Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Faculdade de Engenharia

Pedro Jorge de Albuquerque de Oliveira

Localização e Mapeamento Simultâneos Utilizando Otimização por Inteligência de Enxame

Rio de Janeiro
2019

Pedro Jorge de Albuquerque de Oliveira

Localização e Mapeamento Simultâneos Utilizando

Otimização por Inteligência de Enxame

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Orientadora: Prof.ª Dr.ª Nadia Nedjah
Orientadora: Prof.ª Dr.ª Luiza de Macedo Mourelle

Rio de Janeiro
2019

CATALOGAÇÃO NA FONTE
UERJ / REDE SIRIUS / BIBLIOTECA CTC / B

O48 Oliveira, Pedro Jorge de Albuquerque de.
Localização e mapeamento simultâneos utilizando otimização por inteligência de enxame / Pedro Jorge de Albuquerque de Oliveira. – 2019.
156f.

Orientadora: Nadia Nedjah.
Orientadora: Luiza de Macedo Mourelle.
Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Engenharia eletrônica – Teses. 2. Inteligência coletiva – Teses. 3. Robótica – Teses. 4. Mapeamento digital – Teses. I. Nedjah, Nadia. II. Mourelle, Luiza de Macedo. III. Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia. IV. Título.

CDU 007.52

Assinatura

Data

Pedro Jorge de Albuquerque de Oliveira

Localização e Mapeamento Simultâneos Utilizando

Otimização por Inteligência de Enxame

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Aprovado em: 18 de Janeiro de 2019

Banca Examinadora:

Prof.ª Dr.ª Nadia Nedjah (Orientadora)
Faculdade de Engenharia - UERJ

Prof.ª Dr.ª Luiza de Macedo Mourelle (Orientadora)
Faculdade de Engenharia - UERJ

Prof. Dr. Pedro Carlos da Silva Lara
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca - CEFET-RJ

Prof.ª Dr.ª Rosa Maria Esteves Moreira da Costa
Instituto de Matemática e Estatística - UERJ

Rio de Janeiro
2019

DEDICATÓRIA

Dedico este trabalho à minha mãe, por acreditar em mim, mesmo quando eu não acreditei em mim mesmo, e por sempre me apoiar e ajudar, mesmo nos momentos mais difíceis. Dedico também aos meus avós por terem sempre estado presentes e dispostos a ajudar.

AGRADECIMENTOS

Agradeço a minha orientadora Nadia Nedjah por ter me incentivado a realizar esse curso de mestrado, pela perseverança, e pela enorme quantidade de sugestões e críticas sem as quais o presente trabalho não teria chegado a termo em seu presente estado. Agradeço à professora Luiza de Macedo Mourelle pelas aulas, pelas conversas e pelas sugestões fornecidas para o aprimoramento do texto da presente dissertação. Agradeço também aos professores do PEL-UERJ pelas lições aprendidas nestes dois anos de curso.

Agradeço aos colegas com quem convivi durante esse mestrado: Alexandre, Joelmir, Ramon e Luneque. Obrigado pelas conversas relacionadas com o projeto, pelo auxílio com as dificuldade advindas do uso da LATEX, pelo apoio prestado e pelos momentos de descontração que me ajudaram a perseverar.

Agradeço a CAPES pelo apoio econômico fornecido durante o curso de mestrado, sem o qual não poderia ter desenvolvido este trabalho.

Se um homem não sabe para que porto navega, nenhum vento é favorável.

Lucius Annaeus Seneca (Carta 71 *de Seneca a Lucílio)*

RESUMO

OLIVEIRA, Pedro Jorge de Albuquerque de. *Localização e mapeamento simultâneos utilizando otimização por inteligência de enxame*. 2019. 156f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2019.

O problema de localização e mapeamento simultâneos é de fundamental importância tanto por suas aplicações em potencial para a construção de sistemas robóticos autônomos quanto por seu mérito próprio. De fato, a capacidade de adquirir mapas de regiões desconhecidas por meio de um processo exploratório tem, por exemplo, o potencial de alterar o modo de agir de equipes de resgate e operação em situações de emergência a tal ponto que pode ensejar a construção de equipes robóticas de resgate, que operem de modo totalmente independente, num futuro próximo. Uma aplicação mais imediata seria a ampliação da capacidade de adaptação de robôs industriais, que operam em ambientes que são, majoritariamente, conhecidos *a priori* e estáticos apesar de estarem sujeitos a pequenas alterações como o reposicionamento de obstáculos móveis. Nessa dissertação, apresentamos uma solução para o problema de localização e mapeamento simultâneos que se baseia na otimização por inteligência de enxame. Muito embora diversos trabalhos apliquem métodos clássicos de otimização tais como os quasi-Newton e afins, o uso de técnicas de otimização bio-inspiradas e livres de derivadas foi relativamente pouco explorado nesse campo. Dito isso, escolhemos três técnicas de otimização por inteligência de enxame, quais sejam: enxame de partículas, colônia artificial de abelhas e o algoritmo inspirado nos vaga-lumes . Além disso, e em consonância com o objetivo de construir uma solução eficiente para o problema de localização e mapeamento simultâneos, fizemos uso do conceito de grafos de poses para construir uma solução robusta. Os sistemas de localização e mapeamento simultâneos que são baseados em grafos de poses estão, hodiernamente, no estado da arte e acreditamos que o desenvolvimento de sistemas de processamento de varreduras robustos é de extremo valor para o avanço dessa área da robótica. Concluímos por meio dos experimentos que o uso da técnica de otimização por enxame de abelhas forneceu os resultados mais acurados dentro do escopo da estimação *online*. Essa constatação pode ser verificada pelo fato de que, nos melhores casos, obtivemos ganhos de acurácia entre 12% e 88% na estimativa das componentes translacionais da trajetória do robô em relação a sistemas do estado da arte pelo uso do algoritmo de otimização por colônia artificial de abelhas. O método de otimização inspirado no comportamento de vaga-lumes não obteve resultados tão precisos quanto os do algoritmo de otimização por colônia artificial de abelhas, entretanto em 7 dos 8 conjuntos de dados de domínio público que utilizamos ele foi o método mais veloz, consumindo, em média, apenas 23% do tempo gasto pela otimização por colônia artificial de abelhas por varredura. A otimização por enxame de partículas apresentou uma precisão inferior que a otimização por colônia artificial de abelhas, no entanto, seu consumo de tempo foi intermediário em comparação às duas outras meta-heurísticas empregadas.

Palavras-chave: Localização e Mapeamento Simultâneos; Robótica; Inteligência de enxame.

ABSTRACT

OLIVEIRA, Pedro Jorge de Albuquerque de. *Simultaneous localization and mapping using swarm intelligence optimization.* 2019. 156f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2019.

The problem known in the robotics community as simultaneous localization and mapping is of fundamental importance both in its own right and because of its potential applications in the development of autonomous robots. Indeed, the ability to acquire maps of an unknown environment through exploration has, for example, the potential to completely change the modus operandi of first responders and rescue forces in general, so much so that it may even hold the seeds for the development of completely independent robotic emergency response teams. An application that is more immediate is the adaptation of robotic systems to changes in environments that are for the most part static and known beforehand, such as the operating loci of industrial robots that may be changed at any instant by events such as the placement of movable obstacles. This dissertation presents an approach to solve the simultaneous localization and mapping problem that is based on swarm intelligence optimization methods. While many solutions exist that are based on classical Newton-like optimization techniques, relatively little work has been done with respect to the application of derivative free bioinspired techniques to this particular area of robotics. That being said, we have chosen three techniques to be the subjects of our inquiry, namely particle swarm optimization, artificial bee colony and the firefly algorithm . Furthermore, and in keeping with the intention of creating an effective solution to the problem at hand, we have made use of a pose graph based approach as a means of maintaining the consistency of our estimates across the mapping process. Systems designed to perform SLAM using pose graphs are currently the state of the art and it is our belief that a robust scan matching system is currently of the utmost importance to further the field. Throught the development of this thesis we have concluded that the ABC technique has great potential, having tested it and found it is both fast and efficient in a great range of circumstances. This affirmation is backed by the fact that in the best case scenarios we have obtained accuracy gains in between 12% and 88% regarding the translational estimates of the robot's trajectory, by using the ABC metaheuristic, when compared to state of the art techniques. The firefly algorithm, while not as accurate as the artificial bee colony technique, was faster than the aforementioned bee inspired metaheuristic on 7 out of the 8 public domain datasets. The firefly algorithm consumed, on average, only 23% of the time spent per scan processed by the artificial bee colony optimization technique. Particle swarm optimization has shown an inferior accuracy when compared to the artificial bee colony optimization technique and an intermediate processing time when compared to the other two optimization methods.

Keywords: Simultaneous Localization And Mapping; Robotics; Swarm intelligence

LISTA DE FIGURAS

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

SUMÁRIO

SUMÁRIO

INTRODUÇÃO

A localização e o mapeamento simultâneos (*Simultaneous Localization And Mapping* – SLAM) tem sido considerado como um dos pilares para a construção de sistemas robóticos autônomos (DURRANT-WHYTE; BAILEY, 2006). Essa área de estudo visa desenvolver técnicas que possibilitem a um ente robótico explorar um ambiente que seja inicialmente desconhecido e estimar de modo geminado sua posição e uma representação do ambiente onde o próprio se encontra. O problema de SLAM consiste da determinação simultânea do mapa de um ambiente e da trajetória percorrida pelo ente que faz o mapeamento. À primeira vista, isso aparenta ser um paradoxo, já que se por um lado para que seja possível que qualquer entidade se localize em um ambiente essa deve possuir um mapa preciso o suficiente, por outro lado mapear um ambiente faz necessário conhecer a trajetória realizada por uma entidade que a isso se proponha. Os sensores podem ser de diversos gêneros, sendo os mais usuais: câmeras, bússolas, GPS, *range finders* baseados em tecnologia *laser*, radares e sonares.

Um ponto importante é que o SLAM possui uma miríade de aplicações práticas em potencial, tendo em vista que possibilita o uso de robôs em locais, e situações, que seriam inadequados, ou demasiadamente perigosos, para agentes humanos. Em (THRUN et al., 2003), por exemplo, os autores relatam ter obtido sucesso na aplicação de um sistema autônomo direcionado à exploração de minas abandonadas. Outro exemplo de aplicação é o uso de sistemas robóticos na realização de missões de busca e resgate. Esse tipo de utilização é vislumbrado, por exemplo, em (KLEINER; PREDIGER; NEBEL, 2006) onde os autores discorrem sobre uma técnica para a exploração autônoma de regiões de desastre baseada em etiquetas RFID (*Radio-Frequency IDentification*) que permite a cooperação de robôs por meio dos dados deixados nessas etiquetas.

O desenvolvimento do campo de mapeamento e localização simultâneos é considerado fundamental para a emergência de sistemas robóticos verdadeiramente autônomos. De fato, a aquisição de tal competência tornaria possível que esses sistemas operassem

em ambientes pouco estruturados com pouca, ou nenhuma, intervenção humana no que toca à navegação do ente robótico pelo local onde se encontra. Dentre esses métodos, aqueles que aplicam o conceito de grafos de poses e o alinhamento de varreduras são os que compartilham um relacionamento mais próximo com o que propomos.

Os métodos que aplicam *scan matching* reduzem o problema de localização à uma tarefa de *pattern matching*. Um ente que possua um conjunto de observações do ambiente feitas em instantes sucessivos de tempo torna-se capaz de determinar, ao menos aproximadamente, o seu deslocamento em relação ao ambiente e, consequentemente, sua pose pela obtenção de uma composição de transformações de translação e rotação que levem a uma correspondência entre a observação feita da pose atual e aquelas que fizera anteriormente. Esse tipo de método possui a vantagem de ser, ao menos em princípio, capaz de resolver o problema de localização global. Evidentemente, e como ficará claro ao longo da dissertação, a simples estimativa de poses por meio da composição de transformações relativas entre poses subsequentes está sujeita a um acúmulo de erros, similar ao que se observa em sistemas de carácter integrador, tal qual os que utilizam exclusivamente dados odométricos.

No célebre trabalho (LU; MILIOS, 1997), os autores descrevem uma abordagem baseada em otimização cujo cerne é o armazenamento de todos os *frames* locais de dados além de uma rede de relações espaciais entre eles. Os autores adotaram a definição de que cada *frame* é o conjunto de medições obtidas a partir de uma única pose. As relações espaciais supracitadas, são derivadas pelo uso do alinhamento de varreduras entre pares de *frames*, ou pelo uso de informações odométricas. Por fim, trata-se o problema de determinar todas as poses como uma tarefa de otimização com restrições. A ideia de determinar simultaneamente todas as poses visa garantir a consistência entre todos os *frames* locais.

A construção de um mapa preciso poderia, em princípio, ser feita mediante a disponibilidade de informação de sensores que permitissem ao ente robótico localizar-se em relação a *algum referencial*. Deveras, o uso de dados provenientes de sistemas de GPS, de bússolas ou de medições odométricas trás consigo o potencial de introduzir informação suficiente para que se resolva o problema de SLAM. A presença de ruído, entretanto, juntamente com limitações impostas pelo ambiente, torna inviável a aplicação direta desse tipo de mensuração. O caso do GPS talvez seja o mais evidente, haja vista que em ambi-

entes fechados ou subterrâneos a comunicação com satélites pode ser comprometida, ou mesmo impossibilitada. Bússolas, por sua vez, são extremamente imprecisas na presença de campos magnéticos suficientemente fortes, que podem ser gerados pela presença de linhas de transmissão ou até mesmo por outros condutores energizados que se encontrem nas cercanias, tal qual dita a lei de Ampère-Maxwell. E embora pareça possível utilizar a informação odométrica, tipicamente disponível em sistemas robóticos móveis, para determinar o deslocamento *relativo* ao longo do processo, a sua aplicação direta leva a erros grosseiros.

Do exposto conclui-se que é necessário obter outra fonte de informação para que seja possível corrigir os erros de medição. Para esse fim, os métodos de SLAM , de um modo geral, incorporam dados obtidos por meio de mais de um tipo de sensor, além de informações exógenas na forma de modelos e hipóteses de modelagem, para tentar estimar a verdadeira *pose* do sistema e obter um mapa mais fidedigno do ambiente.

O SLAM pode ser visto como uma situação de inferência e estimação bayesiana simultâneas (FOX; THRUN; BURGARD, 2005). Isso, pois dadas as propriedades dos sensores utilizados, as observações feitas e as ordens dadas ao robô utilizado, aprimora-se o mapa *ad hoc* em uso, tornando-o mais preciso. Por outro lado, dado o mapa mais preciso construído até o momento, as informações dos sensores e as ordens dadas estima-se a posição do robô nesse mapa. A dificuldade do problema de SLAM advém do fato de a tarefa de localização e a de mapeamento estarem intrinsecamente conjugadas.

Em geral, as abordagens mais tradicionais para o tratamento do problema, ora proposto, são baseadas em métodos estatísticos recursivos de estimação de estado. Dentre esses diversas aproximações dos filtros recursivos de Bayes, tais como aquelas de Kalman e suas derivações ou os filtros de partículas com suas várias extensões, levam à soluções iterativas do problema (DURRANT-WHYTE; BAILEY, 2006), que são frequentemente utilizadas quando objetiva-se construir uma representação do mapa com base na posição relativa de certos marcadores (*landmarks*) no espaço.

Os métodos de resolução para o problema de SLAM que se baseiam em grafos se valem de uma formulação do problema em termos de um grafo e tratam o problema de estimação de estado como um problema de otimização. Os membros dessa família de soluções tratam, em geral, do problema de *full SLAM*, em que o objetivo é a estimação da completa trajetória percorrida pelo ente que realiza a tarefa de mapeamento além da

determinação da pose atual e do mapa do ambiente. Apesar disso, diversas variantes capazes de realizar uma otimização incremental, e consequentemente de operar online, foram desenvolvidas ao longo dos anos. Essa sorte de técnicas tende a produzir estimativas mais consistentes do que aquelas que se baseiam em técnicas de filtragem estatística. Isso ocorre pois, técnicas como filtros de Kalman, filtros de partículas, filtros de informação e seus congêneres estão inerentemente sujeitos a inconsistências advindas de erros de linearização ocasionados por estimativas erradas, ou , ao menos, pouco precisas, de poses passadas. O ponto chave é que as técnicas baseadas em filtragem perpetuam esses erros em virtude do modo como operam. Em contraste, as técnicas que se valem de grafos, por vezes denominadas de *smoothers*, são capazes, em virtude de sua solução do problema de *full SLAM*, de revisar estimativas para toda a trajetória baseados em novas evidências produzindo, por conseguinte, resultados de maior acurácia e consistência. Por essa razão, e devido a avanços nos campos da algebra linear concernentes à solução de sistemas lineares esparsos, essa classe de métodos de SLAM tem passado por um "renascimento" e se encontram hodiernamente entre as técnicas consideradas estado da arte. A maior parte dos algoritmos dessa classe funcionam com base em dois componentes principais e complementares: um que é responsável por todo o processo de extração de pontos de interesse e alinhamento de varreduras, e o que diversos autores denominam *back-end*. Ao *front-end* cabe extrair restrições entre as variáveis do problema das informações sensoriais, ou seja realizar todo o processo de alinhamento de varreduras. Ao *back-end* resta a tarefa de otimizar o conjunto das restrições, *i.e.* encontrar a configuração das variáveis do problema que induza o menor erro possível. Que fique claro o significado atribuído à palavra *restrição* no contexto dos algoritmos de SLAM baseados em grafos, que são conjuntos de relações geométricas sob a forma de transformações afins que relacionam poses ou observações. No caso de relações entre poses há tanto restrições derivadas de odometria, aprimoradas ou não pelo uso do alinhamento de varreduras), quanto restrições impostas por observações. As restrições impostas por observação são criadas quando se observa o mesmo local de duas poses diferentes. É nosso objetivo criar um *frontend* baseado em técnicas de inteligência de enxame.

Os algoritmos de otimização bio-inspirados são técnicas cujo funcionamento se baseia em modelos de sistemas naturais, tais como a seleção das espécies, a dinâmica de enxames de pássaros e o comportamento de insetos sociais. Alguns exemplos bastante

conhecidos são os algoritmos genéticos, a otimização por enxame de partículas (*Particle Swarm Optimization* – PSO), a otimização por colônias de formigas (*Ant Colony Optimization* – ACO), algoritmo de otimização baseado no comportamento dos vagalumes (*Firefly Algorithm*) e o algoritmo de otimização por colônia artificial de abelhas (*Artificial Bee Colony – ABC*).

Embora haja exemplos na literatura de aplicação desse tipo de técnica de otimização como parte de métodos destinados à solução do problema de SLAM, sua aplicação aliada a uma abordagem calcada no princípio da máxima verossimilhança foi, aparentemente, pouco explorada. Com isso em mente, propomo-nos, no presente trabalho, a estudar, e comparar, a eficácia de várias técnicas bio-inspiradas de otimização no que tange a sua aplicação à solução do problema de SLAM. No Capítulo 1 define-se o problema de SLAM e o alinhamento de varreduras, e explica-se o conceito de grafo de poses. Descreve-se, também, o problema de correspondência e trata-se da importância da detecção de retornos a locais previamente visitados para sistemas de SLAM que usam grafos de poses.

No Capítulo 2 encontram-se elencados trabalhos relacionados aos temas comumente associados ao SLAM, e mais especificamente à presente dissertação. Esse capítulo, por conseguinte, se encontra dividido em subseções que tratam, respectivamente, de métodos de solução baseados em filtros estocásticos, e do uso de técnicas de otimização nos métodos de SLAM.

O Capítulo 3 trata da fundamentação teórica subjacente ao uso de sistemas de coordenadas projetivas, à definição de pose, e das operações usuais a elas aplicadas. O capítulo se encerra definindo o que são grafos de poses e como esses são otimizados.

O Capítulo 4 trata do sistema de alinhamento de varreduras, dos mapas de ocupação a e do modo como são detectados os retornos a locais já visitados. É objetivo desse capítulo descrever as técnicas que foram aplicadas num grau de profundidade suficiente para que sirva de base para a compreensão da implementação realizada.

O Capítulo 5 trata da inteligência de enxame, mais especificamente das técnicas de otimização pertencentes à essa classe que escolhemos aplicar no presente trabalho, a saber: o PSO, o FA e o ABC. Nesse capítulo são apresentados os algoritmos e comentados os métodos de otimização.

O Capítulo 6 é feita uma descrição geral do sistema proposto. Em seguida é descrito e explicado o sistema de alinhamento de varreduras. Ademais, descreve-se o sistema de manipulação de grafos de poses.O Capítulo 6 traz, também, uma descrição do método empregado para a detecção de retornos a locais que se visitara. O Capítulo 6 se encerra dando as características gerais do robô P3DX.

O Capítulo 7 descreve e comenta os experimentos que utilizamos para validar a abordagem que desenvolvemos. No decorrer desse capítulo, por meio de características intrínsecas aos conjuntos de dados exporemos e discutiremos as respostas características do sistema desenvolvido.

O Capítulo 8 é a conclusão dessa dissertação, onde é examinado, em termos gerais, o que foi realizado. Ainda nesse capítulo discutem-se as possíveis ramificações do presente trabalho e os possíveis trabalhos futuros.

Capítulo 1

LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS

ESTE capítulo define o problema de SLAM e o alinhamento de varreduras que é a base da abordagem proposta. A Seção 1.1 define o problema de SLAM. A Seção 1.2 trata do alinhamento de varreduras. A Seção 1.3 explica o conceito de grafos de poses. A Seção 1.4 descreve o problema de correspondência e trata da importância da detecção de retornos a locais previamente visitados para sistemas de SLAM que usam grafos de poses.

1.1 Descrição Geral

O problema de SLAM lida com as tarefas geminadas de estimação da trajetória de um agente móvel e do mapa do ambiente por onde ele se locomove. Esse problema surge quando um robô não tem acesso a um mapa do ambiente nem à sua própria posição e orientação. Objetivando, por meio da análise de suas medições e dos comandos que lhe foram dados, construir um mapa do ambiente em que navega e localizar-se relativamente a esse mapa. O mapa pode apresentar-se de múltiplas formas, tais como mapas de marcadores (*feature maps*) (SMITH; SELF; CHEESEMAN, 1990; MONTEMERLO et al., 2002), mapas de ocupação (*Occupancy Grid Maps*) (ELFES, 1990; GRISETTI; STACHNISS; BURGARD, 2007; GIL; JULIÁ; REINOSO, 2015), nuvens de pontos (*Point Clouds*) (COLE; NEWMAN, 2006; FIORAIO; KONOLIGE, 2011) ou formulações topológicas (KÜMMERLE et al., 2011; KAESS; RANGANATHAN; DELLAERT, 2008; DELLAERT, 2012).

As formas de representação de mapas citadas podem ser descritas de modo simples. Os mapas de marcadores são compostos por listas das posições de marcadores pré-definidos. Os mapas de ocupação, entretanto, dividem o ambiente em um conjunto

de células (discretização) e a cada célula é atribuído um valor correspondendo à sua probabilidade de estar ocupada, ou seja de conter um obstáculo. Os mapas de nuvens de pontos, por sua vez, são compostos por conjuntos de pontos obtidos por sensoriamento do ambiente e representam-no por um conjunto de amostras de regiões ocupadas. Já os mapas topológicos consistem de um conjunto de vértices e arestas na forma de um grafo. Cada vértice representa um local específico e cada aresta representa o fato de que há um caminho entre dois nós.

