{U}$ should be open. On the other hand, since $\mathcal{F}$ is an open set, the boundary of $\mathcal{U}$, when in contact with $\mathcal{F}$, must be closed. The fourth case considers a point whose frontiers involve $\mathcal{O}$, $\mathcal{F}$, and $\mathcal{U}$, where $\mathcal{O}$ is closed, while the other two boundaries, $\mathcal{F}$ and $\mathcal{U}$ are open sets. As a consequence, the behavior of the set $\mathcal{U}$ is neutral, i.e., it is neither completely closed nor open (HOCKING; YOUNG, 2012). Therefore, $\mathcal{O} \cap \mathcal{U} = \emptyset$, $\mathcal{F} \cap \mathcal{U} = \emptyset$, and $\mathcal{F} \cap \mathcal{O} = \emptyset$.

Figure 3.3: The environment $\mathcal{W}$ considering the unknown space, $\mathcal{U}$. Note that the free, $\mathcal{F}$, and obstacle, $\mathcal{O}$, subspaces define when the boundary of the unknown space should be open or closed, i.e., $\mathcal{U}$ is open in the frontiers with $\mathcal{O}$ and closed in the frontiers with $\mathcal{F}$. In the case of an intersection among the three, the boundary is closed in $\mathcal{O}$ while open in the other spaces (the central point in the figure exemplifies this.).



Logically, by definition, the interior of the region occupied by the robot can be considered explored and inside $\mathcal{F}$. However, its boundaries can now make frontier with $\mathcal{O}$ and also with $\mathcal{U}$. Therefore, while $\mathcal{U}$ is closed in the frontier with $\mathcal{F}$, it is open in the frontiers with $\mathcal{O}$. Thus, similarly to Eq. 3.1, when the robot gets in contact with an unknown region,

$$\boldsymbol{int}(\mathcal{U}) \cap \boldsymbol{int}(\mathcal{A}(\mathbf{x})) = \emptyset \ \textbf{and} \ \mathcal{U} \cap \mathcal{A}(\mathbf{x}) \neq \emptyset. \tag{3.12}$$

That is, the robot interacts with unknown regions in the same way it does with obstacles, through its boundaries. Evidently, the robot must rely on $\mathcal{Z}(\mathbf{x})$ to determine where the unknown regions are,

$$\boldsymbol{int}(\mathcal{U}) \cap \boldsymbol{int}(\mathcal{Z}(\mathbf{x})) = \emptyset \ \textbf{and} \ \mathcal{U} \cap \mathcal{Z}(\mathbf{x}) \neq \emptyset. \tag{3.13}$$

However, it will only happen when $z_{t,\beta} = D$, i.e., the measured distance is equal to the maximum range supported by S_e. In other words, as soon as the robot or the scan are occupy a region, such region becomes known with the exception of its boundaries which belong to obstacles or unknown regions, as stated by Equations 3.1, 3.10, 3.12, 3.13. As a result, when a robot moves toward the boundaries between $\mathcal{F}$ and $\mathcal{U}$, it is in fact going in the direction of unexplored regions.

Unfortunately, the environment remains constrained by the configuration space, $\mathcal{C}$ (see Fig. 3.15 (b)), which follows a pattern similar to $\mathcal{W}$. Thus, $\mathcal{Z}(\mathbf{x})$ defines a closed set which covers a region of the environment. In this way, we can rewrite Eq. 3.2 for the range finder as

$$\mathcal{C}_{obs} = \{\mathbf{x} \in \mathcal{C} \mid \mathcal{Z}(\mathbf{x}) \cap \mathcal{O} \neq \emptyset\}. \tag{3.14}$$

Then, the unknown configuration space, $\mathcal{C}$ can now be rewritten as:

$$\mathcal{C}(t) = \mathcal{C}_{obs}(t) \cup \mathcal{C}_{free}(t) \cup \mathcal{C}_{unk}(t), \tag{3.15}$$

where $\mathcal{C}_{unk}$ is the unknown partition of the configuration space. Another important aspect, which is now made explicit is that, exploration is a time-dependent process, since as the robot visits new regions, the amount of information collected about the environment will rise. In the same way, the amount of unknown information will drop until it reaches a minimum when the exploration is over. This means that the sets encompassing the environment vary as a function of time, t.

Again, we can also take the configuration space into consideration by transferring the dimensions of the robot to obstacles (CONNOLLY; GRUPEN, 1993; ZELEK, 1998) (see Fig. 3.15 (c)) and Equations 3.4, 3.5 and 3.6), which practically projects the configuration space in the map space. Thus, the configuration space is collapsed into to a representation of the environment in the form of thicker obstacles.

Similarly, during the exploration, a robot will acquire information about the free, $\mathcal{F}$, and obstacle regions present in the environment. Information will be also a function of time. As a result, during the exploration at a given instant, t, the map of the environment can be divided into three subsets,

$$\mathcal{M}(t) = \mathcal{O}(t) \cup \mathcal{F}(t) \cup \mathcal{U}(t), \tag{3.16}$$

where all elements are time dependent and $\mathcal{U}$ is an open closed set, just like its high dimensional counter part, $\mathcal{C}_{unk}$. As a consequence, environment assessment is limited to free space and

(thickened) obstacles edges, since S_e usually does not penetrate obstacles. Thus, the actual explored area is usually smaller than the area the robot is meant to explore (see Fig. 3.4 (d)).

According to Eq. 3.16, the exploration is divided into three subset of regions, $\mathcal{O}(t)$, $\mathcal{F}(t)$, and $\mathcal{U}(t)$. Note again that each of these regions can share frontiers with the other two. Even though we are interested in places where the robot can go, we must define clear boundaries for this space. Knowledge with respect to obstacles is important to avoid collisions, while that concerning unknown regions will be used to define the next regions to be explored. The free space and its frontiers with unknown and obstacle regions define the minimum closed set encompassing $\mathcal{F}$. It can be obtained applying the closure operator to the free space, i.e., $\boldsymbol{cl}(\mathcal{F})$. This function returns,

$$\boldsymbol{cl}(\mathcal{F}(t)) = \mathcal{F}(t) \cup \partial_{\mathcal{F}}\mathcal{O}(t) \cup \partial_{\mathcal{F}}\mathcal{U}(t), \tag{3.17}$$

where $\partial_{\mathcal{F}}\mathcal{O}(t) \subseteq \mathcal{O}(t)$ and $\partial_{\mathcal{F}}\mathcal{U}(t) \subseteq \mathcal{U}(t)$ are the closed partition of the obstacle and unknown subspaces that make boundaries with $\mathcal{F}(t)$. These two boundaries closing the free space are fundamental for the robot to know where to go and where to avoid. The exploration process builds a map incrementally, until

$$\boldsymbol{cl}(\mathcal{F}(t)) = \mathcal{F}(t) \cup \partial_{\mathcal{F}}\mathcal{O}(t), \tag{3.18}$$

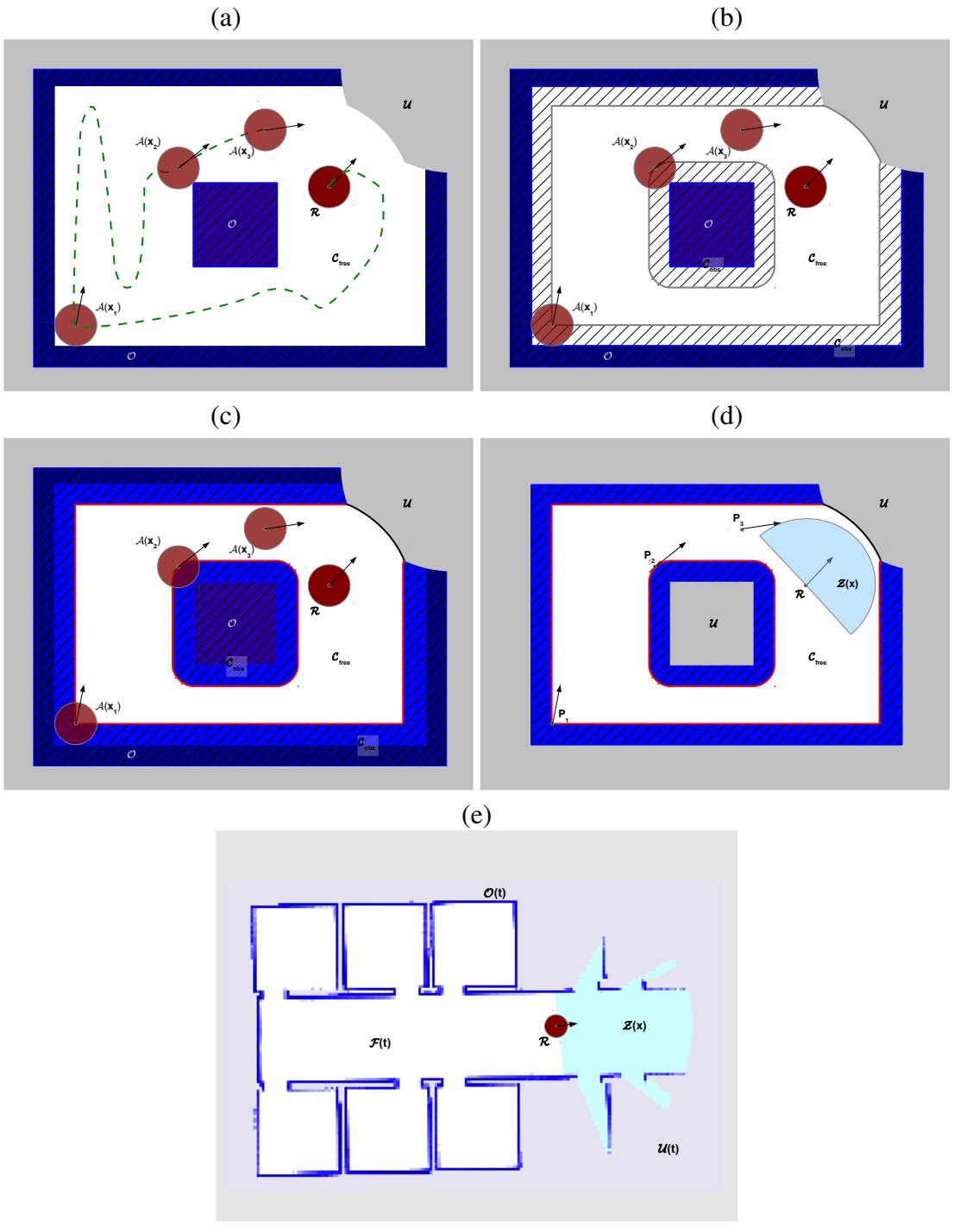
and $\partial_{\mathcal{F}}\mathcal{U}(t) = \emptyset$. That is, in ideal conditions, the exploration ends when the robot is no longer able to access unknown regions through the free space[13].

Note that sensors are not ideal. It means a sensor can perform imprecise or completely wrong measurements, affecting the precision of the map. For this reason, the probability of a point in the map being occupied by an obstacle is usually given as a function of a sequence of sensor readings[14]. Another problem is associated to the representation of environment information in a map. Fig. 3.4 (e) illustrates the effect of a map represented by a discrete 2D regular grid, where environment information is sampled in discrete two-dimensional (2D) cells. The sampling rate of such grid map is limited to the amount of memory and processing power available to the robot. Therefore, part of the information is lost during the discretization of the environment. As a consequence, $\mathcal{R}$ usually constructs a map, $\mathbf{m}$, representing $\mathcal{M}(t)$ which most likely does not correspond to an exact representation of the environment.

[13]Note that this may not hold if S_e is no longer ideal. A laser may erroneously detect free space in unreachable areas. In this case, the exploration ends when the space $\mathbf{cl}(\mathcal{F}(t)) \subseteq \mathcal{F}(t)$ — encompassing the region of free space where the robot is — no longer has frontiers with unknown boundaries, i.e. $\mathbf{cl}(\mathcal{F}(t)) \cap \partial_{\mathcal{F}}\mathcal{U}(t) = 0$.

[14]Note that in practice S_e is also noisy. Therefore, some model must be used to consider such noise, e.g, an inverse sensor model or Histogram in Motion (HIMM) (THRUN; BURGARD; FOX, 2005; MURPHY, 2000).

Figure 3.4: The partially explored map $\mathcal{M}(t)$ of a given environment $\mathcal{W}$ is shown in (a). The map is divided into three subspaces: the unknown region $\mathcal{U}$ (gray), the free space, $\mathcal{F}$ (white), and obstacle space (blue), $\mathcal{O}$. The configuration space for a circular robot is defined in (b) along with the wall-thickened transformation of the map in (c). A robot usually does not have sensors which penetrate obstacles, so Figure (d) presents what exteroceptive sensors, such as range finders, would detect (half-circle in light blue), i.e., the (thickened) boundaries of the obstacles. Figure (e) presents a real exploration process, where the unknown portion of space decreases with time as the robot moves and uses its sensors (light blue) to construct a map over a regular grid.



These definitions, as presented in the two previous sections, are useful in the context of this thesis for a few reasons. Firstly, they enable the specification of the exploration problem in a clear and objective way separating regions of interest which are part of the environment to be explored — i.e., obstacles, free, and unknown regions. Secondly, this strategy for the definitions is compatible with the frontier based approaches which are commonly used for exploration. Lastly, we believe these definitions enable the generalization of the exploration problem to more interesting behaviors, such as integrated exploration.

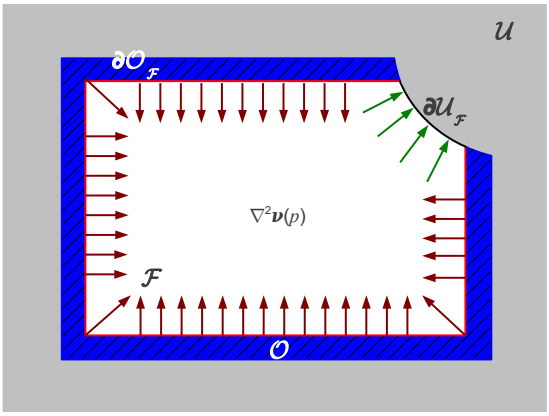
Exploration and the Boundary Value Problem (BVP)

In this section we mathematically formalize exploration as a BVP. For that purpose, it is necessary to model the behavior of a robot, $\mathcal{R}$, while it is mapping a finite environment, $\mathcal{W}$. The physical problem of exploration has two basic premises:

1. the robot should not hit obstacles;
2. and, at the same time, it should move toward unexplored regions.

In such fashion, the boundaries of obstacles should repel the robot, while the boundaries of unknown regions should attract it (see Fig. 3.5). Approaches which are guided by unexplored regions of the environment are commonly called frontier-guided strategies.

Figure 3.5: A figure showing the two basic premises of exploration at the boundaries of a region of interest. The robot must be driven toward the boundaries of unknown regions (green arrows), while at the same time being repelled by obstacles (red arrows). One way to estimate such behavior is to consider a BVP where the Laplace equation is solved inside the free region considering the boundary-values in the frontiers with obstacles and the unexplored regions.



The use of a numeric solution of a BVP for path planning arose with the work of Connolly and Grupen (1993). Prestes et al. (2002), proposed a methodology for the extension of the BVP path planner for the exploration problem. It consists of the application of the numeric solution of the boundary-value problem (BVP) involving the Laplace Equation, considering Dirichlet boundary conditions defined at obstacles and unknown regions.

During the exploration, we saw that the mapped region will correspond to the closure of the free space (see Eq. 3.17), since the robot, through its exteroceptive sensors, perceives $\mathcal{O}(t)$ and $\mathcal{U}(t)$ through their boundaries: the boundaries of obstacle regions, $\partial_{\mathcal{F}}\mathcal{O}(t)$, that make frontier with $\mathcal{F}(t)$; and the boundary, $\partial_{\mathcal{F}}\mathcal{U}(t)$, of the unknown part of the map with the free space, $\mathcal{U}(t)$. These two regions of the map are important, since Dirichlet boundary-conditions are those whose solution $\boldsymbol{\nu}(p)$ is prescribed to every point, p, of the boundaries of the domain. The values of the boundaries must be defined according the two above premises. The adopted approach (PRESTES et al., 2002; PRESTES et al., 2003) was to consider the potential at the boundaries of obstacles as

$$\boldsymbol{\nu}(p) = 1, \text{ on } \partial_{\mathcal{F}}\mathcal{O}(t), \tag{3.19}$$

i.e., a constant function equal to one on the boundary $\partial_{\mathcal{F}}\mathcal{O}$, representing a high potential. Conversely

$$\boldsymbol{\nu}(p) = 0, \text{ on } \partial_{\mathcal{F}}\mathcal{U}(t) \tag{3.20}$$

defines a low potential at the boundaries of unknown regions, $\partial_{\mathcal{F}}\mathcal{U}$, through a constant function equal to zero. Then, the numerical solution for the Laplace equation,

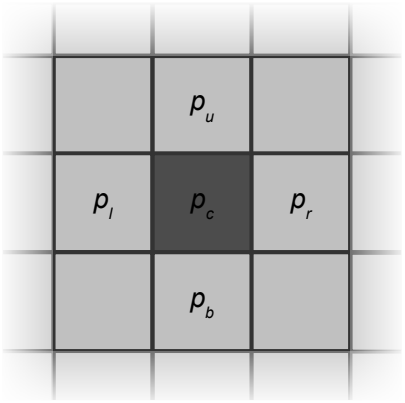
$$\nabla^2\, \boldsymbol{\nu}(p) = 0, \text{ on } \mathcal{F}(t), \tag{3.21}$$

is computed to obtain a scalar field of potential values at the known free space, $\mathcal{F}(t)$. BVP Exploration (PRESTES et al., 2002) uses the gradient descent of this potential field to smoothly guide the robot to unexplored regions in the environment and at the same time repel the robot from the walls.

Recollect that we are working with a discrete regular grid, $\mathbf{m}$, representing the region of interest. The potential is computed iteratively using the Gauss-Seidel method for the numerical approximation of Eq. 3.21 considering the Taylor expansion series for each squared cell of the grid at a position $p \in \mathcal{F}(t)$. This results in

$$\boldsymbol{\nu}(p) = \frac{\boldsymbol{\nu}(p_l) + \boldsymbol{\nu}(p_r) + \boldsymbol{\nu}(p_u) + \boldsymbol{\nu}(p_b)}{4}, \tag{3.22}$$

Figure 3.6: Representation of part of a regular grid, $\mathbf{m}$. The numerical solution of the potential field is computed in relation to a cell centered at $p_c \in \mathcal{F}$, considering the cells on its left, right, bottom and up, p_l, p_r, p_b, and p_u respectively.



where p_l, p_r, p_b, and p_u are respectively position of the cells at the left, right, top and bottom of the reference cell, p (see Fig. 3.6).

Trevisan et al. (2006) proposed a modification of the partial differential equation (PDE) presented in Eq. 3.21 to unbalance the arithmetic average resulting from Eq. 3.22. This distortion was devised to try to move the robot toward and also away from specific regions of interest during the exploration. The suggested example of modification can be written as

$$\nabla^2 \boldsymbol{\nu}(p) - \boldsymbol{\epsilon}\, \overrightarrow{\mathbf{w}} \cdot \nabla\, \boldsymbol{\nu}(p) = 0, \text{ on } \mathcal{F}(t), \tag{3.23}$$

where $\overrightarrow{\mathbf{w}} \in \mathbb{V}^2$ is a normalized perturbation vector defined *a priori*, and, in their implementation, ϵ is set *a priori* to 1. These modifications maintain the absence of local minima, and smooth paths (TREVISAN et al., 2006). The dot product, $\overrightarrow{\mathbf{w}} \cdot \nabla\, \boldsymbol{\nu}(p)$, sets the sign of the potential distortion.

Later, Prestes and Idiart (2009) studied the effect of different $\boldsymbol{\epsilon}$ values in the environment. The perturbation vector was set as the normalized gradient, i.e. $\overrightarrow{\mathbf{w}} = \frac{\nabla\, \boldsymbol{\nu}_h(p)}{||\nabla\, \boldsymbol{\nu}_h(p)||}$, of the harmonic potential considering Eqs. 3.21 and 3.22. Hence, Eq. 3.23 becomes

$$\nabla^2 \boldsymbol{\nu}(p) - \boldsymbol{\epsilon}(p)\, \frac{\nabla\, \boldsymbol{\nu}_h(p)}{||\nabla\, \boldsymbol{\nu}_h(p)||} \cdot \nabla\, \boldsymbol{\nu}(p) = 0, \text{ on } \mathcal{F}(t). \tag{3.24}$$

Note that the Eq. 3.24 now depends on the harmonic gradient which depends on another partial differential equation (PDE) that must be computed numerically. This way, two potentials would have to be computed in order to obtain the numerical solution of Eq. 3.24, what considerably

increases the computational cost of the method. The proposed solution was to adopt a partial solution of the harmonic potential field, iterating Eq. 3.22 a fixed number of times, in order to obtain a reasonable estimate of the normalized gradient. This estimate was used as the distortion vector in Eq. 3.24. Another way to see Eq. 3.24 is through the dot product of these two gradients, that is

$$\boldsymbol{cos}(\gamma) = \frac{\nabla\, \boldsymbol{\nu}_h(p) \cdot \nabla\, \boldsymbol{\nu}(p)}{||\nabla\, \boldsymbol{\nu}_h(p)||\, ||\nabla\, \boldsymbol{\nu}(p)||},$$

which results in

$$\nabla^2\, \boldsymbol{\nu}(p) - \boldsymbol{\epsilon}(p)\, \boldsymbol{cos}(\gamma)\, ||\nabla\, \boldsymbol{\nu}(p)|| = 0, \text{ on } \mathcal{F}(t). \tag{3.25}$$

Observe that the sign of the desired potential distortion in Eq. 3.25 is affected, according to the angle, γ, between these two vectors. A problem with this strategy is that the distortion diminishes as $\boldsymbol{cos}(\gamma) \rightarrow 0$. Besides, it is difficult to determine *a priori* the appropriate distortion vector, since we do not know beforehand where the gradient will point.

Prestes and Engel (2011) circumvented this unintentional behavior by always choosing a normalized distortion vector co-directional to the gradient vector, which also eliminates the need to precompute the harmonic potential field and removes the cosine effect from Eq.3.25, assuming $\boldsymbol{cos}(\gamma) = 1$. Another modification of the algorithm was to consider the sum of the absolute values of the components of the gradient vector, instead of gradient vector norm. This eliminates the need to compute a square root for every point in the grid, reducing the computational cost of the method. These modifications lead to the following equation:

$$\nabla^2\, \boldsymbol{\nu}(p) - \boldsymbol{\epsilon}(p) \left(\left|\left| \frac{\partial \boldsymbol{\nu}(p)}{\partial x} \right|\right| + \left|\left| \frac{\partial \boldsymbol{\nu}(p)}{\partial y} \right|\right| \right) = 0. \tag{3.26}$$

which can also be approximated numerically using Taylor expansion series to

$$\boldsymbol{\nu}(p_c) = \frac{\boldsymbol{\nu}(p_l) + \boldsymbol{\nu}(p_r) + \boldsymbol{\nu}(p_u) + \boldsymbol{\nu}(p_b)}{4} - \boldsymbol{\epsilon}(p) \frac{||\boldsymbol{\nu}(p_r) - \boldsymbol{\nu}(p_l)|| + ||\boldsymbol{\nu}(p_t) - \boldsymbol{\nu}(p_b)||}{8}. \tag{3.27}$$

This last modification transfers the inflexion of the distortion to a scalar value defined by $\boldsymbol{\epsilon}(p)$, which is now only scaled by the norm of the gradient. Knowing the values of the potential maxima, V_{max}, and minima, V_{min}, defined *a priori* at the boundaries, we must constrain $\boldsymbol{\epsilon}(p)$ to ensure the convergence of the solution of the PDE. For that purpose we must keep $V_{min} < \boldsymbol{\nu}(p_c) < V_{max}$ to avoid that $||\boldsymbol{\nu}(p_c)|| \rightarrow \infty$. Therefore, considering such interval, we get

$$V_{max} > \left(\frac{V_{max} + V_{min} + V_{max} + V_{min})}{4} - \boldsymbol{\epsilon}(p) \frac{||V_{max} - V_{min}|| + ||V_{max} - V_{min}||}{8} \right).$$

Considering that $V_{max} - V_{min} > 0$ always holds for $V_{max} \neq V_{min}$:

$$V_{max} > \left(\frac{2V_{max} + 2V_{min}}{4} - \boldsymbol{\epsilon}(p) \frac{V_{max} - V_{min}}{4} \right),$$

$$4V_{max} > 2V_{max} + 2V_{min} - \boldsymbol{\epsilon}(p)(V_{max} - V_{min}),$$

$$2(V_{max} - V_{min}) > \boldsymbol{\epsilon}(p)(V_{max} - V_{min}),$$

$$\boldsymbol{\epsilon}(p) < 2.$$

Similarly, while considering V_{min}:

$$V_{min} < \left(\frac{V_{max} + V_{min} + V_{max} + V_{min})}{4} - \boldsymbol{\epsilon}(p) \frac{||V_{max} - V_{min}|| + ||V_{max} - V_{min}||}{8} \right).$$

$$V_{min} < \left(\frac{2V_{max} + 2V_{min}}{4} - \boldsymbol{\epsilon}(p) \frac{V_{max} - V_{min}}{4} \right),$$

$$4V_{min} < 2V_{max} + 2V_{min} - \boldsymbol{\epsilon}(p)(V_{max} - V_{min}),$$

$$-2(V_{max} - V_{min}) < \boldsymbol{\epsilon}(p)(V_{max} - V_{min}),$$

$$\boldsymbol{\epsilon}(p) > -2.$$

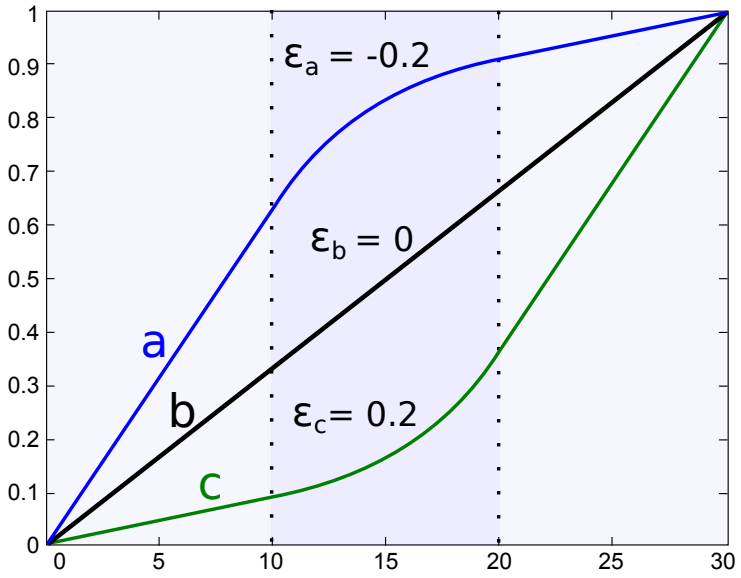
Therefore, the distortion parameter, $\boldsymbol{\epsilon}(p)$, should be constrained to the interval of

$$-2 < \boldsymbol{\epsilon}(p) < 2.$$

If $\boldsymbol{\epsilon} > 0$, the concavity of the potential field increases, while on the assumption that $\boldsymbol{\epsilon} < 0$, the convexity of the field increases. These effects can be observed in the 1D example shown in Fig. 3.7. The potentials of the two boundary points are set as $\boldsymbol{\nu}(0) = 0$ and $\boldsymbol{\nu}(30) = 1$, which are used to compute the intermediate potentials using three different values of $\boldsymbol{\epsilon}$ in the central region of the environment (slightly darker region between two dashed lines). When $\boldsymbol{\epsilon} = 0$, the potential does not present curvature, as shown by the straight line b. However, when $\boldsymbol{\epsilon} < 0$ (curve a), the curve of the potential is flatter near obstacles (high potential) and steeper close to the goals (low potential). On the other hand, when $\boldsymbol{\epsilon} > 0$ (curve c) the slope in the potential is increased near obstacles and decreased close to the goals.

By using only two fixed boundary conditions for obstacles and unexplored cells, BVP exploration is always guided toward the region with the fastest potential value decrease. This

Figure 3.7: Changes in the curvature of the potential field by varying the distortion parameter (ϵ) of in the central region of the chart. Curve b is the standard potential calculated without distortion. Curves a and c present the potential calculated respectively for negative and positive distortion parameters. The darker central region of the chart delimited between two dashed lines are those where the values of the distortion parameter, $\boldsymbol{\epsilon}(p)$, change. Note that when $\boldsymbol{\epsilon}(p) \neq 0$, the potential field is no longer a straight line – i.e., it is distorted.



approach works well when there is no uncertainty in the robot localization. However, an effective integrated exploration strategy balances the navigation in the direction of unexplored regions with revisiting tasks in order to improve the localization of the robot and the quality of the resulting map. Distortions can be applied on BVP exploration to favor the passage over specific regions. This is good, for instance, in sparse environments, where it is possible to keep the robot close to the walls to minimize localization errors. Yet, even in this case the robot always moves toward the regions with smaller potential. As a result, revisiting actions are unlikely to happen. It is worth mentioning that potential distortions were already applied to path-planning in a robot soccer competition (ROMERO et al., 2012) to generate different behaviors for multiple robots using a single solution of the BVP.

We begin revisiting BVP exploration and proposing a generalization of the method to address different problems, such as Active SLAM. Next, we reinforce the impact of loop-closures, introducing a new way to see local potential distortions and boundary conditions to close loops.

BVP Exploration Revisited

BVP Exploration (PRESTES et al., 2002; PRESTES et al., 2003; TREVISAN et al., 2006) comes traditionally from the pure exploration problem. Possibly for this reason, the current works about BVP Exploration are centered on the frontier between known and unknown space. A major aspect behind pure exploration is that localization errors are not considered during the map construction (MAKARENKO et al., 2002). In this case, greedy approaches such as BVP are sufficient for autonomous robot exploration, where mapping errors are restricted to the noise of exteroceptive sensors. Thus, with proper treatment of such noise, the exploration problem can be transposed to movements and trajectories with focus on the maximization of the information gain and mapping speed.

Unfortunately, even for considerably simple real world scenarios, localization errors arise from proprioceptive sensors, ending up in poor mapping. In fact, depending on the environment, even SLAM approaches may fail if the increase in the localization errors are not handled carefully during the mapping process. Therefore autonomous exploration migrates to an Active SLAM problem, where revisits are required and the total exploration time ceases to be the only determinant factor in the evaluation of an exploration strategy, since revisits and loop-closures are required to keep the map accurate. In this case, mapping quality becomes a serious factor in evaluation of techniques.

Currently, BVP Exploration supports two types of behavior: exploration (PRESTES et al., 2002; PRESTES et al., 2003) and wall-following (TREVISAN et al., 2006). In addition, potential distortions were conceived (TREVISAN et al., 2006; PRESTES; IDIART, 2009; PRESTES; ENGEL, 2011) to distort the traditional trajectory drawn from the algorithm. Such behaviors together with potential distortions are unable to lead the robot to regions where the potential is higher than the one in the current robot position. As a consequence, little or no control is possible for revisiting or closing loops, which are fundamental for SLAM approaches.

BVP Exploration has a solid mathematical background from partial differential equations. The flexibility emerging from such foundation can be observed in different fields, where it is applied to solve diverse problems. The core of this Thesis is the conviction that the core concepts behind BVP Exploration are more general than the related approaches presented so far. Our initial thoughts were guided by a fundamental question about BVP Exploration:
— Is it possible to intentionally make a robot go back to previously visited place modifying BVP Exploration in a pragmatic way?
We believe that the methodology we employ here to achieve this objective may assist the roboti-

cist in the design of new solutions to challenging tasks associated to autonomous robot mapping and also different motion problems.

In this Thesis we apply our ideas to the Active SLAM problem, where we have devised different loop-closure behaviors as well as a patrolling behavior which starts as soon as the exploration ends. Such behaviors can be obtained by the modification of the functions defining the potential field along the boundaries. We are interested in general functions which are not strictly constant or step functions (PRESTES et al., 2002; PRESTES et al., 2003; TREVISAN et al., 2006).

In order to generate such behaviors, we modify BVP Exploration to consider virtual boundary conditions which are designed to strategically delimit the environment according to the problem at hand. For that purpose, we also need to modify the functions defining the potential field along these boundaries to

$$\boldsymbol{\nu}(p) = \boldsymbol{f}(p) \quad on \quad \partial_{\mathcal{F}}\mathcal{X}(t). \tag{3.28}$$

where p is a point along one of the boundaries that make frontier with the free space, $\mathcal{F}$. I.e.,

$$\mathcal{X} = \bigcup_{i=0}^{F} \mathcal{X}_i,$$

where F is the total number of different regions, $\mathcal{X}_i$, making frontier with the free space, $\partial_{\mathcal{F}}\mathcal{X}_i$. These regions are not limited to the unknown regions and obstacles. As a matter of fact, the space pertaining a region $\mathcal{X}_i$ can even artificially occupy $\mathcal{F}$, $\mathcal{O}$, or $\mathcal{U}$, increasing or decreasing them and also generating new boundaries and motion patterns. One example of such new region can be the Voronoi diagram of the free space or even a circle surrounding the robot.

We also study the application of potential distortions in innovative ways when compared with previous works (TREVISAN et al., 2006; PRESTES; IDIART, 2009; PRESTES; ENGEL, 2011). In particular, we create local potential distortions in front of the robot, to direct the robot toward loop-closures. These modifications encompass a new BVP Framework which enables different behaviors using a single blueprint, i.e., the BVP, arbitrary boundary conditions, and local potential distortions.

In the next two Chapters we present the background on the SLAM and Active SLAM problems.

4 SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM)

This section formally presents the SLAM problem, describing its derivation using Bayesian filtering as a starting point. Then, different SLAM approaches are discussed, including: graph-based SLAM; Extended Kalman Filters (EKF); and particle filter approaches. We highlight the RBPF approach, with focus on the DP-SLAM algorithm, since the technique was used in both Integrated Exploration strategies described in this thesis.

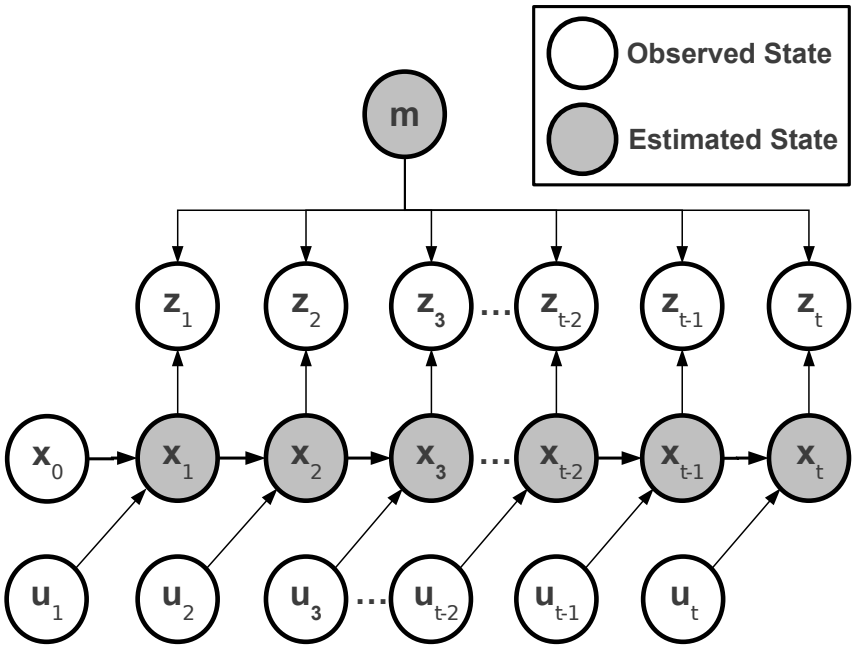
SLAM Introduction

Autonomous robot localization can be a difficult task due to errors in the proprioceptive sensors used to estimate robot poses. In order to reduce localization errors, a robot must use exteroceptive sensor measurements to extract features from the environment, which are matched to a given map to improve pose estimates. Unfortunately, in many practical cases, the accurate map of the environment is not available. Thus, before the accomplishment of any given task, a robot must construct a map and localize itself in the environment at the same time. The key problem is that the robot must concurrently be able to correct localization errors with an on-line map, which is constructed using corrupted information coming from both exteroceptive and proprioceptive sensors. In other words, the robot must use a corrupted map to correct corrupted poses and vice versa. This is why the SLAM problem is sometimes seen as a chicken or egg problem.

The SLAM problem can be seen as a Dynamic Bayesian Network (DBN), as shown in Fig. 4.1. Proprioceptive sensor measurements, $\mathbf{u}_{1:t}$, and exteroceptive sensor observations, $\mathbf{z}_{1:t}$, are the observed variables, while the sequence of robot poses, $\mathbf{x}_{1:t}$, and the map, $\mathbf{m}$, are the hidden variables which must be estimated from the observed variables. SLAM approaches can be classified differently depending on how one sees the problem. From a probabilistic perspective (THRUN; BURGARD; FOX, 2005), SLAM can be classified as on-line and full SLAM approaches.

In on-line SLAM, the current state has sufficient information for the estimation of the next, while the remainder of the trajectory can be ignored. In other words, what matters are the

Figure 4.1: Formulation of SLAM as a Dynamic Bayesian Network (THRUN; BURGARD; FOX, 2005). Observed states (measurements **z** and odometry **u**) are dependent on the hidden states (robot pose **x** and map **m**).



current pose and the map, **m**. The idea is to estimate the joint posterior probability of $\mathbf{x}_t$ and $\mathbf{m}$ at a given instant t, given the observations $\mathbf{z}_{1:t}$ and odometry measurements $\mathbf{u}_{1:t}$, or

$$p(\mathbf{x}_t, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{x}_0). \tag{4.1}$$

This implies the starting pose, $\mathbf{x}_0$, must also be given information.

Conversely, full SLAM estimates the joint posterior probability over the trajectory, $\mathbf{x}_{1:t}$, followed by the robot and the map — instead of just the current robot pose, $\mathbf{x}_t$. That is,

$$p(\mathbf{x}_{1:t}, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{x}_0). \tag{4.2}$$

Note that the on-line SLAM posterior can be derived from the full SLAM approach considering a multiple integral with respect to all the poses in the trajectory

$$p(\mathbf{x}_t, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \int \int \cdots \int \int p(\mathbf{x}_{1:t}, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})\, d\mathbf{x}_0\, d\mathbf{x}_1\, d\mathbf{x}_2\, d\mathbf{x}_3 \cdots d\mathbf{x}_{t-2}\, d\mathbf{x}_{t-1}. \tag{4.3}$$

Integrating Equation 4.3 incrementally we get

$$
\begin{aligned}
\boldsymbol{p}(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) &= \int\int\int \cdots \int\int \boldsymbol{p}(\mathbf{x}_{1:t}, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})\; d\mathbf{x}_0\; d\mathbf{x}_1\; d\mathbf{x}_2 \cdots d\mathbf{x}_{t-2}\; d\mathbf{x}_{t-1} \\
\boldsymbol{p}(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) &= \int\int \cdots \int\int \boldsymbol{p}(\mathbf{x}_{2:t}, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})\; d\mathbf{x}_1\; d\mathbf{x}_2 \cdots d\mathbf{x}_{t-2}\; d\mathbf{x}_{t-1} \\
\boldsymbol{p}(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) &= \int\int \cdots \int \boldsymbol{p}(\mathbf{x}_{3:t}, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})\; d\mathbf{x}_2 \cdots d\mathbf{x}_{t-2}\; d\mathbf{x}_{t-1} \\
&\vdots \\
\boldsymbol{p}(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) &= \int\int \boldsymbol{p}(\mathbf{x}_{t-2:t}, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})\; d\mathbf{x}_{t-2}\; d\mathbf{x}_{t-1} \\
\boldsymbol{p}(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) &= \int \boldsymbol{p}(\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})\; d\mathbf{x}_{t-1}
\end{aligned}
$$

By subsequently integrating the SLAM posterior over the previous pose estimates, we obtain the marginal probability for $\mathbf{x}_t$, given all previous poses $\mathbf{x}_{0:t-1}$.