O SLAM recebeu grande atenção nas últimas décadas por parte de pesquisadores nas áreas de robótica, inteligência artificial, visão computacional e correlatas. Grande parte da razão por trás desse interesse se justifica pelo papel central que uma solução eficiente para o SLAM teria para a construção de sistemas robóticos verdadeiramente autônomos (FOX; THRUN; BURGARD, 2005). Com efeito, o SLAM é essencial para que se possa aplicar sistemas robóticos a ambientes sobre os quais não se possua informação cartográfica, a ambientes em que não se tenha acesso a sistemas de localização externos como sistemas de GPS a tarefas que se deseje completar sem a intervenção humana ou mesmo a ambientes sujeitos a modificações (mesmo que modicas).

O SLAM pode ser visto como uma situação de inferência e estimação Bayesiana simultâneas (FOX; THRUN; BURGARD, 2005), em que o robô adquire um mapa do seu ambiente enquanto se localiza em relação a esse mapa. De um modo geral, as formulações probabilísticas do SLAM veem o problema como uma tarefa de estimação na qual as variáveis são as componentes da pose do robô e o mapa. Um exemplo disso é a solução do problema por meio de filtros de Kalman (JIANG; LI; YU, 2016; PATHIRANAGE; WATANABE; IZUMI, 2007; CHATTERJEE; MATSUNO, 2010) com mapas de marcadores em ambientes bidimensionais, na qual se mantém a pose atual (x, y e θ) e um vetor com as posições dos marcadores observados como variáveis do problema. O SLAM é um problema cuja solução é significativamente mais difícil do que as tarefas de mapeamento e de localização realizadas independentemente (FOX; THRUN; BURGARD, 2005). A dificuldade do SLAM é maior que a do problema de localização já que o mapa é desconhecido e deve ser estimado ao longo do processo. Dir-se-á, também, ser o SLAM mais difícil que o problema de mapeamento com poses conhecidas, haja vista o fato de as poses serem também desconhecidas, o que não só aumenta a dimensionalidade do problema de inferência, mas também acresce à incerteza associada às estimativas realizadas. O SLAM pode ser

formulado de duas maneiras com igual importância prática: *online* e *offline*. A primeira formulação caracteriza-se por buscar distribuição de probabilidade *a posteriori* instantânea da pose e do mapa. Essa formulação é denominada *online* já que envolve apenas a estimação de variáveis que persistem no presente instante de tempo. É interessante notar que muitos algoritmos dedicados ao tratamento do problema de SLAM *online* são incrementais, descartando medições e comandos pregressos a cada passo do seu funcionamento (MONTEMERLO et al., 2002; GRISETTI; STACHNISS; BURGARD, 2007; ZHIRONG; ZIXING; BAIFAN, 2011). A segunda formulação, denotada *offline*, lida com o caso de tratamento dos dados completos em virtude de se realizar posteriormente à coleta de dados, vide (THRUN et al., 2004; WALTER; EUSTICE; LEONARD, 2007; THRUN; MONTEMERLO, 2006). De um modo geral as abordagens *offline* tratam do problema de SLAM completo, muito embora haja atualmente sistemas que são capazes de lidar com o SLAM completo *online*. No SLAM completo busca-se calcular a distribuição *a posteriori* ao longo de toda a trajetória juntamente com o mapa, ao invés de buscar apenas a pose atual. Uma importante observação é a de que sistemas que possuam acesso a toda a trajetória estimada e ao conjunto completo de medições são capazes, ao menos em princípio, de rever estimativas passadas, corrigindo-as conforme tal coisa se mostre necessária à luz das informações disponíveis. Deveras, é fundamental que se note a importância da afirmação anterior, já que sistemas incrementais, por seu funcionamento, podem ser incapazes de recuperar-se de erros cometidos ao longo do processo e, por perpetuá-los, serem ineptos para a aplicação em ambientes com grande ambiguidade morfológica. Os sistemas que se valem de formulações que empregam grafos de poses como forma de garantir a consistência global do mapa construído, embora sensíveis a falsas associações tais como o reconhecimento incorreto de locais ou marcadores já vistos, são capazes de resolver o problema de SLAM completo. Consequentemente, estes podem recuperar-se de erros de estimação cometidos ao longo do processo por meio da revisão das hipóteses de associação extemporâneas (ADAMS, 2007; KAESS; RANGANATHAN; DELLAERT, 2008; KÜMMERLE et al., 2011; DELLAERT, 2012; HESS et al., 2016).

Uma característica chave do SLAM advém da composição do problema de estimação. De fato, o SLAM possui uma componente contínua e uma componente discreta. O problema de estimação contínua é a determinação do local onde se encontram objetos no mapa e da pose do robô. Os objetos citados podem ser marcadores ou regiões de interesse

em um mapa de ocupação detectados por meio de *laser range finders*, de sonares ou mesmo de câmeras. A componente discreta deve-se ao problema de correspondência. A cada vez que um objeto é detectado é imperativo que seja realizado um processo de inferência sobre a relação desse objeto para com aqueles que se tenha detectado previamente. Essa inferência é tipicamente um processo discreto, uma vez que ou o objeto é o mesmo ou não o é, sendo possível enumerar todos os possíveis pareamentos, recentemente foram publicados alguns trabalhos que tratam especificamente de técnicas para a associação de varreduras lidar e da localização de robôs baseada nelas como em (KALLASI; RIZZINI; CASELLI, 2016; TIPALDI; BRAUN; ARRAS, 2014). Muito embora a estimação da completa distribuição a posteriori, isso é a distribuição de probabilidade que captura todas as variáveis do problema, seja o objetivo mais almejável, tal coisa é usualmente impraticável. A dificuldade de se estimar a distribuição completa se deve à alta dimensionalidade do problema, que pode chegar a ter 10^5 variáveis ou mais. Isso é de enorme contraste, por exemplo ao problema de localização em que o espaço de busca é continuo e, geralmente, tridimensional, sendo compreendido pela posição e orientação do robô. Ademais, na maior parte dos casos, o problema de correspondência precisa ser solucionado, já que não se conhece, *a priori*, a relação entre as observações. Com efeito, o tamanho do espaço de buscas para o problema de correspondência cresce exponencialmente em relação ao tempo de exploração, e à quantidade de observações feitas. Por conseguinte, soluções para o SLAM tratam do problema de correspondência com base em aproximações.

Ao longo dos anos, uma miríade de formulações para o problema foi proposta. Os motivos por trás dessa abundância de formulações é, em grande parte, devida ao desejo de maximizar a precisão e eficiência em uma aplicação particular. Nesse trabalho, é proposto um método de resolução para o SLAM baseado num procedimento de alinhamento de varreduras calcado nas meta-heurísticas de otimização conhecidas como: otimização por colônia artificial de abelhas (*Artificial Bee Colony* – ABC), otimização dos vagalumes (*Firefly Algorithm* – FA) e otimização por enxame de partículas (*Particle Swarm Optimization* – PSO). O uso do alinhamento de varreduras, aliado a um sistema otimização de grafos de poses permite que sejam construídos mapas consistentes de grande extensão.

1.2 Alinhamento de Varreduras

O processo de construção de mapas locais é realizado com dois fins essenciais: a determinação da relação espacial entre dois referenciais com base na medição realizada e o fornecimento de subsídio ao procedimento de detecção de retorno a locais que se visitara anteriormente. Em geral, cada varredura é alinhada à sua predecessora imediata e assim são construídos pequenos mapas locais que permitem a determinação da localização quando essa é alterada por pequenos movimentos. A segunda questão diz respeito à capacidade de determinação da relação entre referenciais, tendo-se acesso às varreduras obtidas destes. Com efeito, cada varredura, e por extensão cada conjunto de varreduras obtidas a partir de poses próximas, pode ser considerada uma assinatura do seu referencial, já que por meio dela é possível quantificar a similaridade entre dois referenciais, usando o alinhamento entre as respectivas varreduras. A determinação de relações entre referenciais que não sejam imediatamente subsequentes traz consigo a informação necessária para que sejam construídos mapas globalmente consistentes. Note-se que a consistência local não é um problema acerca do qual haja preocupação tendo em vista que essa pode ser facilmente mantida pois o acúmulo de erros de alinhamento tende a ser pequeno em pequenos trajetos, e em especial em trajetos que sejam aproximadamente retilíneos. Deveras, o mapeamento de ambientes mediante um processo exploratório que se valha de trajetórias retilíneas, ou de trajetórias que possuam pequena curvatura, tende a ser menos problemático do ponto de vista do crescimento da incerteza que o sistema de SLAM possui acerca da posição e da orientação do robô (FOX; THRUN; BURGARD, 2005).

1.3 Grafos de Poses

Os métodos de resolução para o problema de SLAM que se baseiam em grafos de poses se valem de uma formulação do problema de SLAM em termos de um grafo de relações entre diversos referenciais em que o robô se encontrou ao longo do tempo e tratam o problema de estimação de estado como um problema de otimização (LU; MILIOS, 1997; KAESS; RANGANATHAN; DELLAERT, 2008; GRISETTI et al., 2010; KÜMMERLE et al., 2011). Essa categoria membros dessa família de soluções, em geral, realizam a estimação da completa trajetória percorrida pelo agente móvel além da determinação da pose atual e do mapa do ambiente. A nomenclatura grafo de poses se deve ao fato de o sistema que forma a base do

processo de estimação construir um grafo cujos nós são as poses, formadas pela localização e orientação do robô, e cujas arestas são as relações métricas entre essas poses, *i.e.* as transformações que levam ao alinhamento das observações feitas entre esses referenciais. Em verdade, esses métodos produzem grafos com configurações que minimizem o erro incorrido pela posição estimada para os nós mediante todas as restrições entre elas. Com efeito, de posse da trajetória e das varreduras, o problema de SLAM pode ser considerado resolvido tendo em vista que a produção de um mapa métrico nada mais é do que um problema de mapeamento com poses conhecidas.

1.4 Detecção de Retornos e o Problema de Correspondência

Considerando-se que o sistema pode determinar a relação entre informações exteroceptivas e usar esse dado para corrigir erros cometidos por inferências baseadas apenas em dados de origem proprioceptiva resta que se defina quais pares de varreduras há que se testar em busca de relações. De fato, é impraticável que se teste todas as possíveis combinações já que tal conjunto cresceria demasiadamente, e de modo muito veloz, ao longo do processo de mapeamento. Uma heurística que pode ser aplicada é a de que poses que ocorreram dentro de um intervalo curto de tempo estejam relacionadas. Consequentemente, é razoável que se busque relações entre varreduras coletadas em instantes de tempo próximos. Isso melhora a consistência local do sistema, mas é incapaz de corrigir erros que transcendam pequenos períodos de tempo. Um exemplo de ambiente em que esse tipo de sistema falha é o conjunto de dados INTEL que usamos como validação para nosso sistema e que é descrito em maiores detalhes na Seção 6.4. O tamanho de alguns trechos cíclicos do percurso do conjunto de dados supramencionado faz com que pequenos erros de alinhamento ao final dos trechos contidos em pequenas janelas temporais não sejam notados mesmo que esses afetem de modo inequívoco o mapa global. Isso leva à aplicação de buscas por relações métricas entre possíveis alinhamentos extemporâneos. A ideia por trás de tal busca é obter restrições que liguem trechos que estejam espacialmente próximos, mesmo que tenham sido coletados em instantes de tempo separados por intervalos de tempo que podem não cair em uma mesma janela temporal. Uma heurística muito usada é a busca de retorno a locais que se visitara , também chamados de fechamentos de laço, em regiões que caiam dentro de uma elipse de covariâncias, cujo grau de incerteza é um parâmetro

de projeto, da distribuição de probabilidades referente a pose atual. Tal procedimento possibilita que se elimine grande parte das possíveis combinações de poses, entretanto permite, diferentemente do sistema de janelas temporais, que se encontre pareamentos entre referenciais separados por enormes intervalos de tempo. A determinação de relações entre varreduras obtidas desse modo permite que se propague as correções para todo o sistema e contribui fortemente para a elaboração de mapas mais consistentes. Algumas das técnicas mais populares, atualmente, para a lidar com o SLAM são baseadas nas formulações que se valem de grafos de poses. Nas formulações desse tipo, a detecção de um retorno a um local previamente visitado, também conhecido como fechamento de laço, leva à inserção de uma nova aresta entre dois nós que, a princípio, não apresentem conexão direta.

1.5 Considerações Finais do Capítulo

Nesse capítulo definiu-se o problema de localização e mapeamento simultâneo levantando algumas de suas dificuldade. Ademais, foi discutido o papel do alinhamento de varreduras num sistema de SLAM. Também definiu-se o conceito de grafo de poses. Por fim, argumentou-se sobre a necessidade de detectar retornos a locais já visitados e o problema de correspondência. No próximo capítulo serão apresentados alguns trabalhos relacionados à pesquisa realizada.

Capítulo 2

TRABALHOS RELACIONADOS

ESTE capítulo apresenta alguns dos trabalhos relacionados à área da robótica conhecida como localização e mapeamento simultâneos (SLAM). Na Seção 2.1 tratamos dos métodos de solução baseados em filtros estocásticos. Na Seção 2.2 prosseguiremos para o tratamento de técnicas baseadas na aplicação de otimização para nortear o processo. Por fim, na Seção 2.3 faremos uma breve análise do que foi discutido.

2.1 Soluções Baseadas em Filtros Estocásticos

Em geral, as abordagens mais tradicionais para o tratamento do problema ora proposto são baseadas em métodos estatísticos recursivos de estimação de estado. Dentre as diversas aproximações dos filtros recursivos de Bayes, tais como aquelas de Kalman e suas derivações ou os filtros de partículas com suas várias extensões, levam a soluções iterativas do problema (DURRANT-WHYTE; BAILEY, 2006), que são frequentemente utilizadas quando objetiva-se compor um mapa caracterizado pela posição de certos marcadores (*landmarks*) no espaço. Entende-se por estado, nesse problema, o par formado pela pose do robô (ou ente móvel congênere) utilizado para construir o mapa unido ao que se conhece acerca da região explorada.

2.1.1 Filtros de Kalman

Os filtros de Kalman têm por característica distintiva a representação da distribuição *a posteriori* por meio de Gaussianas, que são distribuições unimodais cuja representação pode ser feita de modo compacto usando um número pequeno de parâmetros. Usualmente, técnicas baseadas em filtros de Kalman se valem de marcadores para se localizar no espaço. De fato, o modelo tipicamente usado para o espaço é um vetor de estado com-

pleto, compreendendo a pose do robô e o local dos marcadores (relativamente a algum referencial).

Em (SMITH; CHEESEMAN, 1986; SMITH; SELF; CHEESEMAN, 1990) os autores descrevem os fundamentos da representação e estimação da incerteza no contexto de informações e relações espaciais pertinentes a sistemas robóticos. De fato, os trabalhos em questão contêm a elaboração da fundamentação teórica utilizada na maioria das abordagens probabilísticas à solução de SLAM que os sucederam. Com efeito, os próprios autores descreveram, em (SMITH; SELF; CHEESEMAN, 1990), uma solução baseada em filtros de Kalman. Que esteja clara, também, a restrição de aplicabilidade da abordagem descrita a ambientes com marcadores bem definidos, mesmo que de número desconhecido, *a priori*.

Em (THRUN et al., 2002) o autor descreve o modo como são acomodadas as não linearidades, inerentes ao movimento real, nos filtros de Kalman. O filtro de Kalman resultante é conhecido como filtro estendido de Kalman (*Extended Kalman Filter* - EKF). Neles, tanto o modelo de movimento quanto o modelo de percepção do robô são aproximados usando funções lineares, obtidas por meio de expansões de Taylor. Assim, elimina-se a hipótese de linearidade e substituí-mo-la pela suposição de que os supracitados modelos sejam constituídos por funções diferenciáveis.

Em (MONTEMERLO et al., 2002) fica caracterizado um dos maiores problemas observados na aplicação de filtros estendidos de Kalman à solução do problema de SLAM, a complexidade computacional do método. De fato, os autores afirmam que um passo de atualização de um sistema baseado em EKF leva um tempo proporcional ao quadrado do número de sensores. Fica ressaltado também o fato de ser esse tempo originário do modo interno de representação das incertezas (matriz de covariâncias).

No que tange ao resultado do processo, os filtros de Kalman apresentam a vantagem de estimar a completa distribuição *a posteriori online*, *i.e* é mantido não só o mapa mas também a completa informação acerca da incerteza referente à pose e ao mapa. Note-se que os únicos algoritmos com essa capacidade são os filtros de Kalman e seus derivados, juntamente com os filtros de partículas do tipo Rao-Blackwell. Em relação às fraquezas advindas das hipóteses nas quais se baseiam os filtros de Kalman, a mais severa é a de que o ruído seja Gaussiano. Em verdade, tal hipótese pode levar a falhas práticas em ambientes que apresentem, por exemplo, diversos marcadores de difícil distinção.

Em (PATHIRANAGE; WATANABE; IZUMI, 2007) foi proposto o uso de um modelo *fuzzy* do tipo Takagi-Sugeno aliado a um filtro de Kalman para a solução do problema de SLAM. A ideia consiste do uso de modelos aproximados (locais) para compor um modelo de sistema mais preciso do que seria possível por meio da linearização direta do sistema não linear relacionado. De fato, pelo uso de uma base de regras apropriada, os autores propuseram que o espaço fosse dividido em regiões as quais seriam selecionadas mediante o valor de variáveis como o ângulo da roda e o ângulo de medição. Por meio de experimentos simulados, os autores validaram o método, encontrando resultados que se comparam de modo favorável aos de outros sistemas baseados em filtros de Kalman.

2.1.2 Filtros Estendidos de Informação

Duais dos filtros Estendidos de Kalman, os filtros de informação são algoritmos com operação *offline*. Em virtude de suas características intrínsecas, é um algoritmo melhor adaptado à operação em circunstâncias em que se possa manter todas as informações relevantes na memória e em que se opere sobre todo o corpo de dados (*batch operation*), assumido como sendo de tamanho fixo (FOX; THRUN; BURGARD, 2005).

2.1.3 Filtros de Partículas

Os filtros de partículas aproximam a distribuição de probabilidade *a posteriori* por um número finito de partículas. Conforme dito em (THRUN; BURGARD; FOX, 2000), a ideia central dessa técnica é representar a distribuição *a posteriori* por um conjunto de amostras aleatórias dele. A representação aproximada da distribuição por esse método, ao invés de uma distribuição paramétrica, possibilita que sejam expressos espaços de distribuições estatísticas muito mais variados do que seria possível, por exemplo, se fossem utilizadas apenas Gaussianas (como no filtro de Kalman).

Em (DURRANT-WHYTE; BAILEY, 2006), caracteriza-se a teoria dos filtros de partículas como sendo, em grande parte, baseada numa técnica recursiva de amostragem conhecida como SIS (*Sequential Importance Sampling* - Amostragem Sequencial por Importância). De fato, a implementação de métodos baseados em filtros de partículas tende a ser facilitada pelo fato de quem os implementa não ter a necessidade de efetuar a linearização de modelos não lineares como é o caso com filtros de Kalman. Entretanto, muitas vezes tal linearização é realizada em partes do sistema, como o modelo de percepção,

para tornar o sistema mais veloz. Os filtros de partículas têm por vantagem ainda o fato de ser sua resolução naturalmente adaptativa (THRUN; BURGARD; FOX, 2000), já que os recursos tendem a ser concentrados em regiões mais prováveis do espaço de estados. Isto é, em virtude da amostragem sequencial por importância, regiões com alta probabilidade de ocupação contêm uma concentração maior de amostras. Por outro lado, a eficácia dos filtros de partículas se reduz drasticamente com o aumento da dimensionalidade do problema, já que a densidade de partículas necessária para um funcionamento adequado do sistema não é reduzida. Ou seja, o número de partículas necessário para que haja uma operação satisfatória cresce exponencialmente com o crescimento da dimensionalidade do sistema (na ausência de otimizações) (DURRANT-WHYTE; BAILEY, 2006).

Em (MONTEMERLO et al., 2002), foi proposta uma solução para o problema de complexidade quadrática enfrentado pelo tradicional EKF-SLAM, *i.e* dado um número K de marcadores, o sistema requer um tempo proporcional a K^2 para atualizar as informações armazenadas. Para esse fim, foi tratado o problema de SLAM usando-se como ideia fundamental o fato de que se a trajetória do robô for perfeitamente conhecida, o problema de localizar simultaneamente K marcadores (com localizações interdependentes) ficaria reduzido a K problemas independentes de estimação de um único marcador. Com isso em mente, torna-se possível fatorar a solução do problema. Dessa ideia surgiu o método que os autores denominaram FastSLAM, que numa tradução livre poderia ser expresso como SLAM rápido numa alusão ao ganho de desempenho obtido pela abordagem em relação aos sistemas similares que o precederam. Ademais, o FastSLAM decompõe o problema de SLAM em um problema de localização e em vários problemas de estimação de marcadores. O FastSLAM usa um filtro de partículas de Rao-Blackwell modificado em que cada partícula possui K filtros de Kalman (um para cada marcador) condicionados na trajetória proposta pela partícula. Foi reportada, pelos autores, uma complexidade computacional $O(M \cdot \log K)$, em que M é o número de partículas aplicado, para a implementação baseada em uma estrutura de dados de árvore mencionada no artigo.

Em (GRISETTI; STACHNISS; BURGARD, 2007), foi proposto um sistema batizado de GMapping baseado em filtros de partículas do tipo Rao-Blackwell. É fato notório que diversas implementações de filtros de partículas de Rao-Blackwell trazem consigo um alto consumo de memória, haja vista que cada partícula deve armazenar consigo um mapa do ambiente. É portanto de grande interesse que se utilize o menor número de partícu-

las necessário para o funcionamento eficaz do sistema de SLAM. O sistema proposto em (GRISETTI; STACHNISS; BURGARD, 2007) utiliza uma representação do ambiente do tipo *occupancy grid map*. A noção fundamental usada para reduzir o número de partículas necessário é o teorema de Rao-Blackwell que alega ser possível estimar inicialmente a trajetória do robô e, em seguida, usá-la para calcular o mapa, fortemente dependente dela. Com efeito essa fatoração torna possível que se calcule o mapa de modo eficiente, já que tanto a pose quanto as observações são conhecidas. Uma melhoria significativa oferecida pelo GMapping é sua distribuição proposta para amostragem. Em geral, sistemas baseados em filtros de partículas usam apenas informação do modelo de movimento baseado em odometria. O sistema proposto, no entanto, usa além desses dados, a última medição realizada, focando assim a amostragem nas regiões mais prováveis.

Em (ZHIRONG; ZIXING; BAIFAN, 2011), foi proposto um algoritmo denominado NP-FastSLAM que se baseia no FastSLAM e usa uma abordagem que alia filtros de partículas, filtros de Kalman e *Niche Particle Swarm Optimization* (Otimização por Enxame de Partículas com aplicação de Nichos). O objetivo da introdução do NichePSO ao processo foi desenvolver um sistema que sofresse menos com a depleção de partículas. A depleção de partículas é causada pela perda da diversidade de informação que as partículas representam e a consequente redução na capacidade de estimação do estado do sistema. O NPFastSLAM funciona de modo parecido com o FastSLAM. Mais especificamente, seu procedimento se resume a um processo de amostragem, seguido de um passo de otimização da pose representada por cada particula, que é sucedido por uma avaliação da aptidão de cada partícula. Realiza-se então uma busca pela presença de intersecções nos nichos das partículas e subsequente penalização e mutação das partículas com menor aptidão em cada uma das intersecções de pares de nichos. A supramencionada mutação é feita por um operador de mutação Gaussiano e aplicada à posição das partículas no espaço de poses. Por fim o algoritmo prossegue do modo usual para filtros de partículas, *i.e* realizando a atribuição de pesos às partículas e fazendo uma nova amostragem da distribuição *a posteriori* de poses caso o índice de degeneração seja maior que um limiar previamente definido. Os autores realizaram testes simulados nos quais foram encontrados resultados superiores aos do FastSLAM, embora nenhuma menção tenha sido feita ao esforço computacional requerido pela técnica.

2.2 Técnicas Baseadas em Métodos de Otimização

Em (BESL; MCKAY, 1992), encontramos a técnica denominada ICP (*Iterative Closest Point* – ICP), a qual se destina a determinar a transformação relativa entre duas nuvens de pontos (*point clouds*) que leve à maior sobreposição possível. O alinhamento de varreduras é o processo de busca por uma transformação que leve à máxima superposição entre duas nuvens de pontos. O ICP em particular realiza essa busca aplicando a heurística de que para alinhar duas nuvens de pontos, se faça corresponder a cada ponto de uma nuvem o ponto da outra nuvem que estiver mais próximo.

Em (CENSI, 2008), os autores descrevem um algoritmo de alinhamento de varreduras baseado no uso de uma métrica de distância do tipo ponto a linha (*point to line metric*), que eles alegam fornecer uma estimativa melhor da distância entre o ponto e a superfície cuja mensuração deu origem à varredura. Os autores descrevem uma fórmula fechada para a minimização dessa métrica e afirmam que o algoritmo que a emprega, denominado ICP Ponto a Linha (*Point to Line Iterative Closest Point*), converge de modo quadrático, enquanto o ICP tradicional converge linearmente.

Em (LU; MILIOS, 1997), é analisado o problema de registro de diversos conjuntos de medidas a laser. Na abordagem descrita mantém-se todos os *frames* de dados locais, bem como as relações espaciais entre eles. As relações entre os diversos *frames* são derivadas por meio de *scan matching* entre pares de *frames* ou pelo uso de informação odométrica. As técnicas baseadas em *laser scan matching* podem ser usadas para a construção de mapas de ambientes sem marcadores, *ou estrutura*, conhecidos *a priori*. Essas técnicas inferem a posição e orientação do robô por meio do ajuste da pose considerada até que tenhamos o maior grau de semelhança (segundo alguma métrica de comparação)(LU; MILIOS, 1997). Com efeito, é possível aplicá-las em ambientes sem estrutura conhecida e obter, como resultado, mapas métricos ou mapas topológicos.

Em (CENSI; IOCCHI; GRISETTI, 2005), os autores descrevem uma técnica em que se aplica a transformada de Hough e, em seguida, é realizado o processo de *scan matching* no espaço de Hough. A idéia se baseia no fato de que em locais com segmentos de retas, essas características são acentuadas pelo uso da transformada de Hough melhorando o desempenho da técnica. Além disso, o método descrito permite que se realize a estimação da orientação e da translação de modo separado.

Em (DIOSI; KLEEMAN, 2005), foi proposto um algoritmo para que as medidas obtidas por meio de um sensor a laser fossem alinhadas em coordenadas polares. Como um sensor desse tipo mede as distâncias aos obstáculos mais próximos em uma varredura ao longo de um espaço angular pré-determinado, o uso do método proposto tornaria desnecessária a transformação repetida dos valores medidos para um sistema cartesiano. Ademais, é possível aproveitar o fato de que num sistema polar, deslocamentos ao longo da coordenada angular correspondem a rotações num sistema cartesiano, fato esse que pode ser aproveitado na estimação de orientação em um sistema de SLAM que aplique essa técnica.

Em (NIETO; BAILEY; NEBOT, 2007), é descrito um algoritmo baseado na aplicação conjunta de filtros estendidos de Kalman e *scan matching*. O método, denominado Scan-SLAM foi desenvolvido com a ideia de generalizar o EKF-SLAM para além da aplicação a ambientes com marcadores que possam ser descritos por meio de modelos geométricos simples. Propuseram os autores que os marcadores fossem definidos por *templates* de medidas não processadas. É defendido por eles que os *templates*, assim definidos, podem aproveitar uma parcela maior da informação contida nos *scans* para, por exemplo, melhorar o *template* por meio da informação obtida ao longo do tempo de processo. Do modo como foi definido, é possível realizar a fusão de dados de diversos tipos de sensores e usá-los para criar um *template* em uma dimensão mais alta. Um exemplo usado pelos próprios autores foi a fusão de dados obtidos via *laser range finders* e câmeras.

Em (KAESS; RANGANATHAN; DELLAERT, 2008), foi apresentada uma abordagem ,até então inédita , para o problema de localização e mapeamento simultâneos que se baseia na existência de técnicas que permitem a rápida fatoração incremental de matrizes. O sistema descrito foi denominado iSAM(*Incremental Smoothing And Mapping* – iSAM) e é capaz de fornecer uma solução eficiente e exata por meio da atualização de uma fatoração QR da matriz de informação, que tende a ser esparsa. Por conseguinte, há que se calcular apenas os elementos alterados ao longo do processo. Os resultados experimentais apresentados mostraram que o método é aplicável ao mapeamento tanto com o uso de marcadores quanto sem ele. Foram reportados tempos de execução que se comparam favoravelmente a outros métodos, por exemplo para o *MIT Killian Court dataset* que possui 1941 poses e 2190 restrições entre elas os autores reportaram um tempo de 23.7s para a solução completa e um tempo médio de 12.2ms por passo.

Em (KÜMMERLE et al., 2011) os autores falam do G2O que, parafraseando os autores, é um sistema desenvolvido visando tornar-se um *back end* genérico para aplicação a problemas de otimização de mínimos quadrados não lineares que podem ser representados como grafos. O G2O apresenta um algoritmo similar ao do $\sqrt{SAM}$(*Square Root Smoothing and Mapping* - $\sqrt{SAM}$) que pode ser traduzido como Alisamento e Mapeamento Simultâneos baseados na Raiz Quadrada da matriz de Informação, que é um predecessor do iSAM . Ao ser aplicado ao SLAM, o processo fundamental desse método se baseia na linearização e sucessiva solução do sistema aproximado correspondente ao grafo de poses.

Em (SASAKI; KUBOTA; TANIGUCHI, 2008), os autores propõem que sejam empregados métodos da computação evolucionária, aliados a *Self Organizing Maps*, à solução do do problema de SLAM. Mais especificamente, eles propõem que a construção do mapa seja feita pelo treinamento de uma rede neural. Desse modo, os autores sugerem que o problema de construir um mapa topológico seja tratado como uma tarefa de aprendizagem não supervisionada (*unsupervised learning*). Ademais, no artigo em questão foi descrito o uso de um método pertencente ao ramo da computação evolutiva conhecido pelo nome Estratégia Evolucionária. A Estratégia Evolucionária do tipo $\lambda + 1$ foi usada para possibilitar a localização do robô. A codificação dos indivíduos na população foi definida pelos autores como sendo composta de parâmetros pertinentes à posição e ao ângulo de rotação do robô, no plano de movimento considerado.

Em (MINGAS; TSARDOULIAS; PETROU, 2012), foi usada uma representação do mapa tipo *occupancy grid* e a estimação da pose do robô que melhor explica as observações realizadas se fez por meio de um algoritmo genético que evolui um gene, responsável pela codificação da melhor estimativa de pose do robô que se pôde obter. Com efeito, um robô, cujo corpo seja indeformável e que se mova confinado em um subespaço bidimensional, pode ter até três coordenadas, a saber: x, y e θ, e o espaço de possíveis transformações de pose é, desse modo, reduzido. Ademais, o artigo trata de uma implementação em *hardware* de um algoritmo genético, especificamente adaptado para a solução do problema de estimação de pose. O sistema proposto foi implementado primeiro sob a forma de *software* e em seguida num FPGA. Um FPGA, sigla essa que representa o nome (*Field Programmable Gate Array* – FPGA) tradutível como matriz de portas programáveis por campo, usado na lingua inglesa para que se faça referência a esse tipo de dispositivo. Um FPGA é um circuito integrado projetado para ser configurado por um consumidor ou projetista

após sua fabricação, tratando-se, portanto, de um membro da categoria dos dispositivos de *hardware* reconfigurável. Segundo (MINGAS; TSARDOULIAS; PETROU, 2012) a implementação do algoritmo genético em *hardware* permitiu que um *speedup* de até $14,83$ vezes fosse alcançado em relação à sua versão implementada em *software*. Vale ressaltar que o sistema foi capaz de operar mais velozmente por meio do *hardware* proposto mesmo sendo a frequência de operação do FPGA mais que 20 vezes menor que a do computador em que o algoritmo fora testado. Não só isso, mas o uso de um módulo separado para a tarefa citada deixa o processador principal do robô livre para outras tarefas, tais como planejamento de rotas, caso o sistema esteja realizando SLAM ativo.

2.3 Considerações Finais do Capítulo

Neste capítulo foram apresentados, de modo sucinto, diversos trabalhos sobre o mapeamento e localização simultâneos. Conforme pôde ser observado, as pesquisas no campo em questão têm explorado de forma bastante intensa as técnicas baseadas em otimização, além das abordagens, já bastante consolidadas, apoiadas em filtros de Kalman e filtros de partículas. Os trabalhos baseados em técnicas de otimização possuem diversos representantes que se encontram na classe de algoritmos *offline*. Sendo esta tendência oriunda da grande quantidade de dados considerados em cada passo de otimização, o que aumenta também o esforço de processamento necessário. As técnicas que se direcionam à solução *online* usualmente se valem de métodos que empregam iterações que solucionam partes menores do problema (atualizações locais) ou que se satisfazem com precisão menor ,ao menos nos passos intermediários. O próximo capítulo trata da fundamentação teórica subjacente ao uso de sistemas de coordenadas projetivas, à definição de pose, e das operações usuais a elas aplicadas.

Capítulo 3

LOCALIZAÇÃO

ESTE capítulo trata da fundamentação teórica subjacente ao uso de sistemas de coordenadas projetivas e na Seção 3.1.1, à definição de pose Seção 3.1.2, e das operações usuais a elas aplicadas. Ademais, discorreremos, de modo sucinto, sobre as transformações afins na Seção 3.1.1.2, com vistas à aplicação desse conceito ao processo de alinhamento de varredura que, consoante ficará claro posteriormente, é um dos pilares desse trabalho. O capítulo se encerra definindo, na Seção 3.2, o que são grafos de poses e como esses são otimizados.

3.1 Sistemas de Coordenadas e Transformações

O SLAM objetiva descrever um mapa de uma região e a trajetória da entidade que realiza o mapeamento. Para tanto, é essencial que sejamos capazes de expressar adequadamente toda a gama de movimentos e mudanças de coordenadas necessárias tanto por virtude da mudança de ponto de vista, quanto por imposição da natureza dos sensores utilizados. O caráter da tarefa estabelece uma dicotomia entre o sistema de coordenadas local do robô e aquele em que se expressa o mapa (o sistema global). A análise da relação entre esses dois sistemas é de fundamental importância, não só por ser essencial à tarefa de integrar novos dados obtidos durante o movimento do robô, mas também porque a relação entre dois conjuntos de medidas que contenham sobreposição determina, ao menos aproximadamente, a relação entre os sistemas de coordenadas em que ambos foram obtidos. A determinação de uma transformação que torne máxima a sobreposição entre dois conjuntos de dados permite que se obtenha uma roto-translação que se aproxima do movimento realizado pelo robô (ou seu reverso). De um modo geral o conjunto de isometrias que representam *movimentos rígidos* em n dimensões possui $\frac{n\cdot(n-1)}{2}$ graus de liberdade. No

caso do SLAM, isso significa que a completa determinação de uma *pose* em n dimensões requer n coordenadas translacionais e $\frac{n\cdot(n-1)}{2}$ coordenadas rotacionais.

3.1.1 Coordenadas Homogêneas

Nessa seção trataremos da descrição dos sistemas de coordenadas homogêneas, bastante usados em áreas como a computação gráfica e a visão computacional.

3.1.1.1 Descrição Básica

O problema de definir transformações entre pares de sistemas de coordenadas Cartesianos, pode ser especificado de modo compacto por meio das chamadas *coordenadas homogêneas*. As coordenadas homogêneas são um sistema de coordenadas usado em geometria projetiva, com ampla aplicação no campo de computação gráfica.A ubiquidade desse tipo de representação se justifica pela expressividade que permite a redução de diversas composições, e aplicações, de transformações afins a simples produtos de matrizes. Evidentemente, a simplicidade de composição torna o sistema de coordenadas homogêneas de grande valia pois torna simples a utilização de pacotes de álgebra linear eficientes e bem testados para a implementação de sistemas reais que neles se baseiem.

As transformações afins são as mais gerais dentre as que discutiremos. Os casos particulares aos quais nos dedicaremos, em virtude de sua aplicação ao problema de SLAM do qual tratamos, são: as translações, as rotações e os movimentos de corpo rígido (que nada mais são do que composições de rotações e translações). Que fique claro que são isometrias diretas todos os mapeamentos pertencentes ao supra referenciado grupo.

Definição 1. *Sejam x e y coordenadas cartesianas de um ponto $P \in \mathbb{R}^2$.*

$$\forall t \left\{ t \in \mathbb{R}^* \Rightarrow \exists (u, v) \in \mathbb{R}^2 \mid u = \frac{x}{t} \wedge v = \frac{y}{t} \right\}.$$

Ao conjunto de triplas ordenadas (u, v, t) assim formadas denomina-se coordenadas homogêneas de P.

A Definição 1 segue aquela dada em (WOODS, 1922). É consequência lógica da Definição 1 que $\forall \lambda \in \mathbb{R}^*$, P e $\lambda * P$ representam o mesmo ponto.

Definição 2. *Seja (u, v) um par de razões análogo ao da Definição 1, seja , no entanto, $t \in \mathbb{R} \mid t = 0$. Dir-se-á que o par ordenado (u, v) especifica um ponto no infinito.*

De modo análogo, pode-se definir um conjunto de coordenadas homogêneas para um ponto P em três dimensões, sendo para isso necessário utilizar um espaço homogêneo tetradimensional. As coordenadas do ponto nesse espaço devem seguir a Definição 3.

Definição 3. *Sejam x,y e z coordenadas cartesianas de um ponto* $P \in \mathbb{R}^3$.

$$\forall t \left\{ t \in \mathbb{R}^* \Rightarrow \exists (u, v, w) \in \mathbb{R}^3 \mid u = \frac{x}{t} \wedge v = \frac{y}{t} \wedge w = \frac{z}{t} \right\}.$$

Ao conjunto de quadruplas ordenadas (u, v, w, t) *assim formadas denomina-se coordenadas homogêneas de P,(WOODS, 1922).*

3.1.1.2 Transformações

Uma vez definidas as coordenadas dos pontos de que tratamos num espaço projetivo, torna-se possível definir diversas transformações afins desses pontos por meio de operações matriciais. Com efeito, sejam x um ponto definido num espaço projetivo e M uma matriz. Se denominarmos x' a imagem da aplicação de M a x, teremos definido uma transformação projetiva genérica nesse espaço. Os *movimentos de corpo rígido*, podem ser completamente descritos por isometrias e, por conseguinte, será de particular interesse para o SLAM o conjunto de transformações que pode ser descrito em termos de rotações e translações.

As transformações afins podem ser completamente determinadas mediante a especificação de um número de parâmetros igual a $n \cdot (n + 1)$ se estivermos a lidar com um espaço n dimensional, e podem ser representadas por matrizes da forma vista na Equação 1:

$$M = \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}. \tag{1}$$

Para o nosso caso em particular, todo o trabalho foi realizado em um espaço de dimensão $n = 2$ o que exige que A pertença a um espaço de matrizes 2×2. Deveras, a manipulação dos elementos dessa matriz especifica se, e por qual ângulo, será realizada uma rotação dos pontos aos quais essa será aplicada e também se, e por qual fator, serão ampliadas, ou reduzidas, as componentes de cada ponto que se sujeite a uma tal transformação. Ademais, a constatação de que trabalhamos num espaço bidimensional é suficiente para que se possa verificar que t nada mais é do que uma matriz coluna, sendo $t \in R^{2\times1}$, que representa um vetor de translação. É possível então, e será melhor explicado ao longo do presente capítulo, que se especifique toda, e qualquer manipulação pertencente ao conjunto de roto-translações no plano por meio da especificação de apenas seis das nove componentes de M.

As translações podem ser especificadas por matrizes da forma especificada pela Equação 2:

$$\begin{bmatrix} I & t \\ 0^T & 1 \end{bmatrix}, \tag{2}$$

onde I representa a matriz identidade de dimensão $n \times n$ e t representa uma matriz coluna de dimensão $n \times 1$ com as translações pretendidas, como pode ser visto na Equação 3 para o caso $n = 3$,

$$t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}. \tag{3}$$

Entretanto, para o caso particular em que estamos interessados na nossa implementação os vetores de translação serão da forma expressa pela Equação 4,

$$t = \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \tag{4}$$

onde t_x representa a translação ao longo do eixo x e t_y a translação ao longo do eixo y. O uso da Equação 2, permite que se expresse a trasladação considerada sob a forma de uma matriz.

Na Figura 1, observa-se o efeito de se aplicar uma trasladação de uma unidade de comprimento ao longo do eixo x. A matriz da transformação aplicada é definida pela Equação 5 :

$$T_x = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5}$$

onde T_x é a matriz que representa a transformação realizada, nesse caso uma translação ao longo do eixo x.

De modo inteiramente análogo, a realização de uma translação ao longo do eixo y por uma distância pode ser representada pela matriz T_y definida na equa

$$T_y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Em um espaço bidimensional, como o que se considerou na implementação descrita no Capítulo 6, rotações ficam limitadas a apenas um eixo, e a transformação pretendida pode ser especificada com apenas um parâmetro (o ângulo de rotação em torno de um dos eixos Cartesianos) usando a Equação 7:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{7}$$

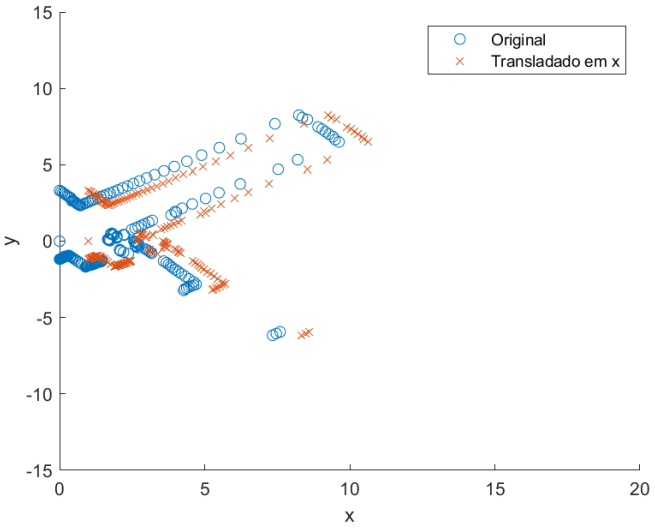


Figura 1: Scan sob efeito de T_x.

onde o ângulo θ é referido à rotação no sentido anti-horário em relação ao eixo das abcissas. No caso bidimensional há apenas rotações em torno da origem (com um sentido que evidentemente há de ser especificado), sendo o conceito de rotação em torno de um eixo empregado em três dimensões sem significado, em virtude da inexistência de um terceiro eixo em torno do qual as presumidas rotações possam se suceder. A situação, no entanto, se torna mais complexa no caso de a rotação ser efetuada em três dimensões. De fato, nesse caso há que se especificar um ângulo de rotação por eixo para que a transformação fique completamente determinada. Realmente, uma rotação em três dimensões pode ser definida em torno de três rotações sucessivas, desde que essas preservem a orientação do espaço (BUTKOV, 1973) podendo-se, por exemplo, adotar a convenção de realizar a composição das matrizes que representam a rotação em torno dos z,y e x respectivamente. As matriz de rotação em torno do eixo x por um ângulo ω é definida pela Equação 8:

$$R_x(\omega) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{bmatrix}. \tag{8}$$

A matriz que representa uma rotação em torno do eixo y por um ângulo ϕ, por sua vez, encontra-se exibida na Equação 9:

$$R_y(\phi) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix}. \tag{9}$$

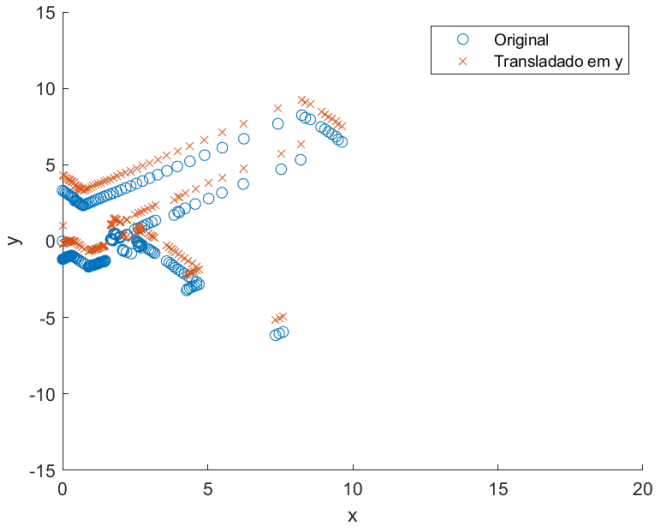


Figura 2: Scan sob efeito de uma translação de uma unidade de comprimento ao longo do eixo y.

A rotação em torno do eixo z por um ângulo κ se define pela Equação 10:

$$R_z(\kappa) = \begin{bmatrix} \cos\kappa & -\sin\kappa & 0 \\ \sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{10}$$

Qualquer rotação num espaço tridimensional pode ser expressa em termos de três rotações sucessivas desde que essas sejam feitas numa ordem que preserve a orientação do espaço usado. Uma convenção que pode ser adotada é a especificada pela Equação 11:

$$R(\omega, \phi, \kappa) = R_z(\kappa) * R_y(\phi) * R_x(\omega). \tag{11}$$

Definidas as transformações de rotação e translação, prossigamos para a especificação das chamadas *transformações de corpo rígido*. Em um espaço tridimensional, toda transformação de corpo rígido pode ser determinada por seis parâmetros *i.e.* três rotações e três translações (BUTKOV, 1973). Note que, no caso particular de encontrar-mo-nos num espaço bidimensional serão necessárias apenas três coordenadas para que se determine unicamente um movimento. De um modo geral, esse tipo de transformação, pode ser identificado, sem perda de generalidade, com o conjunto das roto-translações; pode ser representado por matrizes cuja forma siga a Equação 12:

$$M = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}. \tag{12}$$

A Figura 3 exibe o efeito de uma transformação do gênero supramencionado, em que o efeito observado é obtido pela aplicação de uma translação unitária ao longo dos eixos das abcissas e das ordenadas além de uma rotação por um ângulo de $\frac{\pi}{12}$rad. Essa

transformação exprime-se pela matriz M definida na Equação 13:

$$M = \begin{bmatrix} 0,9659 & -0,2588 & 1 \\ 0,2588 & 0,9659 & 1 \\ 0 & 0 & 1 \end{bmatrix}. \tag{13}$$

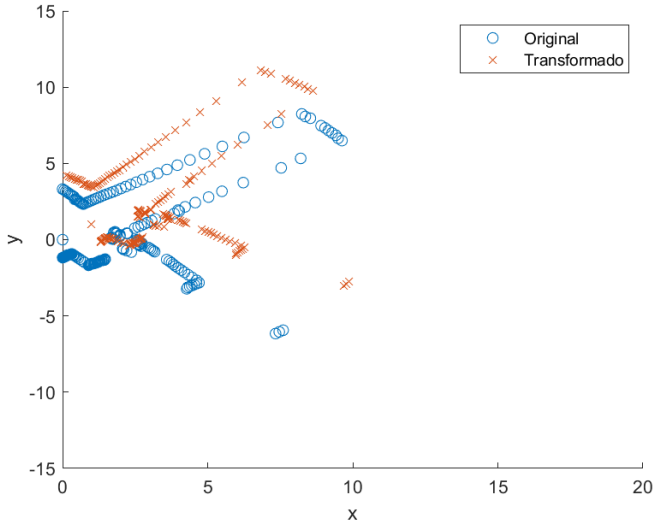


Figura 3: Scan sob efeito da transformação expressa pela Equação 13.