Main approaches to solve the SLAM problem

As stated before, SLAM methods are usually divided in two major groups: full SLAM and on-line SLAM approaches. At every single time step, full SLAM always requires the complete set of measurements performed by the robot to determine the solution for the SLAM problem. Although on-line SLAM approaches can considerably reduce this burden by decreasing the state space, even the on-line SLAM equation[1],

$$
\boldsymbol{p}(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta\, \boldsymbol{p}(\mathbf{z}_t \,|\, \mathbf{x}_t, \mathbf{m}) \int \boldsymbol{p}(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \mathbf{u}_t)\, \boldsymbol{p}(\mathbf{x}_{t-1}, m \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) d\mathbf{x}_{t-1}, \quad (4.4)
$$

cannot be solved in its closed form, because it still has a high dimensional state space, encompassing the robot pose and possibly thousands of features associated to the map. When compared with full SLAM, the main advantage of on-line SLAM approaches is their incremental operation, which discards exteroceptive and proprioceptive sensor measurements as soon as they are used. Unfortunately, this strength is also a limitation. On-line SLAM makes use of the Markov Assumption to discard state variables[2]. An important implication of considering only

[1] This formula can be derived from Eq. 4.1 (see Appendix B) using the Markov Assumption (THRUN; BURGARD; FOX, 2005), Bayes rule, and the Law of Total Probability (see Appendix A). Note that in Eq. 4.4 η is a normalization constant.

[2] The Markov assumption is also called the Complete State Assumption. It states that the knowledge of the current state is enough to estimate the next. In theory, this constraint does not hold in a SLAM problem. For instance, a given robot pose may influence all poses in the trajectory. In practice, however, the Markov Assumption proved to be somewhat robust to such violations (THRUN; BURGARD; FOX, 2005).

the current states is the limitation in the correction capabilities of these algorithms, which may irreversibly affect map and the pose estimations.

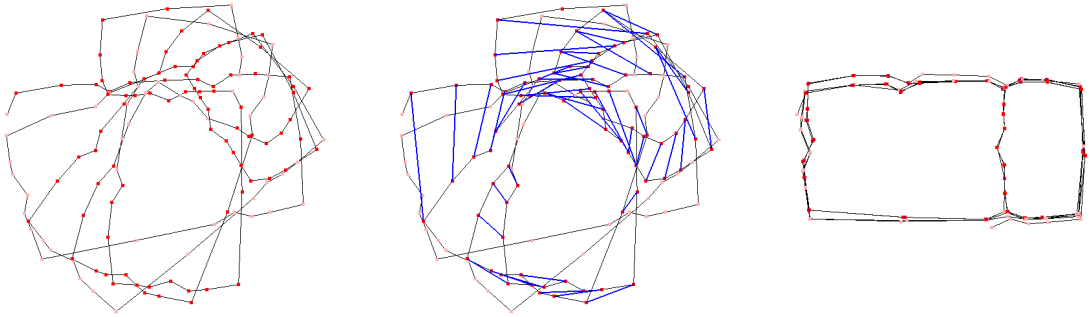
Full SLAM approaches are not limited by the Markov Assumption. Still, due to the fact that they perform optimization over the complete set of observations, full SLAM can be more effective than on-line SLAM at the cost of a high computational power requirements, which usually constrain their use to off-line and batch applications.

Graph-Based Approaches

An example of full SLAM is the GraphSLAM approach (THRUN; MONTEMERLO, 2006), which translates the SLAM problem into a graph, where the nodes are robot poses and the edges are spatial constraints associating two robot poses. These constraints are determined through measurements acquired at two robot locations holding equivalent exteroceptive measurements.

Graph SLAM architecture can be divided into two steps: the front-end and the back-end (GRISETTI et al., 2010). The front-end is heavily dependent on exteroceptive sensors used to construct the graph, such as range finders (GRISETTI et al., 2010), single (ENGEL; SCHöPS; CREMERS, 2014), stereo (TOMONO, 2009) and RGB-D cameras (ENDRES et al., 2012). A major task of the front-end is to define when two different poses in time are the same or in sequence. This is done setting edges between nodes. The back-end is responsible for pose and map corrections. This is usually done using optimization techniques such as the least-squares (GRISETTI et al., 2010; MAFFEI et al., 2015) and simulated annealing (FANFANI et al., 2013) minimization. Furthermore, even though the SLAM problem is not very far from a linear least-squares minimization (ZHAO; HUANG; DISSANAYAKE, 2013), the SLAM problem does not perfectly fit to it. For this reason, sub-map merging is commonly used to try to mitigate eventual errors. Note that full SLAM optimization approaches such as linear least-squares minimization can be stuck in local minima (HUANG et al., 2010) if the cumulative error is high. Fig. 4.2 (a) presents a topological map constructed using pure odometry, while Fig. 4.2 (b) presents the results of the front-end, which connect nodes which are the same (in blue). The last picture presents the results after the optimization using a least-squares approach (see 4.2 (c)).

Figure 4.2: Example of the GraphSLAM approach. In (a) we present a topological map constructed using pure odometry, while (b) presents the function of the front-end: link nodes which are the same (in blue). The last image shows the topological map after optimization using the least-squares approach considering edge data from the front-end (c).

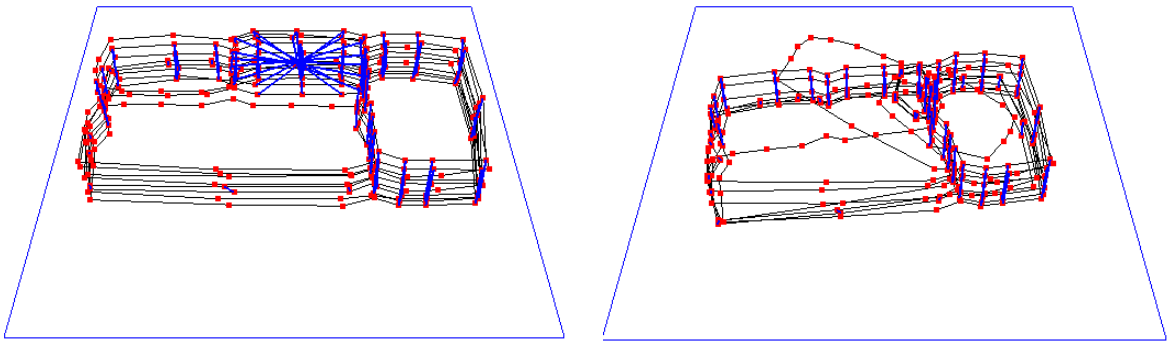


(a) (b) (c)

At this point, we highlight that outliers introduced in the front-end, can compromise the results of the back-end (THRUN; BURGARD; FOX, 2005; AGARWAL et al., 2013) (see Fig. 4.3).

Figure 4.3: The effects of wrong matches in the front end. In (a) we present the ground-truth of a map (elevations are merely to improve visibility) of the environment when the front-end inaccurately creates edges between different poses (edges in blue connected like a star). On the right (b) we see impact on the solution provided by the back-end after least-squares minimization.

(a) (b)



Furthermore, since full SLAM techniques may not be executed in real-time due to the large number of states involved, some graph-based methods divide the environment hierarchically or in subregions to speed-up the performance of optimization algorithms (NI; STEEDLY; DELLAERT, 2007) (KIM et al., 2010) (MCDONALD et al., 2011). Despite the growth in popularity, full SLAM is still a batch process.

Integrated exploration requires real-time processes, and, therefore, on-line SLAM strategies. Only recently (VALLVé; ANDRADE-CETTO, 2015) full SLAM approaches started to be used for integrated exploration. In this thesis we want to determine how the exploration task can improve the result of on-line SLAM algorithms. In particular, we are interested in closing loops. Note in Figs. 4.2 and 4.3 that even for full SLAM approaches, such as GraphSLAM, the detection of loop-closures (blue edges connecting nodes) has an important influence in the results (LATIF; CADENA; NEIRA, 2013; PFINGSTHORN; BIRK, 2015). It is only by correct matching and revisits that a graph-based approach can properly correct SLAM errors.

EKF-SLAM

The original Kalman filter, proposed by Rudolf Kalman (KALMAN, 1960), is a recursive method for estimating the state of a dynamic linear system corrupted by independent Gaussian random noise. At each instant, a measurement of the state of the system is performed considering the presence of stochastically independent Gaussian errors. The robot pose and the measurements of different sensors are modeled through equations (CHOSET et al., 2005)

$$\boldsymbol{p}(\mathbf{x}_{t+1} \,|\, \mathbf{x}_t, \mathbf{u}_t): \quad \mathbf{x}_{t+1} = \mathbf{F}_t \mathbf{x}_t^T + \mathbf{G}_t \mathbf{u}_t^T + \mathbf{v}_t \tag{4.5}$$

$$\boldsymbol{p}(\mathbf{z}_t \,|\, \mathbf{x}_t): \quad \mathbf{z}_t = \mathbf{H}_t \mathbf{x}_t^T + \mathbf{w}_t \tag{4.6}$$

where $\mathbf{F}_t$ is a matrix of size $N \times M$ 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 implications 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 $\mathbf{W}_t$. Note that state and mesurement vectors are transposed for the sake of matrix operations.

The traditional Kalman filter provides the best unbiased estimator for linear systems (ANDERSON; MOORE, 2012). However, the assumption of linear state models, as in Eqs. 4.5 and 4.6, are not always useful in robotics. The Extended Kalman Filter (EKF) Smith, Self and Cheeseman (1990), a pioneer and now popular on-line SLAM approach, is freed from this constraint, by assuming that the posterior probabilities of $\mathbf{x}$ and $\mathbf{z}$ are given by two nonlinear functions $\boldsymbol{f}$ and $\boldsymbol{g}$

$$\mathbf{x}_{t+1} = \boldsymbol{f}(\mathbf{x}_t, \mathbf{u}_t, t) + \mathbf{v}_t \tag{4.7}$$

$$\mathbf{z}_t = \boldsymbol{g}(\mathbf{x}_t, t) + \mathbf{w}_t \tag{4.8}$$

In this case, however, it may not be possible to compute an exact solution to the problem. The idea to circumvent this problem is to approximate the non-linear functions with Gaussians. On EKF-SLAM, the map of the environment consists of a set of distinct features representing information about the physical world — such as beacons, landmarks, or seamarks, which are not the focus of this Thesis.

Even though there are numerous modifications proposed to the Kalman Filter (GUIVANT; NEBOT, 2001; KNIGHT; DAVISON; REID, 2001; TARDOS et al., 2002; FRESE, 2006; PAZ; TARDOS; NEIRA, 2008; JULIER; UHLMANN, 1997; BAILEY et al., 2006; HUANG; MOURIKIS; ROUMELIOTIS, 2008; HUANG; MOURIKIS; ROUMELIOTIS, 2010; NIETO; BAILEY; NEBOT, 2006; KANG et al., 2010; LEE; SONG, 2010), there are problems which limit the broad use of EKF SLAM (THRUN; BURGARD; FOX, 2005). The first problem is that EKF has an update step which is quadratic in the number of features, making it limited to a few thousands. In addition, EKF works better with identifiable features than with similar or indistinguishable ones, which could generate ambiguity or outliers. Another problem is that the update step is usually performed through a deterministic incremental maximum likelihood method, which leads to the solution that is most likely correct without any guarantee the correct one was chosen. EKFs require a form of linearizing a nonlinear function describing a system around a specific point. If the neighborhood of such point cannot be approximated using a straight line, the resulting distribution, a Gaussian around that point, may not correctly represent the nonlinear system. In this case, the resulting approximation error will rise and there is a chance that the filter will diverge, leading to SLAM errors (DU et al., 2011).

In the next section, we will detail the Rao-Blackwellized Particle Filter (RBPF) approach. We start with the definition of its core concepts, followed by two important techniques that are grounded on RBPF: the Fast-SLAM algorithm (MONTEMERLO et al., 2002; MONTEMERLO et al., 2003); and GridSLAM (HAHNEL et al., 2003). Lastly, we present DP-SLAM (ELIAZAR; PARR, 2003), the algorithm we used in our Integrated Exploration Strategy.

Rao-Blackwellized Particle Filters (RBPFs)

The Rao Blackwellization Theorem, also called Rao Blackwellization,provides a way to improve the efficiency of an estimator by taking its conditional expectation with reference to a

sufficient statistic[3]. Put in other words, the idea of RBPFs is to perform the sampling process over some of the sufficient system variables to estimate the remaining ones.

Proposed by Murphy (MURPHY, 1999), the RBPF is a Dynamic Bayesian Network (DBN) which factorizes variables of a probability distribution through a Rao-Blackwellization strategy. This concept was employed for regressions in neural networks, but it was later adapted by Doucet et al. (2000) for building maps in the context of robot motion. RBPFs can be seen as Sequential Monte Carlo approaches applied to the SLAM problem. Conversely to EKF-SLAM, which models the probability distributions as Gaussians, particle filters support non-Gaussian and multi-modal probability distributions. This flexibility made them strong candidates for on-line applications such as Integrated Exploration. Maybe for this reason, RBPFs (MURPHY, 1999) are now consolidated on-line SLAM approaches (MONTEMERLO et al., 2002; MONTEMERLO et al., 2003; HAHNEL et al., 2003; ELIAZAR; PARR, 2003; GRISETTI; STACHNISS; BURGARD, 2005; STACHNISS; GRISETTI; BURGARD, 2005b; STACHNISS; GRISETTI; BURGARD, 2005a; GRISETTI; STACHNISS; BURGARD, 2007; ABDALLAH; GNING; BONNIFAIT, 2008; BLANCO; MADRIGAL; GONZALEZ, 2008; STEUX; HAMZAOUI, 2010; CARLONE et al., 2010; MAFFEI et al., 2013).

Sequential Monte Carlo approaches try to solve the multidimensional SLAM problem using a large number of particles to find an approximate solution. This is done through a recursive Bayesian filtering procedure, sometimes called Sampling Importance Resampling (SIR) (MONTEMERLO et al., 2002).

In terms of SLAM, we have to find the joint conditional probability distribution of

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

The core concept behind RBPFs is that the distribution of the map and that of the trajectory are considered statistically independent. This way, if the robot path is known, the map estimates can be performed independently, i.e.:

$$\mathbf{p}(\mathbf{x}_{1:t}, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}) = \mathbf{p}(\mathbf{m} \,|\, \mathbf{x}_{1:t}, \mathbf{z}_{1:t})\, \mathbf{p}(\mathbf{x}_{1:t} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{4.9}$$

The RBPF tries to solve the SLAM problem through a sampled set of solutions (the particles). Unfortunately, the computational power limits the supported number of particles, which consequently reduces the diversity of solutions the algorithm can hold. Therefore, an-

[3] A statistic is sufficient regarding an statistical model and its associated unknown variable if "no other statistic that can be calculated from the same sample provides any additional information as to the value of the parameter" (FISHER, 1922).

other important step of the filter is then to remove statistically unlikely solutions. This is done by measuring the importance of given solutions, and resampling them accordingly. As the filter execution evolves, the best locally chosen particles will eventually share the same parent. Over time, the best local solutions are maintained and unimportant particles are removed, which, ideally, will result in the best global solution. However, if parents containing the appropriate global solutions are filtered out in the process by their children's poor local choices, the final map will diverge from the globally correct solution — i.e., locally correct measurements executed ahead in time may discard valid solutions back in the trajectory which compromise the global result of the filter. This is a well-known problem of RBPFs called particle depletion (STACHNISS, 2009).

FastSLAM: an insight into RBPF strategies for SLAM

FastSLAM is one of the first algorithms to propose the use of the RBPF as a solution for the SLAM problem (MONTEMERLO et al., 2002). We have seen that the idea behind this approach is to apply a particle filter to estimate the trajectory of the robot and then use it for the estimatation of the map.

A major principle sustaining this concept is the factorization of Eq. 4.9 considering also the conditional independence of the landmarks present in the environment, leading to[4]

$$\boldsymbol{p}(\mathbf{x}_{1:t}, m \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \boldsymbol{p}(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \prod_i \boldsymbol{p}(\mathbf{m}_i \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t}), \tag{4.10}$$

which enables the efficient retrieval of the SLAM posterior by multiplying the posterior of the trajectory by the posteriors of individual landmarks, always assuming that the trajectory is known.In other words, the probability density function of a given part of the map is proportional to the estimated probability of the robot observing the map from a given pose.

Consequently, each RBPF particle is a pair consisting of an individual map, $\mathbf{m}$, tied to unique particular trajectory, $\mathbf{x}_{1:t}$ and its corresponding observations, $\mathbf{z}_{1:t}$. Note that the solution of Eq. 4.10 can be found computing $\mathbf{p}(\mathbf{m} \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t})$ analytically once the trajectory and corresponding observations are available.

However, similarly to EKF-based SLAM, FastSLAM is affected by data associations with respect to different landmark observations. The solution adopted by FastSLAM is to use a maximum likelihood to determine which landmarks are observed at every moment. The im-

[4]The full derivation of this formula is shown in Thrun, Burgard and Fox (2005).

portance of each particle is proportional to $\mathbf{p}(\mathbf{z}_t \mid \mathbf{m}, \mathbf{x}_t)$ — the likelihood of the most recent observation $\mathbf{z}_t$ given the map $\mathbf{m}$ associated to the current particle pose $\mathbf{x}_t$. This way, the filter can be applied recursively in a Bayesian Filtering process. This renders FastSLAM the status of on-line SLAM approach.

Rao-Blackwellized particle filters have a linear computational complexity in the number of particles. However, as the map gets more and more complex, the number of particles required for building precise maps grows considerably. Ultimately, for a large number of particles, the method becomes unusable for real-time applications. Conversely, if the number of particles is reduced, particle depletion presents itself as another potential limitation of the method.

FastSLAM makes use of a SIR (Sampling Importance Resampling) strategy to determine the robot pose, similarly to Monte Carlo localization (MCL) (DELLAERT et al., 1999), but also keeping a map which results from the associated trajectory. In this method, for every map, landmark positions are improved considering an isolated EKF for every landmark. Since the EKF considers only one landmark, the cost of such approach is reduced (even though this improvement in landmark position becomes unpractical as the number of landmarks increases.).

The FastSLAM algorithm (Algorithm 1) is composed by four steps, encompassing a SIR particle filter plus a map estimation step through a naive EKF for each landmark in the map. Next, each of the four steps of the algorithm will be explained in details.

Algorithm 1: Rao-Blackwellized Particle Filter Overview

1. **Sampling** - generates a set of hypothesis over the robot pose using a proposed distribution according to a motion model.
2. **Importance** - weights particles by comparing the current exteroceptive sensor measurements with the on-line constructed particle map.
3. **Resampling** - resamples particles according to their weight to keep good particles with some random diversity to reduce the risk of particle depletion.
4. **Map Estimation** - updates the map of every particle according the observation model using EKFs.

Given the previous generation of particles, $\mathcal{G}_{t-1}$, the **sampling** process is in charge of the generation of the new offspring of particles, $\mathcal{G}_t$. Since the RBPF considers that the pose is given — even though we actually do not have precise information about it — we have to use available information to generate a new pose for every particle. In our case, we do have the control data of the robot, $\mathbf{u}_t$. This information, combined with a proper motion model for the robot, can be used to reduce systematic motion errors, while at the same time providing parameters which

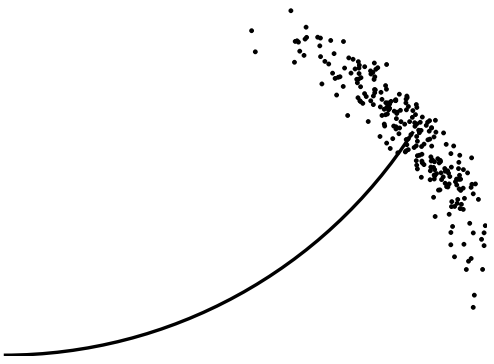
are used to model the motion uncertainty. This is our *proposal distribution*, used to generate particles.

There is an important assumption about the sampling process. If we consider that the previous particles are distributed according to $\boldsymbol{p}(\mathbf{x}_{1:t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})$, and also that the new set of particles will be generated in the same way, i.e., $\boldsymbol{p}(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})$, we can say that the entire trajectory comes from this proposal distribution

$$\boldsymbol{p}(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}). \tag{4.11}$$

Observe that the uncertainty about the robot motion often involves known uncertainty parameters (e.g. variance information), that are important to constrain the space of possible valid poses. In order to enforce that, we can use the uncertainty information provided by the motion model to generate a proposal distribution. Hence, the motion model becomes a probabilistic non-linear function combining odometry information with an amount of Gaussian noise, which is defined according to this uncertainty. Thus, this probabilistic model can be used to sample new particles. Fig. 4.4 (MONTEMERLO; THRUN, 2007), presents an example of sampling process considering a curved trajectory. Note that the translational error is considerably smaller than the rotational error.

Figure 4.4: Example of sampling based on the probabilistic motion model. Figure extracted from (MONTEMERLO; THRUN, 2007).



Weighting is the process that decides which sampled particles will be useful and which will not. It classifies the samples of the proposal distribution $\boldsymbol{p}(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})$ according to their similarity to the posterior distribution $\boldsymbol{p}(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$. In other words, it assigns the importance of each particles with respect to the posterior distribution.

For every particle, $[m]$, its weight, $w_t^{[m]}$, is obtained by measuring the quality of its corresponding map when compared with the target distribution. This is done considering the exteroceptive measurement $\mathbf{z}_t$ in the Ratio,

$$w_t^{[m]} = \frac{\text{posterior distribution}}{\text{proposal distribution}} = \frac{\boldsymbol{p}(\mathbf{x}_{1:t}^{[m]} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})}{\boldsymbol{p}(\mathbf{x}_{1:t}^{[m]} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}, \tag{4.12}$$

between proposal and target probability distributions. Then, by separating $\mathbf{z}_t$ from $\mathbf{z}_{1:t}$, followed by the application of the Bayes Rule we get

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

$$w_t^{[m]} = \frac{\boldsymbol{p}(\mathbf{z}_t \mid \mathbf{x}_{1:t}^{[m]}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}{\boldsymbol{p}(\mathbf{z}_t \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})} \tag{4.13}$$

which corresponds to the weight of the particle. Observe that the denominator is a constant that can be eliminated from Eq. 4.13 due to the fact that the resampling step will renormalize these weights. So we can rewrite Eq. 4.13 as

$$w_t^{[m]} = \boldsymbol{p}(\mathbf{z}_t \mid \mathbf{x}_{1:t}^{[m]}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}). \tag{4.14}$$

The quality of the particle is measured through its degree of innovation, i.e., the difference between real, $\mathbf{z}_t$, and predicted, $\hat{\mathbf{z}}_t$, observations. Recall that FastSLAM landmark position estimates are obtained through an EKF applied in isolation for every landmark. Therefore, sequence of innovations are obtained from the EKF, corresponding to a Gaussian having zero mean with covariance matrix $\mathbf{Z}_t$. Hence, the particle importance is converted into a multi-variate Gaussian considering the difference of innovation means, $\mathbf{z}_t - \hat{\mathbf{z}}_t$,

$$w_t^{[m]} = \frac{1}{\sqrt{(2\pi)^k \mathbf{det}(\mathbf{Z}_t)}} \exp\{-\frac{1}{2}(\mathbf{z}_t - \hat{\mathbf{z}}_t)^T \mathbf{Z}_t^{-1}(\mathbf{z}_t - \hat{\mathbf{z}}_t)\}. \tag{4.15}$$

where k is the dimensionality of the Gaussian. The resampling step uses these weights as inputs to approximate the two distributions.

The **resampling** step changes the proposal distribution using the particle weights to estimate the SLAM posterior. After the weighting process, particles are sampled once again but, this time, giving higher probability of occurrence to particles with higher weights. Usually, resampling is accomplished using a proportional selection algorithm such as roulette wheel selection (LIPOWSKI; LIPOWSKA, 2012). Such selection procedures have a twofold benefit.

First, important particles will have higher weights, and consequently a better chance to be replicated. Second, particles having lower weights will still have moderate chances to survive this random process, maintaining the diversity of the RBPF.

Fig. 4.5 illustrates the SIR particle filter steps. In (a), we see the difference between target (green) and proposal (red dashes) distributions — the proposal distribution is a Gaussian while the target is a multi-modal probability distribution. The sampling step is shown in (b), where samples are represented by black lines with lengths corresponding to individual particle weights. During the sampling process, all particles have the same weight, as indicated in the picture. Another important observation is that, since a Gaussian is used to represent the proposal distribution, the sampling process produces particles that tend to concentrate next to the mean of the Gaussian. The importance, or weighting, step is depicted in (c). We can see that higher weights are given to samples next to the modes of the target distribution, while those next to the valleys of the proposal distribution receive a small importance. In the resampling step, (d), the solution space is sampled into buckets (yellow bars) which will be used to construct a histogram representing the proposal distribution. Bin size represents the sum of the weight of each particle falling in that bucket. As portrayed in (e), the resulting histogram is a good estimate of the target distribution.

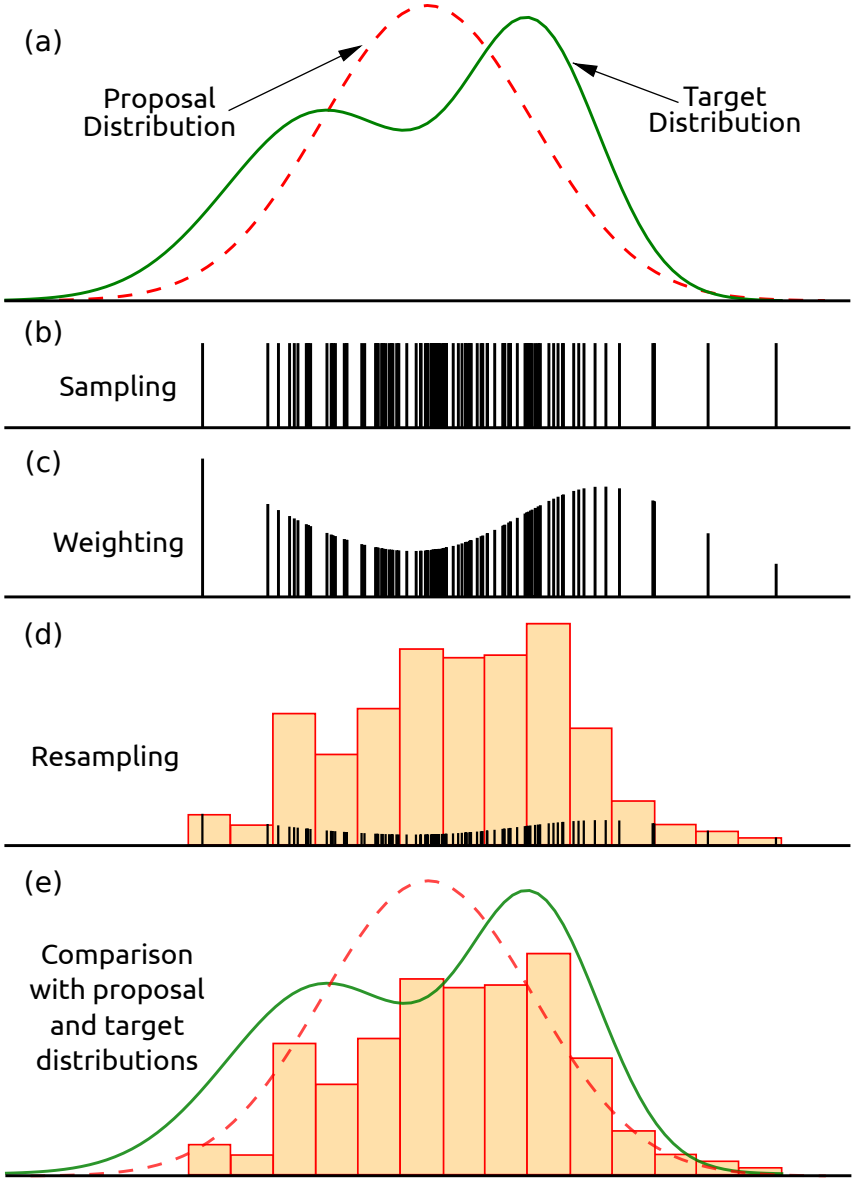
Map Estimation, $p(\mathbf{m} \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$, demands the estimation of landmark positions, given the quantity of landmarks in the environment, M, and the number of particles, P. This step uses one EKF per landmark per map, resulting in $M \times P$ filters, to estimate individual landmark positions. Since only one landmark is considered by the EKF, the execution time for this step is also constant.

It is worth mentioning that modifications were proposed to the FastSLAM algorithm[5]. Perhaps, its most notorious modification is FastSLAM 2.0 (MONTEMERLO et al., 2003), which tries to reduce the problem of particle depletion by modifying the proposal distribution. Considering that exteroceptive sensors are more precise than proprioceptive sensors, such observations can be used to improve the proposal distribution before the resampling step

RBPF methods can be used to estimate a set of landmark positions, and also adapted for occupancy grids. In the latter case, the algorithm estimates the occupancy of all cells in the grid. The two seminal works (THRUN; BURGARD; FOX, 2005) on RBPF methods with occupancy grids are the GridSLAM (HAHNEL et al., 2003) and the DP-SLAM (ELIAZAR; PARR, 2003). GridSLAM efficiently combines the RBPF SLAM algorithm with scan-matching techniques.

[5]Improvements include but are not limited to selective resampling (GRISETTI; STACHNISS; BURGARD, 2005) and diversity recovery (STACHNISS; GRISETTI; BURGARD, 2005b).

Figure 4.5: Example of SIR in a RBPF (Fig. from (MONTEMERLO; THRUN, 2007)).



In the algorithm, every particle has a map and trajectory. Proposed in the same year, DP-SLAM is very similar to GridSLAM, but it aims to compact the map representation. It does that storing the differences observed by particles having similar parents in a single occupancy grid used by all particles. DP-SLAM reconstructs individual grid maps through its ancestry tree, containing the offspring of particles. Next, we present grid-based RBPFs. They are reasonably easy to use and apply, adapting well to the proposed work of integrated exploration using range finders.

GridSLAM: RBPF SLAM using occupancy grids

The GridSLAM algorithm (HAHNEL et al., 2003) is one of the first grid-based implementations of the RBPF for the SLAM problem. The algorithm uses range finder readings combined with a scan-matching registration approach (HAHNEL; SCHULZ; BURGARD, 2002; THRUN; BURGARD; FOX, 2000) to reduce odometry errors. Such refined odometry measurements are then used as inputs for the sampling process in a RBPF SLAM.

A scan-matching (HAHNEL; SCHULZ; BURGARD, 2002; THRUN; BURGARD; FOX, 2000) algorithm aligns laser readings with the occupancy grid, obtaining better prior pose estimates than pure odometry. To avoid the ray-tracing operations required to determine the likelihood, $p(\mathbf{z} \mid \mathbf{x})$, the method considers only the endpoint of the beams (THRUN, 2001), allowing a faster computation of such likelihood.

Fig. 4.6 presents three maps: a map obtained using pure odometry readings (a); another one after the employment of scan-matching registration (b); and the final map (c) obtained with GridSLAM using 100 particles. Such results highlight the power of RBPFs with occupancy grids for on-line SLAM approaches. Note that the improvement of the proposal distribution of the RBPF leads to superior SLAM results (as in FastSLAM 2.0).

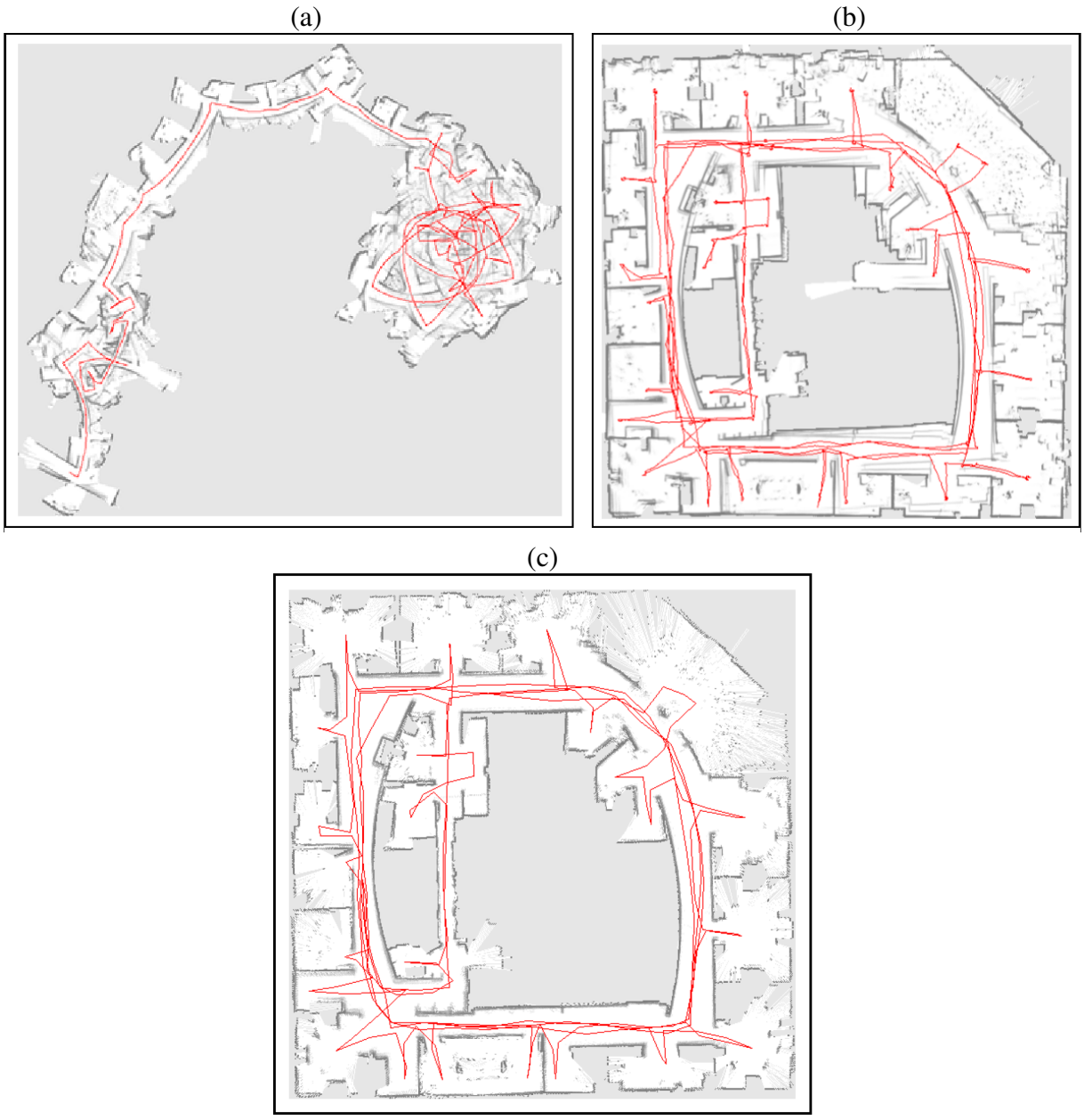
An important limitation of GridSLAM is that the map representation still requires plenty of memory. It is also important to highlight that closing loops in large environments is a challenge for this and other SLAM approaches.

DP-SLAM: Improvements on the data management

Distributed Particle SLAM (DP-SLAM) algorithm[6] (ELIAZAR; PARR, 2003) is another RBPF SLAM over occupancy grids. Unlike GridSLAM, it does not require a full map per particle. Eliazar and Parr (2003) made an important observation about RBPF SLAM approaches: In principle, RBPFs 'solve' the SLAM problem. In practice, they just "replace a conceptual problem with an algorithmic one". They observe that the number of particles required for SLAM is larger than that of pure localization, since pose estimation errors propagate recursively through the trajectory and, consequently, the map. Therefore, the problem changes from solving the SLAM problem to that of finding a way to store enough solutions with the

[6]The name DP-SLAM comes from an inversion of the paradigm where every particle has a map. That is, the DP-SLAM algorithm states that every particle is spread (distributed) over the map.

Figure 4.6: Comparison between a map obtained from pure odometry data (a), corrected with the scan matching (b), and the final GridSLAM map (c) obtained on-line with 100 samples. Figures extracted from (HAHNEL et al., 2003).



available resources. DP-SLAM approaches this problem by storing information from all maps associated to the particles in an efficient map representation, created to merge map information from the different particles.

A key aspect behind DP-SLAM is the notion that for any set of particles, the area, $\mathcal{Z}$, covered by the robot's exteroceptive sensors has aways the same size and always affect at most C cells. As a consequence, the number of different cells that may have conflicting values — e.g. *occupied* or *free* — is at most C. In practice, however, the probabilistic motion model alone

reduces such conflicts to considerably less cells than that. This insight was used to design the DP-Mapping process.

The main challenge is to design a set of data structures that enables efficient updates, while at the same time enabling efficient localization queries. DP-Mapping approaches this problem using two efficient data structures: an ancestry tree and a modified grid map called distributed particle map (DP-Map). The ancestry tree is a family tree encompassing all active particles of DP-SLAM. Its leaves represent current active particles, while the other nodes are the ancestors of these particles. DP-Map is a grid that lists all the particle observations for every cell (using trees to reduce the management costs). When a particle observation made by a child conflicts with that of its ancestors, the branch of the tree corresponding to that cell is updated to include the new particle observation. Therefore, to reconstruct the complete map of a given particle, one must consult the observations made by that particle and its ancestors.

Of course, when a particle does not generate a child, it is removed from the tree since its information will not be inherited. In addition, when a particle generates only one child, the information of both are merged and set as belonging to the parent. However, if child and parent update the same cell, the most current update is kept. Afterward, the child is killed.

One important aspect of DP-SLAM is that its original implementation, also called DP-SLAM 1.0 (ELIAZAR; PARR, 2003), assumes that the grid state was binary, i.e.: occupied or totally empty. Conditioned on the robot position, there is only one possible square in which a laser cast of a given length can terminate. New observations are added to the map whenever the following two conditions are satisfied: the ray cast must not cross an existing obstacle in the map; and the ray cast must not terminate three grid cells from an existing obstacle along the ray. Once an obstacle is inserted in the map, it cannot be removed or updated. The approach has a log-quadratic cost in the number of particles, even though the authors claim that such cost is smaller than that in practice (ELIAZAR; PARR, 2003).

Later, its successor (ELIAZAR; PARR, 2004a), called DP-SLAM 2.0, managed to reduce the computational complexity of the algorithm to a quadratic cost. This was possible due to a more efficient localization method. They also proposed a more realistic probabilistic model for the laser measurement update, which no longer depends on the distance traveled through a grid square.

Eliazar and Parr also developed an hierarchical version of DP-SLAM (ELIAZAR; PARR, 2006) involving two levels of particle filters to try to model and recover from the drift caused by particle depletion. The authors claim that the algorithm maintains a linear time asymptotic complexity. Later, Maffei et al. (2013) proposed a new submap-based RBPF algorithm founded

on the principles of DP-SLAM called Segmented DP-SLAM (SDP-SLAM). It combines optimized data structures to store the maps of the particles with a probabilistic map of segments, representing hypothesis of submaps topologies. While this technique provides better results for large maps, there is no definitive criteria for choosing how to perform the segmentation between consecutive local submaps. Another interesting question which deserves attention is at what point should the algorithm keep (lock) a given set of local maps encompassing a subregion of the map? This last question could be tied to loop-closures, which are addressed by the approaches presented in this Thesis. In fact, when the robot successfully closes a loop, the submaps encompassing such region might be locked.

Discussion on SLAM approaches in the context of this work

DP-SLAM was selected to be used in this Thesis because of its effectiveness and also its capacity to reduce memory and computational costs, enabling the development of real-time active SLAM approaches such as those presented in this document. Still, the techniques presented in this Thesis would work just as well or better with other SLAM algorithms, since they provide important information about loop-closures. For instance, this information could be used to provide input for GraphSLAM and SDP-SLAM. In GraphSLAM, graph connectivity information could be extracted from every closed loop, while in SDP-SLAM, loop-closure could be used to stop the update of the set of submaps encompassing the loop.

The Importance of Loop-Closure for SLAM approaches

In general, SLAM solutions are weak when the prior distribution is weak (OLSON, 2009; MONTEMERLO et al., 2003; HAHNEL et al., 2003). There are many causes of poor priors: poor proprioceptive and exteroceptive sensors and corresponding measurements; weak heuristics solutions associated to matching algorithms (OLSON, 2009); and unpredicted errors in the observation and motion models (THRUN; BURGARD; FOX, 2005)[7]. Nevertheless, errors in a prior can be the result of inconsistencies in previous states. This problem can easily be observed when SLAM approaches have to close loops.