Para os casos em que se tenha, além de uma transformação de corpo rígido uma modificação de escala, define-se as chamadas transformações de similaridade. Na Equação 14, m é um escalar usado para definir a razão entre a norma dos vetores após uma operação de ampliação ou redução e a norma dos vetores no espaço original. De modo mais concreto $m \in \mathbb{R}$ e consiste de uma ampliação, ou redução de escala, sendo desnecessário para sensores como LIDAR que serão aplicados na implementação desse trabalho descrita no Capítulo 6, mas que encontram aplicação mais fértil quando se lida com sistemas como câmeras. Essa definição foi incluída apenas a título de completude da discussão que iniciáramos nessa seção.

$$M = \begin{bmatrix} m * R & t \\ 0^T & 1 \end{bmatrix}. \tag{14}$$

Em (GOLDSTEIN; POOLE; SAFKO, 2002; BUTKOV, 1973), declara-se não só que todo movimento de roto-translação pode ser expresso por uma matriz, mas também que toda matriz dessa sorte é uma matriz ortogonal. Explicitamos essa propriedade a título da completude da discussão.

3.1.2 Pose

O conceito de pose pode ser encontrado em outros trabalhos como (LU; MILIOS, 1997; SMITH; CHEESEMAN, 1986) *inter alia*, e refere-se a um conjunto de coordenadas suficientes para determinar a localização e orientação do robô em relação ao ambiente. No caso de sistemas confinados a se mover em um subespaço bidimensional de $\mathbb{R}^3$, sua pose haverá de conter três coordenadas. Se for considerado um sistema Cartesiano em $\mathbb{R}^2$, a pose poderá ser completamente determinada por duas coordenadas translacionais e um ângulo especificando a orientação do sistema. Em caso de o ente se mover em um subespaço tridimensional de $\mathbb{R}^3$ no entanto, faz-se necessário especificar seis coordenadas, para que fique completamente determinada a posição e a orientação do robô. Nesse caso, a determinação completa se dá por três coordenadas translacionais e três coordenadas rotacionais. Definindo-se uma pose em duas dimensões como $V_b = (x_b, y_b, \theta_b)$ e se considerarmos uma pose $D = (x, y, \theta)$ que representa uma variação da pose V_b que a transforma na pose V_a teríamos que a composição de V_b com D seria denotada por $V_a = V_b \oplus D$, e definida pela Equação 15:

$$\begin{bmatrix} x_a \\ y_a \\ \theta_a \end{bmatrix} = \begin{bmatrix} x_b \\ y_b \\ \theta_b \end{bmatrix} + \begin{bmatrix} x\cos\theta_b - y\sin\theta_b \\ x\sin\theta_b + y\cos\theta_b \\ \theta_a = \theta_b + \theta \end{bmatrix}. \tag{15}$$

Sob a visão de que uma pose absoluta define um sistema de coordenadas, e de que uma pose relativa define uma mudança do sistema de coordenadas, composta por uma translação seguida de uma rotação, a operação de *composição de poses* determina o novo sistema de coordenadas após a transformação.

Analogamente, é possível definir a operação inversa da composição de poses. Essa operação atua sobre duas poses e retorna a pose relativa. Denota-se tal operação por $D = V_a \ominus V_b$, e seu efeito é definido pela Equação 16:

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} (x_a - x_b)\cdot\cos(\theta_b) + (y_a - y_b)\cdot\sin(\theta_b) \\ -(x_a - x_b)\cdot\sin(\theta_b) + (y_a - y_b)\cdot\cos(\theta_b) \\ \theta_a - \theta_b \end{bmatrix}. \tag{16}$$

De posse da operação de diferença de poses, define-se também para uma pose relativa $D_{ab} = V_a \ominus V_b$ sua *pose relativa inversa* D_{ba} pela Equação 17:

$$D_{ba} = \ominus D_{ab} = (0, 0, 0) \ominus D_{ab}, \tag{17}$$

onde a notação $\ominus D_{ab}$ é retirada diretamente de (LU; MILIOS, 1997) e, mesmo que possa ser considerado um abuso de notação, foi mantida por motivos históricos.

Por fim, definiremos também a composição entre uma pose V_b e um ponto $u = (x, y)$ denotada $u' = V_b \oplus u$, cujo resultado será o mesmo ponto expresso noutro sistema de coordenadas. A transformação será análoga à Equação 15. A operação de composição entre uma pose e um ponto em duas dimensões define-se pela Equação 18:

$$\begin{bmatrix} x_a \\ y_a \\ \theta_a \end{bmatrix} = \begin{bmatrix} x_b \\ y_b \\ \theta_b \end{bmatrix} + \begin{bmatrix} x \cdot \cos\theta_b - y \cdot \sin\theta_b \\ x \cdot \sin\theta_b + y \cdot \cos\theta_b \\ \theta_a = \theta_b + \theta \end{bmatrix}. \tag{18}$$

Em luz do que foi explicado na subseção anterior sobre transformações num espaço projetivo, é interessante notar que uma pose, pode ser especificada como uma *transformação de corpo rígido*. De mesma sorte, operações entre poses podem ser especificadas como composições de transformações. Por exemplo, se tivermos uma pose $V_a = (x_a, y_a, \theta_a)$, essa pode ser especificada de modo equivalente pela matriz de transformação exibida na Equação 19,

$$M_a = \begin{bmatrix} \cos(\theta_a) & -\sin(\theta_a) & x_a \\ \sin(\theta_a) & \cos(\theta_a) & y_a \\ 0 & 0 & 1 \end{bmatrix}. \tag{19}$$

Se definirmos uma pose $V_b = (x_b, y_b, \theta_b)$ de modo análogo ao que foi feito na Equação 19, torna-se possível exprimir todas as operações entre poses acima expostas por meio de operações matriciais.

3.2 Grafos de Poses

Os métodos de resolução para o problema de SLAM que se baseiam em grafos de poses se valem de uma formulação do problema de SLAM em termos de um grafo de relações entre diversos referenciais em que o robô se encontrou ao longo do tempo e tratam o problema de estimação de estado como um problema de otimização (LU; MILIOS, 1997; KAESS; RANGANATHAN; DELLAERT, 2008; GRISETTI et al., 2010; KÜMMERLE et al., 2011). Os membros dessa família de soluções tratam, em geral, do problema de full SLAM, em que o objetivo é a estimação da completa trajetória percorrida pelo ente que realiza a tarefa de mapeamento além da determinação da pose atual e do mapa do ambiente. Essas técnicas tendem a produzir estimativas mais consistentes do que aquelas que se baseiam em técnicas de filtragem estatística. Isso ocorre pois técnicas como filtros de Kalman, filtros de partículas, filtros de informação e seus congêneres estão inerentemente sujeitos a inconsistências advindas de erros de linearização ocasionados por estimativas erradas, ou ao menos pouco precisas, de poses passadas. O ponto chave é que as técnicas

baseadas em filtragem perpetuam esses erros em virtude do modo como operam. Em contraste, as técnicas que se valem de grafos, por vezes denominadas de *smoothers*, são capazes, em virtude de sua solução do problema de *full SLAM*, de revisar estimativas para toda a trajetória baseados em novas evidências produzindo, por conseguinte, resultados de maior acurácia e consistência. Vale aqui fazer menção, como um adendo, ao fato de que, segundo (FOX; THRUN; BURGARD, 2005), os filtros de informação também podem fazer esse tipo de correção por operarem *offline* e sobre o conjunto de dados inteiro, embora isso possa consumir um tempo razoavelmente grande em virtude da natureza combinatorial do problema.

A maior parte dos algoritmos dessa classe funcionam com base em dois componentes principais e complementares: um *front-end*, que denominamos de sistema de alinhamento de varreduras, e um *back-end*, que passaremos a chamar de sistema de manipulação e otimização de grafos. Ao sistema de alinhamento de varreduras cabe extrair restrições entre as variáveis do problema (poses, *landmarks* ou similares) das informações sensoriais por meio da detecção de pontos de interesse e do alinhamento das varreduras correspondentes. Ao *back-end* resta a tarefa de otimizar o conjunto das restrições, *i.e.* encontrar a configuração das variáveis do problema que induza o menor erro possível. Com efeito, o sistema que otimiza o conjunto de restrições é o que realiza a otimização da otimização do sistema de grafos de poses. Que fique claro o significado atribuído à palavra *restrição* no contexto dos algoritmos de SLAM baseados em grafos, que são conjuntos de isometrias que relacionam poses ou observações. No caso de relações entre poses há tanto restrições derivadas de odometria (aprimoradas ou não por *scan matching*), quanto restrições impostas por observações. As restrições impostas por observação são criadas quando se observa o mesmo local de duas poses diferentes. Com efeito, se adotarmos a convenção de que as poses serão expressas como transformações num espaço projetivo, e admitirmos que a pose X_i corresponde à i-ésima pose verificada, teríamos então que uma restrição odométrica entre as poses X_i e X_{i+1} assumiria a forma $X_i^{-1} \cdot X_{i+1}$, a situação pode ser vista mais claramente na Figura 4. No caso de restrições provenientes de observações,

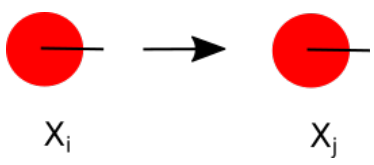


Figura 4: Exemplo de restrição derivada de odometria.

teríamos que a restrição seria definida por $X_i^{-1} \cdot X_j$, para duas poses i e j que estejam relacionadas por uma observação, como ilustrado na Figura 5. Na Figura 5, pode se observar que a aparente diferença entre o que se mede a partir da pose X_i em um referencial egocêntrico e o que se mede da pose X_j, também em um referencial egocêntrico, pode ser explicado se for assumido que houve movimento do robô entre X_i e X_j. Sob a hipótese de que tenha havido movimento, a transformação obtida pelo alinhamento das varreduras realizadas em X_i e em X_j representa a relação espacial entre X_i e X_j e é denominada medição virtual, tendo sido representada por uma seta na Figura 5. Com efeito, ao formar uma representação do grafo de poses teríamos que X_i e X_j seriam nós e a medição virtual, que provê uma restrição entre as poses, seria um arco entre eles. Essa restrição torna possível serem ambas as medições consistentes sob a hipótese de um mundo estático. De fato, o processo que descrevêramos se repete ao longo de todo o processo de SLAM e a configuração mais provável dos nós é determinada pela minimização dos erros incorridos pela violação do conjunto de todas as restrições formadas.

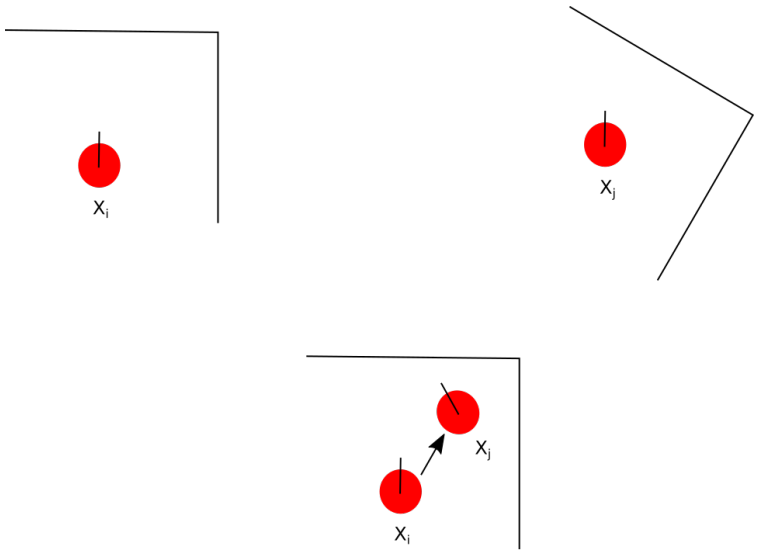


Figura 5: Exemplo de medição virtual.

Uma instância de método dessa sorte pode ser visto no Algoritmo 1 , cuja notação e funcionamento seguem a usada em em (GRISETTI et al., 2010). De modo resumido o Algoritmo 1 funciona de modo iterativo para a otimização da configuração mais provável dos nós. Os argumentos para o Algoritmo 1 são as estimativas iniciais das poses dos nós, as restrições e as suas respectivas matrizes de covariância (incertezas). Os valores que são retornados pelo processo são o conjunto atualizado de poses x^* e a matriz de informação H^* .O procedimento se sucede iterando até que haja convergência. A cada

iteração são calculadas uma matriz de informação e um vetor de informação. Em cada iteração, inicia-se com uma matriz H nula e um vetor b nulo e para cada restrição, esse vetor e essa matriz são atualizados. Resolve-se o sistema linear $H\Delta x = -b$, o que permite que se obtenha o valor do passo de atualização para a configuração dos nós Δx. Ao final de cada iteração faz-se $H^* = H$ e $x^* = x$. Uma vez que haja convergência o processo é encerrado e retorna-se x^* e H^*.

Algoritmo 1 SLAM Minimos Quadrados

Entrada $\check{x} = \check{x_{1:T}}, C = \langle e_{ij}(\cdot), \Omega_{ij} \rangle$

Enquanto não houver convergência **Faça**
  $b := 0$
  $H := 0$
  Para Todo $\langle e_{ij}(\cdot), \Omega_{ij} \rangle \in C$ **Faça**
    $A_{ij} := \frac{\partial e_{ij}}{\partial x_i}$
    $B_{ij} := \frac{\partial e_{ij}}{\partial x_i}$
    $H_{ii} \mathrel{+}= A_{ij}^T \cdot \Omega_{ij} \cdot A_{ij}$
    $H_{ij} \mathrel{+}= A_{ij}^T \cdot \Omega_{ij} \cdot B_{ij}$
    $H_{ji} \mathrel{+}= B_{ij}^T \cdot \Omega_{ij} \cdot A_{ij}$
    $H_{jj} \mathrel{+}= B_{ij}^T \cdot \Omega_{ij} \cdot B_{ij}$
    $b_i \mathrel{+}= A_{ij}^T \cdot \Omega_{ij} \cdot e_{ij}$
    $b_j \mathrel{+}= B_{ij}^T \cdot \Omega_{ij} \cdot B_{ij}$
  Fim Para
  $H_{11} \mathrel{+}= I$
  $\Delta x :=$ **Resolva** $(H \cdot \Delta x = -b)$
  $\check{x} \mathrel{+}= \Delta x$
Fim Enquanto
$x^\star := \check{x}$
$H^\star := H$
$H^\star_{11} - = I$
Retornar $x^\star$, $H^\star$

No Algoritmo 1, define-se cada variável X_i em termos de suas componentes translacionais, que se agrupam no vetor t_i, onde i é o número que denota a pose à qual se faz menção, e de sua orientação θ_i. A variável X_i é definida pela Equação 20:

$$X_{ij} = \begin{bmatrix} t_i \\ \theta_i \end{bmatrix}, \tag{20}$$

onde t_i representa as duas coordenadas translacionais para o plano, e formam um espaço euclidiano, e θ_i representa a orientação, está contida no intervalo $[-\pi, \pi[$ e pertence ao grupo de rotação $SO(2)$. A variável Z_{ij} é a medição virtual entre os nós i e j e representa a transformação que faz os dados obtidos a partir do nó i terem máxima superposição aos

dados adquiridos do nó j. Define-se Z_{ij} pela Equação 21:

$$Z_{ij} = \begin{bmatrix} t_{ij} \\ \theta_{ij} \end{bmatrix}, \tag{21}$$

onde, de modo análogo ao caso de X_i, t_{ij} representa o vetor de translação e θ_{ij} a componente translacional. A função e_{ij} representa a diferença entre a previsão $\hat{z}_{ij}$ que se faria para uma medição virtual entre os nós x_i e x_j dada a estimativa atual e a observação real z_{ij} coletada pelo robô. A função de erro $e_{ij}(x_i, x_j)$ é, com efeito, uma diferença de poses como definida no Capítulo 4 e é definida pela Equação 22:

$$e_{ij}(x_i, x_j) = \begin{bmatrix} R_{ij}^T \cdot (R_i^T \cdot (t_j - t_i) - t_{ij}) \\ \theta_j - \theta_i - \theta_{ij} \end{bmatrix}, \tag{22}$$

onde R_i é a matriz de rotação que tem por ângulo a orientação de x_i, R_j é a matriz de rotação que tem por ângulo a orientação de x_j e R_{ij} é uma matriz de rotação que tem por ângulo a rotação que leva à máxima sobreposição da observação de feita de x_i e aquela feita a partir de x_j. As matrizes A_{ij} são, em verdade, resultado de $\frac{\partial e_{ij}}{\partial x_i}$ e são definidas pela Equação 23:

$$A_{ij} = \begin{bmatrix} -R_{ij}^T \cdot R_i^T & R_{ij}^T \cdot \frac{\partial R_i^T}{\partial \theta_i} \cdot (t_j - t_j) \\ 0 & 1 \end{bmatrix}, \tag{23}$$

onde 0 representa uma matriz linha de zeros com duas componentes e $\frac{\partial R_i^T}{\partial \theta_i}$ é definido pela Equação 24:

$$\frac{\partial R_i^T}{\partial \theta_i} = \begin{bmatrix} -\sin(\theta_i) & -\cos(\theta_i) \\ \cos(\theta_i) & -\sin(\theta_i) \end{bmatrix}. \tag{24}$$

Por fim, as matrizes denotadas B_{ij} são em verdade derivadas da forma $\frac{\partial e_{ij}}{\partial x_j}$ e são definidas pela Equação 25:

$$B_{ij} = \begin{bmatrix} -R_{ij}^T \cdot R_i^T & 0 \\ 0 & 1 \end{bmatrix}, \tag{25}$$

onde o 0 na primeira linha representa uma matriz coluna de dois elementos e o da segunda linha uma matriz linha com duas componentes. Esse tipo de técnica, costuma produzir mapas topológicos, *i.e* mapas focados na representação da relação entre locais de interesse ao invés de sua representação metrica, muito embora seja possível trabalha-los de modo a obter outros tipos de representação espacial como a descrita na Seção 4.2.

3.3 Considerações Finais do Capítulo

Nesse capítulo foram discutidos conceitos fundamentais relativos ao uso de sistemas de coordenadas, às coordenadas homogêneas, que conforme foi explicado são uma representação computacionalmente conveniente das transformações afins, ao conceito de pose e

às operações básicas que a elas podem ser aplicadas. No próximo capítulo, mudaremos nosso foco para as representações do espaço, com ênfase àquelas que foram usadas, ou que possuem forte relação com os métodos que aplicamos.

Capítulo 4

MAPEAMENTO

ESTE capítulo trata do sistema de alinhamento de varreduras, do modo como a informação do mapa foi representada e do modo como são detectados os retornos a locais já visitados. A definição do alinhamento de varreduras se encontra na Seção 4.1. Na Seção 4.2, descreve-se como são representados os mapas. Na Seção 4.1.1 é descrita a busca por vizinhos mais próximos. É objetivo desse capítulo descrever as técnicas que foram aplicadas num grau de profundidade suficiente para que sirva de base para a compreensão da implementação realizada, que se encontra descrita no Capítulo 6.

4.1 Alinhamento de Varreduras

O processo de alinhamento de varreduras (*scan matching*) tem por objetivo determinar a melhor transformação entre duas nuvens de pontos obtidas por meio de algum instrumento de sensoriamento. Em nosso projeto foi usado um (*Laser Ranged Finder* – LRF). O uso do alinhamento de varreduras como forma de estimar o movimento entre duas poses, a partir das quais medições foram realizadas, é eficaz, na medida que se possa confiar na precisão de ambas e na convergência do processo de otimização subjacente, em um tempo razoável. Os algoritmos como o ICP e seus derivados, embora sejam razoavelmente eficientes se forem propriamente implementados, possuem por ponto fraco a sua grande sensibilidade em relação à inicialização já que em (BESL; MCKAY, 1992) se garante que o processo sempre converge para um máximo local. Sobre isso, que nos baste dizer que uma inicialização pouco adequada pode causar a falha, ou a baixa eficiência, do processo de alinhamento. Um modo de mitigar esse problema, e evitar que esses algoritmos convirjam para ótimos locais, é inicializar o algoritmo com uma estimativa que se creia ser próxima do alinhamento ótimo global. Outro caminho, é o uso de um método mais robusto de

busca de alinhamento ao invés da simples aplicação direta do método descrito em (BESL; MCKAY, 1992). Tendo em vista as vantagens proporcionadas pela aplicação de técnicas que não só sejam capazes de explorar de modo mais abrangente o espaço de buscas mas também que explorem o espaço de soluções de modo independente de derivadas, e levando em consideração o fato de que heurísticas diferentes apresentam características diversas que as tornam mais adequadas a resolver tipos particulares de problemas, foi decidido que seria vantajoso para o nosso projeto, e enriquecedor para a pesquisa da aplicação de técnicas de otimização de enxame ao problema do alinhamento de varreduras, que se realizasse uma comparação de técnicas diversas. Em consonância com o objetivo de comparação das meta-heurísticas de otimização de enxame que selecionamos a parte do sistema responsável pelo alinhamento de varreduras foi implementada de modo a possibilitar a aplicação intercambiável das técnicas que foram selecionadas. Em nosso projeto foi utilizada uma representação do mapa como grade de ocupação, e as varreduras foram representadas como nuvens de pontos. Essa escolha se fez, pois permite que se tenha a possibilidade de alinhar varreduras (entre si) com um esforço menor do que o necessário para construir dois mapas locais e alinhá-los, ou ainda do que seria necessário para construir um mapa local e alinhá-lo ao mapa global. Ademais, esse tipo de representação permite que se utilize as vantagens de ambas as representações e, tendo em vista que o sistema preserva toda a informação referente à trajetória, é sempre realizar a conversão entre uma nuvem de pontos em um mapa de ocupação. A desvantagem é que se faz necessária a realização de frequentes buscas por correspondências entre nuvens o que pode acarretar um custo computacional elevado. O fardo computacional da busca por vizinhos mais próximos foi reduzido pelo uso de árvores KD (FRIEDMAN; BENTLEY; FINKEL, 1977; BENTLEY; FRIEDMAN, 1979; BENTLEY, 1980). Dentre as possíveis funções de perda que poderíamos ter selecionado, a denominada função de perda de Cauchy, que se apresenta, em termos do erro quadrático, na Equação 26 :

$$\rho(e) = \log(1 + e^2), \tag{26}$$

onde $\rho(e)$ é a função objetivo após a transformação não linear e e representa a soma dos erros quadráticos, foi a que pareceu apresentar melhores resultados sem onerar excessivamente o processo de alinhamento. Na Figura 6 encontra-se exibida uma comparação entre a soma dos erros quadráticos e a função de perda de Cauchy.

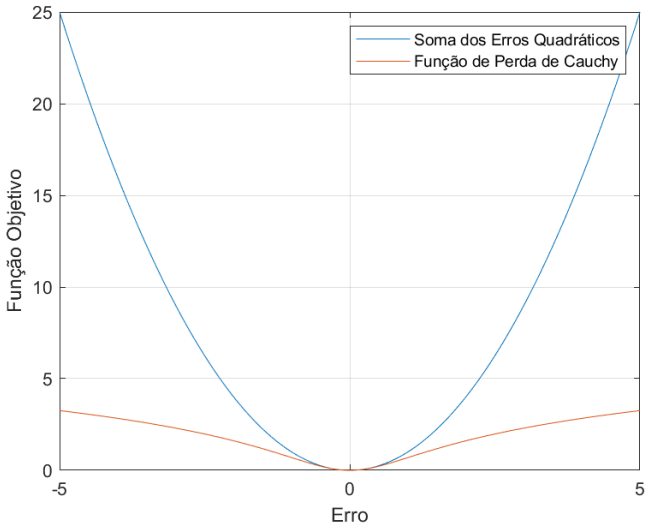


Figura 6: Comparação entre a soma dos erros quadráticos simples e a função de perda de Cauchy.