Loops are challenging because, in a SLAM problem, current states are recursively dependent on all previous states. Even though this is the general case for the SLAM problem, a

[7]Many works (OLSON, 2009; MONTEMERLO et al., 2003; HAHNEL et al., 2003; GRISETTI et al., 2010) make use of strategies to ameliorate prior information.

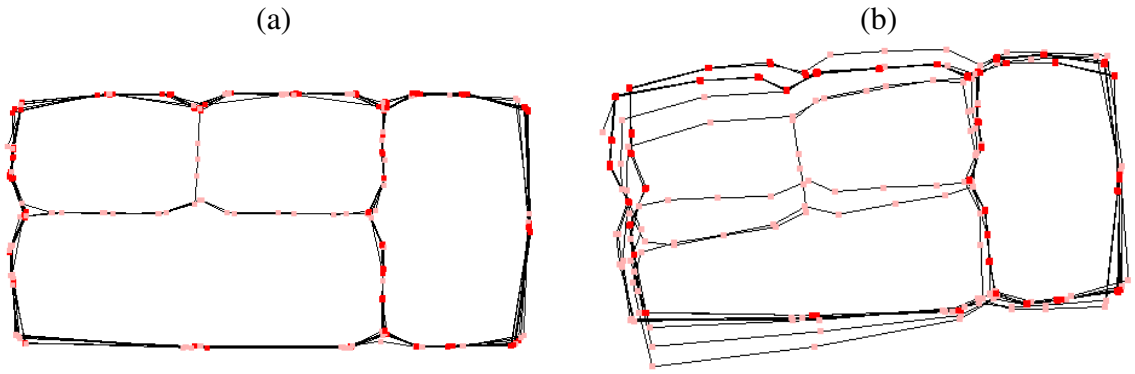
loop can enclose a potentially significant part of the robot trajectory. During the loop-closure unknown or sparse terrain may be crossed, where there may not be sufficient information from the robot sensors to correct pose and map errors. When the loop is detected and appropriately closed in time, observations of previously visited regions make this uncertainty about the joint SLAM posterior drop considerably. If, however, a loop is not closed, this uncertainty may continue to rise, reaching an apex from which the SLAM approach is unable to recover. This problem affects both on-line (STACHNISS; HAHNEL; BURGARD, 2004; STACHNISS; GRISETTI; BURGARD, 2005a; STACHNISS; GRISETTI; BURGARD, 2005b) and off-line SLAM approaches (LATIF; CADENA; NEIRA, 2013; PFINGSTHORN; BIRK, 2015).

In the context of full SLAM approaches (LATIF; CADENA; NEIRA, 2013; PFINGSTHORN; BIRK, 2015), loop detection and closing are critical to correct estimation errors. For instance, a proper loop-closure is capable of major corrections in GraphSLAM, while missing loop information can make such corrections impossible. In addition, incorrect or missing edge information coming from the front-end can completely destroy estimates of the back-end (see Figs. 4.2 and 4.3). In this thesis we do not deal with full SLAM mainly because of the impossibility of running such algorithms in real-time. Nevertheless, they can benefit from a smart exploration algorithm that purposefully close loops — one of the important goals of this thesis. Fig. 4.7 presents a GraphSLAM result for a topological map when the front-end fails to detect a loop[8]. Nodes in light red color are those which were not appropriately connected to any other node beyond the ancestor and successor nodes, while nodes in red are those in which a loop-closure was detected by the SLAM front-end. In particular, observe that when longer loops are not closed properly, they result in noticeable SLAM errors.

On-line SLAM approaches such as the RBPF, in general, are tied to the Markov Assumption (THRUN; BURGARD; FOX, 2005). As such, they assume that the current state is complete and no other state is required to estimate the next. Even though this is effectively done to reduce the number of variables involved in the SLAM problem, this assumption can impact the solution of the algorithm. In terms of particle filters, the current pose of a particle is directly affected by the decisions and mistakes of its ancestors. As the decision about which particles should be kept is local, it may sometimes lead to a bad global solution. If the number of particles is small, the global solution may be excluded by a poor local choice of particles. In this case, the filter diverges from the correct solution (STACHNISS; GRISETTI; BURGARD, 2005b). Fig. 4.8 shows an example of RBPF SLAM where loop closure takes too long and

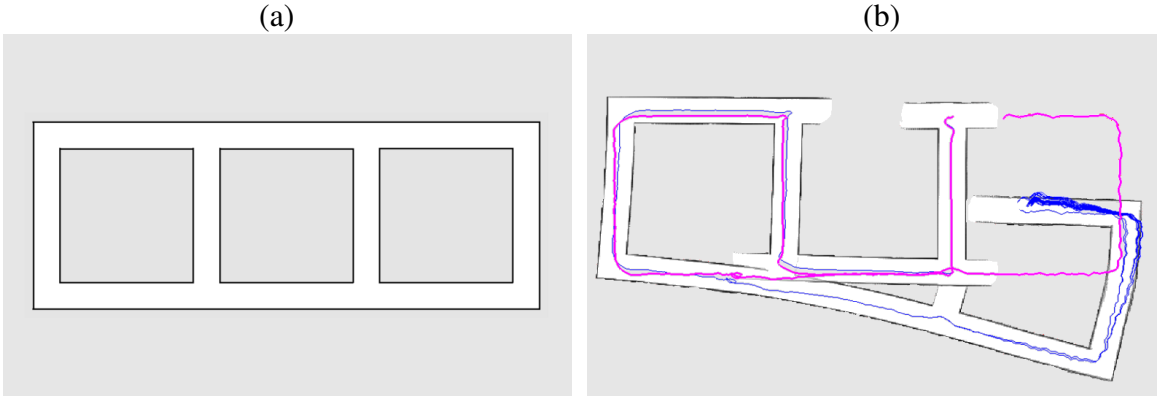
[8]We have worked with GraphSLAM (MAFFEI et al., 2015) in a work focusing on topological maps and the front-end.

Figure 4.7: Picture showing the importance of loop-closures for GraphSLAM. In (a) we present the ground-truth of a map of the environment. Unconnected nodes are shown in light red, while connected nodes are in red. On the right (b) we see impact on GraphSLAM when loop closure fails (on the upper left corner of the map).



fails. Furthermore, under considerable SLAM uncertainty, an exploration algorithm may decide to localize itself by going back to a previously known region. In this case, the particle filter may adjust itself to locally incorrect maps (e.g. going back from a long corridor) leading to incorrect SLAM solutions.

Figure 4.8: Picture showing the importance of loop-closures for DP-SLAM. In (a) we present the ground-truth of a map of the environment. On the right (b) we see impact on a RBPF SLAM (DP-SLAM) when loop closure takes too long and the algorithm fails (in blue we show the particles, while in purple we have the ground-truth).



Hence, we propose an exploration strategy that triggers loop-closures during the entire exploration process to force the decrease of the uncertainty about the joint SLAM posterior. We consider that previous loop-closure techniques designed for the active SLAM process do not properly exploit some key aspects about the nature of this problem, specifically:

1. Possible loops can be found and closed by turning pragmatically to the same side;

2. A loop goes back to a previously known region without turning back using the same path the robot originally came from;
3. Unknown terrain carries important information about eventual loop-closures;

Even though these remarks may look like trivial observations, they provide important insights for decision making while trying to close loops. Two techniques were developed as a result of these observations (MAFFEI et al., 2014; JORGE et al., 2015) and presented in Chapters 6 and 7. Below we present the related work on Active SLAM (Integrated Exploration).

5 INTEGRATED EXPLORATION

Integrated Exploration has been the subject of several interesting works in recent years. Makarenko et al. (2002) introduced the Integrated Exploration problem and its relations with mapping, localization and path planning. As we have mentioned in the introduction (see Chapter 2.1) Integrated Exploration occurs when SLAM results drive the exploration process. Other names for Integrated Exploration are: Simultaneous Planning Localization and Mapping, and Active SLAM.

In this chapter, we introduce the related works on the subject with focus on the state of the art. After the description of each technique, when relevant, we bring aspects we regard important to light, contrasting them with this thesis.

Greedy Approaches

In this section we introduce those techniques which are greedy, i.e., they are not concerned with SLAM improvement, only information gain. Even though Yamauchi (1997)'s and Prestes et al. (2002)'s approaches are not Active SLAM approaches, they are used in many of the techniques in the subsequent sections.

Seminal Work on Exploration

Yamauchi (1997)'s approach explores the environment by successively driving the robot to the closest frontier using a combination of laser and sonar range-finders to minimize the effect of specular reflections on laser range finder readings. Frontier cells are detected through an occupancy grid having prior equal to 0.5 connecting unknown and free regions. Cells with a value larger than the prior are considered obstacles, while those having a smaller value are set as free. The path to the nearest frontier is obtained through a depth-first search algorithm combined with a reactive obstacle avoidance technique to dodge previously undetected obstacles.

BVP Exploration

In the area of potential fields, Prestes et al. (2001, 2002) developed an exploration strategy relying on the Dirichlet boundary-value problem (BVP) resolving the Laplace equation.

Unexplored regions are given low potential (goals), while obstacles are given high potential — the potential of free space is solved numerically using the finite-differences method. When there are no more unexplored frontier cells, the robot stops. The method was improved (PRESTES et al., 2003) by using a local window with size varying according to the smallest sonar reading. This local map is only effective if there are frontier cells inside the local window. Otherwise, the global map needs to be used to calculate the potential field. Later, Silveira et al. (SILVEIRA; PRESTES; NEDEL, 2010) proposed a fast multi-grid approach to solve the BVP performance problems while computing the global potential field. However, this method still needs to be tested in exploration tasks, since its parameterization may be map dependent.

This work and all those involving BVP exploration are the basis for this thesis. Detailed information on BVP Exploration evolution is presented in Chapter 3. Below we describe the first implementation of BVP that is actually worried with the map quality, at least in sparse regions.

BVP Wall-Following

The original BVP Exploration algorithm, presented in Section 3.4, predefines the potential field generated by the exploration method at discovered obstacles, $\mathcal{O}(t)$, and unknown regions, $\mathcal{U}(t)$, through Eqs. 3.19 and 3.20. The resulting gradient of the potential field in front of the robot points toward unexplored regions, while keeping the robot away from obstacles. While the resulting exploratory (greedy) behavior is ultimately required for exploration, it is not enough for integrated exploration where the uncertainty about the SLAM posterior may require revisits before the exploration ends in order to correct the robot pose or the map.

With this problem in mind, Trevisan et al. (2006) proposed a modification of the equations defining the potential values at the boundaries to devise a wall-following behavior,

$$\boldsymbol{\nu}(p) = \boldsymbol{f}(p) \quad on \quad p \in \mathcal{X}, \tag{5.1}$$

where $\mathcal{X} \subset \partial_{\mathcal{F}}\mathcal{U}(t) \cup \partial_{\mathcal{F}}\mathcal{O}(t) \cup \partial_{\mathcal{U}}\mathcal{O}(t)$. Note that they subdivide the boundary region in three important parts: the boundary between free region and obstacle regions, $\partial_{\mathcal{F}}\mathcal{O}(t)$; the boundary between free and unknown regions $\cup\partial_{\mathcal{F}}\mathcal{U}(t)$; with the addition of the fundamental border between obstacles and unknown regions, $\partial_{\mathcal{U}}\mathcal{O}(t)$. The resulting behavior, $\boldsymbol{f}(p)$ works similarly to BVP Exploration. However, the algorithm searches for specific regions where there are obstacles making contact, along its boundaries, with unknown and free regions at the same

time, such that

$$\mathcal{Y} \subset \partial_{\mathcal{U}}\mathcal{O}(t) \cap \partial_{\mathcal{F}}\mathcal{O}(t).$$

When such region is set as a goal, i.e., given a low potential, the robot will tend to follow the boundaries of the obstacles until all their boundaries in contact with the free region are known. Thus, a wall-following behavior can be generated using the following equation

$$\boldsymbol{f}(p) = \begin{cases} 0 & \text{if } p \in \mathcal{Y} \quad \text{or} \quad p \in \partial_{\mathcal{Y}}\mathcal{F}, \\ 1 & \text{if } p \notin \mathcal{Y} \quad \text{and} \quad p \in \mathcal{X}. \end{cases}$$

In the implementation on a discrete grid, the free neighboring cells making boundary with $\mathcal{Y}$ — i.e., $p \in \partial_{\mathcal{Y}}\mathcal{F}$ — are also set as goals. Finally, if $\mathcal{Y}(t) = \emptyset$, the traditional BVP exploration takes place.

This technique set different values at the boundaries depending on the behavior. Nevertheless, the functions defined by the authors are still constant and dependent on the boundary between unknown and known space. Furthermore, such wall-following behavior does not close loops in a controlled way. In this Thesis we are interested in behaviors beyond these to improve SLAM results. For that purpose, we propose that BVP Exploration must use other boundary conditions to better control the loop-closure behavior.

Loop-Driven Approaches

In this section we introduce those techniques that are concerned with loop-closures. Loop-closures are fundamental to reduce the SLAM posterior, as such no true Active SLAM approach will work without a proper loop-closure strategy. We can think of loop-closures as circuits and cycles in a graph[1]. A key observation is that, be it a cycle or a circuit, the loop never goes back from where it came from using the same edges. In terms of exploration, we can describe the loop-closure behavior as traveling an unknown trajectory with a known goal — i.e., the starting position — considering the restriction that you cannot turn around to reach the starting position.

[1] A circuit in a graph $\mathcal{G}$ is a closed trail that begins and ends in the same vertex, but repeating no edges. A cycle is a circuit that repeats no vertexes, except the first (CHARTRAND; ZHANG, 2012).

Loop-Closure through Node Visibility

Stachniss, Hahnel and Burgard (2004) merge a grid-based RBPF and a topological map, $\mathcal{G}$, to enable an active loop-closing behavior during the integrated exploration process. The algorithm constructs $\mathcal{G}$ by adding a new node at pose $\mathbf{x_t}$ if the distance between $\mathbf{x}_t$ and all other nodes in $\mathcal{G}$ exceeds $2.5m$, or if there are no nodes visible from $\mathbf{x}_t$[2]. Once a node is added to $\mathcal{G}$, an edge is created from the newly created node to the last visited one. The authors state that it is useful to re-enter "known parts of the environment and following an encountered loop" to reduce the uncertainty about the SLAM posterior. Then, in order to detect loops, the algorithm measures both the topological, d_t, and the metric distance, d_m, between current node and nearby nodes. The topological distance is given by the sum of the size of the edges, in meters, leading to the node. If $d_t > 20m$ and $d_m < 6m$, the algorithm creates an edge connecting them. In other words, if the distance in the grid is significantly smaller than the distance in the graph, a loop is discovered. This intuitive idea makes it relatively easy to find loops.

They opt to navigate using the map of the particle with the highest accumulated importance factor, that is

$$s^* = \boldsymbol{argmax}_{s \in \mathcal{P}} \sum_{i=1}^{t} \boldsymbol{log}(w_t^{[s]}).$$

where $\mathcal{P}$ is the particle set and $w_t^{[s]}$ is the weight of the particle $s \in \mathcal{P}$ at time-step t.

Another interesting feature of the approach is the use of uncertainty about the pose to define when robots should stay or leave a loop-closure procedure. In particular, the robot stays in the loop until the current entropy, $\boldsymbol{H}_c$, about the pose gets smaller than the entropy, $\boldsymbol{H}_s$, measured when the loop-closure behavior started. Specifically, we stay in the loop as long as:

$$\boldsymbol{H}_c > \boldsymbol{H}_s \geq T_H,$$

where T_H is a threshold value.

These two intuitive ideas can help to reduce the uncertainty about the SLAM posterior[3]. Still, Stachniss, Hahnel and Burgard (2004)'s approach may fail to detect loops, for instance, when there are nearby nodes just around the corner. Besides, when a candidate loop-enabling node appears at instant t, it is likely that the exteroceptive measurement, $\mathbf{z}_t$, will make the entropy of the pose drop (an unintentional loop-closure), even before the start of the loop-

[2] In this case, visibility is treated as a ray casting operation.

[3] In particular, using entropy information to improve the control over the loop closure-behavior can even be adapted to improve both the loop-closure strategies presented in this thesis, which was later implemented in *Ouroboros* 2.0 which is presented in chapter 7.

closure behavior itself. This can happen because when a previously visited pose is found, formerly detected features of the environment may also be detected, reducing the uncertainty about the SLAM posterior. As a consequence, keeping the robot in the loop may just lead to particle depletion at the expense of little or no improvement in the local map. One could argue that the entropy threshold, T_H, would minimize such risk. Even though this is true, it could also mean that in most situations the loop-closing behavior would not be triggered and, instead of being the result of a pragmatic approach, loops would be randomly closed.

Under specific conditions, Ouroboros (see chapter 7), one of the techniques presented in this Thesis, takes a leap of faith that the robot is in a loop. Then, the algorithm generates paths that could potentially close this alleged loop independently of the distance between nearby nodes. These paths use the unknown space and are modified as the map is updated with new exteroceptive information. This enables the search and closure of loops pro-actively instead of just considering obvious node pairs. As a result, Ouroboros has significantly more chances to start the loop-closure behavior before the technique from Stachniss, Hahnel and Burgard (2004). In addition, we know *a priori* where the loop candidate starts and also where it is supposed to finish. This feature may be interesting for a technique that tries to recover the particle diversity.

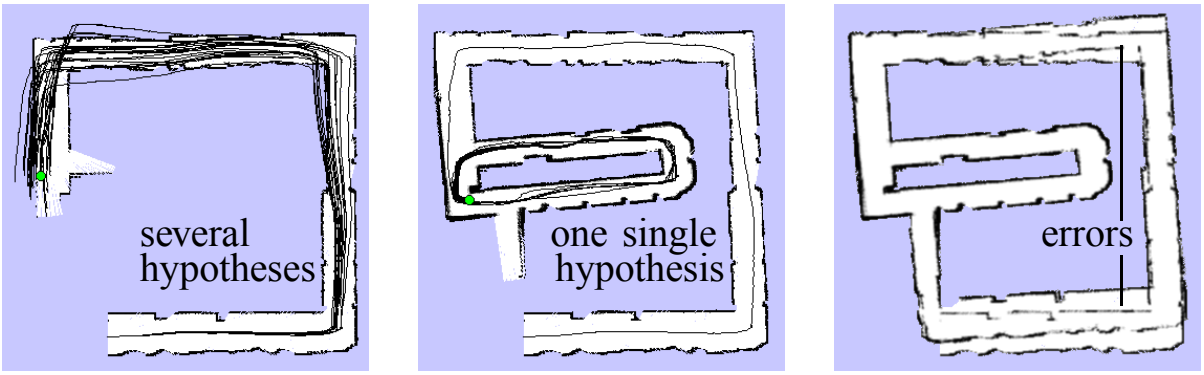
Recovering Particle Diversity in Loops

In this approach (STACHNISS; GRISETTI; BURGARD, 2005b), the authors make use of a previous technique for closing loops (STACHNISS; HAHNEL; BURGARD, 2004) with two important differences. First, the criterion for leaving the loop-closure is no longer the entropy, instead they use the effective number of particles (LIU, 1996), $\boldsymbol{N_{eff}}$, of the RBPF. It can be computed as

$$\boldsymbol{N_{eff}} = \frac{1}{\sum_{i=1}^{N} (w^{[i]})^2} \tag{5.2}$$

where $w^{[i]}$ is the current weight of the *i-th* particle.

Second, they try to overcome the particle depletion problem (see Fig. 5.1) by keeping the diversity of the filter before the robot starts the loop-closure procedure. This problem is particularly important when one decides to close an inner loop (see Fig. 5.1). As one can see in (a), when the robot approaches an inner loop, there is a good diversity of particles to close the outer loop. In (b), as the robot closes the inner loop a given number of times, local decisions enable the proper inner loop-closure. Unfortunately, this locally correct procedure filters out the particles required to close the outer loop (see (c)), leading to a globally incorrect solution.



(a) (b) (c)

Figure 5.1: A particle depletion example (extracted from (STACHNISS; GRISETTI; BURGARD, 2005b)). In (a), before the loop is finished, there are plenty of particles. In (b), we see that when the smaller loop is closed, the filter diversity drops considerably. Finally, in (c), when the robot tries to close the bigger loop, there are no particles containing the global solution.

The rationale behind this strategy is that after a loop-closure procedure the map will be locally consistent, although the global map may not be. Therefore, if one could keep the diversity of the filter before the local map was constructed, the local solution could be found and then "glued" to the previous diversity.

In practice, the filter is stalled at time-step, t_s, before the loop-closing behavior starts, and the particles representing the probability distribution,

$$\boldsymbol{p}(\mathbf{x}_{1:t_s} \mid \mathbf{z}_{1:t_s}, \mathbf{u}_{1:t_s}), \tag{5.3}$$

at t_s are stored. Then, another filter is started to find a set of valid local solutions for the current loop. The robot is kept in the loop considering the stability of the $\boldsymbol{N_{eff}}$. When the robot leaves the loop, these local solutions, represented by

$$\boldsymbol{p}(\mathbf{x}_{t_s+1:t} \mid \mathbf{x}_{1:t_s}, \mathbf{z}_{t_s+1:t}, \mathbf{u}_{t_s+1:t}), \tag{5.4}$$

are importance-sampled and attached to the diversity of particles representing the probability distribution of Eq. 5.3 stored before the start of the loop-closure. As a result, the uncertainty is propagated through the loop and the hypotheses needed to close an outer loop will have an increased chance of being kept.

The authors argue that their method can reduce the number of particles required for the RBPF SLAM. They present results showing that the current method outperforms a previous

approach (STACHNISS; HAHNEL; BURGARD, 2004). This strategy enables the robot to stay in a loop for an allegedly unlimited amount of time.

Still, a major problem of this approach is that the uncertainty of loops may decrease even before the particle diversity is registered or stored. The reason is simple, their loop-closing algorithm only detects nearly-closed or closed loops (as seen in Section 5.2.1). If the distance of nearby nodes in the graph is equivalent to the reach of the exteroceptive sensors, there is a chance that most particles will already have been lost at instant $t_s + 1$, before the relevant diversity is stored. Hence, an approach that controls the loop-closure from start to end, such as Ouroboros (see Chapter 7), could benefit from the proposed particle recovery approach. In particular, any technique that assumes a loop is being closed can perform better than a behavior that loses diversity before the particle recovery process is activated.

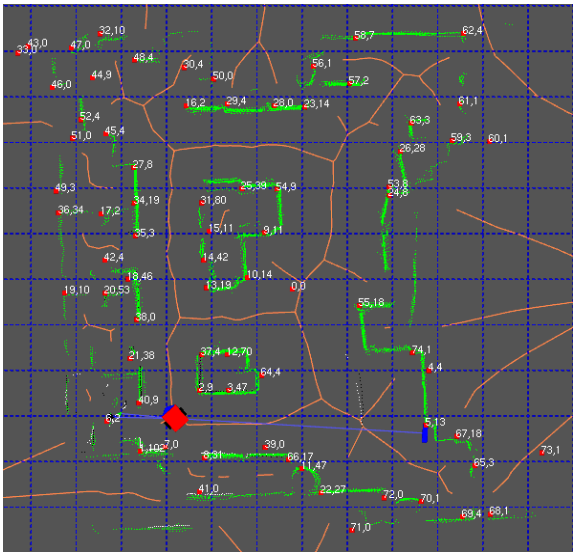
Loop-Based Exploration

Tungadi and Kleeman (2009) developed an exploration technique that uses a Voronoi diagram to detect and close loops. Their algorithm tries to close small loops[4] instead of larger ones. They first create the Voronoi skeleton, $\mathcal{V}(t)$, of the known space over an occupancy grid. Then, they construct an **undirected** graph $\mathcal{G}$, where the nodes, $\mathcal{N}(\mathcal{G})$, are junctions and end points of the Voronoi skeleton and edges are weighted according to their length. After the construction of this graph, a depth-first search is performed to find all paths leading to loop-closure opportunities. Hence, the shortest path is followed by the robot to close a loop. This technique is important because it presents an algorithm that plans the exploration considering all the loops in the environment. Fig. 5.2 presents the Voronoi skeleton in orange and obstacles in green.

A fundamental difference between this algorithm and both the techniques we propose is how we detect and close loops. Potential Rails (see chapter 6) uses a very simple heuristic to close loops that does not require a graph search, only a set of consecutive decisions about when the robot should turn to the left or to the right. On the surface, the other technique presented in this thesis, Ouroboros (see chapter 7), looks similar to the technique proposed by Tungadi and Kleeman (2009), but they are in fact quite different. Ouroboros can be implemented using two boundary conditions inserted in the map. These two new boundary conditions naturally generate a potential field that guides the robot in the direction of a loop closure. We can reduce

[4]This loops cannot be two small otherwise they are ignored

Figure 5.2: Result extracted from Tungadi and Kleeman (2009) algorithm. The Voronoi skeleton is shown in orange, while obstacles are in green.



the computational cost of the relaxation of the potential field by means of a local window, combined with the information of the Voronoi diagram. Even though such optimization is similar to what Tungadi and Kleeman (2009) do, Ouroboros does not require an adjacency list or a compulsory search for all loops in the graph. Once the conditions for the construction of a loop are established, the algorithm just walks over the Voronoi skeleton performing a breadth-first search for the nearest valid loop. Another key aspect is that our search is directed, i.e., the algorithm induces a loop by creating a goal position behind the robot (over the Voronoi skeleton), followed by the construction of a virtual wall between the robot and the goal to block searches backward. Therefore, as soon as the search reaches the robot, we already know it is a possible loop. This way, we guarantee that a tentative loop path will be formed and traveled as soon as it is discovered, allowing Ouroboros to take a decision faster. Besides, as we are supported by BVP exploration, we can also dodge obstacles while constructing the map, which allows the navigation on semi-static environments. Finally, as our decision is local, it scales better for larger maps.

Simulation-based Approaches

Even though loops are crucial for SLAM algorithms and allegedly the goal of Active SLAM, some authors decided to see it as an optimization problem where one among the possible paths the robot can take is elected after the computation of specific heuristics. For this purpose,

such techniques simulate eventual gains in choosing each different path. These techniques actually take advantage of loop-closure strategies described above.

Joint Entropy Based Exploration

Stachniss et al. (STACHNISS; GRISETTI; BURGARD, 2005a) devised an entropy-based approach to alternate between closing loops, re-localization and moving to unknown regions. It uses the same loop-closure strategy devised by Stachniss, Hahnel and Burgard (2004). The idea is to keep the robot inside a loop only while the diversity is not degraded. This Active SLAM approach makes use of a traditional exploration strategy[5] to test the change of entropy of the SLAM posterior. This is done by a simulation of movement toward the different frontiers and measuring the change of entropy. The chosen path is that which returns the most appropriate gain, considering also a cost function to reach each destination.

The joint entropy of the map and the trajectory, given the available observations and the sequence of commands, is given by[6]

$$\begin{aligned} \boldsymbol{H}\big(\boldsymbol{p}(m, \mathbf{x}_{1:t} \,|\, \mathbf{u}_{1:t}, \mathbf{z}_{1:t})\big) &= \boldsymbol{H}[\,\boldsymbol{p}(\mathbf{x}_{1:t} \,|\, \mathbf{u}_{1:t}, \mathbf{z}_{1:t})\,] + \\ + \int_{1:t} \boldsymbol{p}(\mathbf{x}_{1:t} \,|\, \mathbf{u}_{1:t}, \mathbf{z}_{1:t})\, \boldsymbol{H}[\,\boldsymbol{p}(m \,|\, \mathbf{x}_{1:t}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t})\,]\, d\mathbf{x}_{1:t}. & \end{aligned} \tag{5.5}$$

However, the RBPF SLAM posterior consists of a discrete set of weighted particles, which allows the substitution of the integral by a sum. It means the importance of the entropy of each particle will be weighted according to its underlying weight. In other words,

$$\boldsymbol{H}\big(\boldsymbol{p}(m, \mathbf{x}_{1:t} \,|\, \mathbf{u}_{1:t}, \mathbf{z}_{1:t})\big) = \boldsymbol{H}[\,\boldsymbol{p}(\mathbf{x}_{1:t} \,|\, \mathbf{u}_{1:t}, \mathbf{z}_{1:t})\,] + \sum_{i=1}^{N} \omega^{[i]}\, \boldsymbol{H}[\,\boldsymbol{p}(\mathbf{m}^{[i]} \,|\, \mathbf{x}_{1:t}^{[i]}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t})\,]\, d\mathbf{x}_{1:t}. \tag{5.6}$$

where $\mathbf{m}^{[i]}$ and $\mathbf{x}_{1:t}^{[i]}$ are respectively the map and trajectory of the $i-th$ particle at each time step.

Note that we are dividing the entropy of the full SLAM posterior distribution into two entropies: the one associated to the trajectory; and that pertaining to the map. As we are using occupancy grid maps and considering that the posterior of each cell, $m_{x,y}$, belonging to the map, m, is assumed to be conditionally independent from one another, we can compute the map

[5] They use Yamauchi (1997)'s approach. Nevertheless, any other frontier approach could be used.

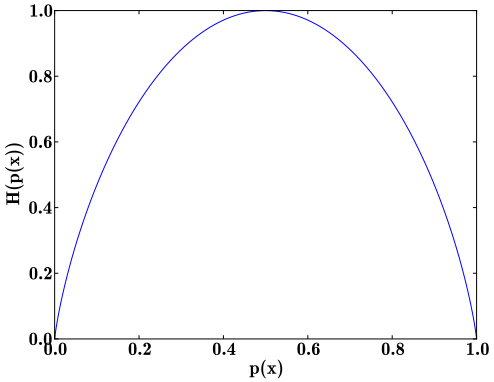
[6] We present how the authors derive the joint entropy of two distributions in Appendix C.

entropy as

$$\boldsymbol{H}(\mathbf{m}) = -\sum_{m_{x,y}\in\mathbf{m}} \left(\boldsymbol{p}(m_{x,y})\,\boldsymbol{log}[\,\boldsymbol{p}(m_{x,y})\,] + (\,1-\boldsymbol{p}(m_{x,y}))\,\boldsymbol{log}[\,(\,1-\boldsymbol{p}(m_{x,y})\,)\,] \right). \quad (5.7)$$

In Eq. 5.7, the entropy of each individual cell is summed to obtain the map entropy. The individual cell probability is associated to the probability that it is free, along with probability that the cell is an obstacle. One may notice that the behavior of such function is not determined when $\boldsymbol{p}(m_{x,y}) = 0$ or $\boldsymbol{p}(m_{x,y}) = 1$. Still, if we constrain the corresponding maxima and minima, it $\boldsymbol{p}(m_{x,y})$ will never reach 0 or 1. The plot of $\boldsymbol{H}(\boldsymbol{p}(m_{x,y}))$ when $0 < \boldsymbol{p}(m_{x,y}) < 1$ is shown in Fig. 5.3.

Figure 5.3: The behavior of a single cell entropy according to Eq. 5.7 and considering that $0 < \boldsymbol{p}(x) < 1$. This is the direct result of treating individual cells as discrete random variables with two outcomes (a Bernoulli distribution).



Observe that the cell entropy achieves its peak when $\boldsymbol{p}(x) = 0.5$, i.e., when we do not know if the cell is free or an obstacle. The authors mention that the map entropy relies on the map discretization. Thus, better refinement leads to better entropy calculation.

The problem now is to compute the entropy, $\boldsymbol{H}[\,\boldsymbol{p}(\mathbf{x}_{1:t} \,|\, \mathbf{u}_{1:t}, \mathbf{z}_{1:t})\,]$, associated to the robot trajectory. This implies that the selection of the optimal trajectory which reduces the entropy requires a test of all possible trajectories leading to all possible goals. Finding these trajectories gets even more complicated with RBPF SLAM approaches, since for every particle there is a map. Furthermore, it means that every pose along the selected trajectory changes the entropy in ways which are not clear until the environment is explored. Stachniss, Grisetti and Burgard (2005a) makes a series of approximations to obtain the entropy of a single map, and also of the trajectory to each frontier considering a frontier-based exploration approach

(YAMAUCHI, 1997). The entropy is then computed considering the average entropy of the pose posteriors, i.e.

$$\boldsymbol{H}(\boldsymbol{p}(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})) \approx \frac{1}{t} \sum_{t'=1}^{t} \boldsymbol{H}(\boldsymbol{p}(\mathbf{x}_{t'} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}))$$

This approach is similar to Roy et al. (1999), but made only over the known space (because this is an exploration).

However, calibrating the weights of linear combination of utility and cost functions requires a careful setting which can be map-dependent (CARLONE et al., 2010). Blanco, Madrigal and Gonzalez (2008) state that the way to compute the path entropy is unclear ans also show that following mathematical treatment of Roy et al. (1999) may sometimes lead to undefined values, specially after loop closing[7]. We highlight that this approach is computationally costly since the changes in the entropy of the pose must take into consideration each of the possible paths for each particle. Consequently, some approximations that overpass the mathematical treatment given in the article are required to make such technique feasible — e.g.: considering only a single particle sampled from the set instead of all of them; making the change of entropy "proportional to the amount of unknown cells enclosed by a beam"; testing only some of the possible paths; and so on.

Expected Map

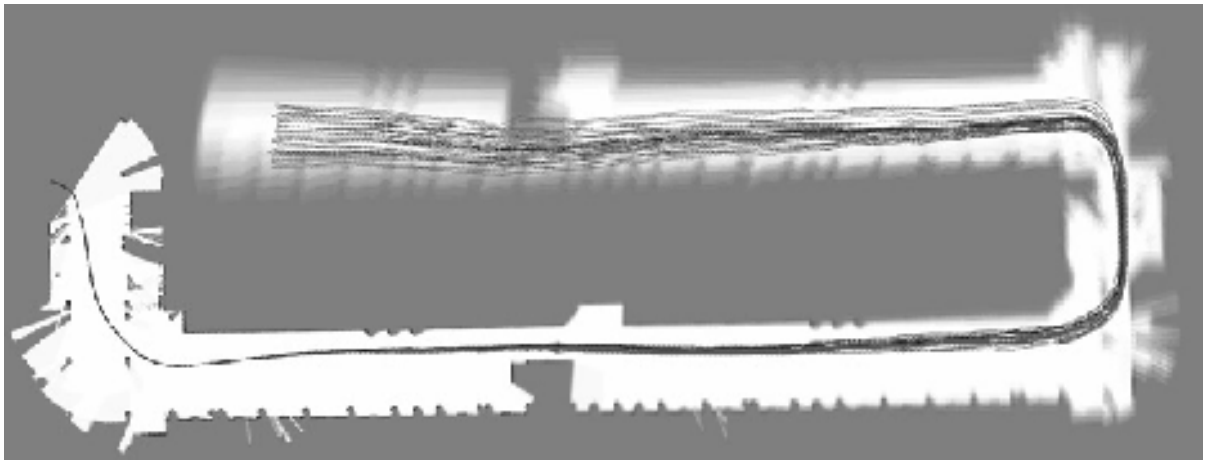
Blanco, Madrigal and Gonzalez (2008) proposed another way to measure the uncertainty about the SLAM posterior. The key aspect of such approach is the construction of an Expected Map ($\mathbf{EM}$), which is obtained from the weighted average of maps considering the contributions of all particles in the RBPF SLAM (see Fig. 5.4).

Hence, given the posterior,$\boldsymbol{p}(\mathbf{x}_{1:t} \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})$, of the robot pose, we can write the posterior of $\hat{\mathbf{m}}$ as

$$\boldsymbol{p}(\hat{\mathbf{m}} \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t}) = \boldsymbol{E}_{\mathbf{x}_{1:t}} [\, \boldsymbol{p}(\mathbf{m} \mid \mathbf{x}_{1:t}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}) \,], \tag{5.8}$$

[7] When particle depletion occurs, which may occur after the closure of a long loop, only one particle may survive. Thus, the covariance of the Gaussian used to implement path entropy becomes 0 leading the entropy to minus infinity — see Roy et al. (1999), Stachniss, Grisetti and Burgard (2005a), and Blanco, Madrigal and Gonzalez (2008).

Figure 5.4: The Expected Map of a RBPF SLAM approach. Extracted from Blanco, Madrigal and Gonzalez (2008)



where $\boldsymbol{E}_{\mathbf{x}_{1:t}}$ is the expectation of the map given the distribution of trajectories. Specifically in the case of the RBPF SLAM the values at each cell,$EM_{x,y}$, of the $\mathbf{EM}$ can be computed as

$$\boldsymbol{p}(\hat{\mathbf{m}}_{x,y} \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t}) = \sum_{i=1}^{N} w^{[i]} \boldsymbol{p}(m_{x,y}^{[i]} \mid \mathbf{x}_{1:t}^{[i]}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}), \tag{5.9}$$

where N is the number of particles and $m_{x,y}^{[i]}$ is the grid cell of the map, $\mathbf{m}^{[i]}$, belonging to the *i-th* particle.

The expected map, $\hat{\mathbf{m}}$, is used to evaluate the consistency of all the individual maps considering the available hypothesis, which also implicitly evaluates the paths which generate such maps. The rationale behind the expected map is that the different possible paths should give preference to those which result in sharp expected maps. However, the authors mention that the direct use of the entropy, as in Stachniss, Grisetti and Burgard (2005a), to evaluate a given map may be inconvenient, since it turns the resulting information dependent on the underlying grid dimensions and resolution. It happens because the entropy of the map becomes a sum of the probability of each cell being occupied or empty which represent Bernoulli distributions (see Eq. 5.7). As a consequence, unknown cells return the highest individual entropy. Therefore, the higher the number of unknown cells, the higher the entropy. This unbounded entropy growth may be problematic for larger maps[8].

For this reason, Blanco, Madrigal and Gonzalez (2008) proposed an information, $\boldsymbol{I}$,

[8]Stachniss (2006) proposed a scaling factor to mitigate such problems in his thesis.

which considers the cell entropy but not directly, i.e.,

$$\boldsymbol{I}(m_{x,y}) = 1 - \boldsymbol{H}(m_{x,y}). \tag{5.10}$$

Consequently, the chart presented in Fig. 5.3 has its concavity reversed, returning the amount of bits required for each cell. Then, the overall map information, in bits, can be computed as

$$\boldsymbol{I}(\mathbf{m}) = \sum_{\forall\, x,y} \boldsymbol{I}(m_{x,y}). \tag{5.11}$$

Such formulation eliminates the importance of unknown cells in the information metric, solving the unbounded growth problem.

While this solves the question regarding what is the amount of data available in a given map (in bits), the authors are also interested in the certainty of the information provided by such map, which is given by the *Mean Information*, $\mathbf{MI}$ or $\bar{\boldsymbol{I}}$.

$$\bar{\boldsymbol{I}}(\mathbf{m}) = \begin{cases} \frac{\boldsymbol{I}(\mathbf{m})}{N_{obs}} & \text{if } N_{obs} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

where N_{obs} is the number of visited cells in the map.

This approach produces values within the interval $[0, 1]$, while at the same time being less dependent on the map size and resolution. Despite that, we highlight that a poor representation of the posterior distribution due to particle depletion might result in a poor map posterior distribution. The certainty provided by $\bar{\boldsymbol{I}}$ may not mean anything in this case.

Kullback-Leibler Divergence as a Measure of Uncertainty

The Kullback-Leibler (KL) divergence (KULLBACK; LEIBLER, 1951; FOX, 2003; LI; SUN; SATTAR, 2013) is used to measure the divergence between two distributions, $\boldsymbol{p}$ and $\boldsymbol{q}$. It can be written as

$$\boldsymbol{\xi}(\boldsymbol{p}, \boldsymbol{q}) = \sum_{x=1}^{k} \boldsymbol{p}(x) \boldsymbol{log} \frac{\boldsymbol{p}(x)}{\boldsymbol{q}(x)}, \tag{5.13}$$

where x represents the bins associated to both discrete distributions. The Kullback-Leibler divergence is also known as KL Distance, Cross Entropy, Relative Entropy, Information Divergence, and Information for Discrimination.

Amigoni and Caglioti (2003, 2004) proposed an information based decision process for

the exploration problem. They rely on the concept of relative entropy to choose among randomly selected positions. Later, Amigoni (2008) performed experiments comparing different exploration strategies and shows that techniques which balance utility and cost are better than those using only utility. Next, Amigoni and Caglioti (2010) proposed a technique that blends information gain and traveling cost to construct maps using salient points in the map as safe zones for the robot. The approach uses a non-informative distribution to compare the different motion policies. Also, the authors prefer to use the Neperian instead of base two logarithms, what is unorthodox since the latter would return bits. The experiments available in these works generally evaluate map construction speed, and not its underlying quality, a critical aspect of Active SLAM approaches. In fact, none of them presents experiments with results associated to SLAM quality. The mathematical derivation of the method is provided in Appendix C.3.

An important aspect about this approach is that instead of a linear combination of cost and information gain (as in Stachniss, Grisetti and Burgard (2005a) and Blanco, Madrigal and Gonzalez (2008)), the authors opted to divide the gain by the cost of reaching different regions. Many comparisons were performed with other techniques (DU et al., 2011). However, we stress that their approach performs considerable computation for a technique that ultimately just explores and closes loops (CARLONE et al., 2010). Furthermore, the authors also stress that the approach is too pessimistic, considering always the worst case scenario. Finally, we highlight that uncertainty reduction does not imply in the correct SLAM result. This, in fact, may be the result of particle depletion (STACHNISS; GRISETTI; BURGARD, 2005b).

Suboptimal approaches

Suboptimal approaches are not worried with the actual optimal path. Such approaches are worried with locally optimal solutions. In fact, in most cases, path simulation has shown that numerous assumptions have to be made to reduce the complexity of the problem (VALLVé; ANDRADE-CETTO, 2015), which make them suboptimal.

Makarenko et al. (MAKARENKO et al., 2002) proposed a navigation strategy based on an utility function which considers the map construction coverage, accuracy, and exploration speed. The localization accuracy is measured using a Kalman filter at a few selected locations during planning. The best positions are used as beacons for the robot. They present partial maps with good initial results.

Juliá et al. (2010) present a hybrid reactive/deliberative method to multi-robot integrated exploration. The design of the reactive and deliberative processes is exploration-centered, where

both have the same importance level. They rely on the concepts of expected safe zone and gateway cell to avoid the presence of local minima and to decide between exploring the current zone or changing to other zones. They also take into account localizability measures to decide if it is necessary to return to previously explored areas.

Attractor Features using Predictive Model Control

Leung, Huang and Dissanayake (2006) devised an Active SLAM approach which uses attractors as auxiliary information to a local planner (n-steps Model Predictive Control with $n = 3$). In this case, attractors are modeled as artificial features strategically placed in the environment according to the corresponding task:

1. exploration: an unknown location point is selected;
2. improve localization: a point near or at a good feature is selected;
3. improve map: a poor feature is selected.

When a task must choose among different points, an heuristic of minimum distance is applied to determine which point can be reached faster.

Presented results are better than greedy approaches, but the authors state that the algorithm does not guarantee complete coverage of the environment. Besides, depending on the velocity and maximum turn-rate, there may be no feasible control options available for the robot. Their solution to the latter problem is to stop the robot and then turn it randomly.

Despite these problems, the approach turns up as an important step in the direction of the elimination of the need for costly global planners with longer horizons. Also, we share the notion that complex behaviors can be obtained by the use of simple features placed in the environment. Even though this is similar to the use of attractor points in Ouroboros (see Chapter 7), we can generate complex behaviors using only two boundary conditions allied with a local (or global) planner (PRESTES et al., 2002): a shield feature and a single attractor point. Another technique in this Thesis, Potential Rails, changes the parameters of the planner through an heuristic algorithm that continuously makes turns which may lead to a loop (see Chapter 6).

Trying to Avoid Path Simulation

Carlone and Lyons (2014) constrained the exploration to the uncertainty about the robot pose. An elegant formulation is proposed where the problem is formally stated, considering the requirements for exploration, collision avoidance, robot limitations (speed), and uncertainty reduction. The exploration problem is translated into a mixed-linear integer programming (an NP-hard problem) where the trajectory is planned T steps ahead. The error about the robot pose considers the Best Linear Unbiased Estimator (BLUE), since it can be computed efficiently while providing a probabilistic setup. The approach is tested in numerical simulations.

While this is an interesting idea, the algorithm does not perform actual loop-closing behaviors. Instead, it forces the robot to turn around when the uncertainty increases, in situations where it might not be necessary. We highlight that returning to previously known regions using the same path may have no practical result for the improvement in the quality of the map. What is more, depending on the SLAM approach, the uncertainty may decrease, but diverging from the correct solution.

Merging Potential Fields with Entropy Information

Vallvé and Andrade-Cetto (2015) devised a decision theoretic approach for exploration which minimizes map and path entropies. The method is executed considering a gradient descent over a potential field, producing better maps than competing strategies. This technique merges potential fields with entropy information, something we have not done yet. The authors state that computing the optimal path in an exploratory process, taking into account the effect of the path in the reduction or increase of joint path and map entropies, is a computationally intractable process. Their suboptimal solution assumes that the robot can reach each region of interest in one single step. This assumption is invalid whenever the robot path has few information, such as long corridors, sparse or symmetric regions, and so on. Still, the authors stress that loop closing reduces uncertainty, which is precisely our goal while modifying the BVP exploration algorithm.

A discussion on the Related Works

In general, most works on Active SLAM provide little or no comparison with other techniques. Furthermore, we can see that most robust strategies actively search for loops. We also noticed that information-based approaches usually require an unrealistic computational cost to test possible paths and decide what region to explore. In addition, such techniques must extrapolate the amount of information obtained when the elected region is unknown. Besides, comparative results, considering simulation, presented contradictory evidences of improvement while comparing the state of art (see (BLANCO; MADRIGAL; GONZALEZ, 2008) and (DU et al., 2011)). A major problem of such approaches is the assumption that the SLAM posterior is always a good estimate for the true SLAM posterior. As Stachniss, Grisetti and Burgard (2005b) noticed, particle depletion can lead to poor posteriors, if this happens none of the approaches would work properly. Moreover, approaches using the Kullback-Leibler divergence often rely on assumptions, e.g., normality and high number of samples, that may not be reasonable for Active SLAM.

However, in the context of SLAM, loop-closure is one of the most important actions required by Active SLAM strategies. While it seems interesting to go back to salient points seeking to reduce the uncertainty about the SLAM posterior, a RBPF SLAM usually fails to construct maps when the robot goes back using previously known paths. Therefore, in this thesis we consider techniques that explore regions using loop-centered behaviors making use of modified BVPs, which can find a loop without explicit loop searches. In addition, our techniques make use of the Voronoi skeleton as a supporting tool to fast propagate the potential field generated by the BVP Exploration at distant regions in the environment and also to give support for the detection and closure of loops. When we combine both the BVP Exploration and the Voronoi skeleton the potential propagation over the Voronoi fastly and naturally attracts the robot toward the smallest loop.

Loop-closures and Mazes

An important aspect of both Ouroboros and Potential Rails is the idea of marking the path traveled by the robot. This technique is well-known from the general public through Hansel and Gretel's fairy tale, where Hansel leaves a trail of bread crumbs to be able to go back home[9].

[9] "Hansel and Gretel" is a fairy tale of German origin published in 1812 by the Grimm Brothers<https://en.wikipedia.org/wiki/Hansel_and_Gretel>.

The different algorithms we propose mark the floor for different reasons: Ouroboros (see Chapter 7) marks the occupancy grid to record what loops were already closed; Potential Rails (see Chapter 6) records time of visit information which is used to calculate what is the potential generated by such region. ’

The Active SLAM problem is somewhat similar to that of finding an exit of a maze. In this context, the Trémaux’s maze solving algorithm — also known as the depth first search (TARJAN, 1972) — makes use of markings on the floor to try to find an exit of a maze. Another maze solving algorithm which uses similar approaches is the breadth first search algorithm (LEE, 1961; MOORE, 1959). Depth first search was used to find loops in an Active SLAM approach by Tungadi and Kleeman (2009) over the Voronoi skeleton to find all loops and the close the nearest one, while Ouroboros (see Chapter 7) makes use of breadth first search. Note that Ouroboros could also use a depth first search, even though it does not guarantee the use of the smallest path to close loops. A longer path could lead to undesirable increase in the uncertainty about the SLAM posterior. This does not make any difference for an approach that detects all loops at once (e.g. (TUNGADI; KLEEMAN, 2009)), but certainly does for one that is actively trying to find only where the smallest loop is — such as Ouroboros.

6 TIME DRIVEN BVP INTEGRATED EXPLORATION (TDBVP) — POTENTIAL RAILS

The first of our integrated exploration approaches is called Time-driven BVP (TDBVP). A key feature of this appoach is that it uses local potential distortions (PRESTES; ENGEL, 2011) to induce the execution of systematic turns aiming to guide the robot toward a loop-closure. Another interesting aspect of this technique is a potential field that exists even after the end of the exploration process. That is, once the first part of the job – i.e., integrated exploration – is finished, the robot returns to previouly visited regions, giving priority to those which were not visited for the longest period of time. It can be seen as a patrolling behavior which takes place as soon as the exploration ends, leading to a smooth transition between map construction and usage.

Time Driven BVP Integrated Exploration

At the time, we hypothesized that if we knew the time of visit of a given cell, we could try to use it to guide the robot toward that region, enabling revisits. For that purpose, we have complemented the boundary conditions of BVP Exploration.

In particular, we define a third boundary condition — besides the other two specified by the traditional BVP exploration (PRESTES et al., 2002) (see section 3.4) — delineated spatially by the region encompassing the Voronoi skeleton, $\mathcal{V}$, of the known free space, $\mathcal{F}(t)$. Theoretically, the Voronoi skeleton is a closed one-dimensional (1D) structure (CHOSET et al., 2005) obtained from a contraction of the closure of the free space, $\boldsymbol{cl}(\mathcal{F})$. When we exclude $\mathcal{V}$ from $\mathcal{F}$, the known free space reduces to a *new* subset of free space

$$\mathcal{F}_{\mathcal{N}}(t) = \mathcal{F}(t) \backslash \mathcal{V}(t).$$

The clear distinction between free cells which belong to the Voronoi skeleton and those of them that do not is critical, since the numerical solution of the TDBVP will be computed only at the region of the known free space, which varies in time, where the Voronoi skeleton is not present, i.e., $\mathcal{F}_{\mathcal{N}}(t)$. As a consequence, new closure of the free space is given by

$$\boldsymbol{cl}(\mathcal{F}_{\mathcal{N}}(t)) = \mathcal{F}_{\mathcal{N}}(t) \cup \partial_{\mathcal{F}_{\mathcal{N}}}\mathcal{V}(\mathbf{t}) \cup \partial_{\mathcal{F}_{\mathcal{N}}}\mathcal{U}(\mathbf{t}) \cup \partial_{\mathcal{F}_{\mathcal{N}}}\mathcal{O}(\mathbf{t})$$

In practice, we contract the free space using a thinning algorithm[1] (GUO; HALL, 1989) to extract the skeleton cells, or center cells $m_c \subset \mathbf{m}$, whose center points are $s \in \mathcal{V}(t)$ — we treat $\mathcal{V} = \partial\mathcal{V}$, even though such algorithms usually operate on regular 2D grids which make this equality false[2].

$\partial_{\mathcal{F}_\mathcal{N}}\mathcal{V}$ will encompass the actual time driven boundary condition. Thus, our intent is to make those center cells not visited for a long time hold lower potential values (i.e. make them more attractive) than recently visited cells. Our approach to this problem is to consider the function which defines the values of the potential on $\mathcal{V}(\mathbf{t})$, as another BVP for the Laplace Equation,

$$\nabla^2 \boldsymbol{\nu}(s) = 0, \quad \text{on } \partial_{\mathcal{F}_\mathcal{N}}\mathcal{V}(t), \tag{6.1}$$

where $\boldsymbol{\nu}(s)$ is the potential at the Voronoi skeleton points, $s \in \mathcal{V}$. However, at this time the potential at each center cell is given as a function of the time-step τ, which is always recorded when a cell is visited by the robot. Naturally, the current position of the robot does not necessarily crosses the Voronoi skeleton cells. Our solution was to update the values of τ of the cells surrounding the current robot position, in a circular region of radius $r = 1\text{m}$[3].

In order to create a gradient descent toward the oldest cell, we define two Dirichlet boundary conditions for this particular BVP problem: the potential at the most recently visited cell, whose center point is s_{new},

$$\boldsymbol{\nu}(s_{new}) = 1;$$

and the potential at the oldest cell, whose center point is s_{old},

$$\boldsymbol{\nu}(s_{old}) = 0.$$

s_{new} and s_{old} are the most recently visited cell and the cell which was not revisited for the longest time respectively.

Since Eq. 6.1 is computed only over $\partial_{\mathcal{F}_\mathcal{N}}\mathcal{V}(t)$, which is a 1D structure[4], the solution of the harmonic function is a straight line between the oldest and the newest cells. Thus, we can compute the potential values of all center cells analytically through

$$\boldsymbol{\nu}(s) = \frac{\boldsymbol{\tau}(s) - \tau_{old}}{\tau_{new} - \tau_{old}}, \quad \text{on } \partial_{\mathcal{F}_\mathcal{N}}\mathcal{V}(t). \tag{6.2}$$

[1] This algorithm guarantees connectivity between the points of the skeleton.

[2] One can consider this equality true for simplicity without significant effects on TDBVP

[3] If the closest center is further away, it will not be updated with the current time-step holding its previous value, which is preset as 0 if it was never visited.

[4] Note that this a one dimensional feature over a 2D space.

where $\boldsymbol{\tau}$ is a function that returns the time of visit of a given cell m_c whose center point is s, while τ_{new} and τ_{old} are respectively the times of visit of s_{new} and s_{old}.

The potential of unmarked center cells is set to a fixed potential value of

$$\boldsymbol{\nu}(s_{unk}) = -0.5,$$

so that when the robot has to choose between unexplored and previously visited regions, this low potential will be stronger, guiding the robot toward unexplored cells. On the other hand, when choosing between two previously visited areas, the potential of the area which was visited the longest time ago will be stronger than the other. As soon as the robot visits the oldest region, it will become the newest while another one becomes the oldest. Hence, in our approach the boundary conditions over the skeleton, i.e. oldest and newest cells, change over time, in such fashion that a potential field will never cease to exist. Therefore, the TDBVP problem will never result in flattening, since one of the boundary conditions always has a potential which is different and smaller than the potential of the walls.

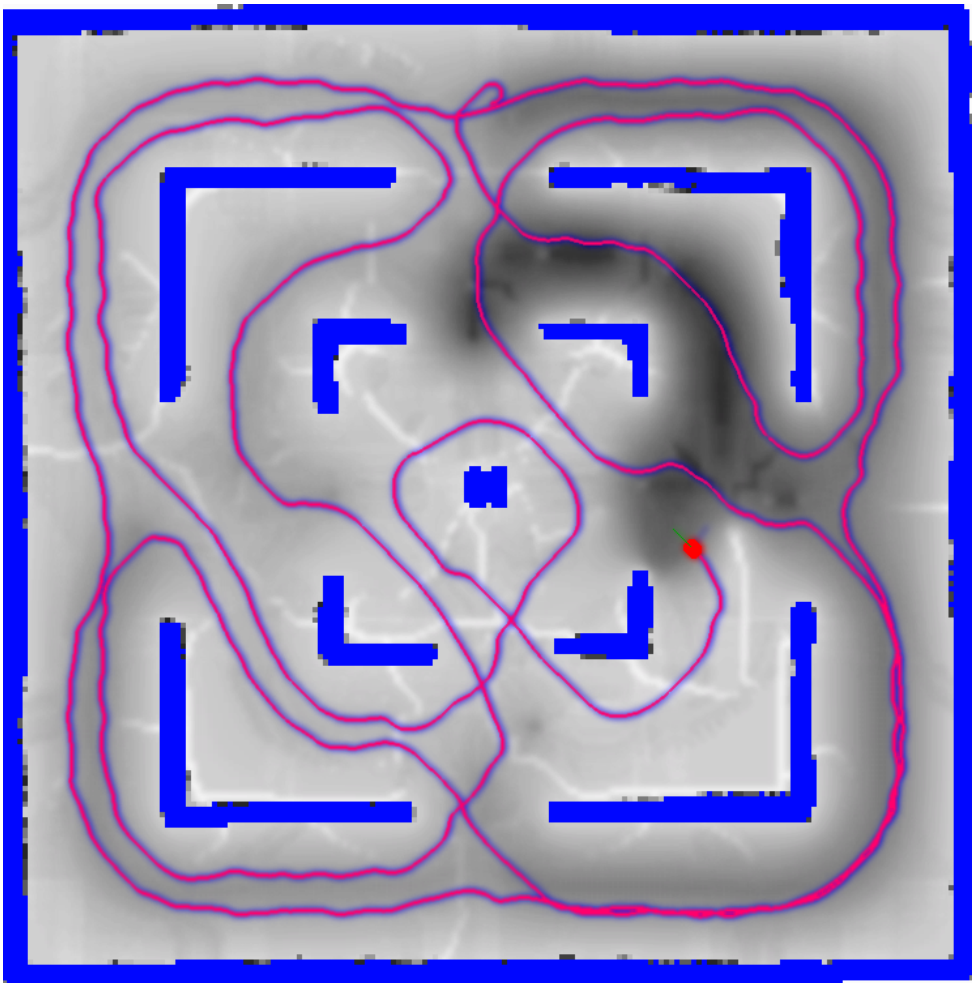
Fig. 6.1 shows an example of our integrated exploration method in a simulated environment. The robot is shown in red, its trajectory in blue, and the obstacles in brown. Grayscale values indicate different potential values: the potential field is zero when the color of the free space is black; and one when the color of the free space is white. Observe that the zone just behind the robot is the brightest area of the map, along with the surroundings of the walls, because their potentials are practically equal to 1. The darkest area of the map is the top right zone, between the two inner walls, which was visited only shortly after the beginning of the experiment — the elder cells claiming for a visit.

Therefore, there are three boundary conditions surrounding $\mathcal{F}_{\mathcal{N}}(t)$: that of obstacles; the one from unexplored cells; and that associated to the time driven boundary at the Voronoi skeleton. Together, they are used to solve the BVP at $F_{\mathcal{N}}(t)$,

$$\nabla^2 \boldsymbol{\nu}(p) = 0, \quad \text{on } \mathcal{F}_{\mathcal{N}}(t), \tag{6.3}$$

where p is the center of a cell, $m \in \mathcal{F}_{\mathcal{N}}(t)$, of the grid, $\mathbf{m}$. Note that this potential also contemplates potential distortions (PRESTES; ENGEL, 2011) defined by two additional components: a scalar value $\boldsymbol{\epsilon}(p)$; and a scaling factor given by the norm of gradient of the potential field, $||\nabla \boldsymbol{\nu}(p)||$.

Figure 6.1: Example of the potential field based on the time of visit. The dark areas are the regions visited the longest time ago.

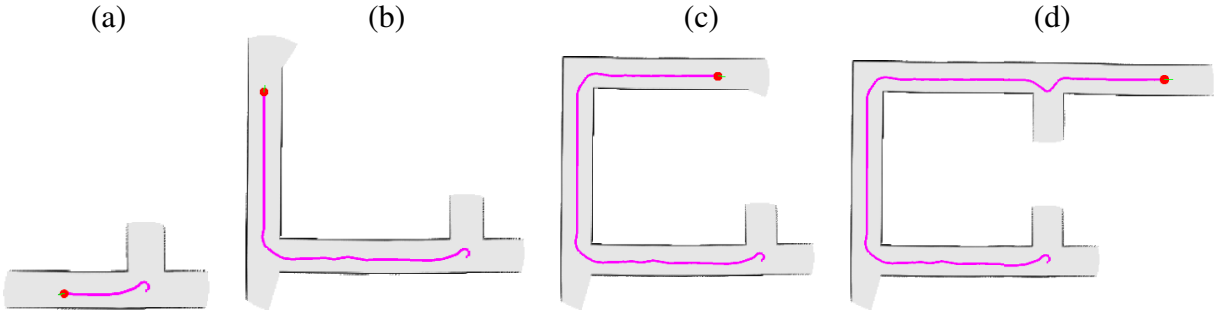


Dynamic Activation of Preferences and loop-closures

One of the conjectures posed in this thesis was that potential distortions (PRESTES; ENGEL, 2011) could indeed generate loop-closure behaviors. Local potential distortions can be implemented to exert a form of control over the robot navigation in situations of indecision. In particular, when a robot reaches an opening that ends in unknown space, all directions are somewhat equally interesting for greedy approaches. The key observation here is to try to discover which direction leads to the nearest potential loop-closure — reducing the uncertainty about the SLAM posterior. The proposed algorithm tries to do this by setting dynamic distortions in front of the robot.

In Fig. 6.2, we show that there are situations where choosing between one direction and another may be the difference between closing a loop or missing it. This is equivalent to

Figure 6.2: A robot (red dot) exploring the environment and constructing a map, (a), using a greedy exploration method — the path followed by the robot is shown in purple. Observe that the robot turned right in the first bifurcation, (b), and again in the corner. The robot has a chance to turn right or going straight in (c). Observe that in (d) the greedy exploration strategy poorly guides the robot forward. Had the robot turned right again in the opening, it would have closed a loop. That is, by using simple a heuristic, loop opportunities would not have been lost.



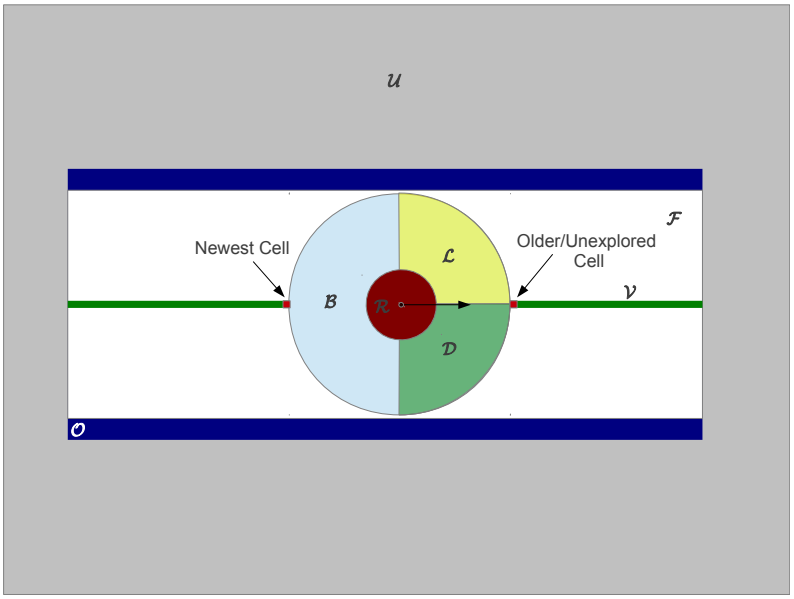
choosing between the reduction of the uncertainty about the SLAM posterior and its continuous increase. Thus, finding a way to systematically turn the robot to the desired direction would be enough to induce important loop-closures. Note, for instance, that the potential fields next to bifurcations can be similar to either side the robot goes. This is the exact case we are interested, since a preference could be used to bend the robot to the desired direction — i.e., a loop-closure. Observe that, despite the fact that potential distortions cannot avoid the gradient descent toward the goal, they might be used to guide the robot to the left or to the right in regions containing bifurcations with similar potential values. Considering that, we decided to use local potential distortions (PRESTES; ENGEL, 2011), considering Eqs. 3.26 and 3.27, with the distortion parameter ϵ changed ONLY in front of the robot, to produce controlled potential distortions and consequently systematic turns during the Active SLAM process.

In particular, ϵ would have different values on the left and on the right of the field of view of the robot. Specifically, both sides could have the same absolute value, but different signs.For that purpose, we divide the region forming half-circle in front of the robot into left, $\mathcal{L}$, and right, $\mathcal{D}$, sides[5]. Then, we apply different distortion parameters on the cells of each region. If we put a positive preference on the cells belonging to the left region (see $\mathcal{L}$ in Fig. 6.3) and a negative distortion parameter on the right (see $\mathcal{D}$ in Fig. 6.3), the potential field in front of the robot will be distorted to guide it to the left. The contrary will happen if we invert the signs of the distortion parameter. Fig. 6.4 presents three illustrative examples created using different configurations of distortion parameters in a cross-section of the potential values between two walls forming a corridor, dashed blue, where the robot is placed in the middle of the corridor.

[5]The circle radius is tipically 1.5m in our implementations.

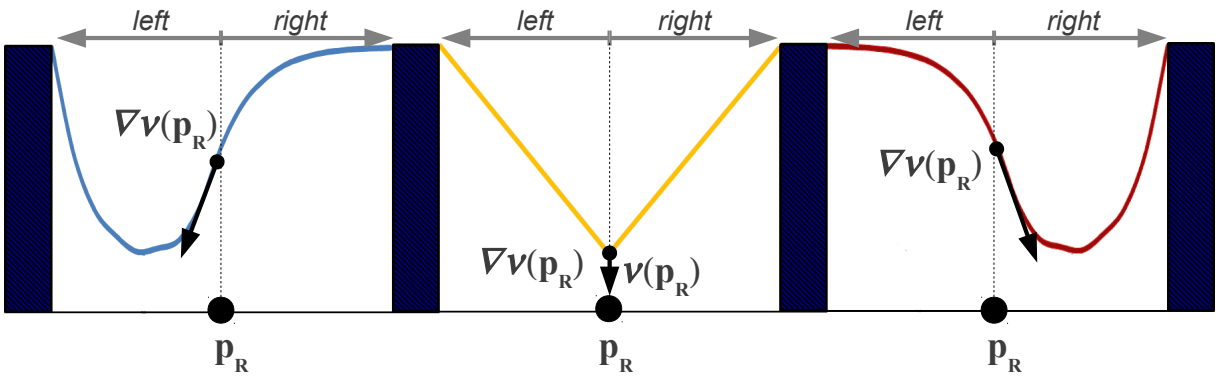
In (a), we have $\epsilon > 0$ on the left, and $\epsilon < 0$ on the right of the robot's field of view. In (b), we present the potential field without potential distortions. Finally, in (c) we have $\epsilon < 0$ on the left and $\epsilon > 0$ on the right of the robot's field of view — the opposite behavior of presented in (a). Note that the robot motion is guided by the gradient of the potential field is pointing the robot to the left, center, and right of the corridor, depending on distortion parameters set in front of the robot.

Figure 6.3: Environment separated into partitions. $\mathcal{U}$, $\mathcal{O}$, and $\mathcal{F}$ correspond to unknown (gray), obstacle (blue), and free space (white) respectively. The Voronoi skeleton, $\mathcal{V}$, is shown in green. The region at the back of the robot, $\mathcal{B}$, is the one with the highest potential value, since $\mathcal{B}$ corresponds to the most recently visited region. It guarantees that the robot will not turn back, unless it is no longer possible to go forward. Regions, $\mathcal{L}$ and $\mathcal{D}$ are those regions where ϵ distorts the potential to the left or to the right, depending on the amount of free space visible to the robot.



If we maintain the configurations of the distortion parameter fixed, the robot will always turn to the same side whenever the robot reaches a bifurcation with similar potentials. This is the case when the robot is in an unknown region consisting of a bifurcation, which is ideal for an Active SLAM strategy trying to close loops. However, if the potential is significantly lower in a region in the opposite direction of that defined by the preferences, the robot will be forced to turn to the region having low potential. This happens, for example, when the robot must choose between previously visited or unexplored regions. These behaviors occur because preferences can only direct the gradient descent, but not avoid it. As a result, there is an emergent balance in

Figure 6.4: Three examples of cross-sections of the potential affected by local potential distortions along a line perpendicular to the two walls forming a corridor. The highest value is the value of the potential field at the walls, while the lowest value occurs at the local minima. In (a) we present the gradient of the potential field considering $\epsilon > 0$ on the left of the robot and $\epsilon < 0$ on the right. In (b), we depict the potential field without local potential distortions. In (c), we see the opposite configuration of (a). Note that the robot position, p_R, is guided by the gradient (black arrows), which changes to the left, center and right of the corridor depending on how the distortions are configured. This strategy enables the robot to tend to turn to the right or to the left in regions having similar potential values.



our approach between the loop closure and the exploratory behaviors. This balance is controlled implicitly by the values of preferences and boundary conditions.

Implementation Details

We store time of visit information in the cells of map of each particle. Time information can be seen as a simple counter of the iterations of the Active SLAM algorithm. We could record this information on the cells of the Voronoi diagram where the robot travels. However, as the skeleton may change position when new obstacles are revealed, we decided to update the half-circle behind the robot, $p_{\mathcal{B}}$ within a given radius, R (see Fig. 6.3). Note that navigation is performed using the map of the particle having the maximum weight, i.e.,

$$s^* = \underset{s \in \mathcal{P}}{\boldsymbol{argmax}}\, w_t^{[s]} \tag{6.4}$$

where $\mathcal{P}$ is the particle set and $w_t^{[s]}$ is the weight of the particle $s \in \mathcal{P}$ at time-step t, while s^* is the particle with maximum weight.

Regarding preferences, note that always using a fixed configuration of preferences may

not be the best solution, because it results in a fixed turning side that may just be the opposite direction of the loop closure. So, we adapt the preferences using a simple algorithm based on the brief history of turns performed by the robot. When the robot sensors detect a large opening on the left or on the right (i.e. a sudden increase in the range of the corresponding laser readings), the method tries to set the preferences to the side of the observed opening. Yet, to avoid changing them too quickly, the preference side is only changed when two subsequent sensor readings detect an opening in that direction (see algorithm 2).

Algorithm 2: Potential Rails preference definition algorithm

```
1  initialize(∀p ∈ L → ε(p) = 0.02, ∀p ∈ D → ε(p) = −0.02, turn = 0)
2  if rangeFinderOpening() = LEFT then
3  |  turn =+1
4  else
5  |  if rangeFinderOpening() = RIGHT then
6  |  |  turn =-1
7  |  else
8  |  |  continue
9  if turn ≤-1 then
10 |  turn = -1
11 |  ∀p ∈ L → ε(p)=-0.02
12 |  ∀p ∈ D → ε(p)=0.02
13 if turn≥1 then
14 |  turn = 1
15 |  ∀p ∈ L → ε(p)=0.02
16 |  ∀p ∈ D → ε(p)=-0.02
17 ∀p ∈ B → update time of visit
18 continue
```

In order to compute the TDBVP potential we let the potential converge in front of the robot, disregarding the nearby Voronoi cells ($p \in \{\mathcal{L} \cup \mathcal{D}\}$) followed by the smoothing of the gradient in front of the robot. An important advantage of our strategy in relation to the BVP exploration, described in Section 3.4, is that our potential field can always be computed using only a small window around the robot. In our approach, the potential of the cells surrounding the robot is primarily affected by the nearby center cells and obstacles. Thus, there is no need to compute the potentials in the whole map, making this computation independent of the size of the environment. In contrast, the potentials in the traditional BVP exploration are affected by cells that can be very far in the environment. For example, when the robot is almost finishing the exploration and only a distant small portion of the map remains unexplored, the potential can take a very long time to converge and start attracting the robot (PRESTES et al., 2003). We

recommend the potential of the cells inside the local window to be initialized with maximum potential value (1.0) to avoid the effect of partial relaxations on attractor potentials. This effect can be easily observed when a wall leaves the local window. When we start setting the potential of every cell to 0 inside the local window, partial relaxations may lead to low potential regions that may be more attractive the actual goals.

Finally, we note that the Voronoi skeleton may be connected to obstacles. Thus, skeleton pruning must be performed, so that the robot is not guided toward the walls. The number of pruned cells must be smaller than the exteroceptive sensor maximum range, and also smaller than the local window.

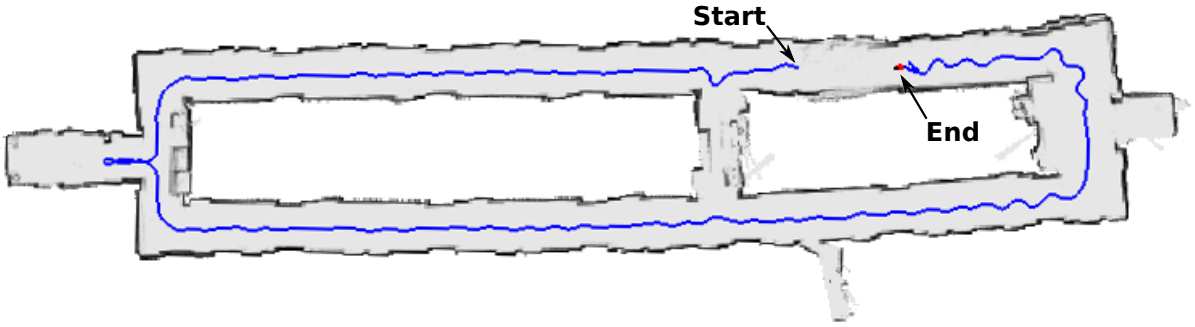
Experiments

We evaluate our method comparing it with the frontier-based BVP exploration in real and simulated environments. Both strategies were combined with a Rao-Blackwellized Particle Filter SLAM algorithm (ELIAZAR; PARR, 2004a), with laser range-finder and occupancy grids (THRUN; BURGARD; FOX, 2005). Experiments were performed on a desktop computer with an Intel® Core™ i5 with 4GB RAM running Ubuntu 14.04 64 bits. The experiments in the real environment were conducted in a building floor of $72m \times 13m$ containing two loops (with $90m$ and $60m$ of length). Fig. 6.5 presents the maps obtained by both methods using 300 particles in SLAM. We can see in Fig. 6.5(a) that the frontier-based BVP exploration did not close the two loops separately, spoiling SLAM quality. This happened because the small passageway at the center of the map is too narrow to produce a sufficient force to attract the robot when compared to those of other near unexplored frontiers.

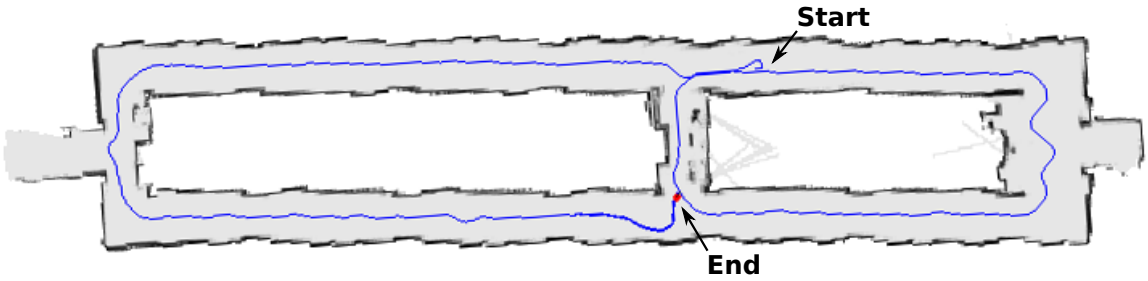
We also observe near the end of the frontier-based exploration that the robot path was not stable. This is a result of the increase in the BVP's convergence time associated to the map growth and the increased uncertainty about the SLAM posterior. The slow update of the potential field combined with constant map changes generate abrupt differences in the gradient of the potential field, which makes the robot oscillate. Fig. 6.5(b) shows the resulting map of the time-driven approach. We see that the robot separately closed both loops using a smooth trajectory. As a result, the uncertainty about the pose and the map is reduced. Lastly, another factor that impacted the underlying results is the fast convergence of the potential field in the local window.

Figure 6.5: Experiments in a real environment. The robot trajectory is shown in blue.

(a) Frontier-based BVP



(b) Time-Driven BVP

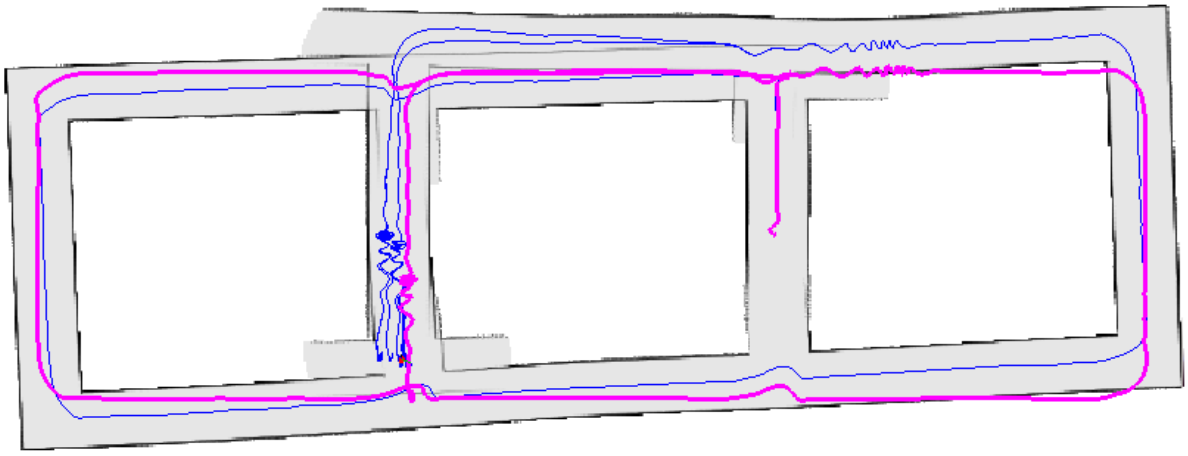


Simulated experiments were performed in two different scenarios, adding uncertainty to the sensors measurements. The first scenario is composed of three adjacent loops (with $72m$ of length each), while the second is composed of nested loops (the larger has $88m$ of length). For each scenario, we chose 10 different starting positions to perform the test for each method. In all simulated experiments, we used 200 SLAM particles — set empirically.

Fig. 6.6 presents the best visual results obtained in the adjacent loops scenario. The trajectories of the remaining particles at the end of the process are shown in blue, while the correct trajectory of the robot is shown in magenta. Fig. 6.6(a) depicts the best map generated by the frontier-based BVP exploration. The robot does not try to revisit known places, in fact, it spends too much time in the external long corridors guided to unexplored areas. As a result, its localization only gets worse with time. Later, when the robot finally returns to visited areas, there is not enough diversity to recover the quality of the map. Another issue is that the robot trajectory oscillates, just like in the real environment experiment. As the region with the lowest potential changes abruptly to a distant part of the map, the potential can take too long to converge. Fig. 6.6(b) shows the best map generated by TDBVP. As expected, the robot tends to close loop by loop, which is good to improve its localization. The estimated trajectory still deviates slightly from the correct one, but the consistency of the map is maintained.

Figure 6.6: Experiments in the adjacent loops scenario. The trajectories of the particles are shown in blue, while the correct trajectory of the robot is shown in magenta.

(a) Frontier-based BVP



(b) Time-Driven BVP

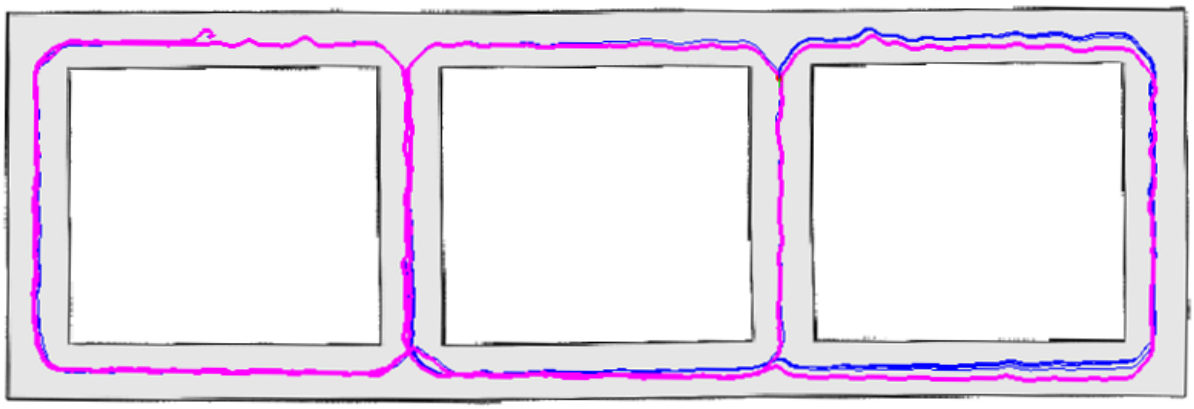


Fig. 6.7 shows the best results obtained in the nested loops scenario. As shown in Fig. 6.7(a), the frontier-based strategy did not prove to be suitable in this situation. The problem is that one badly consolidated loop can directly damage the closure of subsequent ones. In Fig. 6.7(b), we show the map obtained with our method, and, once again, the map is remarkably superior. We can see that, with our integrated exploration strategy, the diversity of the particle filter is preserved till the end of the process, differently from what we observe with BVP exploration.