Observa-se que mesmo para valores pequenos a medida proporcional à soma dos erros quadráticos tende a crescer acentuadamente, o que pode prejudicar a acurácia do alinhamento, de importância superveniente a pequenos acréscimos ao tempo consumido por alinhamento já que a economia desses acréscimos levaria a um consumo maior de tempo no processo de otimização do grafo de poses subjacente, além de dificultar o processo de busca por fechamentos de laço, que descreveremos na seção seguinte, cuja primeira etapa é justamente procurar poses suficientemente próximas de onde se tenha realizado medições passíveis de apresentar medições que evidenciem o retorno a um lugar visitado anteriormente.

4.1.1 Busca por um Vizinho Mais Próximo de um Ponto

No curso do presente trabalho será parte fundamental de diversas das etapas empregadas que se realize a busca dos vizinhos mais próximos de um determinado ponto em um conjunto de pontos. Com o intuito de esclarecer o que dissemos, seja $P = \{p_1, p_2, \cdots, p_N\}$ um conjunto de pontos em $\mathbb{R}^2$, e um ponto q , o problema de busca pelo vizinho mais próximo de q em P consiste de encontrar um ponto p pertencente a P tal que, dentre os pontos de P, p seja o mais próximo de q. O modo mais simples para realizar tal tarefa seria a realização de uma busca exaustiva comparando todos os pontos de P a q. Entretanto essa abordagem é a menos eficiente, o que se torna evidente se o número de buscas do supracitado gênero for grande. Com isso em mente, e como em nosso sistema de SLAM

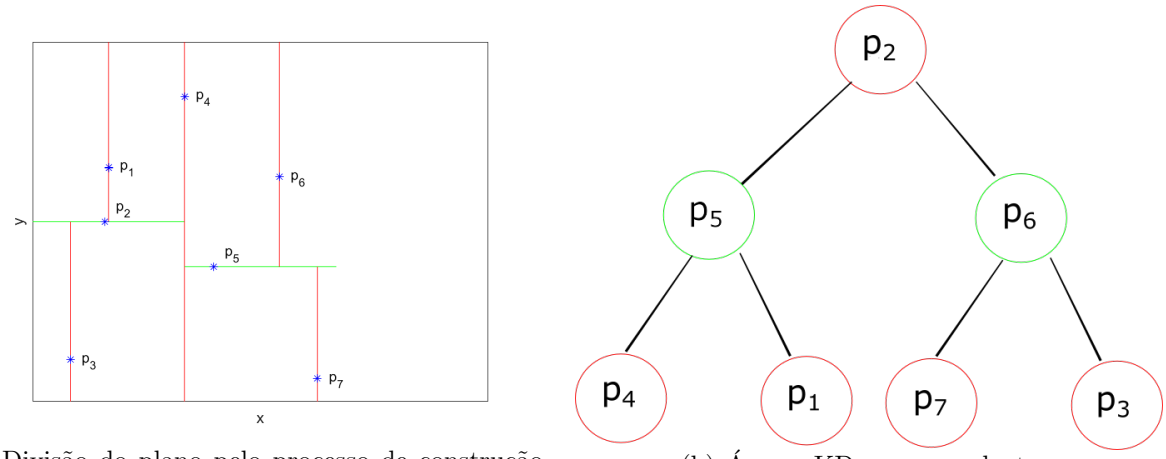
somos levados a realizar diversas buscas por vizinhos mais próximos, sendo cada uma por vizinhos de conjuntos inteiros de pontos em um conjunto de contrapartida, convém que P seja armazenado em uma estrutura de dados que facilite uma busca veloz. Uma das estruturas de dados possíveis e, deveras, a que empregamos é a árvore KD.

4.1.2 Árvores KD

O armazenamento de pontos em uma árvore KD requer que este conjunto seja dividido. Uma das formas de divisão mais comuns faz uso da mediana das coordenadas dos pontos pertencentes ao conjunto que se deseje armazenar, consoante ilustrado na Figura 7. Se fôssemos armazenar um conjunto de pontos P, como o descrito na Seção 4.1.1, tal procedimento iniciaria pela determinação da mediana da primeira coordenada (na ordem de escrita, ou segundo qualquer critério desde que todas as coordenadas sejam consideradas) de todos os pontos de P. O ponto cuja primeira coordenada seja equivalente à mediana será considerado a raiz da árvore. Em seguida, o restante do conjunto será dividido em dois subconjuntos baseado no valor da mediana da primeira coordenada. Os subconjuntos restantes serão divididos com base no valor da mediana da sua segunda coordenada. O processo ora descrito é denominado *binary space partitioning* (Particionamento espacial binário) e gera uma árvore binária balanceada contendo todos os pontos de P. A complexidade computacional de criar uma árvore KD é da ordem de $\mathcal{O}(kN \log_2 N)$ segundo (FRIEDMAN; BENTLEY; FINKEL, 1977). Ou seja, a construção requer um tempo proporcional a ordenação do ponto com relação a cada uma de suas coordenadas , *i.e.* k no caso geral tratado em (FRIEDMAN; BENTLEY; FINKEL, 1977)).

O uso de árvores KD reduz o tempo necessário para que se encontre um conjunto de vizinhos mais próximos significativamente. Em (BESL; MCKAY, 1992) os autores explicitam que a busca por vizinhos mais próximos é o passo mais computacionalmente custoso do algoritmo ICP, descrito em maiores detalhes na Seção 4.1.3, com uma complexidade da ordem de $\mathcal{O}(N_p N_q)$, onde N_p é o número de pontos em P e N_q é o número de pontos cujos vizinhos se busca. Com efeito, uma vez construída a árvore o custo de cada busca tem um custo da ordem de $\mathcal{O}(\log_2 N_p)$ o que reduz o tempo total de busca por vizinhos mais próximos a um tempo da ordem de $\mathcal{O}(N_q \log_2 N_p)$.

A busca aproximada por um ponto p que seja vizinho mais próximo de q pode ser feita de modo algoritmico, iniciando-se a busca pela raiz e comparando cada coordenada



(a) Divisão do plano pelo processo de construção da árvore que armazena P.

(b) Árvore KD correspondente.

Figura 7: Diagrama ilustrando a divisão do plano por uma árvore KD.

do presente nó à coordenada do ponto que se busca, de acordo com a dimensão da qual cada nó encontrado represente uma divisão. Procedendo-se desse modo até que se encontre uma folha da árvore, deve-se então marcar o ponto localizado na folha alcançada como o vizinho mais próximo de q encontrado até o momento (p) e memorizar a distância entre p e q como sendo d. Sobe-se então um nível da árvore e calcula-se a distância entre q e esse nó, se a distância for menor que d atribui-se a p as coordenadas desse novo nó e a d a distância entre p e q. Se da comparação que mencionamos no período anterior for, entretanto, maior que d exclua o ramo filho desse nó que não contém p e repita o processo anterior até que a raíz da árvore seja encontrada. Se a distância entre q e o nó pai de p for igual à d então todo o outro ramo do nó pai de p terá que ser explorado, assim como o resto da árvore. Procedendo dessa maneira retorna-se p ao chegar à raíz da árvore. Esse algoritmo apresenta uma complexidade esperada de ordem $\mathcal{O}(\log N_p)$, para o caso de uma árvore que armazene o conjunto P supracitado. No melhor caso, que ocorre quando ao chega à primeira folha o ponto encontrado apresenta uma distância menor que a de todos os seus nós pais ao ponto que se busca, apenas um a complexidade da busca é de ordem $\mathcal{O}(1)$ (KJER; WILM, 2010). No pior caso, nenhum nó pode ser eliminado da busca e para encontrar um ponto q em um conjunto com N_p pontos o processo terá a complexidade da ordem daquela que se teria ao fazer uma varredura do conjunto, ou seja $\mathcal{O}(N_p)$ (KJER; WILM, 2010).O processo de busca descrito pode ser encontrado em (FRIEDMAN; BENTLEY; FINKEL, 1977; BENTLEY; FRIEDMAN, 1979; BENTLEY, 1980). Ademais, diremos que foi usada a biblioteca nanoflann (BLANCO; RAI, 2014) como base de nossa implementação.

4.1.3 ICP

Um dos métodos mais tradicionais para a resolução dessa tarefa é o denominado ICP (*Iterative Closest Point*), conforme mencionado em (BESL; MCKAY, 1992). As principais vantagens dos métodos baseados em ICP são sua simplicidade de implementação e agilidade, quando implementados por meio de árvores KD (KD-Trees) (FRIEDMAN; BENTLEY; FINKEL, 1977; BENTLEY; FRIEDMAN, 1979; BENTLEY, 1980; ZHANG, 1994). Entretanto, o método assume que as formas cuja correspondência se busca podem ser total e perfeitamente superpostas. É evidente ser mais comum o caso em que as amostras sob análise apresentam apenas uma sobreposição parcial, como quando essas são obtidas com taxas de amostragem diferentes, ou de pontos de vista diferentes. É interessante notar, no entanto, que, objetivando melhorar a robustez da técnica em questão, diversas alterações foram propostas as quais vão desde sugestões de heurísticas de associação de pontos até a implementação de métodos híbridos que empregam técnicas de otimização global para fornecer ao ICP uma melhor estimativa inicial. No Algoritmo 2, encontra-se exposto o funcionamento básico desse método clássico. Nesse algoritmo foi adotada a convenção de que A e B são duas nuvens de pontos distintas, T é uma transformação, w_i são pesos pelos quais cada desvio será ponderado e $d_{\max}$ é o limiar aceitável para o erro.

Algoritmo 2 ICP

Entrada $A = \{a_1, a_2, \cdots, a_i, \cdots, a_N\}, B = \{b_1, b_2, \cdots, b_i, \cdots, b_N\}, N$

1: **Repita**
2: **Para** $i = 1 \cdots N$ **Faça**
3: $p_i = encontraMaisProximo(a_i, T \cdot B)$
4: **Se** $\|p_i - T \cdot B\| < d_{max}$ **Então**
5: $w_i = 1$
6: **Senão**
7: $w_i = 0$
8: **Fim Se**
9: **Fim Para**
10: $T = \min_T \sum_{i=1}^{N} w_i \cdot \|p_i - T \cdot b_i\|^2$
11: **Até** Condição de parada.
12: **Retornar** T

O funcionamento do Algoritmo 2 pode ser descrito como uma aplicação iterada das seguintes etapas:

1. Correspondência: para cada ponto do conjunto de dados de origem, associa-se seu vizinho mais próximo no conjunto de destino.

2. Minimização: a medida de erro (função objetivo utilizada) é minimizada.

3. Transformação: os pontos do conjunto de origem são transformados usando o resultado do processo de minimização.

Elaboremos acerca de cada uma dessas etapas, para esclarecer seu significado. A etapa de correspondência é feita mediante a aplicação de uma busca que procede segundo o que explicamos na Seção 4.1.1. A etapa de minimização consiste de um processo de otimização que se usa para estimar a transformação que minimiza a medida de erro usada. Apesar de nesse trabalho utilizarmos algoritmos de otimização baseados em inteligência de enxame para realizar essa etapa, em (BESL; MCKAY, 1992; ARUN; HUANG; BLOSTEIN, 1987) pode-se encontrar uma solução em forma fechada para a minimização ponto a ponto, como a que foi proposta aqui. A etapa de transformação possui um objetivo bastante evidente dado seu nome, e consiste da aplicação da transformação estimada à nuvem de origem para que essa se sobreponha à nuvem de destino.

O ICP é um método cuja condição de término, que foi deixada em aberto no Algoritmo 2, é geralmente baseada no número máximo de iterações permitido ou num limiar de mudança relativa na medida de erro. Segundo (BESL; MCKAY, 1992), é comum que o algoritmo convirja em poucas iterações, no entanto deve-se ter em mente que diversos problemas podem ocorrer, são eles: a presença de múltiplos mínimos locais, a influência de *outliers* e a sobreposição parcial. A presença de múltiplos mínimos pode levar o ICP a convergir para uma transformação que seja apenas localmente ótima, já que em (BESL; MCKAY, 1992) há garantia de que o algoritmo sempre converge. A presença de *outliers* e de ruído pode levar o erro de sobreposição a nunca se anular, ou guiar o processo para um mínimo diferente do global. A sobreposição parcial pode ser problemática já que varreduras podem não representar a mesma parte de um ambiente, ou a sobreposição entre elas pode não ser suficiente para que se possa adequadamente estimar o movimento relativo entre os referenciais ao quais elas se referem.

4.2 Mapas de Ocupação

Os mapas de ocupação são uma forma de representação espacial baseada na discretização do espaço, *i.e.* faz corresponder uma célula a cada região do espaço, e na classificação de cada "célula" como *cheia* ou *vazia*. Esse tipo de representação espacial tem como hipóteses

a estaticidade do mundo, a independência entre as variáveis aleatórias que representam o mapa e o fato de cada célula estar completamente cheia ou vazia. Ou seja a presença de um obstáculo em uma parte de uma dada região do espaço faz com que se considere a região, como um todo, preenchida e o que se estima é a probabilidade disso ser verdade. Da hipótese de independência entre as diversas células advém ser a distribuição de probabilidade do mapa dada pelo produto das probabilidades de todas as suas células (FOX; THRUN; BURGARD, 2005).

Em geral, o problema de estimar um mapa de ocupação é quebrado em subproblemas menores, quais sejam a estimação de células individuais (FOX; THRUN; BURGARD, 2005). A Figura 8 exibe um mapa de ocupação representando o laboratório de pesquisas da Intel em Seattle. Na Figura 8 vemos claramente a representação do espaço livre e o espaço ocupado. Nesse tipo de mapa é muito comum que se adote a convenção de que espaço branco, ou cinza claríssimo, representa espaço livre, espaço de coloração negra, ou cinza escura, é ocupado por obstáculos e nada, ou pouco, se sabe sobre espaço de cor cinza intermediária.

Figura 8: Mapa do Interior do Laboratório de Pesquisas da Intel em Seattle.

Para a presente discussão adotaremos a convenção usada em (FOX; THRUN; BURGARD, 2005) e repetida a seguir. Sejam t o instante atual, m o mapa, m_i uma célula do mapa no instante atual, x_t a pose atual e z_t a observação feita no instante atual. É

conveniente representar as probabilidades de ocupação no formato de *log-odds*. O cálculo do valor nesse formato pode ser obtido a partir de $p(m_i|z_{1:t}, x_{1:t})$ por meio da Equação 27:

$$l_{t,i} = \log \frac{p(m_i|z_{1:t}, x_{1:t})}{1 - p(m_i|z_{1:t}, x_{1:t})}. \tag{27}$$

A transformação expressa pela Equação 27 se encontra ilustrada na Figura 9. Vale ressaltar que com o uso da forma de *log-odds* há tanto um ganho de desempenho no processo de atualização do mapa, já que são trocados os produtos de valores em ponto flutuante pela adição de seus correspondentes num espaço de logaritmos, quanto um ganho de estabilidade numérica em casos cujos valores de probabilidade envolvidos estejam próximos de 0 ou 1. A obtenção dos valores de ocupação na forma de probabilidades a partir dos valores na forma de *log-odds* pode ser feita pelo uso da Equação 28:

$$p(m_i|z_{1:t}, x_{1:t}) = 1 - \frac{1}{1 + \exp l_{t,i}}. \tag{28}$$

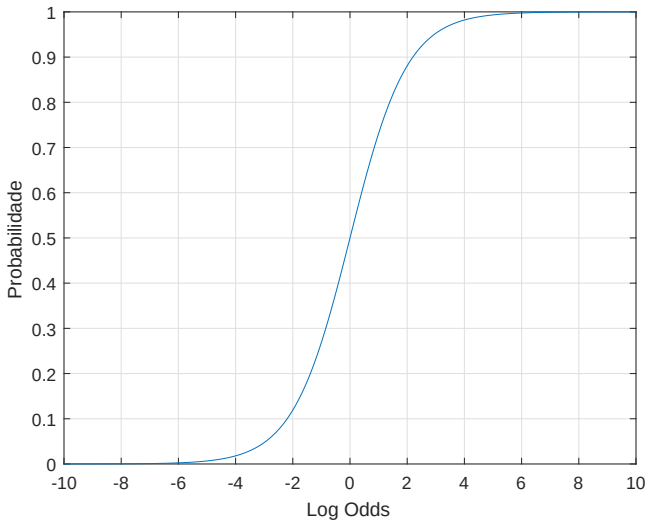


Figura 9: Correspondência entre valores de log *odds* e probabilidades

A Figura 9 representa a curva de correspondência entre os valores de *log odds* e as probabilidades que lhes correspondem, cujo mapeamento segue a Equação 28. Segundo (FOX; THRUN; BURGARD, 2005), a principal vantagem que há em usar esse formato é que são evitada a instabilidade numérica em casos com valores de probabilidade próximos de 0 e 1. Acresceremos ao dito anteriormente que o uso de *log odds* permite simplificação dos processos de atualização dos valores de cada célula do mapa de ocupação. Com efeito, e para além da simplificação das fórmulas usadas para representar a incorporação de informação adquirida durante o processo de exploração a mapas de que já se construíra, o uso

da representação de *log odds* torna possível substituir por operações de adição e subtração as multiplicações e divisões, que são evidentes na lei de Bayes (GRINSTEAD; SNELL, 2006; LAPLACE, 2005). O Algoritmo 3 indica de que modo é feito o mapeamento nesse tipo de representação (FOX; THRUN; BURGARD, 2005). Nesse algoritmo l_0 representa a probabilidade de ocupação *a priori* sob a forma de *log-odds* conforme definido na Equação 27. O modelo de sensor é denominado inverso pois fornece a probabilidade das causas (a existência de obstáculos) condicionada em relação aos efeitos (as observações).

Algoritmo 3 Atualizar Mapa de Ocupação

Entrada $l_{t,i}, x_t, z_t$
1: **Para** todas as células m_i **Faça**
2: **Se** m_i está no campo de visão de z_t **Então**
3: $l_{t,i} = l_{t-1,i} + modeloInversoSensor(m_i, x_t, z_t) - l_0$
4: **Senão**
5: $l_{t,i} = l_{t-1,i}$
6: **Fim Se**
7: **Fim Para**
8: **Retornar** $l_{t,i}$

O modelo mais simples que se pode empregar consiste de uma densidade de probabilidade concentrada na região do espaço que se crê estar ocupada pelo obstáculo detectado. Uma pequena melhoria que se pode fazer é acrescer, ao que foi explicado a hipótese de que nada se sabe acerca do que se encontra atrás (com relação ao ponto de vista do observador) do primeiro obstáculo detectado sendo tão provável que esse espaço se encontre ocupado quanto desocupado. Esse modelo pode ser observado na Figura 10, com o obstáculo suposto, a título de ilustração, a 10 unidades de distância do observador.

A necessidade de considerar um modelo de sensor provém da tarefa em questão fazer necessária uma atualização da crença *a priori* expressa pelo mapa, com vistas ao foi obtido por um processo de sensoreamento segundo um critério racional e consistente. Ademais, há que ser considerado o fato de sensores, por maior que seja a sua precisão, gerarem medições espúrias em certas circunstâncias. A título de exemplo, poderíamos mencionar as reflexões de raios laser em superfícies vítreas, que podem levar à detecção de obstáculos inexistentes. Além de reflexões, medidas realizadas com distancias próximas do maior afastamento que um LRF (*Laser Range Finder*) é capaz de fazer tendem a incluir erros, já que o sensor detectará a fronteira da sua região de alcance como um obstáculo.

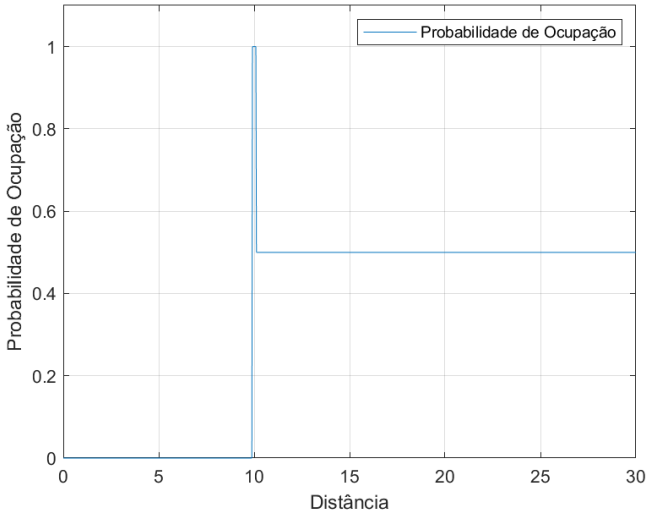


Figura 10: Modelo de ocupação de probabilidade concentrada assumindo obstáculo a 10 unidades de distância.

Em suma, a representação do mapa sob a forma de mapas de ocupação é bastante empregada quando se deseja uma representação métrica do ambiente ou se pretende, por exemplo, mapear um ambiente pouco estruturado. Possui como vantagem o fato de implementação dispensar o uso de extratores de características (*feature extractors*), o que simplifica a implementação. A desvantagem inerente a esse tipo de representação é o consumo de memória, que se torna um fator ainda mais limitante se for desejado generalizar essa técnica para sistemas que construam mapas tridimensionais, e o tempo para sua atualização, sendo tais fatores expostos em (FOX; THRUN; BURGARD, 2005) mas tal coisa vem a custo de precisão.

4.3 Detecção de Retorno a Locais Já Visitados

Um dos obstáculos de solução mais difícil dentro da pesquisa de sistemas de localização e mapeamento simultâneos é o desenvolvimento de um modo de reconhecer que, em dado momento durante o percurso, retornou-se a um lugar que se visitara anteriormente. No contexto do campo de mapeamento e campos afins, esse reconhencimento de lugares previamente visitados é conhecido como *loop closure*, doravante fechamento de laço, e seu nome é uma alusão ao fenômeno que ocorre no caso em que se retorna ao início de um percurso fechado. Com efeito, Edwin Olson em (OLSON, 2008), afirma que a qualidade de um mapa é uma função da quantidade e qualidade dos fechamentos de laço. Realmente, e apesar de sua fundamental dificuldade, essa tarefa é de importância central e ímpar pois

permite que se corrijam erros acumulados em uma longa sequência de medições, reduz a incerteza acerca da pose e, consequentemente, permite que se atualize o mapa de modo consistente por fornecer restrições ao movimento do robô, limitando o erro cometido ao se realizar o que se conhece por *dead reckoning*. Em navegação, *dead reckoning* é o processo de calcular a posição atual de um ente móvel pelo uso de uma posição anterior, também denominada *fix*, atualizada com base numa velocidade estimada, ou conhecida, da direção do movimento e do tempo transcorrido. A detecção de fechamentos de laço é difícil por diversas razões. Por exemplo, diversos dos ambientes a que se proporia a aplicação de sistemas de SLAM podem conter eventos dinâmicos que alteram, ainda que de maneira modica, a aparência do ambiente tais como: pessoas andando e móveis que são trocados de lugar . Nesse trabalho foram usados *laser scanners*, que produzem resultados metricamente precisos. Entretanto, por maior que seja sua precisão, distinguir ambientes baseando-se somente nas varreduras produzidas por esse tipo de sensor é uma tarefa árdua, haja em vista que quase todos os ambientes internos tendem a ser compostos por paredes retilíneas e cantos sendo sua similaridade um empecilho à sua distinção.