Besides the visual analysis, we also compare the methods considering the mean error in the robot position along the trajectory. At each iteration of the process, the particle filter error (ξ) is computed by the mean distance between each particle position and the real robot position (which is known in a simulated experiment). The final mean error $\overline{\xi}$ is the mean of ξ over all iterations.

Figure 6.7: Experiments in the nested loops scenario. The trajectories of the particles are shown in blue, while the correct trajectory of the robot is shown in magenta.

(a) Frontier-based BVP (b) Time-Driven BVP

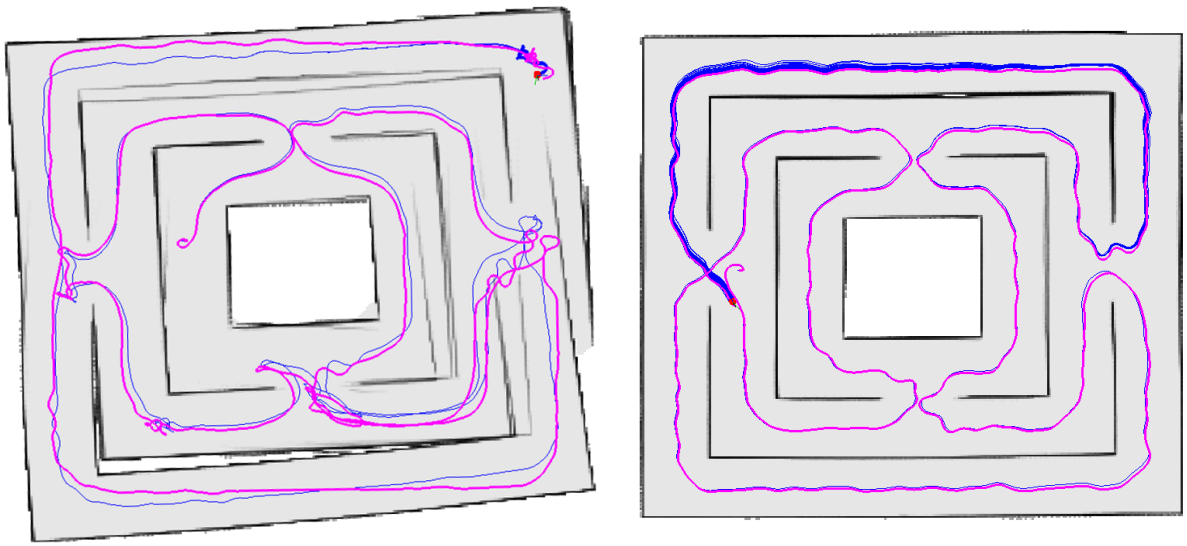


Table 6.1: Results of the experiments in the adjacent loops scenario.

	Frontier-based BVP		Proposed Method		diff
	μ	σ	μ	σ	%
Steps	1947	319.80	1971.1	186.67	+1.24
$\overline{\xi}$	2.3489	1.1733	0.3979	0.1335	-83.06

Table 6.1 shows the results for the adjacent loops scenario (Fig. 6.6). The number of exploration steps was just slightly larger using our method (1.24%). Thus, apparently, our loop closure behavior does not slow down the exploration process too much in this environment. Regarding the mean error measures, our approach presented a gain of 83.06%. Applying the Wilcoxon rank-sum test (WILCOXON, 1945) to the results after 10 runs, we prove that the mean error measures obtained by our method in this environment are significantly smaller than the ones obtained by the frontier based BVP (p-value ≈ 0.00008).

Table 6.2 shows the results for the nested loops scenario (Fig. 6.7). This time, the number of exploration steps was smaller with our approach than with BVP (-1.41%), nonetheless, the difference was not statistically significant. The mean errors are also smaller using the proposed method (56.98% smaller). After running the Wilcoxon rank-sum test for this map, we observe that our approach performed significantly better in this environment than the frontier-based BVP exploration (p-value ≈ 0.00258).

Table 6.2: Results of the experiments in the nested loops scenario.

	Frontier-based BVP		Proposed Method		diff
	μ	σ	μ	σ	%
Steps	1627.4	254.21	1604.4	290.76	-1.41
$\overline{\xi}$	1.4732	1.2783	0.6338	0.1312	-56.98

Finally, we present the histograms of the mean position error in the adjacent loops and in the nested loops scenarios, Fig. 6.8 (a) and Fig. 6.8 (b) respectively. We detect that in our approach, most of the time, 97% in the first scenario and 98% in the second, the error was smaller than $1m$, while in the frontier-based BVP approach these percentages are only 45% in the adjacent loops and 62% in the nested loops.

Finally, Fig. 6.9 presents the algorithm functioning even after the mapping process ends. Note that the potential field, indicated by a grayscale coloring scheme does not end, even after travelling over the environment many times. This is potentially interesting for patrolling.

A Discussion about TDBVP

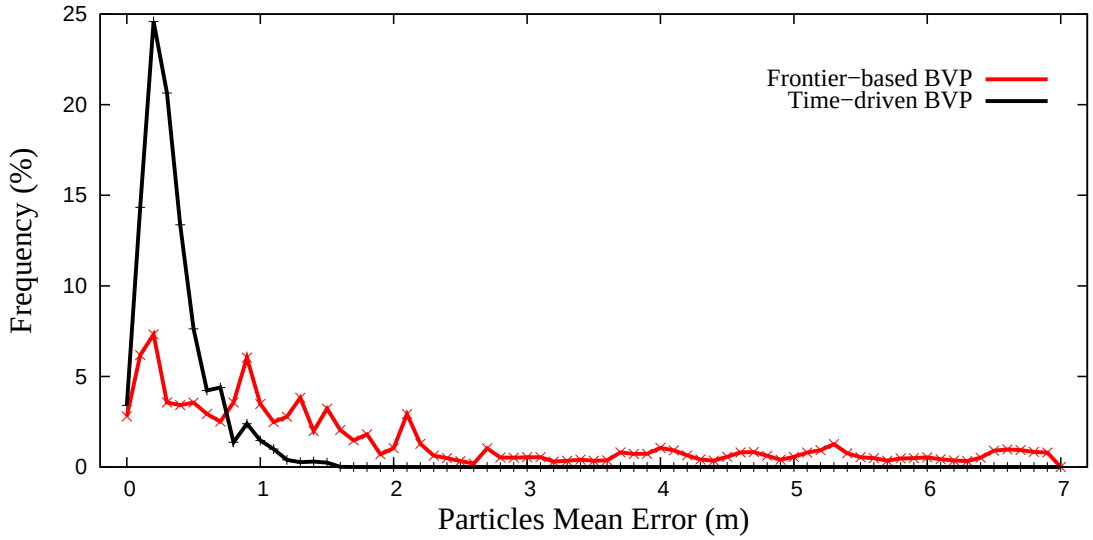
In this chapter we have proposed a strategy that blends harmonic fields and a modified Voronoi Skeleton of the known area. These techniques are used to guide the robot toward unexplored regions or to the "oldest" explored area. Our key contributions are: (i) a potential field that never ceases to exist, even if the whole map is known; (ii) a loop-closing behavior that emerges naturally from the TDBVP equations; (iii) a less expensive approach than traditional BVP-Exploration, as the potential field is computed only inside a local window, instead of the whole map.

Experiments show that TDBVP presents statistically significant improvements in terms of localization errors in simulated environments. We can also see from the results in simulated and real environments that TDBVP presents better map quality than the traditional "greedy" BVP-Exploration. Moreover, our strategy does not demand substantial increase in the exploration time. This is interesting, because the traditional BVP tends to use the shortest path to the goal avoiding revisits that could demand additional time.

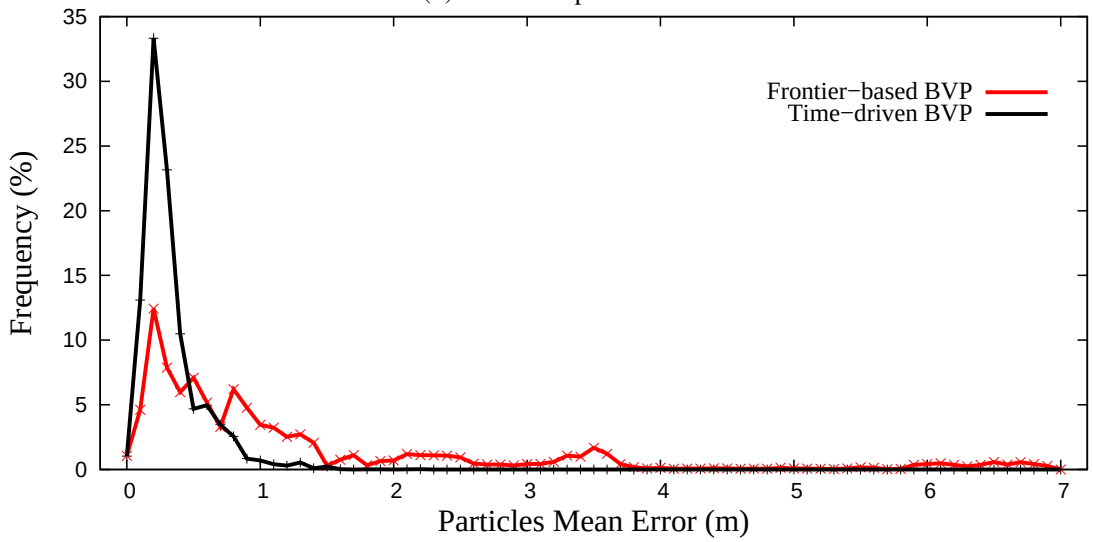
Our idea can be easily integrated to topological SLAM strategies since most of them create a path as the robot moves (be it a skeleton or a given sequence of positions). This approach is a first step in the direction of an algorithm that explores the environment and immediately starts a patrol behavior. However, we must test the algorithm in other environments.

Figure 6.8: Histograms of the mean error for the simulated experiments.

(a) Adjacent loops scenario



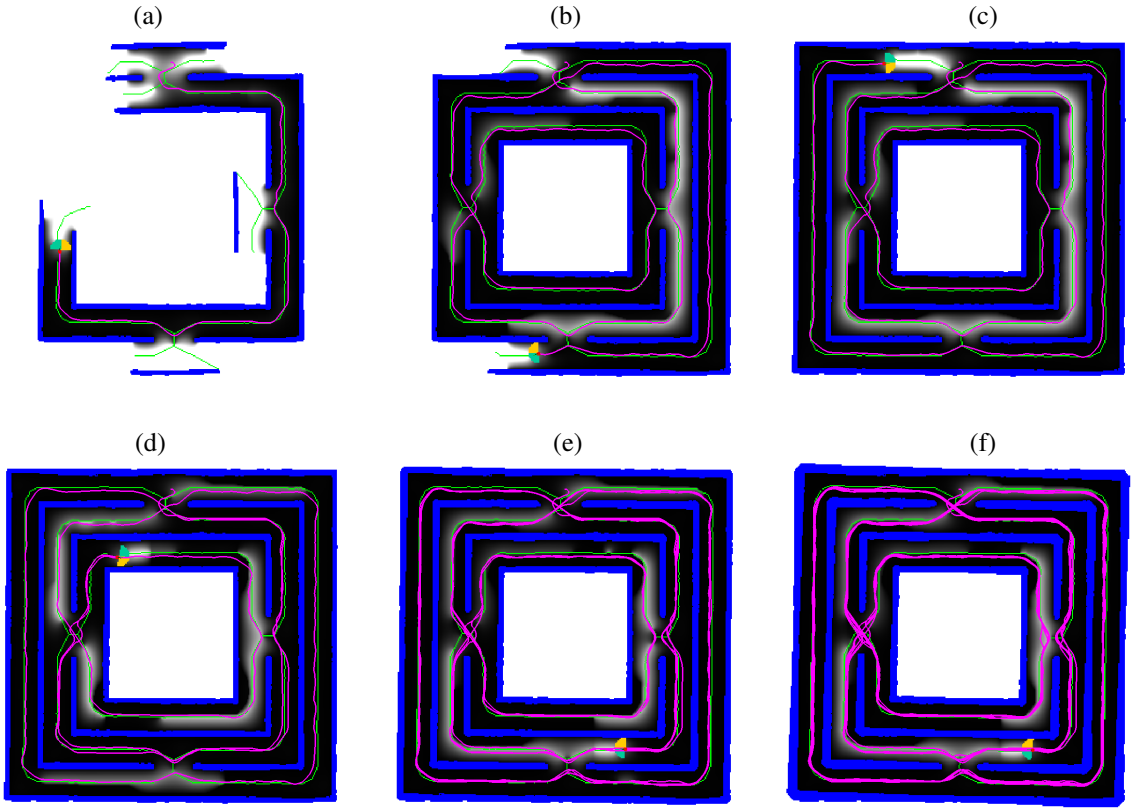
(b) Nested loops scenario



A general problem with the use of a skeleton of the free space is that they may be unstable over time, with branches changing and vanishing due to different factors, including: sensor noise; map and path characteristics; successive particle changes containing considerably different maps[6]; and even the robustness of the skeletonization algorithm. This is undesirable, since TDBVP assumes that a skeleton will always be available inside the local window. Whenever it happens, TDBVP relies on the potential of unexplored cells. If there are not nearby unexplored cells the potential inside the local window flattens due to lack of goals. When the potential is computed over the entire map, it is not a problem, even though the algorithm becomes more computationally expensive, since the numeric solution for the TDBVP must be applied to the

[6]This may happen when the particle filter has more than one mode.

Figure 6.9: Figures portraying the behavior of TDBVP over time. In (a), the robot is still at the beginning of the mapping process, while in (b) the robot is approaching the end of the mapping process. In (c) the robot ends the mapping process. We can see in (d)-(f), from the robot path in purple, that the potential field still drives the robot even after travelling over the environment many times. The potential field is depicted in a grayscale color scheme. The robot is represented by a red circle and the half-circle, in red and green, present the regions where the preferences are changed (on the left and right of the robot).



whole map. This situation can be reproduced in larger and sparser maps, which we decided not to do here since in the next chapter we devised a way to use a local window with varying size[7] — which could be easily adapted here eliminating the problem of absence of a Voronoi skeleton.

There is still space for improvement in the implementation of the method. For instance, the potential may not be smooth enough when many directions are given to the robot at the same time. Furthermore, as the goals are points nearby the robot, the gradient over a discrete grid may not be smooth enough to allow proper navigation. The algorithm that decides the preference

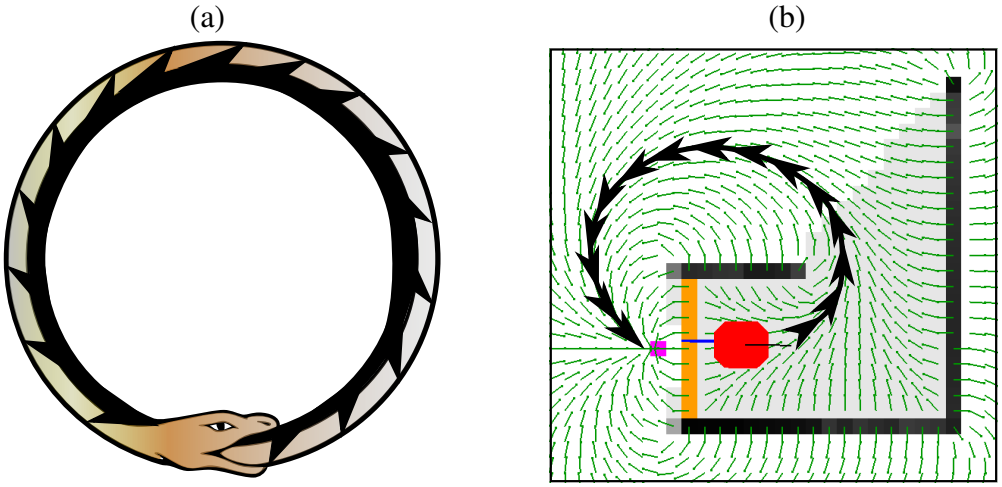
[7]We highlight that the maximum width and height of the local window required to reach a Voronoi skeleton is, in the worst case scenario, half of the available map dimensions, since the Voronoi skeleton cuts the free space at equidistant points of obstacles. This leads to reduced cost when compared to techniques that must search the entire map such as BVP exploration. Also, a simple way to know the required window size is to perform a search from the robot to the skeleton and use it to define the local window.

side during the exploration does not guarantee a turn to the left or to the right since the amount of distortion is also dependent on the environment and the size and distance of the boundaries attracting the robot. For instance, suppose the robot is in a bifurcation and it should turn to the right. If the passage to left has a large unexplored region, the strength of the distortions may fail to make the robot turn to the right, due to the large opening on the left generating a strong attractor potential. This creates a dependence between the distortion parameter values for the cells in front of the robot, on the left and on the right, and the characteristics of the boundaries attracting the robot to guarantee a systematic turn. In summary, controlling turns for arbitrary environments can be difficult and requires further research. Because of these limitations, we pursued other ways to find and close loops, culminating in the research presented in the next chapter.

7 OUROBOROS

We have already seen the impact of loop-closures in SLAM results. The early detection of loop closure opportunities is commonly handled using topological maps (STACHNISS; HAHNEL; BURGARD, 2004) and checking for nearby unconnected nodes in the surroundings of the robot position. Our previous technique, Potential Rails (MAFFEI et al., 2014) (see Chapter 6) induces the robot to make turns using local potential distortions. Even though such technique uses an heuristic that can induce the robot to make turns, there is no explicit loop-closure control.

Figure 7.1: *Ouroboros* symbol (a) depicting a serpent eating its own tail. Influence of tractor point and shield in the potential field (b). In (b), we show: robot (red); obstacles (black); shield (orange); tractor point (pink); loop road (black stylized line); and the potential field (green arrows).



In this chapter we propose a new way to detect and consistently control the loop-closure during the integrated exploration process using potential fields. The name of our technique, *Ouroboros* (see Fig.7.1(a)), comes from its intrinsic meaning — "cyclicality". The technique alternates between going toward the unexplored regions (exploration) and closing loops. The former behavior sets attractor potentials in unexplored areas, while the latter is performed defining the pair "tractor point" (goal) and "shield" (wall blocking the direct influence of tractor point over the robot), see Fig. 7.1(b). We take advantage of the unknown space and the force lines generated by the potential field to make the robot produce a loop-closure behavior, knowing from the start where the loop must begin and end. The approach is new compared to other techniques because none of them, including Potential Rails, properly explore the unknown space to find loops.

Ouroboros Explained

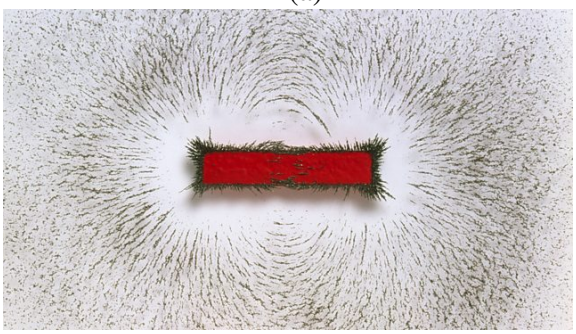
Magnets and Loop-Closures

Loops can be inferred by imagining how the unknown space, $\mathcal{U}$, can connect the different parts of the known space, $\mathcal{F} \cup \mathcal{O}$. This concept allied with the observation of the behavior of the force lines generated by a magnet can provide an interesting insight on how to close loops.

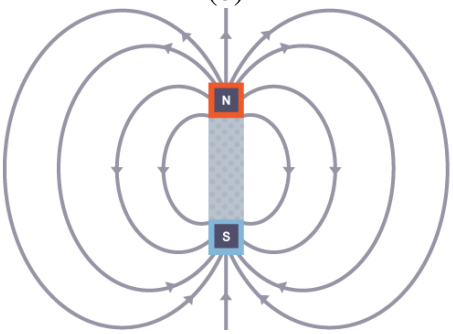
A magnet normally has two poles, each of them with a different polarity. Each pole has a magnetic field that exerts a force on other magnets and ferromagnetic materials. The force lines generated by a magnet are invisible to the naked eye, but we can detect them using iron filings (see Fig. 7.2 (a)[1]). Observe that such lines connect both poles. This happens because both poles attract each other (see Fig. 7.2 (b)). If we could produce the same behavior of a magnet during

Figure 7.2: Magnets with opposite poles generate force lines which are invisible to the naked eye, but are easily revealed by iron fillings in the surroundings of a magnet (a). These force lines eventually go from one pole to the other (b).

(a)



(b)



an exploration process, emulating the force lines of a magnet, we would be able to find out a loop through an unknown path that would lead to the robot back to a previously visited region, without using a previously known path.

[1]Images extracted from <http://www.bbc.co.uk/education/guides/zxxbkqt/revision/2>.

Ouroboros Core: Shields & Tractor Points

Ouroboros is a BVP-based integrated exploration approach with focus on loop-closures. Our conjecture is that BVP Exploration is flexible enough to emulate the behavior of a magnet to perform loop closures. We only need to modify its boundary conditions, redefining those regions having the highest and lowest potential values (see Fig. 7.4) to generate the virtual poles of our magnet. The "north" pole of our virtual magnet can be defined as the region we want the force lines to point to[2] — the destination of our robot, $\mathcal{R}$. This can be done imagining that the robot must go back to its current position to close a loop using a different path.

It can be done by setting an attractor region, i.e., tractor point $\mathcal{T}$, a few cells behind[3]. $\mathcal{R}$, as those having the lowest potential (the darkest region of the environment behind the robot in Fig. 7.4). In particular, it can be seen as a punctual region[4], $\mathcal{T}$, placed over the known free space so it can attract the robot through a potential field. That is, we define a new boundary condition

$$\boldsymbol{\nu}(p) = L \quad \text{on} \quad \mathcal{T}, \tag{7.1}$$

where, during the loop-closure behavior, L will be the lowest potential value in the whole map[5]. This will make this position a goal for the robot. If nothing else is done, the resulting potential field will force the robot to turn around and stop over $\mathcal{T}$. Therefore, a "south" pole must be ‘emulated in a way to force the potential field to generate the force lines over the unknown space as well, making the robot use a different path to try to reach the tractor point — i.e., closing a (if exists) to revisit the tractor point.

That is where the shield, $\mathcal{S}$, comes into play. It is a virtual wall used to block the direct influence of $\mathcal{T}$. Thus, it is always placed between the robot and the tractor point, but as close as possible to the robot. The shield is a region encompassing a straight line perpendicular to the robot heading (or to the Voronoi skeleton[6], $\mathcal{V}$, obtained by contracting the union of free and unknown spaces, $\mathcal{F} \cup \mathcal{U}$). The endpoints of a shield must always be connected to obstacles, otherwise the potential field would circumvent the shield, making the robot turn around using

[2] In a bar magnet the magnetic moment forms a vector field originating in the south pole, toward the north pole.

[3]The shield is always generated between the robot and the tractor point. That is why it is important to leave some space, typically about a dozen cells, between robot and tractor point.

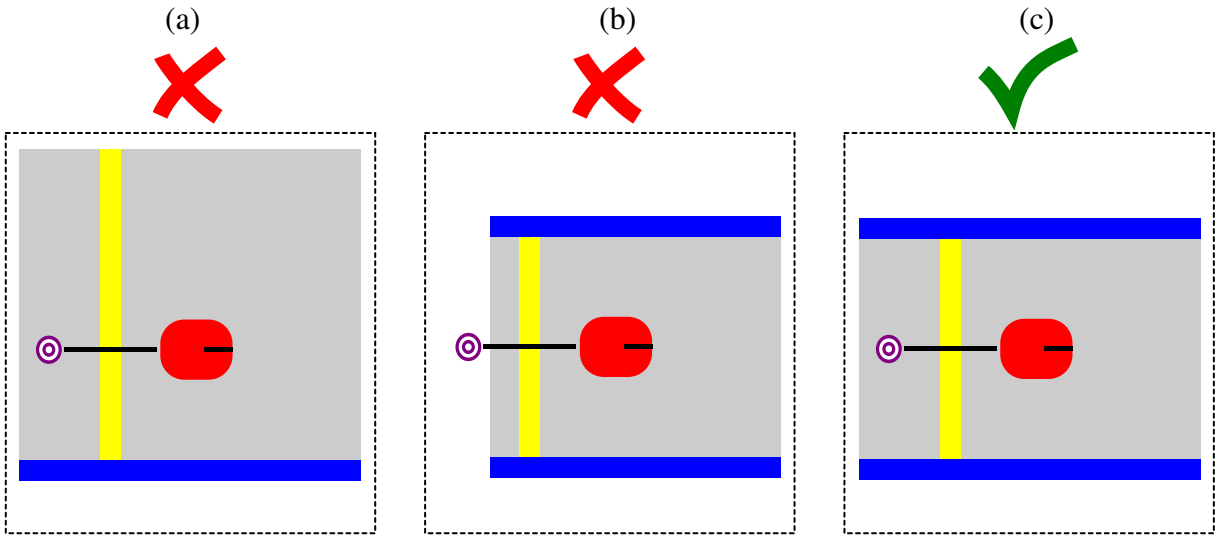
[4]Even though the tractor point can be seen as a point or single cell, we recommend the use of a slightly larger region, such as a small circular or square region, which considerably increases the sink attraction.

[5]The value can be set to a large negative number, and not necessarily 0 as in BVP, to facilitate the propagation of the potential over long corridors. Still, care must be taken during the implementation to avoid numeric problems.

[6]The robot direction sometimes changes abruptly, what sometimes makes the shield unnecessarily tilted in relation to the obstacles or longer than required. The Voronoi skeleton offers better information about where the connecting walls of the shield will be (always perpendicular to the Voronoi skeleton) and, consequently, about the shield orientation in the map.

the same path from where it came from, nullifying our strategy. Fig. 7.3 presents different scenarios for the generation of the shield and the tractor point. As we can see, the only scenario that respects the above mentioned conditions is scenario (c). Note that $\mathcal{S}(t)$ must necessarily

Figure 7.3: Example of invalid/valid scenarios for the generation of shield and tractor point. Robot (red), shield (yellow), tractor point (purple), obstacles (blue), known region (gray).



connect two opposing walls behind the robot, otherwise the robot WILL turn around instead of trying to go back to its current position using a different path — see the orange line behind the robot in Fig. 7.4. The potential field of the shield and its boundaries, $\partial\mathcal{S}$, will be the same of the obstacle space, $\mathcal{O}$ and its boundaries, $\partial\mathcal{O}$, during the loop-closure behavior, resulting in

$$\boldsymbol{\nu}(p) = H \quad \text{on} \quad \mathcal{S}(t) \cup \mathcal{O}(t), \tag{7.2}$$

where H will be the highest potential value in the whole map[7].

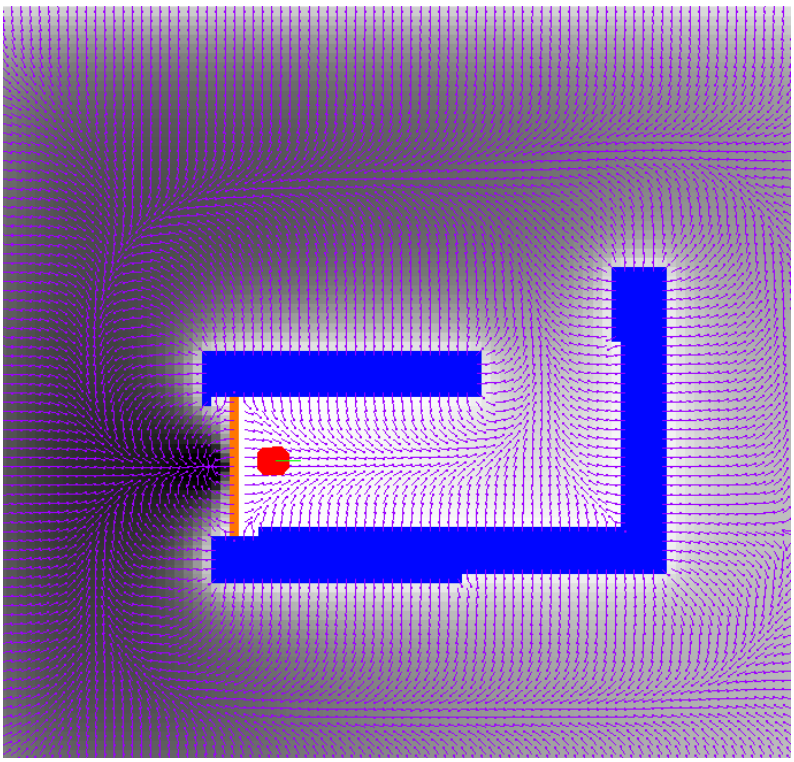
That is, the unknown regions cease to be the goal as in a traditional greedy exploration. Instead, during the loop-closure behavior, the potential field is obtained through the numeric solution of the Laplace Equation (or one of its variants) in the regions formed by the unknown and free space, i.e.,

$$\nabla^2\, \boldsymbol{\nu}(p) = 0, \text{ on } \mathcal{F}_{\mathcal{N}}(t) \cup \mathcal{U}(t), \tag{7.3}$$

where $\mathcal{F}_{\mathcal{N}}(t)$ is given by

$$\mathcal{F}_{\mathcal{N}}(t) = \mathcal{F}(t) \setminus \left(\mathcal{S}(t) \cup \mathcal{T}(t) \right). \tag{7.4}$$

Figure 7.4: Picture presenting the emulation of a loop-closure behavior for a robot (red) using BVP with modified boundary conditions in a given environment. The obstacles are shown in blue and the potential field is iterated in known and unknown regions — brighter cells indicate high potentials, while darker ones express low potentials. The gradient (purple arrows) shows that the robot indeed would move in the direction of the loop. The orange line behind the robot represents a virtual wall, or shield, that blocks the direct influence of the potential field of the tractor point, forcing a loop-closure using a different path.



When the robot returns to the tractor point, closing a loop in the process, both tractor point and shield are disabled, and all the cells near the path traversed by the robot during the loop closure are marked to avoid closing the exact same loop more than once. The next tractor point must be placed in an unmarked area, therefore, each closed loop is associated to just one tractor point. For this reason, even if it is possible to generate a tractor point and a shield, the loop-closure process will not be started unless it is the first time a loop is closed in that area.

Ouroboros Potential Field

Using a grid as the map representation for the BVP exploration is an easy solution to implement, but with a major drawback regarding the update cost of the potential field. When

[7]The potential of the walls does not necessarily need to be 1 as in BVP. It may higher to increase the strength of the attractor potential guiding the robot in narrow passages.

the nearby attractor cells vanish and the available ones are distant from the robot, the entire potential field may change. In this case, the convergence time of BVP computed over grids may drastically increase.

Our approach to circumvent this problem is to avoid the propagation of the potential over the full grid. Instead, the global potential is just propagated over the center cells of a Voronoi diagram. At this time, to compute the Voronoi diagram, we perform the thinning[8] (GUO; HALL, 1989) of known and unknown space. Knowing the center cells, we compute the potential field over them, by setting the goal as the lowest potential, and the robot as the highest. In practice, since we only have punctual boundary conditions, the goal and the robot, this is equivalent to compute a wavefront propagation algorithm (LAVALLE, 2006) over the center cells, starting at the goal.

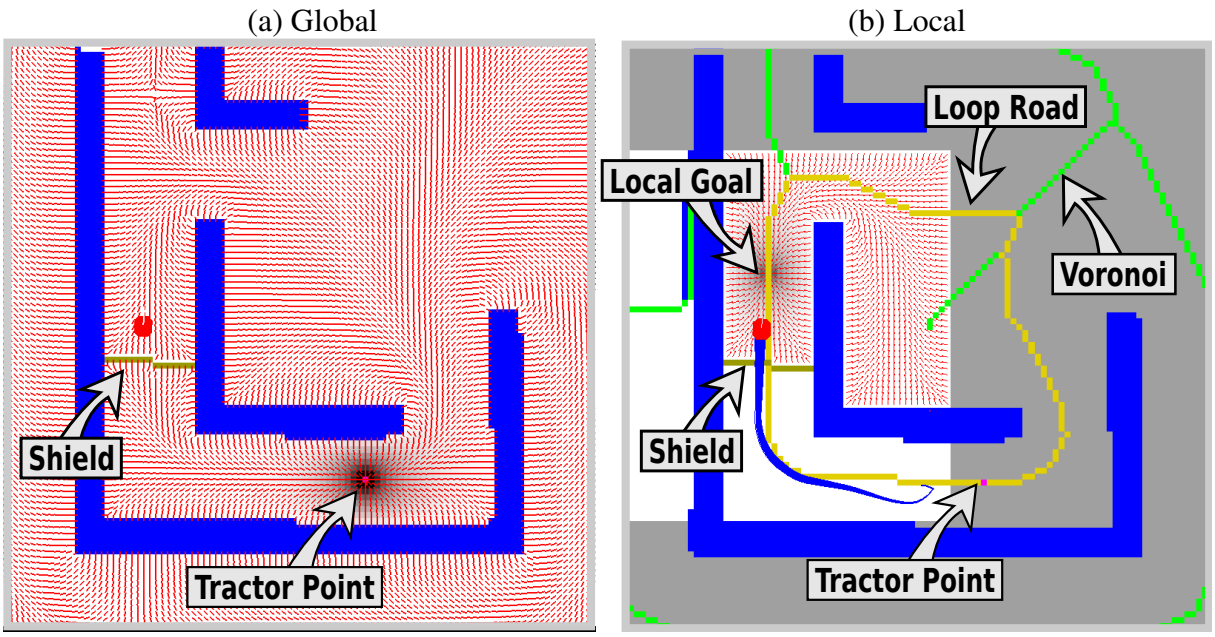
With the information of the global potential obtained from the center cells, we can compute the potential field inside a local window surrounding the robot via finite difference methods, as in (PRESTES et al., 2002). For this purpose, we set the potentials of the center cells as boundary values of the local potential field along with the obstacles located inside the window. This local potential field can be used by the robot to smoothly navigate toward the goal, while avoiding obstacles. Another advantage in using a local window, is that we can set a single local goal at the cell with the lowest potential at the robot's surroundings, rather than fixing the potentials of every center cell inside the window. This not only simplifies the computation of the potential, but the local potential field becomes smoother.

Since the size of the local window is stable, the cost to update the local potential is low. However, the Voronoi inside the local window may disappear if the robot gets too far from obstacles. In this case, our solution is to grow the local window until it becomes possible to find a valid local goal. This idea is similar to Prestes et al. (PRESTES et al., 2003). Still, in the worst case, the maximum size of the local window will be the maximum distance from the Voronoi to the walls.

Fig. 7.5 illustrates differences between the global and the local computation of the potential field. The global form is easy to implement, given that it just requires the definition of the tractor point and the shield. The implementation complexity of the latter form is higher due to the maintenance of additional structures, such as the Voronoi diagram, and the necessity of more rigor in the definitions of loops and local goals. However, computing the potential field using BVP in a large grid has severe implications in terms of cost and quality of the results, which is circumvented through the use of a local window.

[8] Any other strategy which generates a skeleton of the map guaranteeing connectivity could be employed.

Figure 7.5: Global vs Local computation of the potential field.



Another important aspect of Ouroboros is that the only region with a low potential is the tractor point, while in Potential Rails the entire Voronoi skeleton is a dynamic boundary condition. This difference results in the requirement of a pruning process for Potential Rails, in order to avoid the navigation through small branches connected to obstacles. This is not required for Ouroboros, since it will find the way to the goal, and these branches will be discarded because they do not lead to any loop or unexplored region.

Furthermore, as Ouroboros brings the unknown space into play, it requires the use of grid cells beyond the known regions. In order to avoid computation over the entire map, we simply add a small border of unknown cells (10 cells) to the boundaries of the map, so that the potential or the Voronoi skeleton can propagate through the unknown regions and close loops. Note that if you are using the full approach this border may have to be even larger, bu once you use a Voronoi skeleton, a one or two cells is enough, depending on how the algorithm is implemented.

Ouroboros Blueprint: the Algorithm

Ouroboros is presented in Algorithm 3. It is initialized without shield or tractor point (line 1). The idea is to initially move the robot to the frontiers of known space, since, at the beginning, it is not possible to set the goal or build a shield in the area behind the robot (not

visited yet).

Algorithm 3: *Ouroboros* Integrated Exploration

```
1  initialize(previousTractorPoint = ∅, previousShield = ∅)
2  while TRUE do
3      updateVoronoi()
4      robotCell = moveToVoronoi(robotPose)
5      if ∃ previousTractorPoint then
6          tractorPoint = moveToVoronoi(previousTractorPoint)
7          if dist(robotCell, tractorPoint) < Threshold then
8              disableTractorPoint()
9      else
10         tractorPoint = generateNewTractorPoint()
11     if ∃ tractorPoint then
12         propagatePotential(goal=tractorPoint, block=robot)
13         if ∃ previousShield then
14             loopRoad = buildLoopPassingByPreviousShield()
15         else
16             loopRoad = buildLoop()
17     if ∃ loopRoad then
18         shield = buildNewShield()
19     if ∃ shield then
20         propagatePotential(goal=tractorPoint, block=shield)
21     else
22         if ∃ unexploredVoronoiCells then
23             propagatePotential(goal=unexplored regions)
24         else
25             exit()
26     defineLocalGoal()
27     computeLocalPotential()
28     followPotentialField()
29     previousTractorPoint=tractorPoint;
30     previousShield=shield;
```

The first step in the main loop is to compute the Voronoi diagram with the current map of the environment (line 3). The Voronoi cells are computed through the thinning of the map (using both known and unknown cells, discounting obstacles). We start from a box slightly larger than the known area, to allow the existence of paths through unexplored regions. This continuous update is important because the Voronoi diagram changes as the known space increases. Additionally, since we use a RBPF for SLAM, the map can suffer drastic transformations in subsequent iterations, specially if the particles dispersion is large. That said, all the information placed over the Voronoi cells must be updated at each iteration, starting by the cell corresponding to the robot pose – moved to the nearest Voronoi cell (line 4).

The second step is the update of the tractor point. If there is a previous tractor point, we move it toward the Voronoi skeleton to guarantee that the tractor point is always over it (line 6). Otherwise, we try to generate a new tractor point behind the robot (line 10). As the tractor point must be a reachable cell, we disable it if it falls over an obstacle when the map changes. Lastly, if the robot reaches the tractor point we also disable it, since the loop was closed (line 8).

The third step is the construction of the loop road with the associated new shield. Given that the tractor point is valid, we try to build a road connecting it to the robot via two different paths – one connecting it to the robot's front and other connecting it to the robot's back. To do this, we compute the potential field over the Voronoi cells, considering the tractor point as the single goal, and the robot as a blocker (line 12). By blocking the potential propagation in the robot cell, we can easily detect the existence of a loop by checking if the potential originated in the tractor point reaches the robot from more than one direction. When this is not the case, the loop is invalid, and the robot can keep exploring the environment. Otherwise, the loop road is built by following the gradient descent of the potential field until reaching the tractor point, after departing from the Voronoi cells in the robot's 8-neighborhood.

It is worth noting that the loop road must be updated at each iteration, due to variations in the Voronoi diagram. There are two different conditions to construct the loop road: one is when the robot is already closing a loop (line 14) and other when it is not (line 16). The latter is rather simple, and corresponds to the strategy described above. The former, on the other hand, requires some extra precautions, because the resulting loop must connect the tractor point, the robot and the shield used, and stored, in the previous loop closing iteration. If these constraints are not respected, there is no guarantee that the robot will close the previously established loop. In extreme cases, the robot may end-up confined in adjacent loops. Thus, when such conditions cannot be satisfied we do not generate a loop road.

The fourth step is the update of the shield, which must be built over the loop road, orthogonally to the Voronoi. To keep the shield close to the robot, the method starts searching for a valid shield in the cells just behind the robot. The search continues until a valid shield is built or the tractor point is reached, meaning that we do not have an appropriate loop. A shield is valid when its two endpoints are connected to obstacles and when it crosses only one Voronoi branch. This is important to ensure that no path, other than the one assigned to the loop road will be blocked by the shield.