Tendo em vista a necessidade de constantemente buscar locais já visitados que sejam próximos da pose em que o robô se encontra, um problema que seria muito custoso computacionalmente, o sistema mantém uma árvore KD cujos nós referenciam as poses registradas ao longo do percurso. Esses nós da árvore mantém referências para os nós do grafo de poses o que permite que se obtenha de modo ágil um conjunto de pontos registrados na memória do sistema que sejam os mais próximos da presente pose. A cada instante de tempo o sistema inicia sua procura por candidatos a fechamentos de laço começando pela busca na supracitada árvore KD. Em seguida, utiliza-se a estimativa da matriz de covariâncias entre a pose atual e a candidata, extraída da matriz de informação do sistema, para calcular a distância de Mahalanobis (MAHALANOBIS, 1936) entre a pose atual e as possíveis candidatas a determinar um fechamento de laço. A distância de Mahalanobis d_{ij} entre x_i e x_j é definida pela Equação 29:

$$d_{ij} = \sqrt{(x_i - x_j)^T \cdot \Omega_{ij}^{-1} \cdot (x_i - x_j)}, \tag{29}$$

onde Ω_{ij} é a matriz de covariâncias entre as variáveis de interesse x_i e x_j.

Como é usual na literatura, adota-se a hipótese de que as poses podem ser aproximadas por Gaussianas multidimensionais, de modo que pode se utilizar uma distribuição χ^2 com dois graus de liberdade para obter um limiar para aceitar-se ou rejeitar-se a hi-

pótese de que duas poses sejam o mesmo objeto. Os dois graus de liberdade vem do fato de que para todos os efeitos, no caso de buscarmos fechamentos de laço, a orientação das poses é irrelevante, haja vista que um ente móvel pode retornar a um ponto de onde partira por um número infinito de direções.

4.3.1 Associação de Dados

A associação de dados é um problema que trata do modo como se determina as relações entre observações obtidas em momentos diferentes. Uma discussão no tocante à associação de dados é importante por tratar dos pormenores referentes à delimitação, e posterior reconhecimento, de locais, bem como de características distintivas desses. No problema de SLAM, em particular, esse problema é central para que seja possível reconhecer correlações existentes entre medições exteroceptivas diferentes, mais importante se torna quando o intuito é estabelecer relações entre medições que se encontrem separadas por um grande intervalo de tempo. É patente que a capacidade de fechar laços, essencial para limitar o acúmulo de erros proveniente da composição sucessiva de poses incertas, e construir mapas consistentes. As principais dificuldades que se apresentam, diremos que são de dois tipos principais: aquelas referentes às características do sensor e aquelas referentes às características do ambiente em que o processo se sucede. É importante notar que as abordagens aplicadas ao problema de associação de dados se subdividem, essencialmente, em dois campos: há as que utilizam um processo de busca no espaço de configurações possíveis, e objetivam determinar a localização, ou mais frequentemente, a localização relativa, do robô que leve à máxima sobreposição entre os dados observados e aqueles registrados no mapa e há aquelas que realizam a busca no espaço de correspondências e procuram uma hipótese de correspondências consistente com o que foi observado, para posteriormente calcular a localização do robô. Sobre as abordagens ambientadas no espaço de configurações, dir-se-á serem aplicáveis a conjuntos de dados sem pré-processamento (*raw data*). As abordagens que operam no espaço de correspondências se baseiam, usualmente, em características que possam ser identificadas mediante o processamento de dados provindos de sensoreamento. No caso de varreduras de LIDAR é comum que se utilize características como: linhas e cantos, dentre outras (KALLASI; RIZZINI; CASELLI, 2016; NUNEZ et al., 2006; TIPALDI; BRAUN; ARRAS, 2014; KALLASI; RIZZINI, 2016).

4.3.2 Máxima Verossimilhança

O método mais simples de realizar associação de dados se baseia no uso da probabilidade de que esses sejam realizados. Isso é feito calculando a probabilidade de que cada possível característica do ambiente, que seja de conhecimento do agente, seja um par adequado à medição verificada. Esse método é frequentemente referenciado pelo nome de método do vizinho mais próximo (*Nearest Neighbour* – NN).

De um modo geral a estimação probabilidade de uma associação pode ser realizada para qualquer tipo de distribuição de probabilidade, desde que a função densidade de probabilidade possa ser calculada para qualquer medição possível. Na prática, há que se estabelecer de antemão alguns pressupostos para que seja possível representar a supramencionada probabilidade de modo analítico, são eles: o conhecimento da densidade de probabilidade do erro mensuração conhecido e o conhecimento, *a priori*, das características do ruído de medição, por exemplo seu desvio padrão e média. De modo mais concreto, seja o ruído de medição normalmente distribuído e sejam sua média e desvio padrão conhecidos. Seja a Equação 30:

$$f_{ij} = \frac{1}{(2 \cdot \pi)^{\frac{n}{2}} \sqrt{S_j}} \cdot \exp\left(-\frac{1}{2} \cdot \nu_{ij}^T \cdot S_j^{-1} \cdot \nu_{ij}\right), \tag{30}$$

onde f_{ij} é a função densidade de probabilidade, ν_{ij} é a inovação para o par $\{i, j\}$, S_j é a matriz de covariância da inovação, e n é a dimensão do *vetor de inovação* que é definido como a diferença entre a medição verificada e a que se estimara dada a associação $e_l = \{i, j\}$ que é definida pela Equação 31:

$$e_l = \max_j f_{ij}. \tag{31}$$

Associações baseadas no princípio da máxima verossimilhança podem ser equipadas com a habilidade de rejeitar medições espúrias ou de aceitar medições de características anteriormente desconhecidas. Para tal, basta que se adote um limiar para a probabilidade de uma mensuração inválida ou um marcador desconhecido até então. A seleção desse tipo de limiar pode ser feita de modo rigoroso considerando-se que o expoente da equação 30 é da forma dada pela Equação 33 e segue uma distribuição χ^2.

A distribuição χ^2 com k graus de liberdade é a distribuição da soma do quadrado dos valores de k variáveis independentes e normalmente distribuídas e padronizadas (MULHOLLAND; JONES, 1968). Essa distribuição possui uma função densidade de probabili-

dade dada pela Equação 32:

$$p(x) = \frac{1}{2^{\frac{k}{2}} \cdot \Gamma(\frac{k}{2})} \cdot \exp(-\frac{x}{2}) \cdot x^{\frac{k}{2}-1}, \tag{32}$$

onde $\Gamma(\frac{k}{2})$ é a função gama. Duas propriedades interessantes da distribuição χ^2 são o fato de sua média ser igual ao seu número de graus de liberdade k e sua variância ser $2 \cdot k$ (MULHOLLAND; JONES, 1968). Ademais, que se note ser a distribuição χ^2 nula para qualquer $x < 0$. A Figura 11 ilustra o comportamento da densidade de probabilidade de uma distribuição desse gênero com a mudança do número de graus de liberdade k. Na Figura 11, pode-se observar que, diferentemente da distribuição Gaussiana, a χ^2 não é simétrica. Entretanto, é possível observar que o aumento do número de graus de liberdade faz com que essa distribuição se aproxime de uma Gaussiana, consoante esperado pelo teorema do limite central.

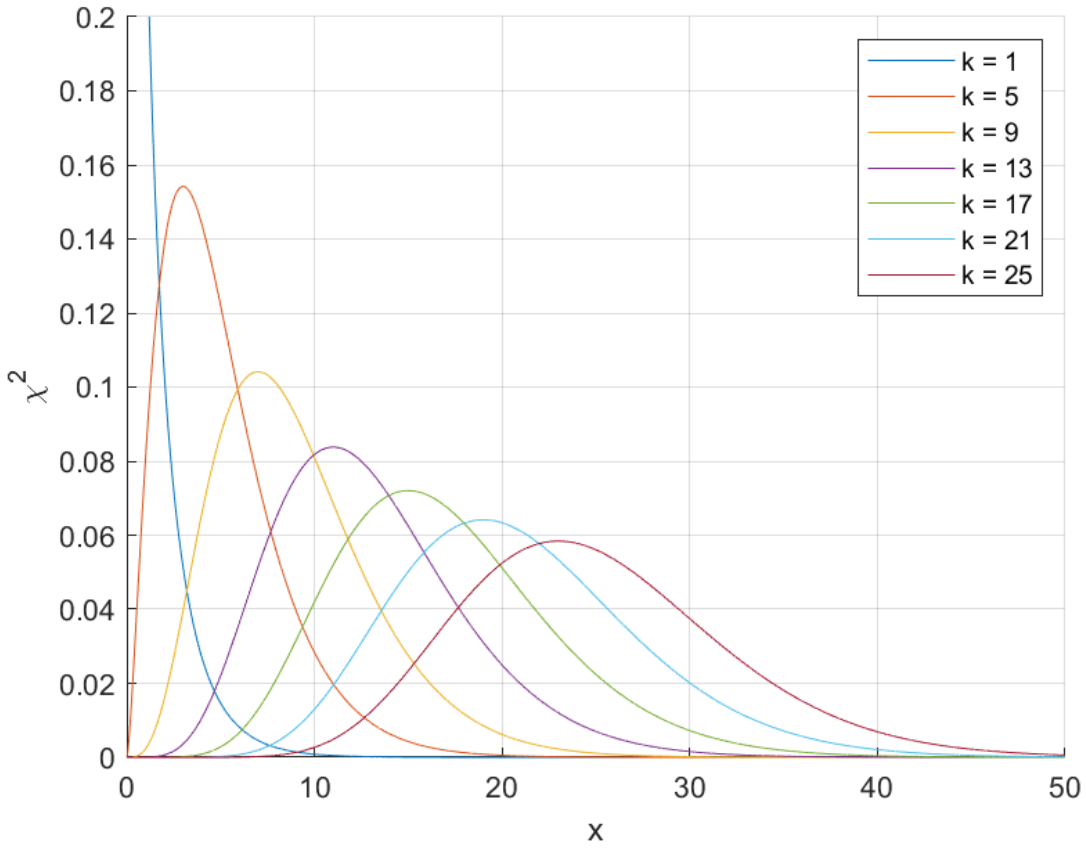


Figura 11: Função densidade de probabilidade da distribuição χ^2 para diferentes graus de liberdade k.

O método da compatibilidade individual não é, no sentido estrito da expressão, um modo pelo qual se possa rigorosamente determinar uma associação de dados. Em

verdade a abordagem consiste da eliminação de pareamentos que sejam estatisticamente improváveis, ou incompatíveis, do conjunto de possíveis associações. Advém desse *modus operandi* o fato de alguns autores se referirem ao método como sendo uma forma de redução de ambiguidade, em contraste à expressão gerenciamento de ambiguidade que se refere àqueles métodos, como a máxima verossimilhança, que fazem de modo explícito escolhas acerca dos pareamentos que consideram associações de dados válidas.

Isso pode ser realizado notando que, como dissemos na última alínea de seção predecessora, o expoente da Equação 30, a qual descreve a distribuição das inovações, segue uma distribuição do tipo χ^2. Seja M_{ij} definido pela Equação 33:

$$M_{ij} = \nu_{ij}^T \cdot S_{ij}^{-1} \cdot \nu_{ij}, \tag{33}$$

onde M_{ij} é o quadrado da distância de Mahalanobis (MAHALANOBIS, 1936), definida na Equação 29. O teste de compatibilidade individual consiste então de uma verificação acerca da satisfação da Equação 34:

$$M_{ij} < \gamma_k, \tag{34}$$

por parte de M_{ij}.

O valor de γ_k é completamente determinado se forem dados dois parâmetros: a dimensão do vetor de inovação (k) e a taxa de aceitável de associações corretas. Dado que M_{ij} é distribuído segundo uma distribuição χ^2 basta que se determine um valor de x tal que $\gamma_k = \int_{-\infty}^{x} p(x)dx$, onde $p(x)$ é definida pela Equação 32. Quanto à taxa de acertos, é bastante comum que se encontre o uso do limiar definido para que se obtenha 95%.

4.4 Considerações Finais do Capítulo

Nesse capítulo foram apresentados os conceitos básicos no tocante ao alinhamento de varreduras, à representação do mapa e à detecção de retornos a locais que se visitara. Na descrição do alinhamento de varreduras trata-se da função objetivo e do uso de árvores KD para busca de correspondências entre as varreduras. Na parte sobre descrição da representação do mapa foi descrito o uso de mapas de ocupação e a forma como são manipulados para a incorporação da informação espacial adquirida ao longo do processo de construção do mapa. O capítulo se encerra descrevendo conceitualmente como são detectados os retornos a regiões previamente observadas. O capítulo seguinte será dedicado à descrição das técnicas de otimização por inteligência de enxame que empregamos.

Capítulo 5

TÉCNICAS DE OTIMIZAÇÃO

ESTE capítulo apresenta as técnicas de otimização de enxame que serão empregadas para o desenvolvimento do trabalho proposto. Na Seção 5.1, são descritas as técnicas de otimização por inteligência de enxame. Na Seção 5.2 são realizados experimentos que visam caracterizar o comportamento das técnicas de otimização por inteligência de enxame que escolhemos.

5.1 Otimização De Enxame

A inteligência de enxame é um campo da inteligência artificial que se inspira no comportamento coletivo de seres como formigas, abelhas, vermes, revoadas de pássaros, e cardumes, dentre outros. Muito embora os indivíduos sejam, usualmente, pouco sofisticados em seu *modus operandi*, a aparente coordenação de seus comportamentos individuais faz surgir um complexo padrão de interação entre o grupo e o ambiente em que ele se encontra. O processo de busca é uma interação e dicotomia entre as etapas conhecidas como *exploration* e *exploitation* (KENNEDY, 2006). Para algoritmos desse gênero denota-se exploração (*exploration*) o processo de descobrimento das variadas soluções que podem existir no *espaço de busca*; e *exploitation* ou, numa tradução livre, *usufruto* o processo de focar a busca nas cercanias das boas soluções já encontradas, tirando, assim, proveito da informação já adquirida. Todo e qualquer algoritmo que se baseie em alguma sorte enxame depende diretamente do fluxo de informação nesse enxame para seu bom funcionamento. Sobre isso, e do mais alto interesse para a aplicabilidade de um algoritmo, diremos que a localidade, isso é a formação de pequenos grupos que se comunicam mais intimamente entre si do que com o restante dos membros do enxame, fomenta a exploração simultânea de diversos ótimos locais pelo aumento da diversidade das fontes de informação do enxame.

5.1.1 Otimização por Enxame de Partículas

Um algoritmo de otimização por enxame de partículas (*Particle Swarm Optimization* - PSO) mantém um conjunto de entes denominados partículas, cada qual representando uma solução codificada em potencial. Sob esse aspecto a noção de enxame possui nos algoritmos de inteligência de enxame um papel análogo ao desempenhado pela população em sistemas que se valem de computação evolucionária. A ideia central sendo a de que a inter-relação entre a experiência individual de cada partícula interaja com a experiência coletiva, direcionando, dessa maneira, a busca pelo espaço de soluções. O processo de busca ocorre iterativamente e cada partícula, ao longo de sua trajetória pelo espaço de buscas, mantém um registro da melhor posição encontrada até a presente iteração durante sua história. Deveras, a preservação desse tipo de memória de cada partícula é uma das grandes distinções entre o PSO e algoritmos de computação evolutiva, por exemplo. Essa interação entre indivíduo é bando quantificada, e descrita, em termos da variação da velocidade dos membros do enxame. A velocidade de cada partícula é recalculada, a cada iteração, como uma composição de três componentes distintas: uma componente social, uma componente cognitiva e uma componente de inércia. A chamada componente de inércia exerce um papel análogo à componente resultando da força homônima na mecânica Newtoniana, *i.e.* opõe-se à alteração da tendência de movimento. Essa componente introduz, portanto um efeito que se opõe a mudanças drásticas na trajetória da partícula, levando a uma preservação do presente curso pelo espaço de buscas. Denomina-se componente cognitiva a parcela da velocidade de cada partícula devida à diferença entre a atual posição e a melhor já encontrada. A componente cognitiva tem por objetivo causar a busca por novas soluções em potencial nas cercanias das soluções boas já encontradas, tornando essas polos de atração para as partículas. Por fim, a chamada componente social é responsável por direcionar os indivíduos do enxame para as melhores posições encontradas de que eles tenham conhecimento, quer tenham essas posições sido descobertas por eles próprios ou por um vizinho com o qual haja comunicação desobstruída. Vale ressaltar nesse ponto que a distinção entre as topologias das vizinhanças empregadas pelo sistema de otimização possuem um efeito especialmente exacerbado sobre a componente social da velocidade.

5.1.1.1 Algoritmo e Parâmetros Básicos

Até o presente momento concentra-mo-nos em discutir de modo razoavelmente abstrato o impacto dos diversos parâmetros presentes no PSO. Para continuarmos a discussão, no entanto faz-se necessário que introduzamos uma nomenclatura adequada. Com isso em mente, partimos então para uma definição em alto nível do algoritmo e das equações de atualização de posição e da velocidade. Sejam $x_i(t)$ a posição da partícula i no instante t, $v_i(t)$ a componente da velocidade da partícula i no instante t, w o coeficiente de inércia, c_1 o coeficiente cognitivo, c_2 o coeficiente social, n o tamanho do enxame e f a função objetivo. O algoritmo básico para o PSO com *global best* (melhor global) pode então ser enunciado como se pode observar no Algoritmo 4 (KENNEDY; EBERHART, 1995; KENNEDY, 2006).

Algoritmo 4 PSO

Entrada f, n, c_1, c_2, w
1: **Inicializa** o enxame
2: **Repita**
3: **Para** $i = 1, 2, ..., n$ **Faça**
4: **Se** $f(particula_i) < f(pbest_i)$ **Então**
5: $pbest_i = particula_i$
6: **Fim Se**
7: **Se** $f(pbest_i) < f(gbest)$ **Então**
8: $gbest = pbest_i$
9: **Fim Se**
10: **Fim Para**
11: **Para** $i = 1, 2, .., 3$ **Faça**
12: **Atualizar** a velocidade segundo a Equação 35.
13: **Atualizar** a posição segundo a Equação 36
14: **Fim Para**
15: **Até** Seja satisfeita alguma condição de parada
16: **Retornar** $gbest$

O funcionamento do Algoritmo 4 pode ser explicado de modo sucinto como a sucessão dos seguintes passos: a inicialização e a iteração de um processo de movimentação do enxame na busca por soluções de boa qualidade. A inicialização do enxame consiste da geração aleatória de partículas (soluções codificadas) que se espalhem pelo espaço de busca. Enquanto nenhuma condição de parada tiver sido satisfeita, repete-se o processo de movimentação do enxame. O processo de movimentação do enxame depende da qualidade dos membros do enxame e das experiências passadas de cada partícula. A cada iteração, avalia-se a qualidade da solução codificada por cada partícula e, se essa for a melhor

solução encontrada por ela atribui-se sua posição à variável *pbest*. A cada vez que uma partícula for avaliada com relação à sua qualidade, compara-se o *pbest* dessa partícula à melhor posição encontrada durante todo o processo pelo enxame e, se a qualidade do *pbest* da presente partícula for considerada melhor que a do melhor global *gbest*, atribui-se ao *gbest* o valor do *pbest*. Em seguida, cada partícula tem sua velocidade atualizada e, por fim, a posição da partícula é atualizada. Ao final do processo de otimização o *gbest* é retornado como a solução do problema.

No Algoritmo 4 a atualização da velocidade de cada partícula é feita segundo a Equação 35:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (pbest_i(t) - x_i(t)) + c_2 \cdot r_2 \cdot (gbest(t) - x_i(t)), \qquad (35)$$

onde $pbest_i(t)$ a melhor posição na história da partícula até o instante t, $gbest(t)$ é o melhor global até o instante t e r_1 e r_2 são números pseudo-aleatórios amostrados de uma distribuição uniforme no intervalo $[0, 1]$. Sobre os números aleatórios r_1 e r_2, diremos ainda que são no para cada dimensão do problema o passo de atualização há de gerar um desses números de modo independente. A atualização da posição de cada partícula é realizada segundo a Equação 36:

$$x_i(t+1) = x_i(t) + v_i(t+1). \qquad (36)$$

No Algoritmo 4 não especificou-se quais são as condições de parada a serem satisfeitas, embora isso seja fundamental para qualquer técnica de otimização aproximada. A definição de uma condição de parada adequada pode ser feita de diversos modos, sendo os mais comuns: a especificação de um número máximo de iterações, a definição de uma tolerância para o valor da função objetivo, a definição de um limiar de variação do valor da função objetivo e a definição de um limiar de variação do argumento da função objetivo. Os casos em que se especifica limiares de variação são, em verdade, medidas de estagnação e esses limiares estão tipicamente associados a um número de iterações máximo para que as quantidade por eles limitadas os violem antes que o processo de otimização como um todo seja terminado. Com essas definições estabelecidas, podemos então tentar quantificar a importância de cada um dos parâmetros citados para a convergência e eficiência do processo. Diversos trabalhos foram realizados sobre a seleção de parâmetros para um processo eficiente de otimização usando o PSO. Os parâmetros denotados c_1 e c_2 são também conhecidos como coeficientes de aceleração. Como esse nome implica, valores

baixos de c_1 e c_2 levam a trajetórias mais suaves, e permitem que as partículas explorem o espaço de buscas por mais tempo, afastando-se temporariamente de regiões boas, antes de serem atraídas para uma condição de equilíbrio. De modo inverso, valores altos desses dois parâmetros podem levar a trajetórias mais abruptas e fazer com que as partículas convirjam mais velozmente para boas soluções, mas também podem fazê-las saltar para além regiões boas do espaço de soluções.

5.1.1.2 Comportamento do Algoritmo

Em (ENGELBRECHT, 2006) encontramos referência a modelos simplificados do PSO canônico em que são suprimidas certas componentes da sua velocidade. Com efeito, a supressão de determinadas componentes que influenciam na trajetória das partículas pelo espaço de buscas permite que se tenha uma noção melhor de sua contribuição para a qualidade dos resultados obtidos. A esses modelos simplificados o autor de (ENGELBRECHT, 2006) denomina *Cognition Only Model* (modelo puramente cognitivo), *Social Only Model* (modelo puramente social) e *Selfless Model* (modelo altruísta). Esses modelos simplificados são instrutivos no sentido de que ilustram de modo inequívoco até que ponto a alteração dos três parâmetros até aqui mencionados, qual sejam: o coeficiente de inércia, o coeficiente cognitivo e o coeficiente social; afeta o comportamento do enxame e o desempenho do algoritmo de modo geral. De acordo com (KENNEDY; EBERHART, 1995), o modelo puramente cognitivo é mais suscetível a falhas que o modelo completo. Ainda de acordo com os autores de (KENNEDY; EBERHART, 1995), esse tipo de modelo de velocidade leva a um comportamento de intensa busca próxima das regiões inicialmente populadas, degenerando-se o PSO para um caso de busca aleatória em paralelo. Segundo (KENNEDY, 1997; ENGELBRECHT, 2006) o modelo puramente cognitivo apresenta ainda uma menor velocidade de convergência, como confirmado por (CARLISLE; DOZIER, 2000). O chamado modelo puramente social, foi laudado em (KENNEDY; EBERHART, 1995; CARLISLE; DOZIER, 2000; ENGELBRECHT, 2006) por ser mais veloz e mais eficiente que o modelo completo e o modelo puramente cognitivo. Esse modelo representa a tendência das partículas de imitar suas vizinhas mais bem-sucedidas. Por fim, Kennedy menciona o modelo altruísta. O modelo altruísta nada mais é que o modelo social sob a restrição de que a melhor partícula de uma vizinhança não possa ser a própria partícula, ou seja quando um membro de uma vizinhança procura quem é a partícula mais bem-sucedida nas cercanias ela se abstém da competição. Esse modelo foi descrito como sendo mais

rápido que o puramente social para alguns problemas em (KENNEDY, 1997), embora tenham sido reportados resultados insatisfatórios para o caso de otimização em ambientes dinâmicos em (CARLISLE; DOZIER, 2000).

5.1.2 Otimização dos Vaga-Lumes

O algoritmo de otimização baseado no comportamento de vaga-lumes (*Firefly Algorithm* - FA) é um algoritmo da classe de métodos de otimização baseados em inteligência de enxame (FISTER; YANG; BREST, 2013). Seus passos estão definidos no Algoritmo 5. Em sua essência é um método meta-heurístico, estocástico e bio-inspirado. A inspiração do FA, provém da bioluminescência dos vaga-lumes (*fireflies*). De fato, é justamente o cintilar periódico de luzes, característica marcante desses insetos, que, aliada à característica física do decaimento da intensidade luminosa e foi usada para a elaboração dessa estratégia de otimização. Em (FISTER; YANG; BREST, 2013) é dito que o decaimento da intensidade luminosa I segue uma relação do tipo $I \propto \frac{1}{r^2}$, onde r foi usado pelos autores para indicar a distância à fonte de luminescência. A comunicação entre insetos sociais já foi utilizada, com sucesso, por diversas vezes como inspiração para o desenvolvimento de técnicas de otimização, como nos casos de formigas e abelhas (DORIGO; CARO; GAMBARDELLA, 1999; KARABOGA, 2005). É importante reiterar que em algoritmos de enxame, os processos de tomada de decisão de cada entidade ocorrem, tipicamente, de forma descentralizada e, por essa razão é essencial que haja comunicação entre os membros do enxame para que se possa ter um comportamento coeso do conjunto.

5.1.2.1 Algoritmo e Parâmetros Básicos

Os parâmetros do FA são β_0, γ e n. O coeficiente β_0 é a atratividade de um vaga-lume quando a distância a ele é nula. A constante γ determina a taxa de decaimento da atratividade de um vaga-lume em termos da distância a ele, como pode ser visto na Equação 37. No Algoritmo 5, cada vaga-lume foi denotado por s e a função objetivo por f.

A essência do funcionamento do FA é a combinação do movimento aleatório com a atração de cada vaga-lume pelos outros membros mais luminosos do enxame. Com efeito, o vaga-lume j (s_j) é atraído ao vaga-lume i (s_i) e seu movimento é definido pela Equação 37:

$$ {s_i}^{t+1} = {s_i}^{t} + \beta_0 \cdot \exp(-\gamma \cdot r_{ij}^2) \cdot (s_j - s_i) + \alpha \cdot \epsilon_i, \tag{37} $$

Algoritmo 5 FA

Entrada β_0,γ,n
1: **Inicializa** o enxame
2: **Repita**
3: $\alpha^{(t)} = novoAlfa()$
4: **Para** $i = 1, 2, ..., n$ **Faça**
5: **Calcule** $f(s_i)$
6: **Fim Para**
7: **Ordena** os vaga-lumes s pelo valor de $f(s)$.
8: **Atribui** a s^* o vaga-lume mais apto.
9: **Move** os vaga-lumes segundo a Equação 37.
10: **Até** Seja satisfeita alguma condição de parada
11: **Retornar** s^*

onde γ é a absorção luminosa, isso é o quanto a luz se difunde pelo meio em que se propaga entre um vaga-lume e outro. O parâmetro $\alpha \in [0, 1]$ é um dos responsáveis pelo termo aleatório $\alpha \cdot \epsilon_i$ da Equação (37) e é amostrado de uma distribuição uniforme nesse intervalo. O valor ϵ_i é amostrado de uma distribuição Gaussiana. O valor de r_{ij} é a distância entre os vaga-lumes s_i e s_j, como definido pela métrica Euclidiana em duas dimensões definida na Equação 38:

$$r_{ij} = \|s_j - s_i\| = \sqrt[2]{\sum_{k=1}^{2}(s_{jk} - s_{ik})^2}. \tag{38}$$

É importante notar que, no algoritmo, o índice k no somatório da Equação (38), varia sempre no intervalo de 1 ao número máximo de dimensões do problema. O passo $novoAlfa()$ é utilizado para permitir uma variação adaptativa de α, de modo a reduzir o grau de aleatoriedade do movimento dos vaga-lumes com o passar das iterações.

5.1.2.2 Comportamento do Firefly Algorithm

O enxame de vaga-lumes é inicializado de modo aleatório, e o processo de busca é norteado pela intensidade luminosa, que é proporcional à qualidade da solução, dos vaga-lumes do enxame, como percebida pelos seus semelhantes. A essa qualidade, como vista pelos outros vaga-lumes, denomina-se atratividade (FISTER; YANG; BREST, 2013). Um fenômeno interessante, no que tange a comparação entre o FA e outros algoritmos de otimização, é que o FA apresenta dois comportamentos assintóticos notáveis, mediante o valor de γ selecionado. O primeiro quando $\gamma \to 0$, já que a atratividade se torna constante e igual a β_0 em todo o espaço de busca, e o algoritmo se comporta como um caso particular de PSO. O segundo comportamento assintótico notável, ocorre quando temos $\gamma \to \infty$, o que

faz o FA se comportar como um caso paralelo de Simulated Annealing (FISTER; YANG; BREST, 2013). Donde se conclui que todo FA apresenta um comportamento intermediário entre essas duas técnicas. No FA descrito em (GAO et al., 2013), são mencionados critérios diversos de parada , tais como a qualidade da pior solução no enxame, a qualidade da melhor solução no enxame e o número de avaliações da função de aptidão. Sobre a primeira e a segunda condições, há que se dizer que caso haja a imposição de um limiar como esse, é de se esperar que haja a convergência do algoritmo com um número menor de iterações, e consequentemente com um número menor de avaliações da função objetivo. No entanto, essa convergência vem conjugada à uma perda de precisão, *i.e.* pode haver a convergência prematura na presença de um máximo local.

5.1.3 Colmeia Artificial de Abelhas

O algoritmo de otimização baseado no comportamento de colméias de abelhas (*Artificial Bee Colony* - ABC) foi sugerido em (KARABOGA, 2005). O algoritmo baseado no comportamento de forrageamento observável em uma colmeia de abelhas produtoras de mel (KARABOGA, 2005). No modelo empregado pelo ABC a colmeia é composta de três grupos de abelhas: as operárias (*employed bees*), as abelhas em espera (*onlookers*) e as abelhas exploradoras (*Scouts*).

No ABC, cujos passos aparecem enunciados no Algoritmo 6, cada fonte de alimento representa uma solução em potencial, e sua correspondente quantidade de néctar representa a qualidade dessa solução, medida em termos da função objetivo. O número de abelhas operárias é igual ao número de soluções no enxame.

Cada ciclo do processo de busca do ABC consiste de três etapas: o envio das abelhas operárias para as fontes de alimento e a avaliação da quantidade de néctar coletado por cada uma; a seleção de fontes de alimento pelas abelhas operárias após o compartilhamento da informação das abelhas operárias; a seleção das abelhas exploradoras e seu sucessivo envio a destinações contendo possíveis fontes de alimento. Durante a fase de inicialização do ABC, um conjunto aleatório de fontes de alimento é selecionado, aos quais sucessivamente são enviadas abelhas que avaliam suas quantidades de néctar. A inicialização das fontes de alimento pode ser feita como definido pela Equação 39:

$$x_{ij} = l_j + r_1 \cdot (u_j - l_j), \tag{39}$$

Algoritmo 6 ABC

Entrada n,N

1: **Envie** as abelhas exploradoras para fontes iniciais de alimentos escolhidas usando a Equação 39.
2: **Repita**
3: **Envie** operárias para as fontes de alimentos e determine sua quantidade de néctar.
4: **Selecione** fontes vizinhas para cada operária usando a Equação 39.
5: **Calcule** a probabilidade de preferência das fontes pelas abelhas em espera usando a Equação 40.
6: **Envie** as abelhas em espera para fontes de alimentos e determine suas quantidades de néctar.
7: **Interrompa** o usufruto das fontes exauridas.
8: **Mova** aleatoriamente as abelhas exploradoras usando a Equação 39.
9: **Memorize** a melhor fonte de alimentos descoberta até a presente iteração.
10: **Até** Seja satisfeita alguma condição de parada
11: **Retornar** a melhor fonte de alimentos descoberta até a presente iteração.

onde x_{ij} é a dimensão j da fonte de alimento i, r_1 é um número pseudo-aleatório uniformemente distribuído pertencente ao intervalo $[0, 1]$, l_j é o limite inferior ao longo da direção j do subconjunto do domínio da função objetivo onde se realiza a busca e u_j é o limite superior do supracitado conjunto.

Após seu retorno à colmeia, essas abelhas compartilham o que foi descoberto acerca das fontes de alimento que exploraram com as outras abelhas, que se encontram na região de dança da colmeia. Na segunda fase, completado o compartilhamento da informação referente à qualidade das fontes de alimento descobertas, as abelhas operárias retornam às fontes de alimento que visitaram na iteração predecessora, haja vista o fato de tal localidade se encontrar registrada na memória da abelha operária, e então escolhe uma nova fonte de alimento nas cercanias daquela à qual se dirigira. Na terceira fase, cada abelha em espera escolhe uma fonte de alimentos baseada na dança das abelhas operárias. É importante ressaltar que a probabilidade de seleção de uma fonte de alimentos aumenta com o aumento da quantidade de néctar lá detectada. Deveras, a probabilidade de seleção de uma determinada fonte p_i é definida de acordo com a Equação 40:

$$p_i = \frac{f(x_i)}{\sum_{k=1}^{N} f(x_k)}, \tag{40}$$

onde $f(x_i)$ denota o valor da função objetivo na fonte de alimento i e N representa o número de fontes de alimento.

É importante notar que cada abelha operária, ou em espera, altera a fonte de alimento em sua memória com uma dada probabilidade, e essa modificação é mantida

desde que a fonte de alimentos modificada seja mais profícua que a original. O algoritmo de seleção pode, por conseguinte, ser caracterizado como guloso. A alteração da fonte de alimento é determinada pela Equação 41:

$$v_{ij} = x_{ij} + r_{ij} \cdot (x_{ij} - x_{kj}), \tag{41}$$

onde r_{ij} é um número pseudo-aleatório uniformemente distribuido tal que $r_{ij} \in [-1, 1] \subset \mathbb{R}$, $k \in \{1, \cdots, N\} \subset \mathbb{N}$, $i \in \{1, \cdots, N\} \subset \mathbb{N}$, $j \in \{1, \cdots, N\} \subset \mathbb{N}$, x_{ij} é a coordenada j da fonte de alimento i e x_{kj} é a coordenada j de uma fonte de alimentos selecionada aleatoriamente.

Após o término do processo de exploração por todas as abelhas operárias, as abelhas em espera selecionam fontes de alimento para si próprias de modo probabilístico, com base no que é relatado pelas operárias relativamente à qualidade das fontes de alimento exploradas.

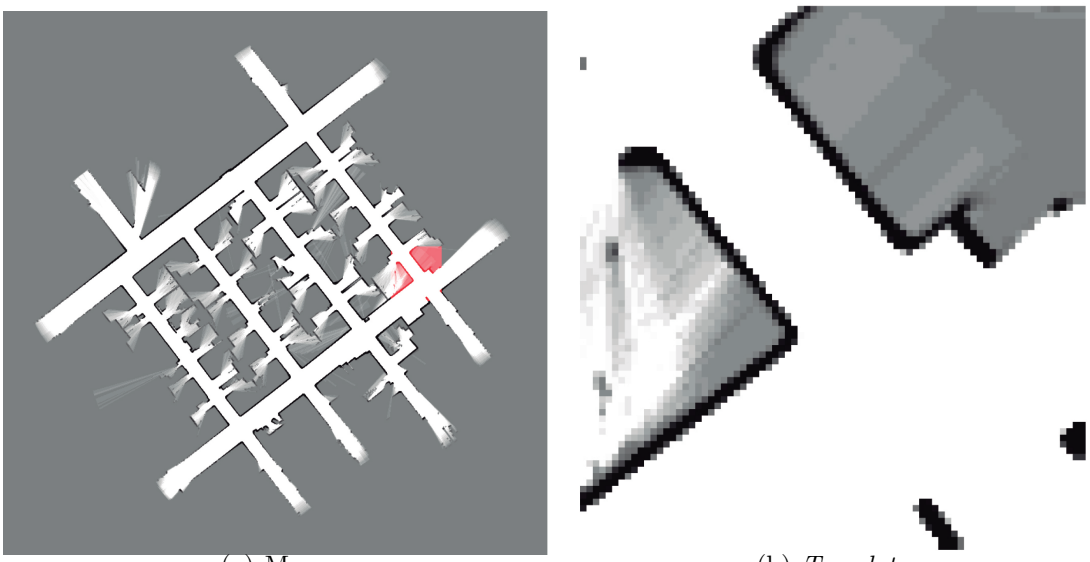
O processo de seleção de fontes de alimento pelas abelhas é controlado por um parâmetro de controle denominado limite, ou seja, se a solução representada por uma fonte de alimentos não apresenta uma melhora em sua qualidade após uma quantia pré-fixada de tentativas, aquela fonte de alimentos é abandonada e a abelha operária que dela se encarregava torna-se uma exploradora. Com essa explanação, evidencia-se que as abelhas em espera e as empregadas são responsáveis pelo processo de *exploitation* (usufruto), já as exploradoras controlam o processo de exploração.

De um modo geral, o algoritmo possui como parâmetros de controle o tamanho do enxame, o limite, o número de abelhas em espera, geralmente estipulado como constituindo metade do enxame, o número de abelhas operárias, usualmente fixado em 50% do tamanho do enxame e o número de exploradoras, que normalmente é fixado como uma (KARABOGA, 2005). Para além disso, no Algoritmo 6 não foi explicitado um critério de parada. Há critérios diversos de parada que podem ser adotados, tais como: a qualidade da pior solução no enxame, a qualidade da melhor solução no enxame e o número de avaliações da função de aptidão.

5.2 Caracterização Inicial

Objetivando avaliar o desempenho, sob um ponto de vista majoritariamente *qualitativo*, das meta-heurísticas expostas, realizamos um ensaio com cada uma delas, no qual se-

lecionamos o mapa da Figura 12 como objeto de teste e utilizamos a métrica do erro médio quadrático como modo de avaliar a qualidade das soluções providas. Com cada meta-heurística repetimos os testes 3000 vezes afim de tornar razoável a hipótese de que os erros sejam aproximadamente normalmente distribuídos. Realizamos então um recorte incluindo um canto, observado de um ponto de vista em particular e, em seguida, tentamos determinar a correspondência do recorte por meio da pose que induzisse o menor erro. No primeiro teste utilizamos o recorte visualizado como aparece na Figura 12(b), ausente rotação, embora tenhamos deixado o problema de determinação em aberto para a técnica de otimização. O recorte foi obtido pegando a região com canto inferior direito no pixel de coordenada $(459; 693)$ que pode ser vista na Figura 12(b).



(a) Mapa (b) *Template*

Figura 12: Mapa e *template* usados para a avaliação preliminar das técnicas de enxame que serão empregadas.

A região de onde o recorte foi retirado encontra-se marcada por um tom avermelhado na Figura 12(a). A região em destaque na Figura 12(a) encontra-se ampliada na Figura Figura 12(b) para que se possa observá-la em mais detalhes. Ela se assemelha aos variados outros cantos visíveis na Figura 12(a), o que torna visível o carácter multimodal do problema de otimização subjacente à tarefa de localização. Para todos os testes que realizamos impusemos um limite de 1000 iterações. Realizamos inicialmente o experimento computacional com o PSO usando 20 partículas com coeficientes de inércia e cognitivo tendo ambos o valor de $1,49$ e com a inércia fixada em $0,7$. O número de avaliações

da função de aptidão ao longo de cada teste foi coletado no histograma que se encontra exibido na Figura 13.

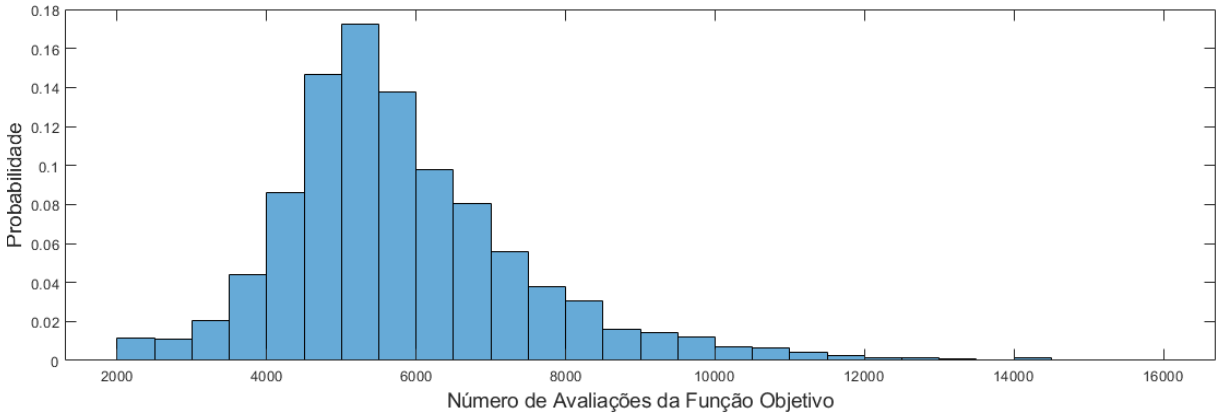


Figura 13: Histograma de chamadas da função objetivo realizadas pelo PSO.

Na Figura 13 pode-se observar a distribuição do número de chamadas à função de aptidão até que houvesse convergência ou a interrupção do algoritmo. O número médio de $1243,5000$ chamadas, com um desvio padrão de $388,9200$ chamadas. A acurácia ao longo de todas as chamadas pode ser ilustrada de modo sucinto pelo *boxplot* exibido na Figura 14. Pôde-se verificar, conforme Figura 14, que sob as condições impostas o PSO obteve como mediana de x, y e θ os valores de $454,1600$;$496,4200$ e $0,2651$ respectivamente. Esses valores indicam que o erro percentual absoluto médio cometido pelo PSO foi de $1,0544\%$ em sua coordenada x, $28,3660\%$ em sua coordenada y e o algoritmo cometeu um Erro absoluto em θ foi de $0,2651$ radianos.

Em seguida, realizamos os mesmos testes com o FA usando 20 vaga-lumes com α, β e γ tendo os valores de $0,5$; $0,2$ e $1,0$ (os valores padrão no código disponibilizado pelo autor). O número de avaliações da função de aptidão ao longo de cada teste foi coletado no histograma que se encontra exibido na Figura 15, pode-se observar a distribuição do número de chamadas à função de aptidão até que houvesse convergência ou a interrupção do algoritmo.

O número médio de chamadas observado foi $1,4408 \cdot 10^5$, com um desvio padrão de $4,1265 \cdot 10^3$ chamadas. A acurácia ao longo de todas as chamadas pode ser ilustrada de modo sucinto pelo *boxplot* exibido na Figura 16, onde pôde-se verificar que sob as condições impostas o FA obteve como mediana de x, y e θ os valores de $459,0000$; $547,0000$ e $0,0000$, respectivamente. O que indica que o FA cometeu um erro percentual absoluto

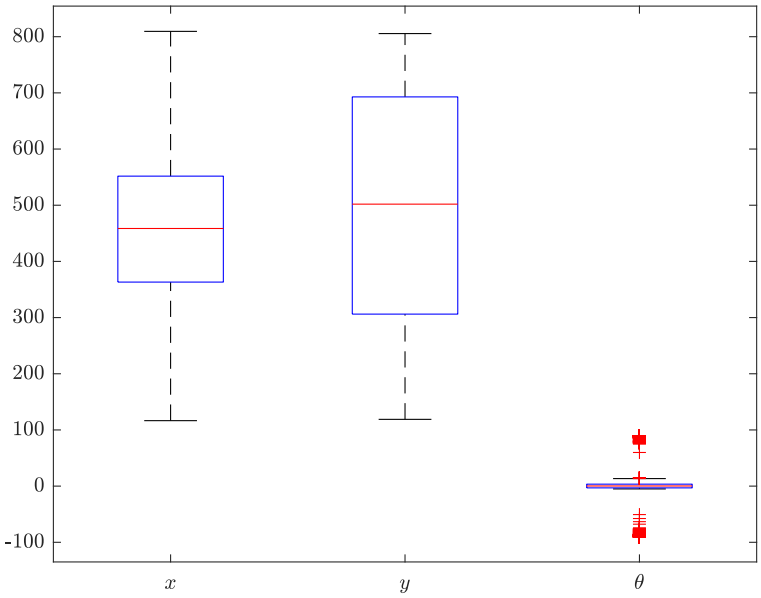


Figura 14: *Boxplot* das roto-translações estimadas pelo PSO.

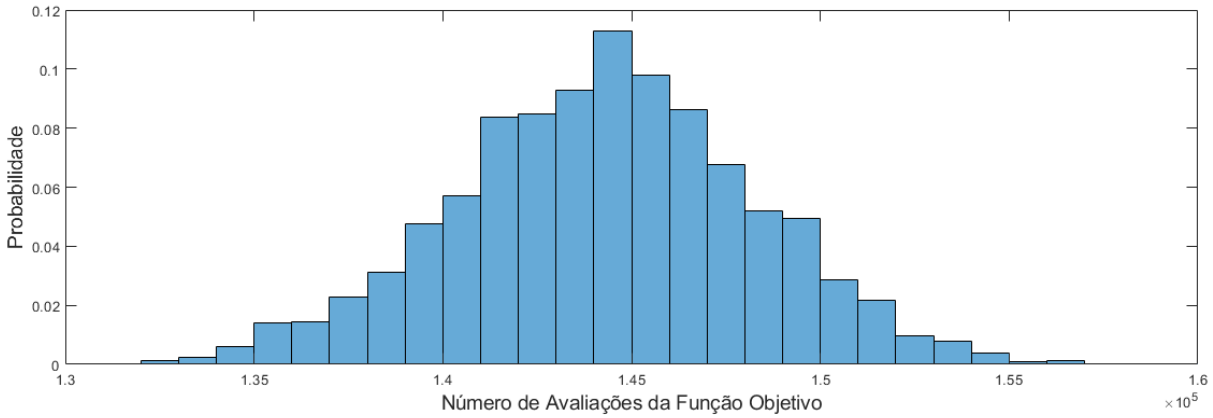


Figura 15: Histograma de chamadas da função objetivo realizadas pelo FA.

médio de $0,0000\%$ na componente x, um erro percentual absoluto médio de $21,0670\%$ na componente y e um Erro absoluto de $0,0000$ radianos.

Por fim, realizamos os mesmos testes com o ABC usando uma colônia com 20 abelhas com 10 fontes de alimento e limite de usufruto de 100. O número de avaliações da função de aptidão ao longo de cada teste foi coletado no histograma que se encontra exibido na Figura 17, onde pode-se observar a distribuição do número de chamadas à função de aptidão até que houvesse convergência ou a interrupção do algoritmo.