The fifth step is the propagation of the potentials, either from the tractor point considering the new shield, or from the unexplored cells that are over the Voronoi skeleton. If the method is successful in all three previous steps it propagates the potential from the tractor point

using the shield as blocker (just like the potentials propagation in the loop road construction). Or else, it propagates the potentials from the Voronoi cells in the border of the free space, i.e., $p \in \{\partial_{\mathcal{U}}\mathcal{F}(t) \cap \mathcal{V}(t)\}$. When there are no more reachable unexplored Voronoi cells, the method ends because the exploration of the environment is complete.

Finally, in the sixth step we define a local goal inside a local window surrounding the robot (line 26) and compute the traditional BVP inside such region (line 27), as explained in Section 7.1.3. The local goal is always placed in the Voronoi cell with the smallest potential, i.e. the one closest to the global goal (tractor point or unexplored cells), inside the region encompassed by a circle of 2m radius around the robot — the 2m value, or 20 cells, was chosen since it maintained motion smoothness in a variety of scenarios. Note that the farther the local goal is from the robot, the smoother the potential field computed with BVP. Therefore we must keep some distance between the local goal and the robot to avoid abrupt changes in the gradient at the robot position, what could derail the navigation process — during empirical tests, a distance of 2m, or 20 cells, was sufficient for that purpose. Lastly, the robot moves according to the gradient descent of the local potential field (line 28).

Comparing Ouroboros with a greedy approach

The evaluation of *Ouroboros* was made through the comparison with the traditional BVP exploration, in experiments using a Pioneer 3-DX mobile robot, equipped with a SICK LMS 200 laser range-finder.Experiments were performed on a desktop computer with an Intel® Core™ i7-4500 CPU with 16GB RAM running Ubuntu 14.04 64 bits. For each technique we considered four simulated environments, presented in Fig. 7.6. Environment A contains four adjacent loops of different lengths ($44m$, $46m$, $58m$ and $78m$), while environment B contains three adjacent loops of same length ($72m$). Environment C, of size $23m \times 25m$, has many rooms along a corridor without loops. Lastly, environment D is a sparse square scenario, of $20m \times 20m$, containing multiple small obstacles randomly distributed in the space, simulating an unstructured environment with trees. Each experiment configuration was performed 15 times. We also performed experiments in a real scenario, whose results are presented in this section. Both methods used a regular grid with cells of $10 \times 10cm^2$ as the map representation (STACHNISS, 2009)). In all experiments with *Ouroboros*, the potentials were computed over a local square window of 61×61 cells, centered in the robot, yielding $3m$ of range. In addition, the two methods were combined with the same RBPF SLAM strategy considering 300 particles in all experiments. Note that once again we navigate over the map having maximum weight (see Eq. 6.4).

Figure 7.6: Environments used in the experiments, with obstacles in black and free-space in gray.

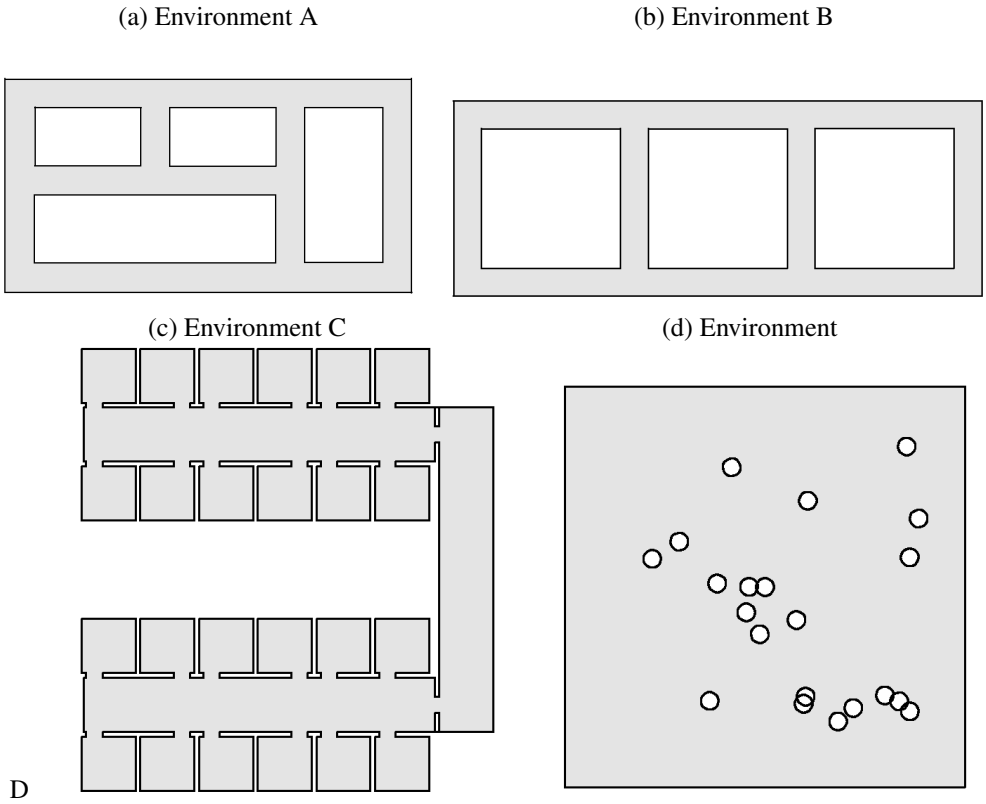
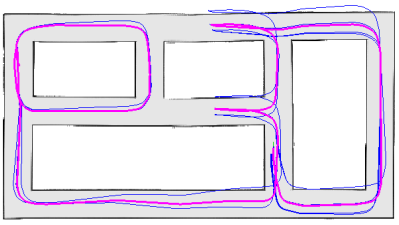


Table 7.1 show the results of the experiments according to three metrics: the error (Euclidean distance) between the particles positions and the real robot positions; the total time of exploration required for each method; and the final area size in comparison to the ground truth, from where we can estimate the percentage of the map left uncovered (or erroneously defined as covered). To help in the analysis of results, we present, in Fig. 7.7, one resulting map for each configuration. The chosen maps were the ones associated with the median error of each set of runs, in other words, for each configuration there were seven better maps and seven worse maps than the one shown here.

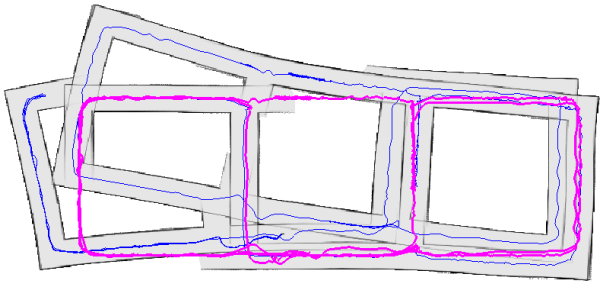
In a general analysis, *Ouroboros* presented the best results in Environments A and B – those with multiple loops. environments C and D were more problematic, yet the results were also good. In C, the robot tried to close loops in numerous occasions, but it was unsuccessful given the absence of loops. Nonetheless, it could sequentially visit each room of the scenario. In D, *Ouroboros* had difficulty to build consistent valid loops because it could not place the shield between such small obstacles. On the other hand, this avoided closing loops in cluttered areas (when the loop road is too close to obstacles).

Figure 7.7: Results obtained by each method in the simulated environments, with the real robot path (pink) and the estimated path (blue).

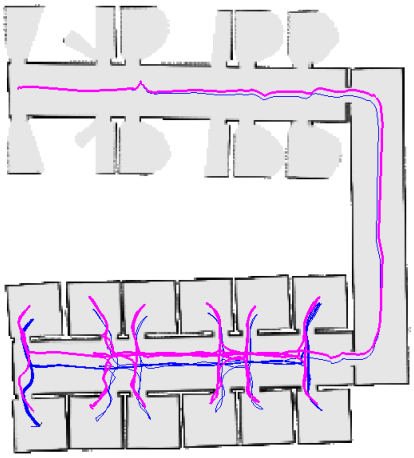
(a) BVP A



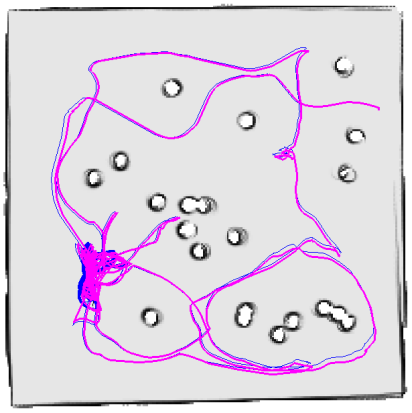
(b) BVP B



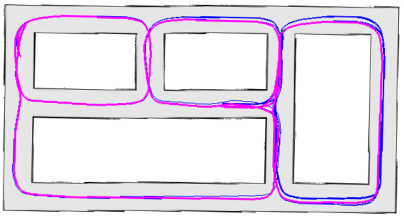
(c) BVP C



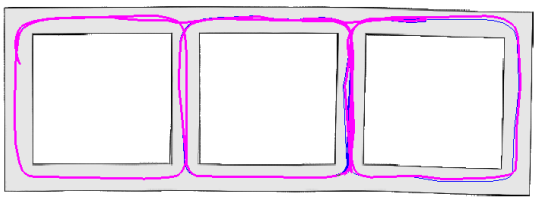
(d) BVP D



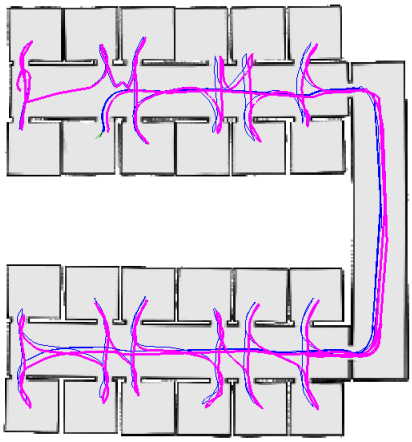
(e) *Ouroboros* A



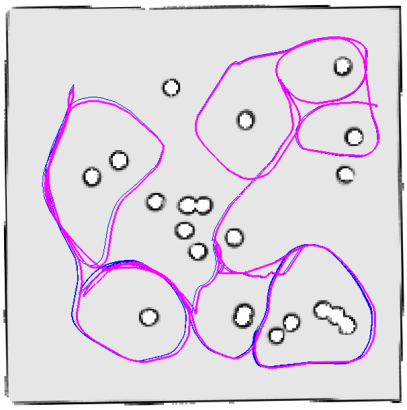
(f) *Ouroboros* B



(g) *Ouroboros* C



(h) *Ouroboros* D



Comparing the performances of *Ouroboros* and BVP, we noticed that BVP consistently left a non negligible portion of uncovered area in environment C, as we could see in Fig. 7.7(c). This is supported by the results of the final area sizes in relation to the ground-truth, as shown in Table 7.1, where BVP ended with almost 10% of uncovered area in such case. In Environment B, the coverage error of BVP is large due to the huge flaws in the map construction, resulting from its greedy and naive navigation strategy.

Table 7.1: Results of the simulated experiments.

(a) Environment A

	BVP		*Ouroboros*		diff
	mean	std	mean	std	%
Position Error (m)	1.01	0.61	0.40	0.25	-60.4
Total Time (s)	4881.9	957.7	4399.3	399.7	-9.9
Final Area Size (%)	105.2	1.31	99.7	0.25	—

(b) Environment B

	BVP		*Ouroboros*		diff
	mean	std	mean	std	%
Position Error (m)	2.54	2.34	0.49	0.25	-80.5
Total Time (s)	8585.4	2796.8	5231.7	835.6	-39.1
Final Area Size (%)	141.8	39.2	100.8	0.32	—

(c) Environment C

	BVP		*Ouroboros*		diff
	mean	std	mean	std	%
Position Error (m)	0.57	0.22	0.37	0.18	-35.1
Total Time (s)	3549.7	182.0	4985.8	431.1	+40.5
Final Area Size (%)	91.1	0.57	99.9	0.36	—

(d) Environment D

	BVP		*Ouroboros*		diff
	mean	std	mean	std	%
Position Error (m)	0.27	0.13	0.20	0.10	-25.4
Total Time (s)	5823.9	891.3	3691.3	400.4	-36.6
Final Area Size (%)	100.1	0.43	98.8	0.34	—

Regarding the total time of exploration, Table 7.1 shows that *Ouroboros* was generally faster than the BVP exploration. Although its focus at closing loops leads the robot to continuously revisit known regions, delaying the exploration, such revisits tend to avoid leaving small gaps on the map, which is one of the main problems of BVP. BVP roughly covers the whole area pretty fast, but leaves openings in the map that requires further inspection. This leads to

different problematic scenarios. At best, the robot must travel long distances to complete the exploration. However, it can turn into an unstable behavior, where the goal neither is strong enough to attract the robot, nor too weak to allow the convergence of the potential field (this happens in environment D). But it can be worse: if the attractive region is too small, too far from the robot, or beyond several narrow passages, the potential field may suddenly flatten, compromising the exploratory process (the robot stops its navigation since there is no gradient descent to follow). Indeed, this was the cause for the premature ending of the BVP exploration in environment C.

Fig. 7.8 presents plots depicting the variation of the time associated to the potential field update during the exploration process. We can see along the x axis that, with exception to environment C, *Ouroboros* always finished sooner than BVP. In relation to the potential field update time (y axis), *Ouroboros* was also faster than BVP. The reason is that computing the potential field over the full grid becomes too costly as the size of the known area grows. In the beginning of the exploration, BVP is always faster than *Ouroboros* due to the overhead in the computation, caused by the inclusion of additional borders cells belonging to the unknown space. This is crucial to allow the propagation of the potential field and the Voronoi over unexplored cells. However, this pattern changes after a few minutes of exploration. Note that, despite using a local window, the *Ouroboros* iteration time slightly varies due to the variations in the local window size and the global Voronoi computation.

Regarding the error between the estimated path and the real robot path, *Ouroboros* consistently obtained better results, as we can see in Table 7.1. Additionally, we performed the Wilcoxon rank-sum test (WILCOXON, 1945) over the results to confirm that, in the tested environments, *Ouroboros* presented significant improvements in the error quality when compared to BVP (p-values of 0.00002 in A, 0.00001 in B, 0.0006 in C and 0.04 in D). Fig. 7.9 shows a box-plot to compare the errors concentration during the process. The error is represented in the vertical axis, highlighting the median, the quartiles and the maximum and minimum errors, discarding outliers. We can observe once again that the largest difference between *Ouroboros* and BVP occurs in the multi-loops scenarios, specially in Environment B, where the upper quartile of *Ouroboros* is below the lower quartile of BVP.

Finally, Fig. 7.10 presents examples of visual results obtained with the real robot in a building floor of $72m \times 13m$ containing two adjacent loops. Ouroboros closed both loops sequentially, then it visited the front and back entries of the building to complete the exploration.

Figure 7.8: Total time of exploration in seconds (x axis) and time per iteration in seconds of the potential field update (y axis) during the 15 runs in each simulated scenario, for Ouroboros (green) and BVP (red).
[ht]

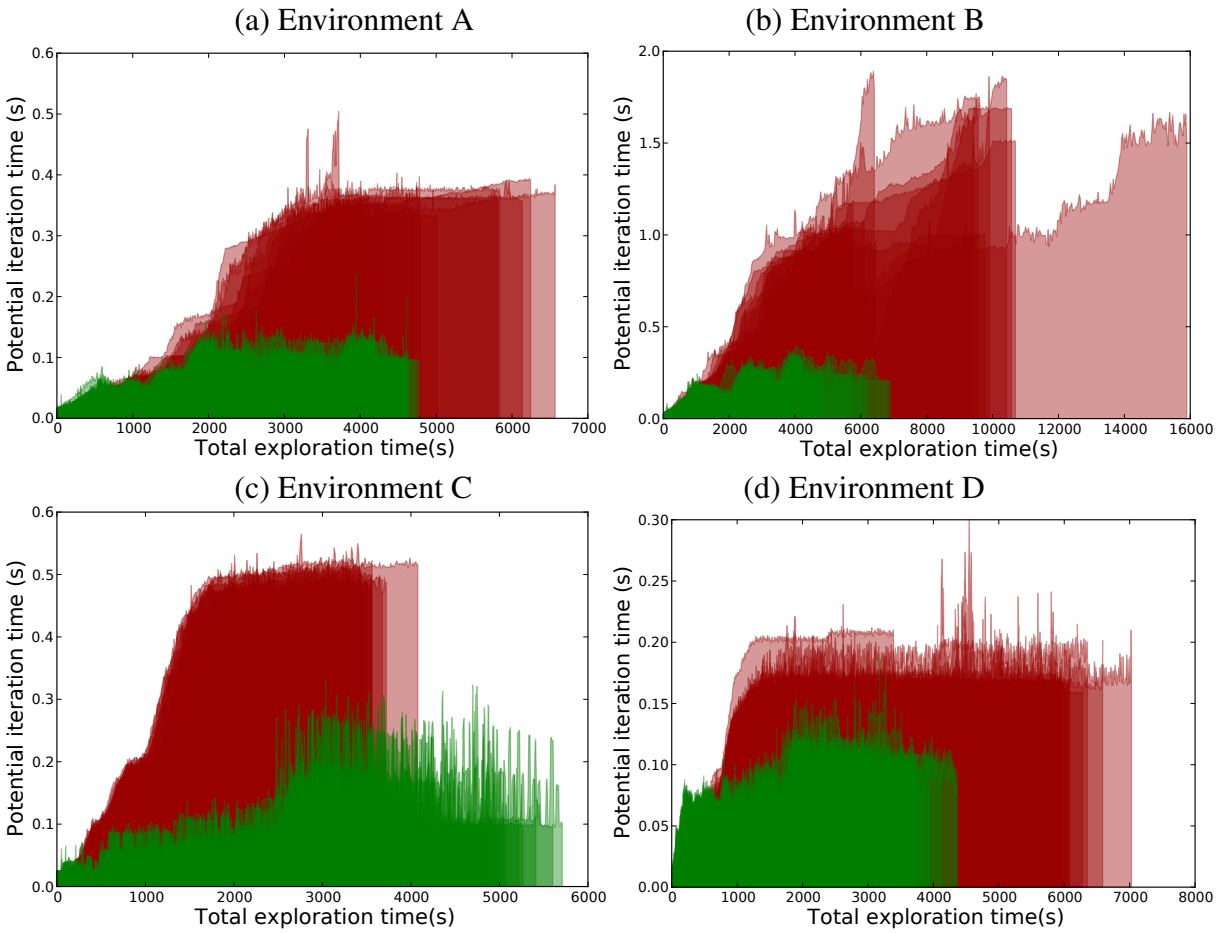


Figure 7.9: Box-plot of the errors for *Ouroboros* and BVP in all simulated environments.

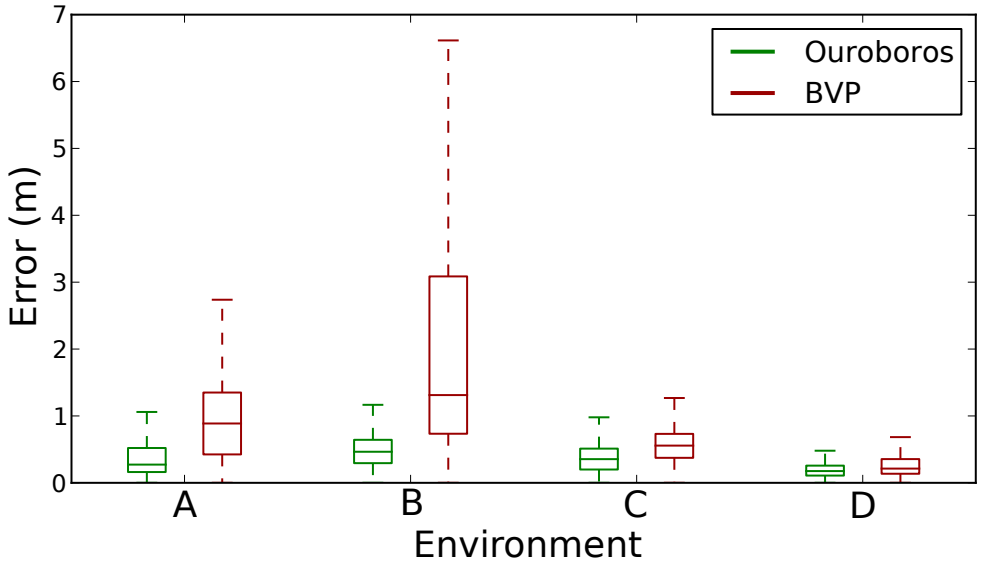
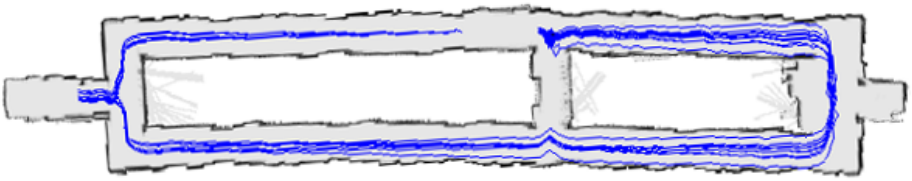
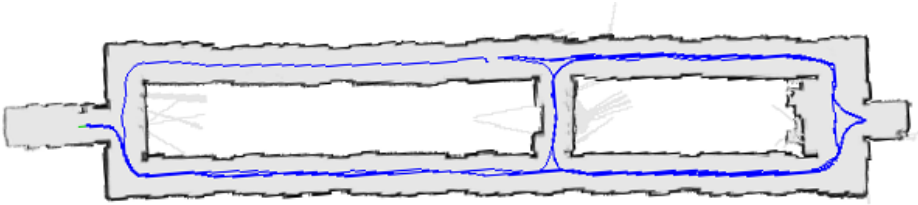


Figure 7.10: Examples of results obtained in a real environment, showing the particles paths in blue.

(a) BVP



(b) Ouroboros



Contrarily, BVP did not close the two small loops, and skipped some passages, what considerably increased the uncertainty of the SLAM process. Despite this fact, both methods obtained good results. Nevertheless, the purpose of such tests were to check if *Ouroboros* was suitable for real world applications, and the answer was positive.

A Brief Discussion about Ouroboros

So far in this chapter we have proposed a new integrated exploration strategy, in particular a new loop-closure behavior using potential fields. Our technique has proved to be effective in four simulated scenarios when compared to the frontier-based exploration. Such scenarios included realistic structured and unstructured environments. We also tested our approach in a real environment where the technique performed equally well.

Our key contributions are:

(i) A new loop-closure strategy using potential fields that naturally goes toward the smallest loop found (given a minimum size constraint), given the robot position and the knowledge it currently has about the environment;

(ii) A way to use the Voronoi skeleton to provide information about a local window surrounding the robot. This enables the computation of the BVP locally, leading the robot to the closest unexplored regions or in the direction of the nearest possible loop closure opportunity;

(iii) A strategy to automatically scale the local window if the Voronoi skeleton is not inside the local window;

There are some important considerations on using our strategy. We have successfully used fixed shields to explore different types of maps. Yet, if the fixed shield is placed at some region of the map that does not contain loops, the robot may be trapped into the loop-closing behavior for a while, until it detects the absence of loops. Finally, our approach does not require Voronoi edge pruning, because the robot will not walk over leaves that are known to lead it to obstacles (high potential).

Ouroboros manages to close loops by choosing the nearest path that may close a loop. When we compare it with Potential Rails, we note that *Ouroboros* generates a single local goal in the local window, while Potential Rails generates the potential in all cells of the Voronoi skeleton inside it. In addition, *Ouroboros* scales the local window when the Voronoi skeleton disappears, even though this could also be extended to Potential Rails as well. Finally, Potential Rails uses only the potential at the free space and potential distortions to close loops, while *Ouroboros* takes advantage of the propagation of the potential field over unknown regions to detect and close loops. The latter method is more robust than the former, since a local potential distortion may not be enough to force the turn, depending on the map characteristics. Still, when using *Ouroboros* and the exploration is finished, the potential ceases to exist, while in Potential Rails the potential field never ceases to exist and the robot continues the revisit in a behavior similar to patrolling.

Additionally, *Ouroboros* can be combined with utility metrics to avoid undesired conditions, such as staying too long in a loop. Undesired consequences in the former condition include failure to converge to the correct solution when the opportunity comes, while in the latter case, the problem is that the RBPF may spread particles for too long, risking the divergence from the correct solution. In particular, when *Ouroboros* closes two distant and unconnected loops, the local solution after the second loop closure may result in a locally correct but globally incorrect solution. This happens because the RBPF may have spread too much, making the local solution, after the second loop closure, filter out the global one. Thus, a form of measuring the uncertainty of the RBPF SLAM would be a great asset to the *Ouroboros* loop closure strategy. In the next section, we improve *Ouroboros* focusing on measuring the uncertainty and taking action to consistently keep a compact set of RBPF SLAM particles, surrounding the actual global solution, during the entire active SLAM process.

Improving *Ouroboros*: *Ouroboros* 2.0

In this section, we revisit key aspects introduced and discussed in Section 5.2.1 related to the technique proposed by Stachniss, Hahnel and Burgard (2004), providing further details about the method, pointing out possible improvements and identifying where such strategy fits with respect to the loop closure strategy adopted by *Ouroboros*. Both techniques have strengths regarding their underlying loop closure characteristics. By improving and combining them, we are able to circumvent limitations in both techniques. As a result of this merging process, *Ouroboros* 2.0 was born.

The most important characteristic of *Ouroboros* is its capacity to move through unknown regions to close loops. Still, as is, it is not concerned on how much time it should stay in a loop – e.g., considering a given uncertainty metric. It only tries to reach the position the robot was, considering the map of the best RBPF SLAM particle at the time the possible loop closure was first detected. This position and also those which compose the loop are not available across all the different particles and its corresponding maps. The loop closure path is defined online according to the map of the best SLAM particle. The outcome of such strategy is that the actual end of the loop may be, in fact, some meters ahead or before the previously defined loop closure position. One possible result is that closing a loop does not guarantee RBPF SLAM uncertainty reduction, another is that *Ouroboros* may keep the robot in the loop when it is no longer needed.

On the other hand, Stachniss, Hahnel and Burgard (2004) have developed a technique that does not actively search for loops. It implies the algorithm will often detect loops randomly and often try to close them when the robot is already close to the loop closure position, where the RBPF SLAM may already have detected a map matching and reduced the uncertainty about the SLAM posterior. Instead, it passively waits until a loop closure situation occurs, which may never happen depending on the exploration strategy and the map used. This is a key disadvantage of such method, since several loop closure opportunities may be lost. Nevertheless, it can close loops once the robot gets nearby a previously visited region. The algorithm constructs a graph of connected poses, $\mathcal{G}$, for every particle by adding a new node at pose $\mathbf{x_t}$ if the distance between $\mathbf{x}_t$ and all other nodes in $\mathcal{G}$ exceeds $2.5m$, or if there are no nodes *visible* from $\mathbf{x}_t$. Once a node is added to graph $\mathcal{G}$, an edge is created from the newly created node to the last visited one. In order to detect loops, the algorithm measures both the topological, d_t, and the metric distance, d_m, between current node and nearby nodes. The topological distance is given by the sum of the size of the edges, in meters, leading to the node. If $d_t > 20m$ and $d_m < 6m$, the algorithm creates an edge connecting them. In other words, if the distance in the grid is signifi-

cantly smaller than the distance in the graph, a loop is detected. Another interesting feature of the approach is the use of uncertainty about the pose to define when robots should stay or leave a loop-closure procedure. In particular, the robot stays in the loop until the current entropy, $\boldsymbol{H}_c$, about the pose gets smaller than the entropy, $\boldsymbol{H}_s$, measured when the loop-closure behavior started. Specifically, we stay in the loop as long as:

$$\boldsymbol{H}_c > \boldsymbol{H}_s \text{ and } \boldsymbol{H}_c > T_H, \tag{7.5}$$

where T_H is a threshold value.

Algorithm 4 displays the loop closure algorithm implemented from Stachniss, Hahnel and Burgard (2004), as we have implemented in our tests. Given the robot position, the algorithm searches for the closes nodes, through function *getNearestNode*. If none is available, or the distance from the nearest node is greater than 2.5m, for the current particle, a new node is added to the graph and then connected to the previously visited node. This function creates a new node and edge connection separately for the graph of every particle to keep the same graph topology for all particles, simplifying navigation of loops when the map, and consequently the exploration graph, change abruptly. At every iteration, through function *findLoopNodesNearby*, the algorithm searches for nearby nodes that are still not connected to the current node – again considering the current particle used. If the nearby nodes have a considerably large closest distance in the graph, $d_g > 20m$, while its distance in the grid map is small, $d_m < 6m$, the node is added to a list of available loop nodes. When the list of loop nodes is not empty, the algorithm considers them possible loop closure opportunities. The function called *getNearestLoopNode* uses the loop node that is closer to the current node. Then, the nodes forming the loop are extracted by the *getLoopPath* function. Finally, if the uncertainty criterion of line 11 is not met, the robot either follows a loop or explores the environment, through functions *followLoop*(path) and *explore*, respectively. We have decided to use the same exploration strategy used by Ouroboros, *explore*, while using traditional BVP to travel from node to node to keep both navigation strategies smooth and also to eliminate the creation of nodes when the nearest node cannot be seen.

It is important to highlight that *node visibility* is treated as a ray casting operation, which is ultimately dependent on the range finder used by the robot. As such, it is not appropriate to connect the loop closure node to the node corresponding to the current robot position while the closest path connecting them are still separated by unknown regions. The reason for this is the possible presence of still undetected obstacles blocking a direct edge connection. Thus, when

Algorithm 4: *Stachniss* et al. 2004 Loop closure algorithm

```
1  initialize(path = ∅, currentNode = ∅, Nodes = ∅, robotPosition = (0.0, 0.0))
2  while TRUE do
3      currentNode =getNearestNode(robotPosition)
4      if currentNode = ∅ or distance(currentNode, robotPosition)> 2.5m then
5          currentNode=robotPosition
6          addNodetoGraph(currentNode)
7      Nodes = findLoopNodesNearby(currentNode)
8      if Nodes≠ ∅ then
9          loopNode = getNearestLoopNode(Nodes, currentNode)
10         path =getLoopPath(LoopNode, currentNode)
11         while H_c > H_s and H_c > T_H do
12             followLoop(path)
13     else
14         explore()
```

a robot explores a given region, it has a field of view of at most[9] D, where D is the maximum distance which can be detected by the range finder. Of course, as the robot approaches such region again during a loop closure, its field of view will also be constrained by D. Therefore, the maximum distance at which a loop closure may happen through node visibility is $d_m < 2D$. Recall that the amount of superposition in the different maps available in the particles of the RBPF is what makes the SLAM posterior converge to the correct solution. This modification makes the algorithm less passive, since it detects the loop earlier than the original $d_m = 6m$. Besides, when a candidate loop-enabling node appears at instant t, it is likely that the exteroceptive measurement, $\mathbf{z}_t$, will make the entropy of the pose drop (an unintentional loop-closure), even before the start of the loop-closure behavior itself. This can happen because when a previously visited pose is found, formerly detected features of the environment may also be detected, reducing the uncertainty about the SLAM posterior[10]. As a consequence, keeping the robot in the loop might lead to particle depletion at the expense of little or no improvement in the local map. For the minimum loop size, we maintain 20m ((STACHNISS; HAHNEL; BURGARD, 2004)).

There is a strong connection between this technique and *Ouroboros*, since the latter

[9]We say "at most", because there may be occlusions or ignored regions the robot left behind while exploring the environment. For instance, when a robot, provided with a $180°$ field of view range finder, starts the exploration process in the middle of a corridor facing one of the two possible directions, consider the following two possible actions: 1) performing a $360°$ turn to explore its surroundings; 2) going straight ahead in the direction it is facing. In the former case, there is a radius around the initial robot position covered by the robot sensors. In the latter, only half a circle will be covered by the $180°$ range finder as the robot moves forward without exploring backwards.

[10]For instance, if we choose $d_m = 6m$ and we know that $D = 5m$, in the worst case scenario, when the loop is detected by the algorithm, we will already have an overlap of $4m$ containing information that could be used by the RBPF to reduce the SLAM posterior.

knows what path to follow in order to start the search and the technique from Stachniss, Hahnel and Burgard (2004) provides a mean to control the uncertainty considering all the SLAM particles. Therefore, one straightforward way to merge them is to switch the simple exploration behavior in algorithm 4 with a Ouroboros Active SLAM iteration, i.e., we invoke algorithm 3 instead of a standard greedy exploration behavior step – see algorithm 5 keeping its state during the entire Active SLAM. As such, once the *Ouroboros* algorithm reaches a previously visited

Algorithm 5: *Ouroboros* 2.0 algorithm

```
1  initialize(path = ∅, currentNode = ∅, Nodes = ∅, robotPosition = (0.0, 0.0))
2  while TRUE do
3      currentNode =getNearestNode(robotPosition)
4      if currentNode = ∅ or distance(currentNode, robotPosition)> 2.5m then
5          currentNode=robotPosition
6          addNodetoGraph(currentNode)
7      Nodes = findLoopNodesNearby(currentNode)
8      if Nodes≠ ∅ then
9          loopNode = getNearestLoopNode(Nodes, currentNode)
10         path =getLoopPath(LoopNode, currentNode)
11         while H_c > H_s and H_c > T_H do
12             followLoop(path)
13     else
14         OuroborosStep()
```

node, the algorithm keeps the robot in the loop until the uncertainty drops below $\boldsymbol{H}_s$, recorded when the loop was first detected, or the uncertainty is less than T_H. In addition, we perform selective resampling if the *Ouroboros* Step is active and traditional resampling when the robot reaches a region nearby a previously visited node. This is important narrow the solution while approaching a previously visited region and also keeping a conservative diversity of particles at the start of the loop closure procedure, when information is sparse or insufficient to narrow down the most accurate SLAM particles. In this way, we run less risk to have an uncertainty decrease before the actual loop closure with possible loss of the global solution.

In the next section we perform tests comparing the two versions of *Ouroboros* and the technique from Stachniss, Hahnel and Burgard (2004).

Comparing *Ouroboros* 2.0 with other Loop Closure Techniques

In this section we compare the proposed loop closure techniques against the loop closure strategy proposed by Stachniss, Hahnel and Burgard (2004). We use an experimentation set similar to Section 7.2 to compare the three techniques, with differences mainly relative to the computer used and the metric for map consistency.

Experiments were performed using a Pioneer 3-DX mobile robot, equipped with a SICK LMS 200 laser range-finder with a field of 180° configured to provide 181 measurement points separated by 1°. Simulations were performed using MobileSim[11] from Adept Mobile Robots™. Experiments were performed on a desktop computer with an Intel® Core™ i7-4790 CPU with 8GB RAM running Ubuntu 14.04 64 bits. Experiments were performed once again considering the four simulated environments[12], depicted in Fig. 7.6 and presented in Section 7.2. Each experiment configuration was performed 15 times. All methods used a regular grid with cells of $10\times10cm^2$ as the map representation (STACHNISS, 2009). In addition, all methods were combined with the same RBPF SLAM strategy considering 300 particles and using the same exploration strategy, but varying its loop closure capabilities.

However, the expected results are now distinct from the first experiment, since now we are comparing three loop closure techniques. Environment A and B are those that contain loops. However, the environment A has small corridors which facilitates the loop detection conditions from Stachniss, Hahnel and Burgard (2004) strategy. The same does not happen in environment B where loop closure positions are too distant to be detected by such strategy. This does not matter to *Ouroboros* and *Ouroboros* 2.0 which are naturally attracted to the loop closure position. Still, *Ouroboros* does not consider metrics to maintain the robot in the loop, it just reaches the end of the loop to restart the exploration behavior once again. *Ouroboros* 2.0 starts similarly, but when it detects a loop closure nearby, it will force the uncertainty to drop. Thus, we expect *Ouroboros* 2.0 to present better results in terms of localization and mapping errors, mainly in environment B. While similar or slightly improved results are expected for environment B. Note that environment C does not contain loops, while environment D contains sparse obstacles that make the creation of shields difficult. In these maps, if improvements are available, they may not be significant.

Results once again are presented considering the mean error (Euclidean distance) be-

[11] Available at <http://robots.mobilerobots.com/wiki/MobileSim>

[12] Environment A contains four adjacent loops of different lengths ($44m$, $46m$, $58m$ and $78m$), while Environment B contains three adjacent loops of same length ($72m$). Environment C, of size $23m\times25m$, is a scenario with rooms and long corridors, without loops. Environment D is a sparse square scenario, of $20m\times20m$, containing multiple small obstacles randomly distributed in the space, simulating an unstructured environment with trees.

tween the particles positions and the real robot positions and the total time of exploration required for each method. However, at this time we use a slightly different set of measurements with respect to the map matching metric. In particular, in Section 7.2 we adopted the amount of free space of the resulting maps to compare *Ouroboros* with BVP. At the time it was enough, since BVP, in the presence of successive narrow passages and long loops, runs into numerical problems and potential flattening, halting the exploration process before the map construction is complete, which is unlikely to happen in *Ouroboros*[13]. Still, at this time we are comparing techniques that do not leave the map incomplete. Therefore a new map matching metric is required. As a matter of fact, the best way to approach map matching is to compare the ground-truth map with those resulting from the exploration process of each technique. Here, we consider a metric $\mathbf{e_{map}}$, such that

$$\mathbf{e_{map}}(\mathbf{m_{gt}}, \mathbf{m}) = \frac{TP}{TP + FN + FP}, \tag{7.6}$$

where TP is the number of true positives, which consider the number of discrete map cells correctly assigned as free space or obstacle, while FP and FN, are respectively the number of cells which were incorrectly marked as part of the map (free or obstacle space) and those which were incorrectly marked as not belonging to the map (marked as unknown cells). We call the prevalence $\mathbf{e_{map}}$, the map matching ratio. The map matching ratio only considers correct maps those which are in the correct position and orientation in relation to the initial robot pose. Therefore, such metric penalizes both position and rotation errors in the SLAM process.

For the fifteen experiments using each of the loop closure strategies we present boxplots for each different environment with the mean position error (Fig. 7.11); the map matching ratio (Fig. 7.12); and the total exploration time (Fig. 7.14). In Figs. 7.11, 7.12, and 7.14 the red mark represents the median, the blue box delimits the first and third quartiles, while the black marks delimit minima and maxima for each technique and environment. We also present Table 7.2 with values for mean, standard deviation and median for each environment and technique to complement the available figures. Finally, we perform the signed Wilcoxon rank-sum test to analyze the significance of these results providing the sum of the ranks of the differences above or below zero (whichever is smaller), t, and the two-sided p-value for the test[14].

[13]The distance between the robot and the local goal of *Ouroboros* is small making such case unlikely to appear.
[14]<http://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.wilcoxon.html>

Results

We start analyzing the results considering the mean position error. From Fig. 7.11 displaying the boxplots for environments A-D, we observe that the median of the mean error in the fifteen experiments is always smaller with *Ouroboros* when compared to the loop closure strategy from Stachniss, Hahnel and Burgard (2004) (see Table 7.2), independently of the environment. The major problem with the strategy from Stachniss, Hahnel and Burgard (2004) is that the error increases a lot before a loop is closed, culminating in large errors and often in divergence from the correct solution. *Ouroboros* can decrease the position error by going back to previously visited regions right from the start of the exploration, what makes its position error decrease faster than the strategy from Stachniss, Hahnel and Burgard (2004). Still, it seems that keeping the robot in the loop forcing the uncertainty decrease, as the strategy from Stachniss, Hahnel and Burgard (2004) does, is also important. As *Ouroboros* does not ensure the uncertainty reduction, the position error is not guaranteed to decrease. In environment A, where the loops are small and the technique from Stachniss, Hahnel and Burgard (2004) often detects and close the loops, the difference between these two techniques is not significantly different in terms of localization analyzing the Wilcoxon rank-sum results, $(t = 1.5139, p = 0.1300 > 0.05)$. In environment B, as expected, the strategy from Stachniss, Hahnel and Burgard (2004) is not sufficient to effectively detect and close large loops, and the position error becomes statistically significant $(t = 2.0117, p = 0.044 < 0.05)$. This is expected, since closure through proximity leads to dependence on range finder range and the loop size, since two thresholds are used to detect loops: the distance to the loop closure position; and the distance in the graph. Therefore, the method fails to close large loops depending on the values of such parameters. In environment A, these parameters seem to have been sufficient to close small loops, while in environment B, the technique from Stachniss, Hahnel and Burgard (2004) fails to intentionally detect and close loops. Instead, it keeps searching for unexplored regions, which only makes the position error increase. Scenarios C and D are either small or with the absence of actual loops, what leads to no significant differences in the mean position errors, with $(t = -0.1867, p = 0.8519 > 0.05)$ and $(t = 0.3526, p = 0.7244 > 0.05)$ for environments C and D respectively.

Analyzing the mean error in *Ouroboros* 2.0 in Fig. 7.11, the first and third quartiles in the box plots indicate that *Ouroboros* 2.0 presents considerably better localization results when compared to *Ouroboros* or the strategy from Stachniss, Hahnel and Burgard (2004). In particular, we observe statistically significant localization improvements for all environments when we

Table 7.2: Results of the simulated experiments.

(a) Environment A

	Stachniss et al. 2004			*Ouroboros*			*Ouroboros 2.0*		
	mean	std	median	mean	std	median	mean	std	median
Position Error (m)	0.513	0.226	0.413	0.394	0.152	0.377	0.335	0.133	0.289
$\mathbf{e_{map}}$	0.790	0.146	0.860	0.803	0.055	0.789	0.827	0.055	0.842
Total Time (s)	3551	356	3592	4426	290	4424	5269	469	5472

(b) Environment B

	Stachniss et al. 2004			*Ouroboros*			*Ouroboros 2.0*		
	mean	std	median	mean	std	median	mean	std	median
Position Error (m)	0.676	0.281	0.533	0.491	0.173	0.415	0.350	0.162	0.311
$\mathbf{e_{map}}$	0.722	0.118	0.766	0.787	0.058	0.774	0.850	0.048	0.867
Total Time (s)	4051	417	3982	4477	525	4262	5262	613	5682

(c) Environment C

	Stachniss et al. 2004			*Ouroboros*			*Ouroboros 2.0*		
	mean	std	median	mean	std	median	mean	std	median
Position Error (m)	0.384	0.156	0.410	0.392	0.156	0.337	0.254	0.134	0.206
$\mathbf{e_{map}}$	0.636	0.047	0.640	0.641	0.037	0.644	0.670	0.036	0.682
Total Time (s)	4864	386	4737	4720	448	4653	4881	434	4891

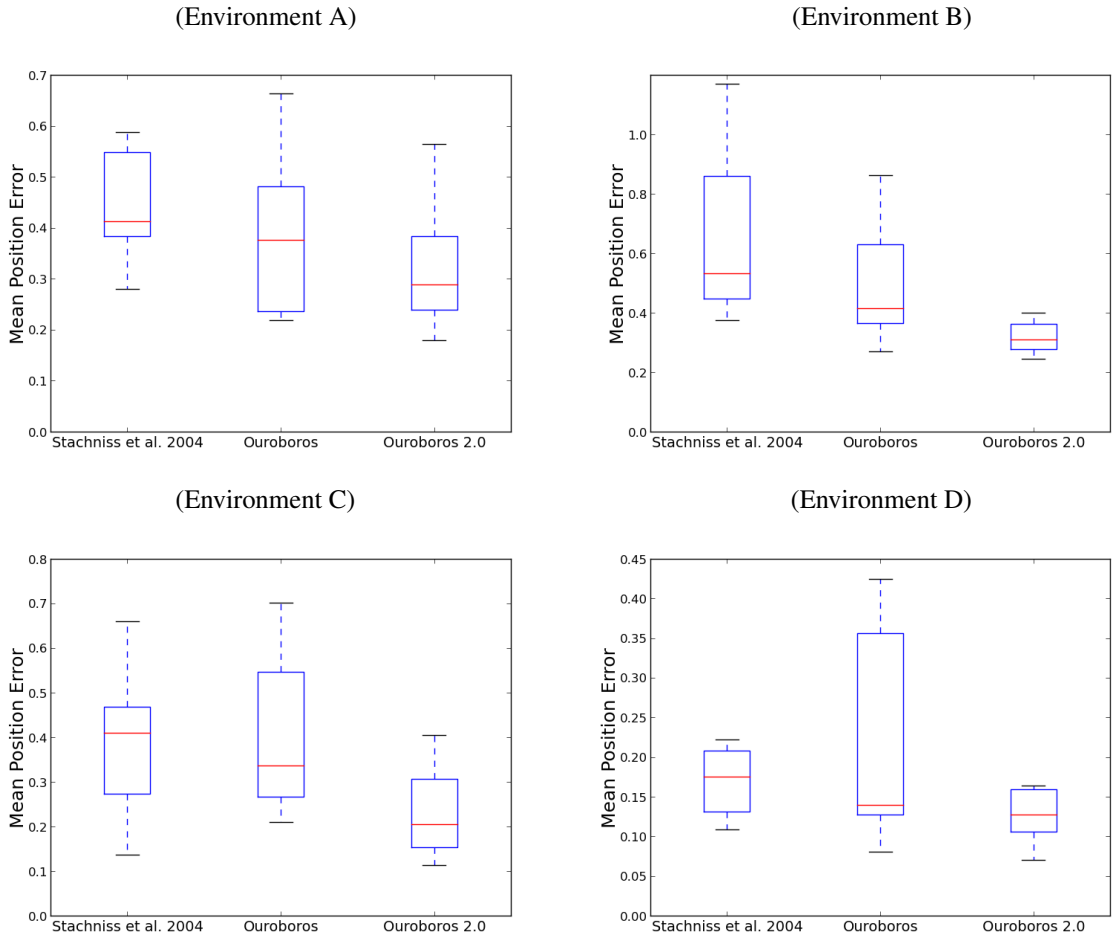
(d) Environment D

	Stachniss et al. 2004			*Ouroboros*			*Ouroboros 2.0*		
	mean	std	median	mean	std	median	mean	std	median
Position Error (m)	0.199	0.092	0.175	0.212	0.121	0.140	0.140	0.067	0.127
$\mathbf{e_{map}}$	0.930	0.028	0.938	0.926	0.031	0.935	0.939	0.021	0.948
Total Time (s)	2418	285	2476	3652	345	3638	3889	440	3832

compare the technique from Stachniss, Hahnel and Burgard (2004) with *Ouroboros* 2.0, where the Wilcoxon signed rank-sum test presents $(t = 2.5509, p = 0.0107 < 0.05)$, $(t = 4.0856, p = 4.3965e - 05 < 0.05)$, $(t = 2.2191, p = 0.0265 < 0.05)$, and $(t = 2.4265, p = 0.0152 < 0.05)$ for environments A-D respectively. Even though environment A presents statistically significant results favoring *Ouroboros* 2.0, again the most significant result occurs in environment B — the map with three large loops. The small size and absence of loops reduces the significance of the results for environments C-D. *Ouroboros* 2.0 represents the merging of the previous two techniques analyzed, having an imediate loop search and also a guaranteed uncertainty reduction at every loop, explaining the decrease in position error presented by *Ouroboros* 2.0.

Table 7.2 also presents the means, standard deviations and medians of the c ratio for each method and environment. Still, at this time, it is not easy to detect significant differences among methods in terms of mapping quality by only looking at the medians or the means. Similarly, the boxplots for the map matching ratio, shown in Fig. 7.12, present results that requires further analysis before coming to a conclusion — in contrast with the boxplots for localization errors. Comparing the strategy from Stachniss, Hahnel and Burgard (2004) with Ouroboros,

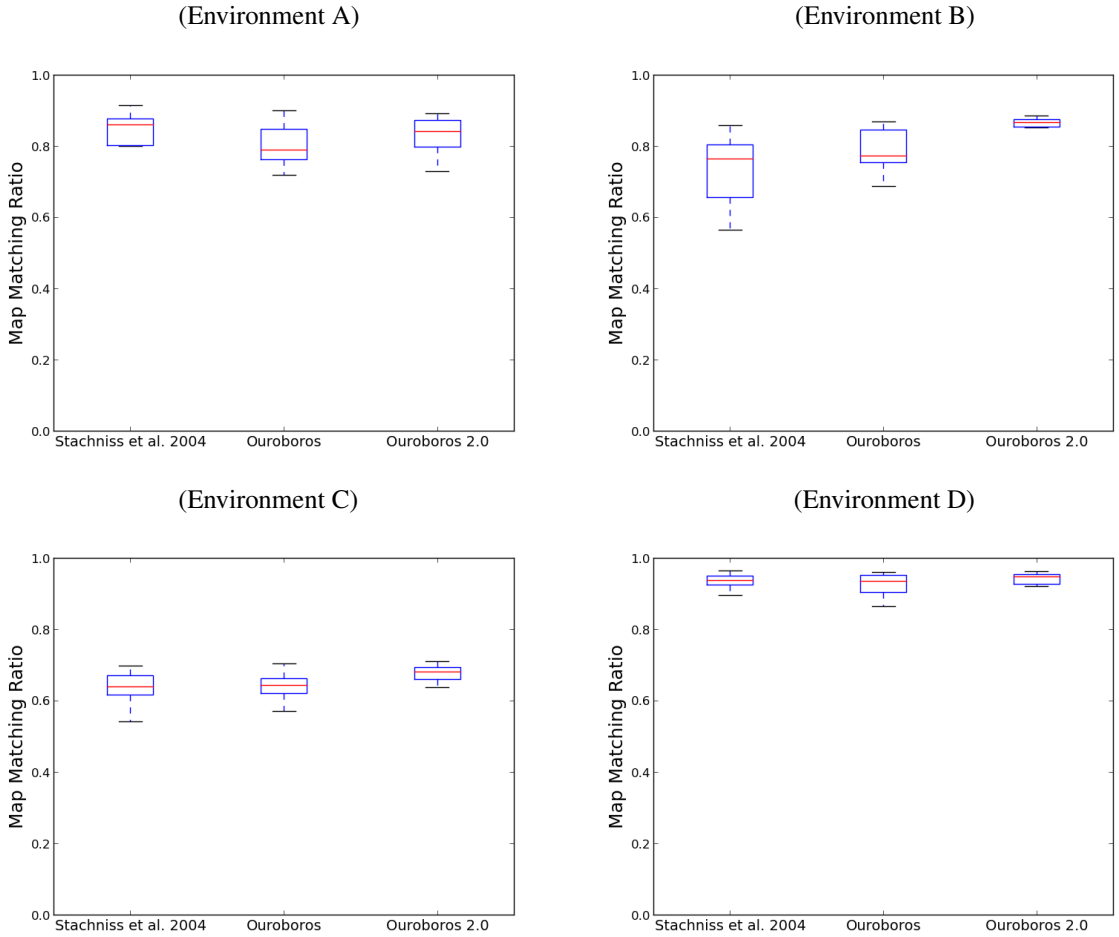
Figure 7.11: Mean position error considering environments A-D using the following techniques: (STACHNISS; HAHNEL; BURGARD, 2004), *Ouroboros*, and *Ouroboros* 2.0.



no statistically significant difference in terms of map matching ratio is found in environment A, $(t = 1.3066, p = 0.1914 > 0.05)$, even though it seems that mapping results may favor loop closure through node visibility over *Ouroboros*. Still, in environment B, the opposite happens, $(t = -1.2651, p = 0.2058 > 0.05)$, since *Ouroboros* is actually better at finding the loop closure than the strategy from Stachniss, Hahnel and Burgard (2004). Still, as *Ouroboros* does not guarantee uncertainty reduction, it was unable to reduce mp matching errors in a statistically significant way. There are no significant differences in environments C and D, with $(t = -0.1866, p = 0.8519 > 0.05)$ and $(t = -0.0622, p = 0.9504 > 0.05)$ respectively — there are no actual loops or th evironment is small. A similar trend happens when we compare *Ouroboros* 2.0 with the strategy from Stachniss, Hahnel and Burgard (2004). However at this time, results show that the map matching ratio of the two ditributions are now closer in environment A $(t = 0.1867, p = 0.8519 > 0.05)$. While a statistically significant difference favors *Ouroboros* 2.0 in environment B, $(t = -3.8367, p = 0.0001 < 0.05)$, as expected. Still,

in environment C, results also favored *Ouroboros* 2.0 $(t = -2.1776, p = 0.0294 < 0.05)$. While this was not expected, we expeculate that the loop closure stage provides a more stable motion than a pure exploratory behavior. This, combined with the uncertainty reduction, played a role in such differences. In environment D, no statistically significant difference was reported, $(t = -0.9333, p = 0.3507 > 0.05)$, where uncertainty does not drop considerably before the whole map is explored, leading to little or no need to close loops. The map matching ratio is particularly superior with the two versions of *Ouroboros* for environments containing large loops. On the other hand, the result was not statistically different for small loops or environments with no loops at all. This, together with the presented position error results, show that both versions of *Ouroboros* tend to be best suited for closing loops in unkown environments.

Figure 7.12: Map matching ratio considering environments A-D using the following techniques: (STACHNISS; HAHNEL; BURGARD, 2004), *Ouroboros*, and *Ouroboros* 2.0.



Even though the statistical test could not show the prevalence of *Ouroboros* versions over the competing strategy in terms of map matching ratio, only when we look at the localization errors in environment A, the standard deviations for the mapping and localization errors in environments A and B (see Table 7.2) show larger values with the strategy from Stachniss, Hahnel and Burgard (2004), revealing a pattern in the distributions of map match. A distribution with larger standard deviation probably has bad results canceling the effect of good map matches. For this reason, here we add one last metric to complete the analysis of mapping results: mapping failure, which we consider all cases where the solution diverges from the correct map topology. The RBPF, when a large loop is successfully closed, even with high uncertainty, it leads to good mapping results. However, the risk of map failure increases considerably when closing large loops and the uncertainty about the SLAM posterior is high. This is what happens when the strategy from Stachniss, Hahnel and Burgard (2004) is used to close loops, since it does not actively search for them. It failed trice in environments A and B, while *Ouroboros* failed twice in environments B and D, while no mapping failure was reported with *Ouroboros* 2.0. Fig. 7.13 presents the mapping failures of all techniques. There is one caveat with constant active loop closure. When *Ouroboros* closes subsequent loops, and the loops are somewhat distant from each other, there is a chance that the correct solution from the first and the subsequent loop closures are locally correct, but globally incorrect. Thus, when the uncertainty about the SLAM posterior increases considerably, a form of returning to a previously visited region may be useful to force to narrow down the solution around the global solution once again. Even though *Ouroboros* can close subsequent loops, it has no mechanism to detect, or take action, when the uncertainty about the SLAM posterior increases. This led the technique to fail in environment D, where the map is small, but the amount of relevant data, obstacles, are scarce and sparsely divided. On the other hand, the same does not happens with the other two techniques, where the loop closure through node visibility provides a mechanism to measure the RBPF SLAM uncertainty and also going back to previously known regions when the uncertainty about the SLAM posterior increases considerably. This makes the techniques from Stachniss, Hahnel and Burgard (2004) and *Ouroboros* 2.0 better suited for sparse environments when compared to the first version of *Ouroboros* and explains the poor results of *Ouroboros* in terms of localization when we look at Fig.7.12 at the boxplots of environment D. Therefore, we can conclude that active loop closure plays a key role in the successful construction of maps containing large loops, such as environments A and B, while the control of uncertainty is also important in sparse environments. Lastly, regarding mapping results, a combination of both methods has shown robuster results than both techniques separately.

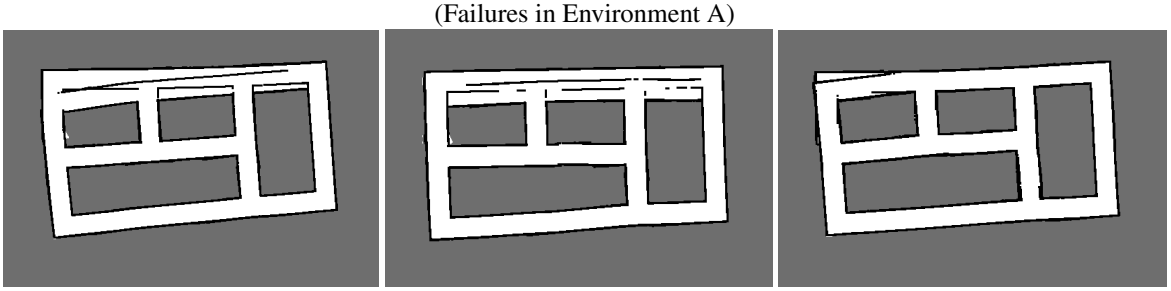
Another important aspect of the whole exploration procedure is the exploration time. Taking less time to explore the environment imply a more efficient method. However, as we have seen, it does not mean the method is also effective for the reduction of localization and mapping errors. This is because a small exploration time is most likely the result of less loop closures and less uncertainty reductions, affecting the effectiveness of the exploration strategy. We proceed with the analysis of the total exploration time for each technique. The statistical significance of the differences in exploration time among the technique proposed by Stachniss, Hahnel and Burgard (2004) and both versions of *Ouroboros* are obtained using, once again, the Wilcoxon signed rank-sum tests. The comparison with *Ouroboros* yields for environments A-D respectively ($t = -4.4174, p = 9.9889e-06$), ($t = -2.5509$,$p = 0.0107$), ($t = 1.0577, p = 0.2902$), and ($t = -4.6663$,$p = 3.067e-06$)., while for *Ouroboros* the same comparison yelds ($t = -4.6663$,$p = 3.067e-06$), ($t = -3.9197$,$p = 8.8668e-05$), ($t = 0.1244$,$p = 0.9010$), and ($t = -4.6248, p = 3.7495e-06$) respectively for environments A-D. We observe that there is significantly less time spend on the exploration with the technique from Stachniss, Hahnel and Burgard (2004) than with both versions of *Ouroboros* for environments A, B, and D, where there are loops present. The same does not happen in environment C, where both versions of *Ouroboros* have spent more time than the competing technique, even though both differences are not statistically relevant, which is expected due to the absence of loops. Nevertheless, in environments A and B, we show that just passively detecting nearby loops results in less loop closures, resulting in a smaller total exploration time, negatively affecting both localization and mapping results. Environment D shows that, for small environments, the mapping result seems to be independent of the amount of tiny loops closed. In other words, when fast mapping is required for a small environment, circling around small obstacles in the environment may be of little use to improve the quality of mapping, even though it will most likely affect the robot localization during the exploration procedure, as we have seen before. *Ouroboros* 2.0 presented the most robust SLAM results, working well in a variety of scenarios, while the other two techniques presented poor results in specific scenarios. Finally, we observe from the results that

active SLAM results can indeed be improved by superior loop closure strategies. Exploration time raised questions on what the minimum acceptable loop size would be, since detecting such value could be important to reduce the exploration, saving energy for other tasks. Unfortunately, the value of the minimum acceptable loop size is dependent on the robot motion error, while the relevance of such information is also environment dependent.

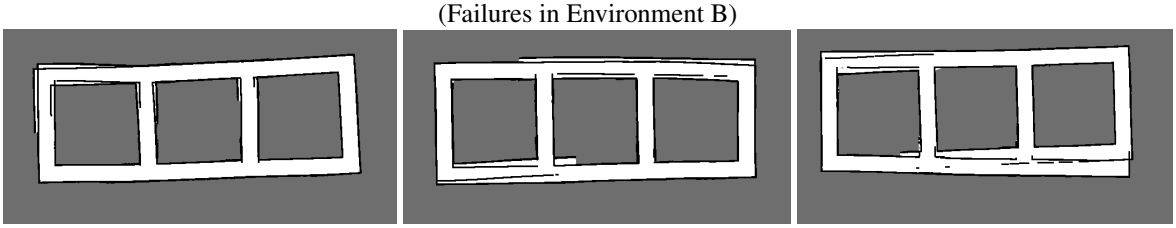
Figure 7.13: Mapping failures for each technique and method in environments A-D. Note that there is no mapping failures for *Ouroboros* 2.0 in any of the experiments.

Stachniss, Hahnel and Burgard (2004) Failures

(Failures in Environment A)

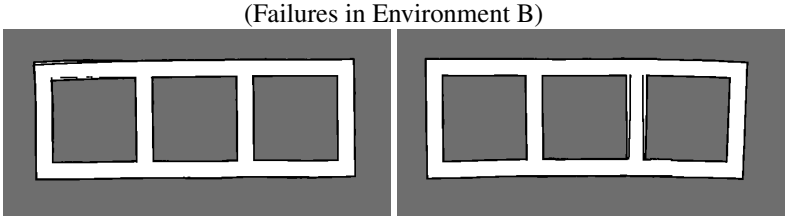


(Failures in Environment B)



***Ouroboros* Failures**

(Failures in Environment B)



(Failures Environment D)

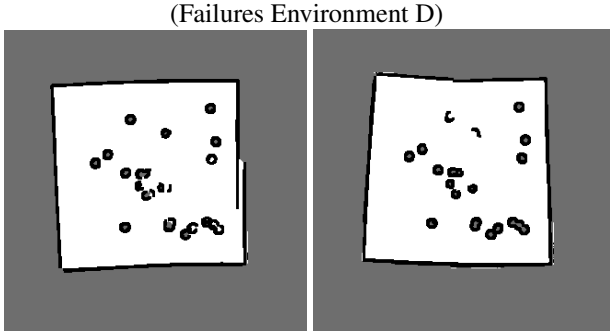
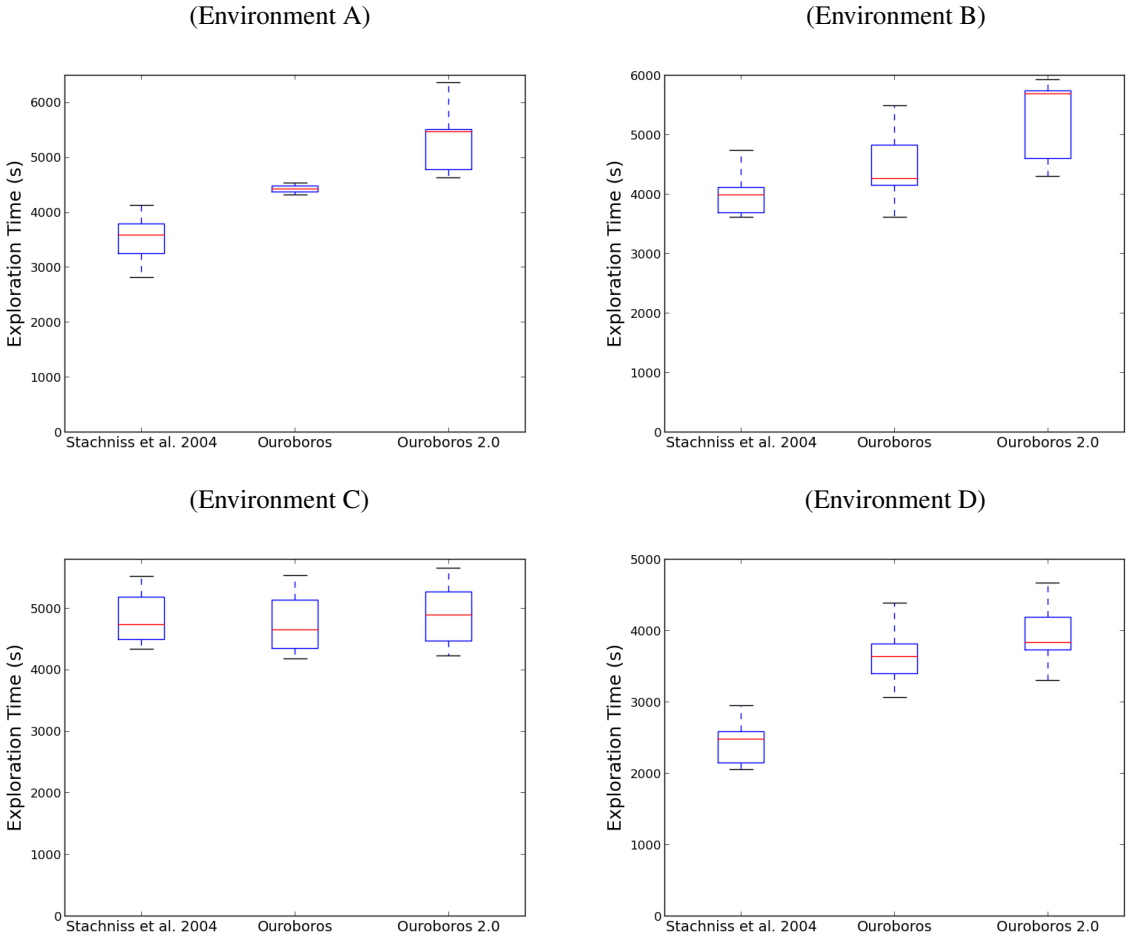


Figure 7.14: Total exploration time considering environments A-D using the following techniques: (STACHNISS; HAHNEL; BURGARD, 2004), *Ouroboros*, and *Ouroboros* 2.0.



8 CONCLUSION

In this document we have shown that BVP Exploration can be further modified to improve the results of Active SLAM approaches. A BVP naturally emerges (ASCHER; MATTHEIJ; RUSSELL, 1994) in numerous problems beyond those associated with exploration. We argue that the flexibility and strong mathematical background of the BVP Exploration can be a powerful tool to devise new and complex behaviors for autonomous robots motion.

In this Thesis, we have introduced a more general formalization for the exploration problem using the BVP which enables the use of new boundary conditions for different purposes, specially in the context of integrated exploration. The current implementations of BVP Exploration are limited to greedy behaviors (PRESTES et al., 2002; PRESTES et al., 2003). Trevisan et al. (2006) suggested that dynamic boundary conditions could be used to modify the exploratory behavior of the algorithm. However, their approach is still greedy and is only dynamic in the sense of position changing, while the functions defining the values of the potential at the boundaries remain constant and are restricted to the surroundings of the boundary between known and unknown regions.

Similarly, local potential distortions (TREVISAN et al., 2006; PRESTES; IDIART, 2009; PRESTES; ENGEL, 2011) were also devised to change the behavior of the BVP Exploration and path-planner and, for that purpose, the partial differential equation used by the method was modified. However, their approaches do not focus on integrated exploration. As a matter of fact, the resulting work culminated in the development of an equation which included a potential distortion parameter, but its application was reduced to greedy approaches that still does not allow intentional revisits due to the gradient descent process. Another aspect of all current BVP approaches is that the potential field flattens when the algorithm finishes the exploration, since there is no longer a goal. As a result the robot stops and the exploration ends. These limitations are a major problem for integrated exploration approaches, since they eliminate the possibility of revisits. In particular, loop-closures are quintessential for SLAM approaches (STACHNISS; HAHNEL; BURGARD, 2004; STACHNISS; GRISETTI; BURGARD, 2005b). Our proposal generates loop-closure behaviors through the effective application of local potential distortions and boundary conditions. We also propose a way to perform time-based revisits using dynamic boundary conditions, which can be seen as an uninterrupted patrolling behavior.

We have devised two Active SLAM techniques, *Ouroboros* and Potential Rails (see Chapters 6 and 7), which are presented as evidences that it is possible to close loops using

local potential distortions and new boundary conditions. We have also demonstrated that these techniques can perform revisits continuously using dynamic boundary conditions whose values are defined by a time-based function. We present qualitative and quantitative results showing that the proposed approaches have systematically outperformed the traditional BVP algorithm in simulated and real environments. Regarding *Ouroboros*, we have also managed to include a more robust loop closure strategy using the information about the uncertainty of the SLAM posterior, similarly to the technique from Stachniss, Hahnel and Burgard (2004). This new approach, named *Ouroboros* 2.0, uses information from all particles to detect loop closures and also control the time the robot must stay in the loop, while at the same time keeping the active loop closure properties of the original *Ouroboros* algorithm. This avoids particle depletion and also enables the complementary detection of loops using a graph structure which is created for all SLAM particles. The result is robuster than *Ouroboros* or Potential Rails.

In this thesis, we also managed to devise a way to take advantage of the global information contained in a Voronoi skeleton to generate the potential field inside a local window, which reduced the computational cost of the algorithm. The approach evolved from a fixed window size in Potential Rails to a variable size window in Ouroboros, since the Voronoi skeleton may suddenly disappear during the exploration. We show with quantitative results that this approach is more efficient than the traditional BVP algorithm. Even in the worst case scenario, our approach only needs a local window which has at most half of the dimensions of the rectangle encompassing the map, while BVP requires the entire map. In particular, we have shown that a robust loop closure strategy can indeed improve SLAM results considerably. Such improvements may come at the cost of additional exploration time.

There are still interesting new aspects of the exploration that could be addressed. The algorithm could include uncertainty information to generate other types of boundary conditions. Potential distortions naturally keep the robot on the left, right, or the middle of a corridor. This could be used in multi-robot exploration to cut down abrupt changes of direction when one robot has to dodge the other. We just need to keep the robots apart using dynamic boundary conditions and local windows. The loop-closure behavior controlled by position of the best particle has shown poor behavior when compared to one that performs qualitative SLAM measures, such as the uncertainty about the SLAM posterior. This can be seen by the poor behavior of *Ouroboros* when compared to *Ouroboros 2.0*. A future extension of this work could be the use of other SLAM measures to further improve the control of our loop-closure procedures. Also, one could argue that the environments are small. Still we highlight that the laser range finder had its maximum range diminished to increase the difficulty Active SLAM process. Also,

we need to test the algorithm in sparse environments performing proper modifications if necessary, including partial relaxations and additional modifications of the exploratory behavior to improve the robustness of the method even further.

We consider the concept of shields and tractor points useful for exploration and navigation. In fully explored maps, the effect of shields and tractor points may be helpful to continuously revisit and update the map. There is also the possibility to combine the effect of different shields and tractor points working together. *Ouroboros* defines submaps which actually encompass cycles in the environment. It is possible to see a map as a set of cycles and edges connecting them. We also could try to use the algorithm with different SLAM approaches such as SDP-SLAM and GraphSLAM to measure the benefits of our exploration approaches in these algorithms.

Even though a loop-closure can result in locally correct maps, there is a possibility that they are not globally correct (STACHNISS; GRISETTI; BURGARD, 2005a). There is a chance that returning to previously closed loops using specific paths in specific orders could improve the global map construction results. Furthermore, a shield next to the tractor point source alone can create movements clockwise and counterclockwise in a loop – depending on the side the tractor point source is and the robot position along the loop. Another possible improvement could the analysis of the loop path in search of features to improve localization and associate their quality for SLAM to trigger, or not, the loop closure behavior. This is important because the loop path may exist, but with no features to actually improve the SLAM posterior. The use of the shield, or the failure to generate one, across the path could be an indication of a good or bad loop path. However, all these possibilities need to be further investigated.

We could merge *Ouroboros* and Potential Rails in a new and better Active SLAM approach that controls the loop properly while at the same time keeping the patrolling behavior when the exploration ends. We are planning to apply *Ouroboros* to outdoor environments and 3D environments where approaches using BVP are still not real-time.

Finally we believe it is also possible to further explore the Potential Rails algorithm, specially with respect of the value if the the distortion parameter to guarantee proper turns in arbitrary environments. A fixed value for the potential distortion may not work in arbitrary environments when the distance and potential value of two different are too discrepant. For that purpose, a method could be devised to compute the distortion parameter value on-the-fly. Furthermore, we could also separate exploration potential and revisits to avoid this problem, starting the latter after the exploration of the map. This may give additional control to the local potential distortions, since the revisit potential will not fight with the potential of unexplored

regions. When this interaction happens, the robot may take too many time exploring known regions, increasing the exploration time unnecessarily. Lastly, the interaction between dynamic boundary conditions, specially those which are next to one another, should be further investigated with respect to convergence time and also stability. The same is true for distinct nearby local potential distortion regions.

Appendices

AppendixA MATHEMATICAL FORMULAS

Here we present basic formulas used throughout this work. Most definitions are extracted from the comprehensive work of Thrun, Burgard and Fox (2005).

Conditional Independence

Given two variables x and y, if they are statistically independent, then the joint probability of x and y can be written as

$$\boldsymbol{p}(x, y) = \boldsymbol{p}(x)\boldsymbol{p}(y). \tag{A.1}$$

Conditional Probability

When a random variable carries information about another random variable, we can condition the probability of the latter to the former. A conditional probability can be written as

$$\boldsymbol{p}(x \mid y) = \frac{\boldsymbol{p}(x, y)}{\boldsymbol{p}(y)}, \tag{A.2}$$

if $\boldsymbol{p}(y) > 0$.

Theorem of Total Probability

A consequence of the conditional probability is the Theorem of Total Probability. In the continuous case it is stated as

$$\boldsymbol{p}(x) = \int \boldsymbol{p}(x \mid y)\boldsymbol{p}(y)\, dy, \tag{A.3}$$

which in our case is reduced to the discrete case:

$$\boldsymbol{p}(x) = \sum_{y} \boldsymbol{p}(x \mid y)\boldsymbol{p}(y)\, dy. \tag{A.4}$$

Bayes Rule

Fundamental to our work is the Bayes Rule, which associates a conditional probability $\boldsymbol{p}(x \mid y)$ to its mirrored statistical counterpart $\boldsymbol{p}(y \mid x)$, It arises from the manipulation of Eq. A.2 followed by the application of the Law of Total probability

$$\boldsymbol{p}(x \mid y) = \frac{\boldsymbol{p}(y \mid x)\boldsymbol{p}(y)}{\boldsymbol{p}(y)} = \frac{\boldsymbol{p}(y \mid x)\boldsymbol{p}(y)}{\int \boldsymbol{p}(y \mid x')\boldsymbol{p}(x')\, dx}, \tag{A.5}$$

which also has a discrete form

$$\boldsymbol{p}(x \mid y) = \frac{\boldsymbol{p}(y \mid x)\boldsymbol{p}(y)}{\boldsymbol{p}(y)} = \frac{\boldsymbol{p}(y \mid x)\boldsymbol{p}(y)}{\sum_{x'} \boldsymbol{p}(y \mid x')\boldsymbol{p}(x')\, dx}. \tag{A.6}$$

Note that the denominators of Eqs. A.5 and A.6 can be seen as normalization constants.

AppendixB RECURSIVE ON-LINE SLAM DERIVATION

The joint probability of the SLAM posterior can be written as a recursive computation of the distribution at time $t-1$ (MONTEMERLO; THRUN, 2007). The key supposition is to assume the notion of complete state. As a result, the current state becomes the best predictor for next. This way, measurements and poses in previous time-steps are no longer required for their estimation. Such temporal processes are called Markov Chains (THRUN; BURGARD; FOX, 2005) — a special type of stochastic process.

Therefore, the probability of the pose $\mathbf{x}_t$ can be written as the conditional probability of $\mathbf{x}_t$ given the previous pose $\mathbf{x}_{t-1}$ and odometry measurement $\mathbf{u}_t$ that moved the robot from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$

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

This is a motion model, which is used to describe the robot movement and its underlying uncertainties.

In the same way, the measurement model (or observation model) describes the behavior and uncertainty of the exteroceptive sensors as a conditional probability function associating the measurement $\mathbf{z}_t$ with the robot position, $\mathbf{x}_t$, and the map $\mathbf{m}$.

$$\boldsymbol{p}(\mathbf{z}_t \,|\, \mathbf{x}_t, \mathbf{m}) \tag{B.2}$$

With these two probability functions, it is possible to derive the recursive formula for the on-line SLAM posterior distribution. First, we rewrite the posterior distribution of SLAM, from equation 4.1, using the Bayes rule.

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

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

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

Similarly to equation B.2, of the measurement model, $\mathbf{z}_t$, depends only on $\mathbf{x}_t$ and $\mathbf{m}$, obtaining:

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

$$\boldsymbol{p}(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta\, \boldsymbol{p}(\mathbf{z}_t \,|\, \mathbf{x}_t, \mathbf{m})\, \boldsymbol{p}(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}), \tag{B.3}$$

where η is a normalization constant.

The Law of Total Probability can be applied to the last term of equation B.3 in order to make the distribution $\mathbf{x}$ conditional to $\mathbf{x}_{t-1}$.

$$\boldsymbol{p}(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \int \boldsymbol{p}(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})\, \boldsymbol{p}(\mathbf{x}_{t-1} \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) d\mathbf{x}_{t-1} \tag{B.4}$$

After, applying the definition of conditional probability to the first term on the right side of B.4, we obtain

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

Then, replacing this result on B.4, we get

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

Now, we can replace the first term of the integral by the motion model defined by equation B.1, since $\mathbf{x}_t$ only depends on $\mathbf{x}_{t-1}$ and $\mathbf{u}_t$. We can also merge the last two terms, excluding $\mathbf{u}_t$ from the set $\mathbf{u}_{1:t}$, since the odometry at instant t is independent of $\mathbf{x}_{t-1}$ or $\mathbf{m}$ at the instant $t-1$.

$$\boldsymbol{p}(\mathbf{x}_t, \mathbf{m} \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \int \boldsymbol{p}(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \mathbf{u}_t)\, \boldsymbol{p}(\mathbf{x}_{t-1}, \mathbf{m} \,|\, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) d\mathbf{x}_{t-1} \tag{B.5}$$

Finally, we obtain a recursive formula for the joint posterior distribution of $\mathbf{x}_t$ and $\mathbf{m}$, given by the combination of the observation and motion models distributions at instant $t-1$. Then, Eq. B.3 becomes

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

Note that generally there is no closed form solution for the equation above, so approximations and assumptions must be made to reduce the dimensionality of the problem.

AppendixC ENTROPY OF BIVARIATE DISTRIBUTIONS

The SLAM posterior is a bivariate posterior distribution involving the map and the robot poses. Many approaches rely on the SLAM entropy directly (STACHNISS; GRISETTI; BURGARD, 2005a) or indirectly (BLANCO; MADRIGAL; GONZALEZ, 2008; DU et al., 2011) to select possible exploration goals. Here, we derive a suitable formula for the entropy of bivariate posterior distribution which separates the entropy for each distribution.

Entropy

Entropy is a measure of disorder or uncertainty about a random distribution (COVER; THOMAS, 1991). It can be written as

$$\boldsymbol{H}(\mathcal{X}) = -\sum_{x \in \mathcal{X}} \boldsymbol{p}(x)\boldsymbol{log}(x). \tag{C.1}$$

An entropy measurement is dependent on the log base. For instance, when the log base is 2, entropy is measured in $bits$. When the log base is the natural logarithm, e, it is measured in $nats$. In this thesis, unless specified otherwise, logarithms are expressed using base 2.

Entropy of a Bivariate Joint Distribution

The entropy, $\boldsymbol{H}$, of a joint probability distribution can be used to determine its underlying uncertainty:

$$\boldsymbol{H}(x, y) = \boldsymbol{E}_{x,y}\{-\boldsymbol{log}\big(\,\boldsymbol{p}(x,\, y)\big)\}. \tag{C.2}$$

By applying the definition of conditional probability to Eq. C.2, it becomes

$$\boldsymbol{H}(x, y) = \boldsymbol{E}_{x,y}\{-\boldsymbol{log}\big(\,\boldsymbol{p}(x)\,\boldsymbol{p}(y \,|\, x)\big)\}. \tag{C.3}$$

Then, adopting the property of logarithms, $\boldsymbol{log}(ab) = \boldsymbol{log}(a) + \boldsymbol{log}(b)$, in Eq. C.3, we obtain

$$\boldsymbol{H}(x, y) = \boldsymbol{E}_{x,y}\{-\boldsymbol{log}\big(\,\boldsymbol{p}(x)\big) - \boldsymbol{log}\big(\,\boldsymbol{p}(y \,|\, x)\big)\}. \tag{C.4}$$

Applying the additive property of expectation, $\boldsymbol{E}\{a + b\} = \boldsymbol{E}\{a\} + \boldsymbol{E}\{b\}$, into Eq. C.4

we get

$$\boldsymbol{H}(x,y) = \boldsymbol{E}_{x,y}\{-\boldsymbol{log}[\,\boldsymbol{p}(x)]\} + \boldsymbol{E}_{x,y}\{-\boldsymbol{log}[\,\boldsymbol{p}(y\,|\,x)]\}. \tag{C.5}$$

Afterward, knowing that $\boldsymbol{H}(x) = \boldsymbol{E}_x\{-\boldsymbol{log}\big(\,\boldsymbol{p}(x)\big)\}$ and substituting in Eq. C.5, the result is

$$\boldsymbol{H}(x,y) = \boldsymbol{H}(x) - \boldsymbol{E}_{x,y}\{\boldsymbol{log}\big(\,\boldsymbol{p}(y\,|\,x)\,\big)\}. \tag{C.6}$$

Finally, applying the definition of the expected value of a continuous bivariate distribution, we obtain

$$\boldsymbol{H}(x,y) = \boldsymbol{H}(x) + \int_{x,y} -\boldsymbol{p}(x,y)\,\boldsymbol{log}\big(\,\boldsymbol{p}(y\,|\,x)\,\big)\,dx\,dy. \tag{C.7}$$

The integral of Eq. C.7 can be modified using again the definition of conditional probability,

$$\int_{x,y} -\boldsymbol{p}(x,y)\,\boldsymbol{log}\big(\,\boldsymbol{p}(y\,|\,x)\,\big)\,dx\,dy = \int_{x,y} -\boldsymbol{p}(x)\,\boldsymbol{p}(y\,|\,x)\,\boldsymbol{log}\big(\,\boldsymbol{p}(y\,|\,x)\,\big)\,dx\,dy. \tag{C.8}$$

Since $\boldsymbol{p}(x)$ is independent of $\boldsymbol{p}(y)$ in the above equation, we can rewrite Eq. C.8 as

$$\int_{x,y} -\boldsymbol{p}(x)\,\boldsymbol{p}(y\,|\,x)\,\boldsymbol{log}\big(\,\boldsymbol{p}(y\,|\,x)\,\big)\,dx\,dy \;=\; \int_x \boldsymbol{p}(x) \int_y -\boldsymbol{p}(y\,|\,x)\,\boldsymbol{log}\big(\,\boldsymbol{p}(y\,|\,x)\,\big)\,dy\,dx, \tag{C.9}$$

which uncovers the entropy $\boldsymbol{H}(y\,|\,x)$. That is,

$$\int_x \boldsymbol{p}(x) \int_y -\boldsymbol{p}(y\,|\,x)\,\boldsymbol{log}\big(\,\boldsymbol{p}(y\,|\,x)\,\big)\,dy\,dx \;=\; \int_x \boldsymbol{p}(x)\,\boldsymbol{H}(y\,|\,x)\,dx \tag{C.10}$$

Replacing Eq. C.10 into Eq. C.7, we obtain a different way to write the joint entropy:

$$\boldsymbol{H}(x,y) = \boldsymbol{H}(x) + \int_x \boldsymbol{p}(x)\,\boldsymbol{H}(y\,|\,x)\,dx. \tag{C.11}$$

Kullback-Leibler Divergence for Active SLAM

Fox (2003) has used the KL Divergence to estimate the number of samples of particle filters with respect to the localization problem. Briefly, the strategy assumes that there is a way to estimate the number of particles required to model a distribution with probability $1-\delta$ that the distance between the true distribution and the estimated distribution is less than a given

value, ε. One way to compare two distributions is to use the KL divergence[1].

Suppose that N samples are drawn from a probability distribution with k different bins of a discrete distribution, where $\mathcal{X} = \{x_1, \cdots, x_k\}$ define the number of samples drawn for each bin. The maximum likelihood estimate of this discrete multinomial distribution is given by $\hat{\boldsymbol{Q}} = \mathcal{X}/N$, which gives the estimated probability, $\hat{\boldsymbol{q}}$, for each bin. Thus, the log likelihood ratio statistic, Λ, can be written as

$$\boldsymbol{log}(\Lambda) = \sum_{i=1}^{k} x_i \boldsymbol{log}\left(\frac{\hat{\boldsymbol{q}}(x_i)}{\boldsymbol{q}(x_i)}\right),$$

where $\boldsymbol{q}$ is the true probability distribution. But $x_i = N\boldsymbol{q}(x_i)$, so

$$\boldsymbol{log}(\Lambda) = N\sum_{i=1}^{k} \hat{\boldsymbol{q}}(x_i) \boldsymbol{log}\left(\frac{\hat{\boldsymbol{q}}(x_i)}{\boldsymbol{q}(x_i)}\right).$$

As a result

$$\boldsymbol{log}(\Lambda) = N\boldsymbol{\xi}(\hat{\boldsymbol{q}}, \boldsymbol{q}).$$

Assuming we can ensure that the maximum likelihood estimators are locally unique, consistent, and asymptotically normal[2] (FOX, 2003), we know that

$$2\,\boldsymbol{log}(\Lambda) = \boldsymbol{\chi}^2_{k-1} \quad \text{as} \quad N \to \infty,$$

where $\boldsymbol{\chi}^2_{k-1}$ is a chi-square distribution with $k-1$ degrees of freedom[3]. In this way, assuming that the conditions above hold, and that the probability, $\boldsymbol{p}_\varepsilon(\xi(\hat{\boldsymbol{p}}, \boldsymbol{p}) \leq \varepsilon)$, with a small KL-distance value, we can correlate the chi-square distribution and the number of samples by

$$\xi(\hat{\boldsymbol{q}}, \boldsymbol{q}) \leq \varepsilon \tag{C.12}$$

$$2N\xi(\hat{\boldsymbol{q}}, \boldsymbol{q}) \leq 2N\varepsilon \tag{C.13}$$

$$2N\boldsymbol{log}(\Lambda) \leq 2N\varepsilon \tag{C.14}$$

$$\boldsymbol{\chi}^2_{k-1} \leq 2N\varepsilon \tag{C.15}$$

[1]The KL-distance is not metric because it is not symmetric ($\boldsymbol{\xi}(\boldsymbol{p}, \boldsymbol{q}) \neq \boldsymbol{\xi}(\boldsymbol{q}, \boldsymbol{p})$) and does not respect the triangle inequality. From Eq. 5.13, we can see that $\boldsymbol{\xi}(\boldsymbol{p}, \boldsymbol{q}) = \infty$ when, for some x, $\boldsymbol{q}(x) = 0$ and $\boldsymbol{p}(x) > 0$, since $\boldsymbol{log}(\boldsymbol{p}(x)/0) = \infty$. Still, it is only zero when the two distributions are the same and it is always positive. For this reason it is commonly used as a standard measure of divergence between two distributions.

[2]<http://www.math.umd.edu/~slud/s701.S14/WilksThm.pdf>

[3]This asymptotic behavior is valid when the number of independent variables is small, in relation to the number of samples, otherwise it may result in approximation errors (PORTNOY, 1988).

Of course, if we allegedly know the probability distribution function, we can correlate the probability of this distribution to its quantiles, $1 - \delta$

$$\boldsymbol{p}(\boldsymbol{\chi}^2_{k-1} \leq \boldsymbol{\chi}^2_{k-1,1-\delta}) = 1 - \delta = \boldsymbol{p}(\xi(\hat{\boldsymbol{q}}, \boldsymbol{q}) \leq \varepsilon)$$

where $0 \leq 1 - \delta \leq 1$.

Then, if we make $2N\varepsilon \geq \boldsymbol{\chi}^2_{k-1,1-\delta}$, we get

$$N = \frac{1}{2\varepsilon}\boldsymbol{\chi}^2_{k-1,1-\delta}.$$

which links the number of samples and quantiles directly, with a given probability distribution. This in turn can be seen as the number of samples required to keep $\boldsymbol{p}(\xi(\hat{\boldsymbol{q}}, \boldsymbol{q}) \leq \varepsilon)$. The quantiles of the chi-square distribution can be obtained through the Wilson-Hilferty transformation (WILSON; HILFERTY, 1931; FOX, 2003), which in turn can be used to compute the number of samples as

$$N = \frac{k-1}{2\varepsilon}\left(1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}\mathfrak{Z}_{1-\delta}\right)^3, \tag{C.16}$$

where $\mathfrak{Z}_{1-\delta}$ is the upper quantile of the standard normal distribution, obtained from statistical tables.

The problem of the method is then to compare each sample with the true belief, which is unknown. Fox decided to use the distribution obtained during the sampling phase of the particle filter as the true distribution. Still, the method does not need the KL-distance directly, it only needs to know the total number of bins with support ($\boldsymbol{p}(x) > 0$) to obtain the required number of particles using Eq. C.16. Parameters such as bin size and ε were determined empirically through a series of experiments. It is worth mentioning that the method does not guarantee convergence[4].

This approach was adapted for Active SLAM, in a reverse process (CARLONE et al., 2009; CARLONE et al., 2010; DU et al., 2011), where, instead of computing the number of samples, which is set fixed, the problem becomes to estimate the confidence level of the pose posterior (CARLONE et al., 2009), $\boldsymbol{p}(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})$. Hence, one must discover $\mathfrak{Z}_{1-\delta_t}$, which

[4]Of course, the particle filter, even in its plain version is a sequential Monte Carlo approach which also does not guarantee convergence.

can be obtained by modifying Eq. C.16

$$\beth_{1-\delta_t} = \left(\left(\frac{2N\varepsilon}{k_t - 1} \right)^{\frac{1}{3}} + \frac{2}{9(k_t - 1)} - 1 \right) \sqrt{\frac{9(k_t - 1)}{2}}, \tag{C.17}$$

and used to obtain the cumulative distribution function of the Normal distribution at $\beth_{1-\delta_t}$. This is equivalent to the area below the normal curve and can be employed as an estimate about the certainty of the pose posterior[5]. Having this parameter one can measure the uncertainty about possible future poses and use it to decide what path should be taken. Note that again, the number of non-empty bins, k_t at time t, is responsible for changes in the uncertainty.

[5]This is the z_a parameter in statistical tables. The cumulative distribution is usually given by a or $0.5 + a < 1$

REFERENCES

ABDALLAH, F.; GNING, A.; BONNIFAIT, P. Box particle filtering for nonlinear state estimation using interval analysis. **Automatica**, Elsevier, v. 44, n. 3, p. 807–815, 2008.

AGARWAL, P. et al. Robust map optimization using dynamic covariance scaling. In: **Robotics and Automation (ICRA), 2013 IEEE International Conference on**. USA: IEEE Press, 2013. p. 62–69. ISSN 1050-4729.

AMIGONI, F. Experimental evaluation of some exploration strategies for mobile robots. In: **Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on**. IEEE Press, 2008. p. 2818–2823. ISSN 1050-4729.

AMIGONI, F.; CAGLIOTI, V. An information-based criterion for efficient robot map building. In: **Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2003. VECIMS '03. 2003 IEEE International Symposium on**. [IEEE Press, 2003. p. 184–189.

AMIGONI, F.; CAGLIOTI, V. An information-based exploration strategy for environment mapping with mobile robots. **Robotics and Autonomous Systems**, v. 58, n. 5, p. 684–699, 2010. ISSN 0921-8890.

AMIGONI, F.; CAGLIOTI, V.; GALTAROSSA, U. A mobile robot mapping system with an information-based exploration strategy. In: **ICINCO (2)**. 2004. p. 71–78.

ANDERSON, B. D.; MOORE, J. B. **Optimal filtering**. Courier Corporation, 2012. 46-50 p.

ASCHER, U. M.; MATTHEIJ, R. M.; RUSSELL, R. D. **Numerical solution of boundary value problems for ordinary differential equations**. Siam, 1994. 1 p.

BAILEY, T. et al. Consistency of the ekf-slam algorithm. In: **Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2006. p. 3562–3568. ISBN 1-4244-0258-1. Available from Internet: <http://dx.doi.org/10.1109/IROS.2006.281644>. Accessed in: 28 apr. 2017.

BHAT, S.; MEENAKSHI, M. Vision based robotic system for military applications – design and real time validation. In: **Signal and Image Processing (ICSIP), 2014 Fifth International Conference on**. IEEE Press, 2014. p. 20–25.

BLANCO, J. L.; MADRIGAL, J. A. F.-.; GONZALEZ, J. A novel measure of uncertainty for mobile robot slam with rao-blackwellized particle filters. **Int. Journal of Robotic Research**, Sage Publications, Inc., Thousand Oaks, CA, USA, v. 27, p. 73–89, 2008. ISSN 0278-3649.

CARLONE, L. et al. An application of kullback-leibler divergence to active slam and exploration with particle filters. In: **Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on**. IEEE Press, 2010. p. 287–293. ISSN 2153-0858.

CARLONE, L.; LYONS, D. Uncertainty-constrained robot exploration: A mixed-integer linear programming approach. In: **Robotics and Automation (ICRA), 2014 IEEE International Conference on**. IEEE Press, 2014. p. 1140–1147.

CARLONE, L. et al. Reverse kld-sampling for measuring uncertainty in rao-blackwellized particle filters slam. In: **Workshop on Performance Evaluation and Benchmarking for Next Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems**. IEEE Press, 2009.

CASPER, J.; MURPHY, R. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. **Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on**, v. 33, n. 3, p. 367–385, June 2003. ISSN 1083-4419.

CHARTRAND, G.; ZHANG, P. **A first course in Graph Theory**. Mineola, New York: DOVER PUBLICATIONS, INC., 2012. 11-13 p. ISBN 13:978-0-486-48368-9.

CHOSET, H. et al. **Principles of Robot Motion: Theory, Algorithms, and Implementations**. Cambridge, MA: MIT Press, 2005.

CONNOLLY, C. I.; GRUPEN, R. A. The applications of harmonic functions to robotics. **Journal of Robotic Systems**, Wiley Subscription Services, Inc., A Wiley Company, v. 10, n. 7, p. 931–946, 1993. ISSN 1097-4563. Available from Internet: <http://dx.doi.org/10.1002/rob.4620100704>. Accessed in: 28 apr. 2017.

COVER, T. M.; THOMAS, J. A. Entropy, relative entropy and mutual information. **Elements of Information Theory**, John Wiley & Sons, Inc. Hoboken, NJ, p. 12–49, 1991.

DELLAERT, F. et al. Monte carlo localization for mobile robots. In: **Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 1999.

DOUCET, A. et al. Rao-blackwellised particle filtering for dynamic bayesian networks. In: **Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI)**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000. p. 176–183. ISBN 1-55860-709-9. Available from Internet: <http://dl.acm.org/citation.cfm?id=647234.720075>. Accessed in: 28 apr. 2017.

DU, J. et al. A comparative study on active slam and autonomous exploration with particle filters. In: **Advanced Intelligent Mechatronics (AIM), 2011 IEEE/ASME International Conference on**. IEEE Press, 2011. p. 916–923. ISSN 2159-6247.

ELIAZAR, A.; PARR, R. Dp-slam: fast, robust simultaneous localization and mapping without predetermined landmarks. In: **Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003. p. 1135–1142. Available from Internet: <http://dl.acm.org/citation.cfm?id=1630659.1630822>. Accessed in: 28 apr. 2017.

ELIAZAR, A. I.; PARR, R. Dp-slam 2.0. In: **Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2004. v. 2, p. 1314–1320. ISSN 1050-4729.

ELIAZAR, A. I.; PARR, R. Learning probabilistic motion models for mobile robots. In: **Proceedings of the twenty-first international conference on Machine learning**. New York, NY, USA: ACM, 2004. (ICML '04), p. 32. ISBN 1-58113-838-5. Available from Internet: <http://doi.acm.org/10.1145/1015330.1015413>. Accessed in: 28 apr. 2017.

ELIAZAR, A. I.; PARR, R. Hierarchical linear/constant time slam using particle filters for dense maps. In: **Proceedings of the 18th Advances in Neural Information Processing Systems (NIPS)**. Cambridge, MA, USA: MIT Press, 2006. p. 339–346.

ENDRES, F. et al. An evaluation of the rgb-d slam system. In: **Robotics and Automation (ICRA), 2012 IEEE International Conference on**. IEEE Press, 2012. p. 1691–1696. ISSN 1050-4729.

ENGEL, J.; SCHöPS, T.; CREMERS, D. Lsd-slam: Large-scale direct monocular slam. In: FLEET, D. et al. (Ed.). **Computer Vision – ECCV 2014**. Springer International Publishing, 2014, (Lecture Notes in Computer Science, v. 8690). p. 834–849. ISBN 978-3-319-10604-5. Available from Internet: <http://dx.doi.org/10.1007/978-3-319-10605-2_54>. Accessed in: 28 apr. 2017.

FANFANI, M. et al. Samslam: Simulated annealing monocular slam. In: WILSON, R. et al. (Ed.). **Computer Analysis of Images and Patterns**. Springer Berlin Heidelberg, 2013, (Lecture Notes in Computer Science, v. 8048). p. 515–522. ISBN 978-3-642-40245-6. Available from Internet: <http://dx.doi.org/10.1007/978-3-642-40246-3_64>. Accessed in: 28 apr. 2017.

FISHER, R. A. On the mathematical foundations of theoretical statistics. **Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences**, The Royal Society, v. 222, n. 594-604, p. 309–368, 1922. ISSN 0264-3952.

FOX, D. Adapting the sample size in particle filters through kld-sampling. **The international Journal of Robotics Research**, SAGE Publications, v. 22, n. 12, p. 985–1003, 2003.

FRESE, U. Treemap: An o(log n) algorithm for indoor simultaneous localization and mapping. **Autonomous Robots**, Kluwer Academic Publishers, Hingham, MA, USA, v. 21, n. 2, p. 103–122, sep 2006. ISSN 0929-5593. Available from Internet: <http://dx.doi.org/10.1007/s10514-006-9043-2>. Accessed in: 28 apr. 2017.

GAYLE, R.; LIN, M.; MANOCHA, D. Constraint-based motion planning of deformable robots. In: **Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on**. IEEE Press, 2005. p. 1046–1053.

GRISETTI, G. et al. A tutorial on graph-based slam. **IEEE Intelligent Transportation Systems Magazine**, IEEE Press, Piscataway, NJ, USA, v. 2, n. 4, p. 31–43, winter 2010. ISSN 1939-1390.

GRISETTI, G.; STACHNISS, C.; BURGARD, W. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In: **Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2005. p. 2443–2448.

GRISETTI, G.; STACHNISS, C.; BURGARD, W. Improved techniques for grid mapping with rao-blackwellized particle filters. **IEEE Transactions on Robotics**, IEEE Press, Piscataway, NJ, USA, v. 23, n. 1, p. 34–46, feb. 2007. ISSN 1552-3098.

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**, IEEE Press, Piscataway, NJ, USA, v. 17, n. 3, p. 242–257, jun 2001. ISSN

1042296X. Available from Internet: <http://dx.doi.org/10.1109/70.938382>. Accessed in: 28 apr. 2017.

GUO, Z.; HALL, R. W. Parallel thinning with two-subiteration algorithms. **Commun. ACM**, ACM, New York, NY, USA, v. 32, n. 3, p. 359–373, mar 1989. ISSN 0001-0782.

HAHNEL, D. et al. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In: **Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2003. v. 1, p. 206–211.

HAHNEL, D.; SCHULZ, D.; BURGARD, W. Map building with mobile robots in populated environments. In: **Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on**. IEEE Press, 2002. v. 1, p. 496–501 vol.1.

HASAN, K.; ABDULLAH-AL-NAHID; REZA, K. Path planning algorithm development for autonomous vacuum cleaner robots. In: **Informatics, Electronics Vision (ICIEV), 2014 International Conference on**. 2014. p. 1–6.

HOCKING, J. G.; YOUNG, G. S. **Topology**. 3rd. ed. NY: Dover Publications, Inc., 2012. 384 p. ISBN 978-0486656762.

HUANG, G. P.; MOURIKIS, A. I.; ROUMELIOTIS, S. I. Analysis and improvement of the consistency of extended kalman filter based slam. In: **Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2008. p. 473–479. ISBN 978-1-4244-1646-2. ISSN 1050-4729. Available from Internet: <http://dx.doi.org/10.1109/ROBOT.2008.4543252>. Accessed in: 28 apr. 2017.

HUANG, G. P.; MOURIKIS, A. I.; ROUMELIOTIS, S. I. Observability-based rules for designing consistent ekf slam estimators. **International Journal of Robotics Research**, Sage Publications, Inc., Thousand Oaks, CA, USA, v. 29, n. 5, p. 502–528, April 2010. ISSN 0278-3649. Available from Internet: <http://dx.doi.org/10.1177/0278364909353640>. Accessed in: 28 apr. 2017.

HUANG, H.-M. et al. A framework for autonomy levels for unmanned systems (alfus). **Proceedings of AUVSI Unmanned Systems 2005**, Citeseer, 2005.

HUANG, S.; DISSANAYAKE, G. Convergence and consistency analysis for extended kalman filter based slam. **Robotics, IEEE Transactions on**, v. 23, n. 5, p. 1036–1049, Oct 2007. ISSN 1552-3098.

HUANG, S. et al. How far is slam from a linear least squares problem? In: **Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2010. p. 3011–3016. ISSN 2153-0858.

JO, K. et al. Development of autonomous car 2014 part i: Distributed system architecture and development process. **Industrial Electronics, IEEE Transactions on**, v. 61, n. 12, p. 7131–7140, Dec 2014. ISSN 0278-0046.

JO, K. et al. Development of autonomous car 2014 part ii: A case study on the implementation of an autonomous driving system based on distributed architecture. **Industrial Electronics, IEEE Transactions on**, v. 62, n. 8, p. 5119–5132, Aug 2015. ISSN 0278-0046.

JORGE, V. et al. Ouroboros: Using potential field in unexplored regions to close loops. In: **Robotics and Automation (ICRA), 2015 IEEE International Conference on**. IEEE Press, 2015. p. 2125–2131.

JULIÁ, M. et al. A hybrid solution to the multi-robot integrated exploration problem. **Engineering Applications of Artificial Intelligence**, v. 23, n. 4, p. 473–486, 2010. ISSN 0952-1976. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0952197609001791>. Accessed in: 28 apr. 2017.

JULIER, S. J.; UHLMANN, J. K. A new extension of the kalman filter to nonlinear systems. In: **Proceedings of the 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls**. Bellingham, WA, USA: SPIE, 1997.

KALMAN, R. E. A new approach to linear filtering and prediction problems. **Transactions of the ASME–Journal of Basic Engineering**, v. 82, n. Series D, p. 35–45, 1960.

KANG, J.-G. et al. Augmented ekf based slam method for improving the accuracy of the feature map. In: **Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2010. p. 3725–3731. ISBN 978-1-4244-6674-0. ISSN 2153-0858. Available from Internet: <http://dx.doi.org/10.1109/IROS.2010.5652938>. Accessed in: 28 apr. 2017.

KIM, B. et al. Multiple relative pose graphs for robust cooperative mapping. In: **Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2010. p. 3185–3192. ISSN 1050-4729.

KNIGHT, J.; DAVISON, A.; REID, I. Towards constant time slam using postponement. In: **Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2001. v. 1. Available from Internet: <http://dx.doi.org/10.1109/IROS.2001.973391>. Accessed in: 28 apr. 2017.

KULLBACK, S.; LEIBLER, R. A. On information and sufficiency. **Ann. Math. Statist.**, The Institute of Mathematical Statistics, v. 22, n. 1, p. 79–86, 03 1951. Available from Internet: <http://dx.doi.org/10.1214/aoms/1177729694>. Accessed in: 28 apr. 2017.

LATIF, Y.; CADENA, C.; NEIRA, J. Robust loop closing over time for pose graph slam. **The International Journal of Robotics Research**, v. 32, n. 14, p. 1611–1626, 2013. Available from Internet: <http://ijr.sagepub.com/content/32/14/1611.abstract>. Accessed in: 28 apr. 2017.

LATOMBE, J.-C. Probabilistic roadmaps: An incremental sampling approach to approximate the connectivity of robot configuration spaces. In: **Cognitive Informatics, 2008. ICCI 2008. 7th IEEE International Conference on**. [S.l.: s.n.], 2008. p. 2–2.

LAVALLE, S. M. **Planning Algorithms**. Cambridge U. Press, 2006. ISBN 0521862051.

LEE, C. An algorithm for path connections and its applications. **Electronic Computers, IRE Transactions on**, EC-10, n. 3, p. 346–365, Sept 1961. ISSN 0367-9950.

LEE, J. et al. Modeling autonomous military robots using hybrid system framework. In: **Information and Communication Technology Convergence (ICTC), 2010 International Conference on**. 2010. p. 429–430.

LEE, S.-J.; SONG, J.-B. A new sonar salient feature structure for ekf-based slam. In: **Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2010. p. 5966–5971. ISBN 978-1-4244-6674-0. ISSN 2153-0858. Available from Internet: <http://dx.doi.org/10.1109/IROS.2010.5650169>. Accessed in: 28 apr. 2017.

LEUNG, C.; HUANG, S.; DISSANAYAKE, G. Active slam using model predictive control and attractor based exploration. In: **Proceedings of IEEE/RSJ IROS**. Piscataway, NJ, USA: IEEE Press, 2006. p. 5026–5031.

LI, T.; SUN, S.; SATTAR, T. Adapting sample size in particle filters through kld-resampling. **Electronics Letters**, v. 49, n. 12, p. 740–742, June 2013. ISSN 0013-5194.

LIPOWSKI, A.; LIPOWSKA, D. Roulette-wheel selection via stochastic acceptance. **Physica A: Statistical Mechanics and its Applications**, v. 391, n. 6, p. 2193 – 2196, 2012. ISSN 0378-4371. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0378437111009010>. Accessed in: 28 apr. 2017.

LIU, J. S. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. **Statistics and Computing Journal**, Springer Netherlands, Dordrecht, Netherlands, v. 6, n. 2, p. 113–119, jun 1996. Available from Internet: <http://dx.doi.org/10.1007/BF00162521>. Accessed in: 28 apr. 2017.

LIU, Y. et al. Sensory navigation of autonomous cleaning robots. In: **Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on**. 2004. v. 6, p. 4793–4796 Vol.6.

LOZANO-PEREZ, T. Spatial planning: A configuration space approach. **Computers, IEEE Transactions on**, C-32, n. 2, p. 108–120, Feb 1983. ISSN 0018-9340.

MADHAVAN, R. et al. 2014 humanitarian robotics and automation technology challenge [humanitarian technology]. **Robotics Automation Magazine, IEEE**, v. 21, n. 3, p. 10–16, Sept 2014. ISSN 1070-9932.

MADHAVAN, R. et al. 2015 humanitarian robotics and automation technology challenge [humanitarian technology]. **Robotics Automation Magazine, IEEE**, v. 22, n. 3, p. 182–184, Sept 2015. ISSN 1070-9932.

MAFFEI, R. et al. Segmented dp-slam. In: **Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on**. Piscataway, NJ, USA: IEEE Press, 2013. p. 31–36. ISSN 2153-0858.

MAFFEI, R. et al. Integrated exploration using time-based potential rails. In: **Robotics and Automation (ICRA), 2014 IEEE International Conference on**. Piscataway, NJ, USA: IEEE Press, 2014. p. 3694–3699.

MAFFEI, R. et al. Using n-grams of spatial densities to construct maps. In: **Inteligent Robotics and Systems (IROS), 2015 IEEE International Conference on**. Piscataway, NJ, USA: IEEE Press, 2015.

MAKARENKO, A. A. et al. An experiment in integrated exploration. In: **Proceedings of IEEE/RSJ IROS**. Piscataway, NJ, USA: IEEE Press, 2002. p. 534–539.

MAXWELL, P.; LARKIN, D.; LOWRANCE, C. Turning remote-controlled military systems into autonomous force multipliers. **Potentials, IEEE**, v. 32, n. 6, p. 39–43, Nov 2013. ISSN 0278-6648.

MCDONALD, J. B. et al. 6-dof multi-session visual slam using anchor nodes. In: LILIENTHAL, A. J.; DUCKETT, T. (Ed.). **Proceedings of the 5th European Conference on Mobile Robots (ECMR)**. Orebro, Sweden: 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). ISBN 3540463992.

MONTEMERLO, M. et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In: **Proceedings of the AAAI National Conference on Artificial Intelligence**. Edmonton, Canada: AAAI, 2002.

MONTEMERLO, M. et al. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: **Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

MOORE, E. F. **The shortest path through a maze**. USA: Bell Telephone System., 1959.

MURPHY, K. P. Bayesian map learning in dynamic environments. In: **Proceedings of the 12th Advances in Neural Information Processing Systems (NIPS)**. Cambridge, MA, USA: MIT Press, 1999. p. 1015–1021.

MURPHY, R.; PRATT, K.; BURKE, J. Crew roles and operational protocols for rotary-wing micro-uavs in close urban environments. In: **Human-Robot Interaction (HRI), 2008 3rd ACM/IEEE International Conference on**. 2008. p. 73–80. ISSN 2167-2121.

MURPHY, R. R. **An Introduction to AI Robotics (Intelligent Robotics and Autonomous Agents)**. The MIT Press, 2000. ISBN 0262133830. Available from Internet: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0262133830>. Accessed in: 28 apr. 2017.

NASKAR, S. et al. Application of radio frequency controlled intelligent military robot in defense. In: **Communication Systems and Network Technologies (CSNT), 2011 International Conference on**. 2011. p. 396–401.

NI, K.; STEEDLY, D.; DELLAERT, F. Tectonic sam: Exact, out-of-core, submap-based slam. In: **Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE Press, 2007. p. 1678–1685. ISSN 1050-4729.

NIETO, J. I.; BAILEY, T.; NEBOT, E. M. Scan-slam: Combining ekf-slam and scan correlation. In: CORKE, P. I.; SUKKARIEH, S. (Ed.). **Proceedings of the 5th International Conference of Field and Service Robotics**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. (Springer Tracts in Advanced Robotics, v. 25), p. 167–178. ISBN 978-3-540-33452-1.

OKEREAFOR, D. et al. Improving security and emergency response through the use of unmanned vehicles. In: **Emerging Sustainable Technologies for Power ICT in a Developing Society (NIGERCON), 2013 IEEE International Conference on**. USA: IEEE Press, 2013. p. 263–269.

OLSON, E. B. Real-time correlative scan matching. In: **Robotics and Automation, 2009. ICRA '09. IEEE International Conference on**. Piscataway, NJ, USA: IEEE Press, 2009. p. 4387–4393. ISSN 1050-4729.

PAZ, L. M.; TARDOS, J. D.; NEIRA, J. Divide and conquer: Ekf slam in. **IEEE Transactions on Robotics**, IEEE Press, Piscataway, NJ, USA, v. 24, n. 5, p. 1107–1120, oct. 2008. ISSN 1552-3098.

PETTERSSON, I.; KARLSSON, I. Setting the stage for autonomous cars: a pilot study of future autonomous driving experiences. **Intelligent Transport Systems, IET**, v. 9, n. 7, p. 694–701, 2015. ISSN 1751-956X.

PFINGSTHORN, M.; BIRK, A. Generalized graph slam: Solving local and global ambiguities through multimodal and hyperedge constraints. **The International Journal of Robotics Research**, 2015. Available from Internet: <http://ijr.sagepub.com/content/early/2015/07/09/0278364915585395.abstract>. Accessed in: 28 apr. 2017.

PORTNOY, S. Asymptotic behavior of likelihood methods for exponential families when the number of parameters tends to infinity. In: **The Annals of Statistics**. 1988. v. 16(1), p. 356–366.

PRESTES, E. et al. Towards a core ontology for robotics and automation. **Robotics and Autonomous Systems**, v. 61, n. 11, p. 1193 – 1204, 2013. ISSN 0921-8890. Ubiquitous

Robotics. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0921889013000596>. Accessed in: 28 apr. 2017.

PRESTES, E.; ENGEL, P. M. Exploration driven by local potential distortions. In: **Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE Press, 2011. p. 1122–1127. ISSN 2153-0858.

PRESTES, E. et al. Exploration method using harmonic functions. **Robotics and Autonomous Systems**, v. 40, n. 1, p. 25–42, 2002.

PRESTES, E.; IDIART, M. Sculpting potential fields in the bvp path planner. In: **Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on**. IEEE Press, 2009. p. 183–188.

PRESTES, E. et al. Exploration technique using potential fields calculated from relaxation methods. In: **Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on**. Piscataway, NJ, USA: IEEE Press, 2001. v. 4, p. 2012–2017 vol.4.

PRESTES, E. et al. Bvp-exploration: further improvements. In: **Proceedings of IEEE/RSJ IROS**. Piscataway, NJ, USA: IEEE Press, 2003. p. 3239–3244.

ROLLINSON, D.; BUCHAN, A.; CHOSET, H. State estimation for snake robots. In: **Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on**. Piscataway, NJ, USA: IEEE Press, 2011. p. 1075–1080. ISSN 2153-0858.

ROMERO, R. A. et al. Locally oriented potential field for controlling multi-robots. **Communications in Nonlinear Science and Numerical Simulation**, v. 17, n. 12, p. 4664 – 4671, 2012. ISSN 1007-5704. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S1007570411005971>. Accessed in: 28 apr. 2017.

ROSS, P. Robot, you can drive my car. **Spectrum, IEEE**, v. 51, n. 6, p. 60–90, June 2014. ISSN 0018-9235.

ROY, N. et al. Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. In: **Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on**. Piscataway, NJ, USA: IEEE Press, 1999. v. 1, p. 35–40 vol.1. ISSN 1050-4729.

SHIM, I. et al. An autonomous driving system for unknown environments using a unified map. **Intelligent Transportation Systems, IEEE Transactions on**, v. 16, n. 4, p. 1999–2013, Aug 2015. ISSN 1524-9050.

SICILIANO, B.; KHATIB, O. (Ed.). **Springer Handbook of Robotics**. Berlin, Heidelberg: Springer, 2008. ISBN 978-3-540-23957-4. Available from Internet: <http://dx.doi.org/10.1007/978-3-540-30301-5>. Accessed in: 28 apr. 2017

SIEGWART, R.; NOURBAKHSH, I. R. **Introduction to Autonomous Mobile Robots**. Scituate, MA, USA: Bradford Company, 2004. ISBN 026219502X.

SILVEIRA, R. **Configurable flows.** Thesis (PhD) — Universidade Federal do Rio Grande do Sul. Instituto de Informática. Programa de Pós-Graduação em Computação., 2015.

SILVEIRA, R.; PRESTES, E.; NEDEL, L. Fast path planning using multi-resolution boundary value problems. In: **Proceedings of IEEE/RSJ IROS**. Piscataway, NJ, USA: IEEE Press, 2010. p. 4710–4715. ISSN 2153-0858.

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, chp. Autonomous robot vehicles, p. 167–193. ISBN 0-387-97240-4. Available from Internet: <http://dl.acm.org/citation.cfm?id=93002.93291>. Accessed in: 28 apr. 2017.

SPONG, M. W.; HUTCHINSON, S.; VIDYASAGAR, M. **Robot modeling and control**. USA: Wiley New York, 2006.

STACHNISS, C. **Exploration and mapping with mobile robots.** Thesis (PhD) — University of Freiburg, 2006.

STACHNISS, C. **Robotic Mapping and Exploration**. 1st. ed. Springer, 2009. (Springer Tracts in Advanced Robotics). ISSN 1610-7438. ISBN 978-3-642-01096-5.

STACHNISS, C.; GRISETTI, G.; BURGARD, W. Information gain-based exploration using rao-blackwellized particle filters. In: **Proceedigs of Robotics: Science and Systems**. Cambridge, USA: 2005.

STACHNISS, C.; GRISETTI, G.; BURGARD, W. Recovering particle diversity in a rao-blackwellized particle filter for slam after actively closing loops. In: **Proceedings of the**

2005 IEEE International Conference on Robotics and Automation (ICRA). Piscataway, NJ, USA: IEEE Press, 2005. p. 655–660.

STACHNISS, C.; HAHNEL, D.; BURGARD, W. Exploration with active loop-closing for fastslam. In: **Proceedings of IEEE/RSJ IROS**. Piscataway, NJ, USA: IEEE Press, 2004. v. 2, p. 1505–1510.

STEUX, B.; HAMZAOUI, O. E. tinyslam: A slam algorithm in less than 200 lines c-language program. In: **Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on**. 2010. p. 1975–1979.

TARDOS, J. D. et al. Robust mapping and localization in indoor environments using sonar data. **The International Journal of Robotics Research**, SAGE Publications, Thousand Oaks, CA, USA, v. 21, n. 4, p. 311–330, 2002. Available from Internet: <http://ijr.sagepub.com/cgi/doi/10.1177/027836402320556340>. Accessed in: 28 apr. 2017.

TARJAN, R. Depth-first search and linear graph algorithms. **SIAM journal on computing**, SIAM, v. 1, n. 2, p. 146–160, 1972.

THRING, M. Automation in the home. **Electronics and Power**, v. 14, n. 11, p. 440–441, November 1968. ISSN 0013-5127.

THRUN, S. A probabilistic online mapping algorithm for teams of mobile robots. **International Journal of Robotics Research**, v. 20, n. 5, p. 335–363, 2001.

THRUN, S.; BURGARD, W.; FOX, D. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In: **Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on**. USA: IEEE Press, 2000. v. 1, p. 321–328 vol.1. ISSN 1050-4729.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)**. Cambridge, MA, 2005. ISBN 9780262201629.

THRUN, S.; MONTEMERLO, M. The graph slam algorithm with applications to large-scale mapping of urban structures. **The International Journal of Robotics Research**, Sage Publications, Inc., Thousand Oaks, CA, USA, v. 25, n. 5-6, p. 403–429, 2006. Available from Internet: <http://ijr.sagepub.com/content/25/5-6/403.abstract>. Accessed in: 28 apr. 2017

TOMONO, M. Robust 3d slam with a stereo camera based on an edge-point icp algorithm. In: **Robotics and Automation, 2009. ICRA '09. IEEE International Conference on**. Piscataway, NJ, USA: IEEE Press, 2009. p. 4306–4311. ISSN 1050-4729.

TREVISAN, M. et al. Exploratory navigation based on dynamical boundary value problems. **J. Intell. Robotics Syst.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 45, n. 2, p. 101–114, feb 2006. ISSN 0921-0296. Available from Internet: <http://dx.doi.org/10.1007/s10846-005-9008-2>. Accessed in: 28 apr. 2017.

TSAI, Y.-C.; HUANG, H.-P. Motion planning of a dual-arm mobile robot in the configuration-time space. In: **Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on**. Piscataway, NJ, USA: IEEE Press, 2009. p. 2458–2463.

TUNGADI, F.; KLEEMAN, L. Loop exploration for slam with fusion of advanced sonar features and laser polar scan matching. In: **Proceedings of IEEE/RSJ IROS**. Piscataway, NJ, USA: IEEE Press, 2009. p. 388–394.

VALLVÉ, J.; ANDRADE-CETTO, J. Potential information fields for mobile robot exploration. **Robotics and Autonomous Systems**, v. 69, p. 68 – 79, 2015. ISSN 0921-8890. Selected papers from 6th European Conference on Mobile Robots. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0921889014001675>. Accessed in: 28 apr. 2017.

VALLVé, J.; ANDRADE-CETTO, J. Potential information fields for mobile robot exploration. **Robotics and Autonomous Systems**, 2014. ISSN 0921-8890.

VOTH, D. A new generation of military robots. **Intelligent Systems, IEEE**, v. 19, n. 4, p. 2–3, Jul 2004. ISSN 1541-1672.

WILCOXON, F. Individual comparisons by ranking methods. **Biometrics Bulletin**, International Biometric Society, v. 1, n. 6, p. pp. 80–83, 1945. ISSN 00994987.

WILSON, E. B.; HILFERTY, M. M. The distribution of chi-square. In: **Proceedings of the National Academy of Sciences of the United States of America**. 1931. v. 17 (12), p. 684–688.

YAMAUCHI, B. A frontier-based approach for autonomous exploration. In: **Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on**. IEEE Press, 1997. p. 146–151.

YAMAUCHI, B. M. **PackBot: a versatile platform for military robotics**. 2004. 228-237 p. Available from Internet: <http://dx.doi.org/10.1117/12.538328>.

ZELEK, J. S. A framework for mobile robot concurrent path planning and execution in incomplete and uncertain environments. In: **Proceedings of the AIPS-98 Workshop on Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments, Pittsburgh, PA**. 1998. p. 931–946.

ZHANG, Y.-P. et al. Resolving local minima problem of potential field. In: **Proceedings of the 9th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry**. New York, NY, USA: ACM, 2010. (VRCAI '10), p. 339–346. ISBN 978-1-4503-0459-7. Available from Internet: <http://doi.acm.org/10.1145/1900179.1900250>.

ZHAO, L.; HUANG, S.; DISSANAYAKE, G. Linear slam: A linear solution to the feature-based and pose graph slam based on submap joining. In: **Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on**. Piscataway, NJ, USA: IEEE Press, 2013. p. 24–30. ISSN 2153-0858.