Pudemos verificar no corpo de dados coletado um número médio de $2,0082 \cdot 10^5$ chamadas, com um desvio padrão de $37,6600$ chamadas. A acurácia ao longo de todas as chamadas pode ser ilustrada de modo sucinto pelo *boxplot* exibido na Figura 18, onde

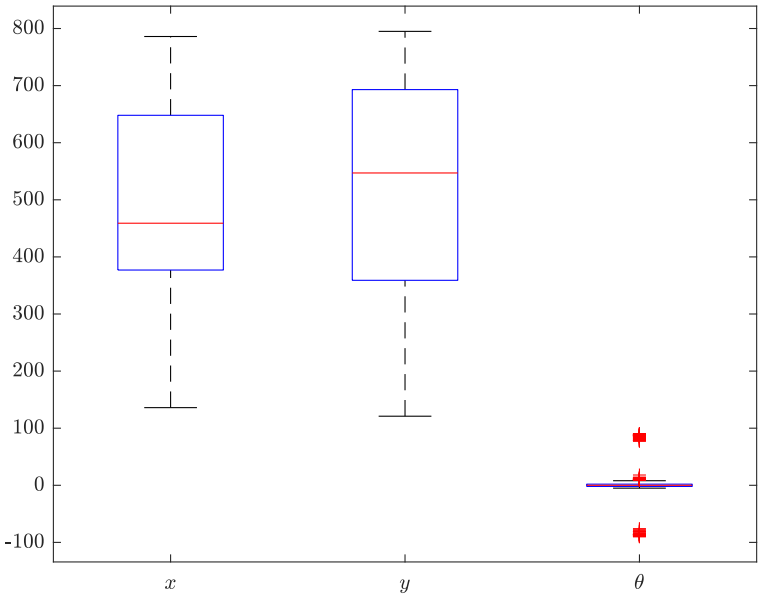


Figura 16: *Boxplot* das roto-translações estimadas pelo FA.

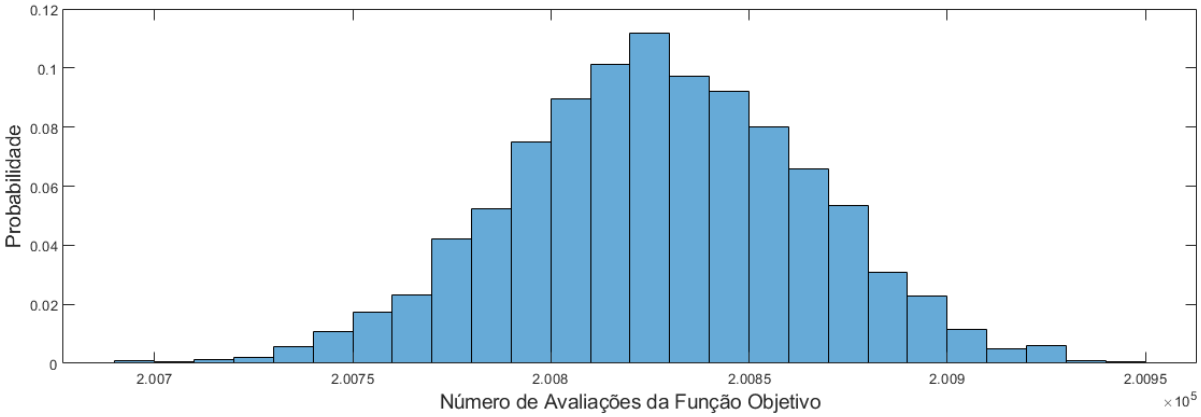


Figura 17: Histograma de chamadas da função objetivo realizadas pelo ABC.

observa-se que sob as condições impostas o ABC obteve como mediana de x, y e θ os valores de $459,4200$; $691,5900$ e $0,3148$. Isso significa que o ABC cometeu um erro percentual absoluto médio de $0,0915\%$ em sua coordenada x, um erro percentual absoluto médio de $0,2035\%$ e um erro absoluto em θ de $0,3148$ radianos.

Comparando os resultados, verificamos que o PSO mostrou-se o mais eficiente em termos de número do número de avaliações da função objetivo até que alguma condição de parada se satisfizesse, consoante o que se encontra ilustrado na Figura 19, sendo o ABC o segundo colocado e o FA o terceiro.

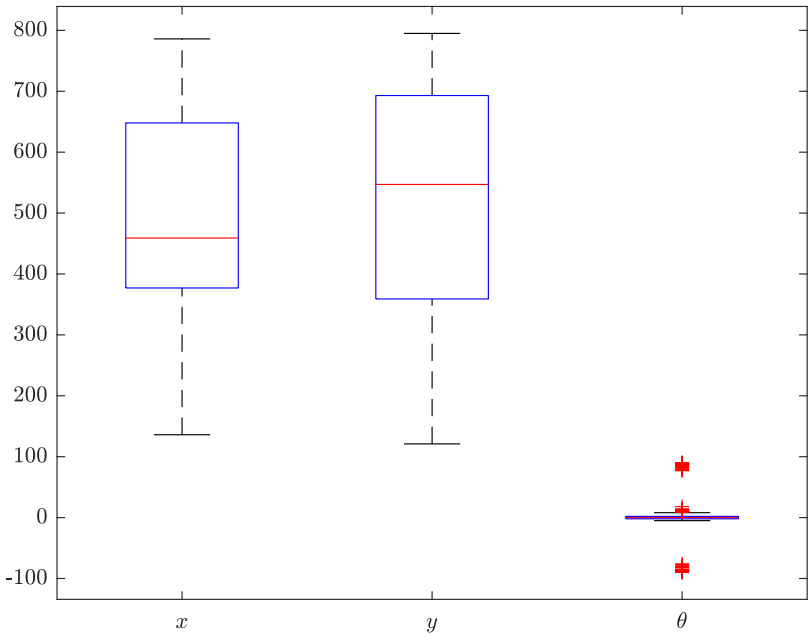


Figura 18: *Boxplot* das roto-translações estimadas pelo ABC.

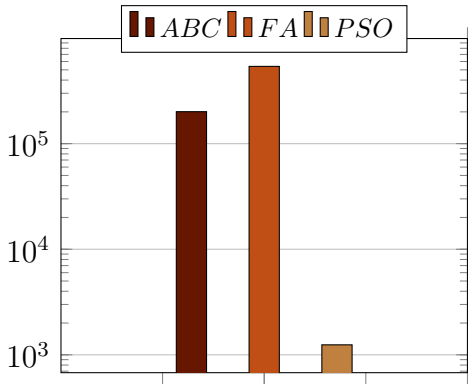


Figura 19: Comparação da media de avaliações da função de aptidão realizada por cada meta-heurística.

Verifica-se também que o desvio da mediana dos resultados, ao longo do processo de experimentação seguiu o padrão exibido na Figura 20, onde mostra-se num diagrama de barras a comparação dos valores da mediana de cada meta-heurística. O ABC mostrou-se, no presente teste, a meta-heurística mais precisa dentre as três que exibimos aqui, com o FA em segundo lugar e o PSO em último.

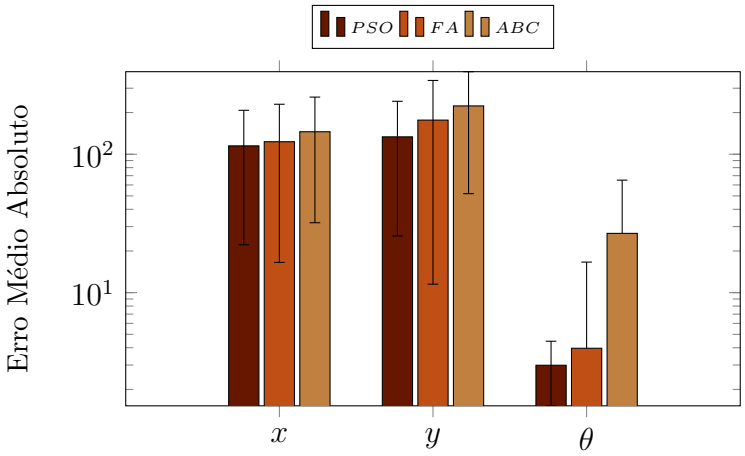


Figura 20: Comparação do valor absoluto do desvio percentual da mediana em relação às coordenadas reais do recorte.

Os resultados dessa breve experimentação mostram que os algoritmos selecionados apresentam, entre si, diversos *trade-offs* entre o custo computacional e a consistência e acurácia dos resultados, mostrando-se promissores para maior investigação. Com base nos testes desse capítulo espera-se que, na implementação final o PSO seja o algoritmo mais rápido e o menos preciso. O ABC aparenta realizar um *trade-off* interessante tendo exibido resultados que o apontam como o mais preciso dos três métodos de otimização. O FA exibiu um desempenho medíocre não sendo nem o mais rápido nem o mais preciso. Essas conclusões são claramente sustentadas pelos valores apresentados dos erros percentuais médios absolutos das coordenadas translacionais, e pela comparação dos valores de erros absolutos rotacionais.

Um carácter interessante de se observar para um problema como o SLAM é a consistência dos resultados fornecidos pelas meta-heurísticas, afinal de pouca valia é uma técnica que fornece resultados que variem demasiadamente. Ao longo dessa subsecção exibimos nas Figuras 14, 16 e 18 que ilustravam sob a forma de *boxplots* a dispersão dos resultados obtidos por cada técnica em cada coordenada. Na Figura 20 exibimos no diagrama de barras o desvio padrão observado ao longo desse experimento para cada coordenada e técnica com o propósito de facilitar a interpretação dos resultados pelo leitor.

5.3 Considerações Finais do Capítulo

Nesse capítulo fizemos uma descrição de alto nível acerca das meta-heurísticas de otimização que empregaremos. Procedendo dessa maneira, a extensão dos conceitos aqui definidos

e ilustrados se dará de modo natural e simples, ao caso particular do qual trataremos nos próximos capítulos, qual seja o uso de técnicas de otimização dessa sorte para a resolução do problema de localização e mapeamento simultâneos. Ademais, os testes realizados na Seção 5.2 permitem que especulemos acerca da adequação das meta-heurísticas ao problema de alinhamento de varreduras no contexto do SLAM. Sobre isso, esperamos que o PSO, tendo apresentado a maior velocidade, seja mais apropriado para situações em que o tempo de execução seja o fator limitante e que o ABC seja o mais adequado quando se dispuser de mais tempo e desejar-se uma maior acurácia dos resultados. O FA apresenta um comportamento intermediário e esperamos que sua aplicação seja também de precisão e velocidade medíocres em relação aos extremos observados nos casos do PSO e do ABC. No próximo capítulo, será descrito o método proposto para a solução do SLAM.

Capítulo 6

MÉTODO PROPOSTO

NE sse Capítulo, será feita uma descrição do sistema implementado. Na Seção 6.1 é feita uma descrição geral do sistema. Na Seção 6.2 é descrito o sistema de alinhamento de varreduras, que é o cerne do trabalho exposto por ter papel central no funcionamento dos outros dois subsistemas. Ademais, na Seção 6.3 é descrito o sistema de manipulação de grafos de poses. Na Seção 6.4 é feita uma descrição do método empregado para a detecção de retornos a locais que se visitara. O presente Capítulo se encerra dando as características gerais do robô P3DX, na Seção 6.5.

6.1 Descrição Geral

O sistema de SLAM, conforme implementado, é dividido em quatro partes: a responsável pelo alinhamento de varreduras, a responsável pela atualização do grafo de poses, a encarregada da otimização do grafo de poses e a destinada à detecção de retornos a locais que se visitara. O Algoritmo 7 fornece uma descrição de alto nível do sistema proposto.

O método proposto tem carácter incremental, e a cada iteração do Algoritmo 7 é processada uma varredura. O ciclo de processamento se inicia pela realização de uma varredura explicitada pelo método **Realizar_Varredura**(), que ordena ao robô que faça a aquisição de dados. A varredura obtida é armazenada na lista de varreduras l_v. A varredura $l_v(t)$ é então alinhada a $l_v(t-1)$ (varredura obtida no instante imediatamente anterior). O processo de alinhamento entre duas varreduras sucessivas produz uma estimativa do movimento que se sucedeu entre as poses das quais foram obtidas as varreduras $l_v(t)$ e $l_v(t-1)$ como a diferença de poses ΔP_s (diferença de poses sequencial). Em seguida, a diferença de poses é usada para que se atualize o grafo de poses G pela inclusão de um nó que será ligado ao último nó criado por meio de uma aresta. Concluída a atualização, G

Algoritmo 7 SLAM Proposto

1: $G := \emptyset$; $l_v := \emptyset$
2: $\mu_\lambda := 0$; $\sigma_\lambda := 0$;$\lambda := 0$
3: $\Delta P_s := \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$
4: $l_v(0) :=$ **Realizar_Varredura**()
5: $t := 1$
6: **Repita**
7: $l_v(t) :=$ **Realizar_Varredura**()
8: $\Delta P_s :=$ **Alinhar_Varreduras**$(l_v(t), l_v(t-1))$
9: $G :=$ **Otimizar_Grafo**(**Atualizar_Grafo**$(G, \Delta P_s, t, t-1)$)
10: $G :=$ **Buscar_Retorno**(G, l_v)
11: $G :=$ **Otimizar_Grafo**(G)
12: $t := t + 1$
13: **Até Receber** ordem de parada.
14: $M :=$ **Gerar_Mapa_Global** (G, l_v))
15: **Retornar** M

é otimizado. Terminado o processo de otimização de G, realiza-se uma busca de retorno, a qual recebe como argumentos G e l_v e tem por efeito a adição de arestas entre nós a partir dos quais se observe as mesmas regiões do espaço. Concluída a busca de retorno, otimiza-se novamente G e encerra-se o processamento de $l_v(t)$. Essa sequencia de passos se repete até que seja interrompido o SLAM por um comando externo. Ao receber um comando externo para encerrar o SLAM, gera-se o mapa global M a partir de G e de l_v.

O caso da primeira iteração é especial já que ainda não se tem nenhum referencial ou varredura pregressa. Assim sendo, define-se que a pose inicial será usada como referência e a primeira varredura realizada é armazenada sem que haja toda a sequencia de passos descrita no parágrafo anterior. Com efeito a primeira varredura impõe a introdução do primeiro nó do grafo.

6.2 O Alinhamento de Varreduras

Nessa seção será dado foco ao sistema de alinhamento de varreduras. A Seção 6.2.1 descreve o sistema de alinhamento de varreduras. A Seção 6.2.2 lista os parâmetros usados pelas técnicas de otimização de enxame que foram utilizadas.

6.2.1 Descrição

O sistema de alinhamento de varreduras pode operar usando dois modos diferentes de funcionamento: varredura a varredura e varredura a mapa. Explicaremos primeiro o

funcionamento dos modos de operação para depois esclarecer o que leva cada modo a vigorar.

Algoritmo 8 Alinhar_Varreduras(v, v_r)

Entrada μ_λ, σ_λ, ϵ
1: $l_c :=$ **Buscar_Correspondências**(v, v_r)
2: $\begin{bmatrix}\Delta P & \lambda\end{bmatrix}^T :=$ **Minimizar** a Equação 42
3: **Se** $\lambda > \epsilon$ **Então**
4: $\quad m :=$ **Renderizar_Mapa_Local** (v_r)
5: $\quad \begin{bmatrix}\Delta P & \lambda\end{bmatrix}^T :=$ **Minimizar** a Equação 44 com argumentos (v, m)
6: $\quad \lambda :=$ **Calcular** novo λ usando a Equação 42 com argumento ΔP
7: **Fim Se**
8: $\mu_\lambda^* := \mu_\lambda$
9: $\mu_\lambda := t^{-1}[\lambda + (t-1)\mu_\lambda]$
10: $\sigma_\lambda := \frac{t-2}{t-1}\sigma_\lambda + \frac{1}{t}(\lambda - \mu_\lambda^*)^2$
11: $\epsilon := \mu_\lambda + \sigma_\lambda$
12: **Retornar** ΔP

O alinhamento de varreduras está especificado pelo Algoritmo 8. O procedimento recebe duas varreduras: a varredura sob investigação v e a varredura de referência v_r. Em seguida realiza-se um processo de busca por correspondências entre v_r e v, o qual denotamos **Buscar_Correspondências**(v, v_r). O método **Buscar_Correspondências**(v, v_r) retorna uma bijeção entre v_r e v, tal que a cada ponto de v associa-se o ponto mais próximo pertencente a v_r. Essa bijeção é retornada como uma lista de pares ordenados denominada l_c. A tarefa de alinhar as varreduras então se reduz à determinação de um ponto ótimo da Equação 42, que é a função objetivo do processo quando esse opera no modo de alinhamento varredura a varredura, pela aplicação de uma das técnicas de enxame descritas no Capítulo 6, quais sejam: ABC, FA e PSO. A aplicação dos métodos de otimização foi denotada **Minimizar** no Algoritmo 8. O processo de otimização retorna dois resultados: uma pose relativa (ΔP) e o valor da função objetivo obtido para a solução retornada (λ). Se o valor de λ estiver abaixo de um limiar ϵ, o alinhamento é aceito como válido. Caso contrário, constrói-se um mapa local m a partir da varredura de referência, usando a metodologia descrita no Algoritmo 3, e usa-se a Equação 44 como função objetivo para realizar um alinhamento do tipo varredura a mapa. O resultado do processo de otimização será, novamente um par formado por ΔP e λ. O valor de λ retornado, entretanto, não pode ser utilizado para a atualização do limiar ϵ por não se encontrar quantificado pela mesma função objetivo. Usa-se em seguida a Equação 42 para obter um valor coerente de λ que será usado para atualizar o limiar de erro ϵ.

Após a obtenção de ΔP e do λ correspondente, atualiza-se a média e o desvio padrão dos valores obtidos para λ ao longo do SLAM, denotados μ_λ e σ_λ, respectivamente. Subsequentemente armazena-se o valor μ_λ, antes de alterá-lo, em μ_λ^* para que se possa usá-lo ao atualizar o valor de σ_λ. Atualiza-se μ_λ e depois σ_λ. Atualiza-se o limiar superior aceitável ϵ. Feito isso, retorna-se o valor de ΔP.

A função objetivo para o alinhamento varredura a varredura é a especificada na Equação 42:

$$f(\Delta P) = \log\left(1 + \sum_{i=1}^{N} \left\| \left(\Delta P \oplus \begin{bmatrix} v_{x_i} \\ v_{y_i} \end{bmatrix}\right) - \begin{bmatrix} v_{rx_i} \\ v_{ry_i} \end{bmatrix} \right\|^2\right), \tag{42}$$

onde ΔP é uma pose relativa e a operação de composição entre pose e ponto denotada pelo operador $\oplus$ é a definida pela Equação 18 na Seção 3.1.2.

A utilização do alinhamento do tipo varredura a mapa ocorre sempre que λ não se encontrar abaixo do limiar aceitável ϵ após a realização do alinhamento varredura a varredura. Nesse modo de operação, v_r é renderizado como um mapa local de ocupação m. O processo de alinhamento de varredura a mapa é realizado mediante a aplicação das mesmas técnicas de enxame, apenas alterando-se a função objetivo que passa a tomar a forma descrita em (OLIVEIRA; NEDJAH; MOURELLE, 2017). Sejam v_{x_i} e v_{y_i} as coordenadas Cartesianas do i-ésimo feixe de v e ΔP uma pose relativa; define-se as coordenadas resultantes da composição de ΔP e o ponto (v_{x_i}, v_{y_i}) pela Equação 18 da Seção 3.1.2. As coordenadas resultantes são denotadas (ξ_i, ζ_i), e definidas pela Equação 43:

$$\begin{bmatrix} \xi_i \\ \zeta_i \end{bmatrix} = \Delta P \oplus \begin{bmatrix} v_{x_i} \\ v_{y_i} \end{bmatrix}, \tag{43}$$

onde (ξ_i, ζ_i) são as coordenadas do obstáculo transformadas por ΔP, no sistema de coordenadas do mapa local. O processo de alinhamento da varredura ao mapa então se reduz ao processo de minimizar a função objetivo explicitada na Equação 44:

$$f(\xi_i, \zeta_i) = \sum_{i=1}^{n} \frac{255 - m(\xi_i, \zeta_i)}{255} + \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} \frac{m(x_r + k, y_r + l)}{255}, \tag{44}$$

onde as únicas variáveis que não foram explicitadas até aqui são: x_r e y_r. O par ordenado (x_r, y_r) representa a posição do robô no mapa local, já que essa evidentemente tem que estar vazia. A variável m representa o mapa local de ocupação, como dito anteriormente. O mapa local de ocupação é representado como uma matriz de inteiros de 8 bits o que garante uma gama de valores entre 0 e 255 para cada célula. Foi estabelecido que o valor 0 corresponderia a locais com 100% de probabilidade de estarem ocupados e o valor 255

a locais cuja probabilidade de ocupação é nula. A escala escolhida distribui linearmente todas as probabilidades entre 0 e 100% na faixa de 255 a 0, respectivamente. Na Equação 44, o primeiro somatório é realizado com respeito a todos os feixes de raio laser presentes na varredura que se deseja localizar e o segundo somatório penaliza todas as 9 células na região onde o robô se localiza.

Em ambos os modos de operação o procedimento de minimização usado pelo alinhamento está sujeito a três condições de parada. A primeira sob a forma de um limite ao número de iterações máximo de cada técnica aplicada, pré-fixado em 1000. A segunda é o limite de estagnação, que interrompe a execução das técnicas de otimização se o valor da melhor solução, aquela com o menor valor da função objetivo, não se alterar por mais do que a tolerância estabelecida em 10^{-6} ao longo de 50 iterações sucessivas. A terceira é a de que considera-se concluída a otimização se o valor absoluto da melhor solução for inferior à tolerância de 10^{-6}.

6.2.2 Parâmetros

Os parâmetros das meta-heurísticas foram ajustados empiricamente com base em testes de alinhamento repetidos 30 vezes por cenário com cada uma das meta-heurísticas. Cada caso foi construído com base em uma varredura de um dos conjuntos de dados que selecionamos para experimentação no Capítulo 7 aleatoriamente (foi usado um gerador de números aleatórios para que a seleção se desse sem qualquer tendenciosidade humana). Uma vez selecionadas as 10 varreduras. A cada varredura aplicou-se 3 tipos de transformação: uma rotação pura, uma translação pura e uma roto-translação. A partir dos 30 cenários resultantes, foram geradas duas variantes: uma sem ruído extra e uma com ruído Gaussiano de média nula e variância unitária.

Os testes que foram realizados com o ABC levaram à escolha dos parâmetros especificados na Tabela 1. Vale ressaltar que também levou-se em consideração o tempo de execução do processo de otimização e não apenas a acurácia.

Tabela 1: Tabela de parâmetros do ABC.

Parâmetro	**Valor**
Tamanho da Colônia	20
Fontes de Comida	10
Limite	10

Os parâmetros selecionados para o FA foram os que se pode observar na Tabela 2. Novamente observa-se que além de tentar maximizar a acurácia tentou-se minimizar o tempo de execução.

Tabela 2: Tabela de parâmetros do FA.

Parâmetro	**Valor**
Número de Vagalumes	20
α	$0,75$
β	$0,2$
γ	$1,0$

Os parâmetros selecionados para o PSO foram os que se pode observar na Tabela 3. Novamente observa-se que além de tentar maximizar a acurácia tentou-se minimizar o tempo de execução.

Tabela 3: Tabela de parâmetros do PSO.

Parâmetro	**Valor**
Tamanho do Enxame	50
Inércia	$0,7$
Coeficiente Cognitivo	$1,49$
Coeficiente de Inércia	$1,49$

6.3 Manipulação e Otimização do Grafo de Poses

Na Seção 3.2, foi descrito o que é a realização de SLAM com base em grafos de poses e foi explicado como é usada a matriz de informação H juntamente com o vetor de informação b para que se realize a otimização iterativa do grafo de poses. Aliás, o Algoritmo 1 explicita os passos para que se obtenha a matriz H e o vetor b partindo de um conjunto de restrições C presentes nas arestas de um grafo G. A diferença fundamental entre o Algoritmo 1 e o Algoritmo 12 é que o primeiro engloba também um procedimento de alteração da configuração espacial dos nós e o segundo não o faz.

Nessa seção, assume-se que o grafo encontra-se igualmente bem definido sendo dados sua matriz de informação H e pelo vetor de informação b ou o grafo, haja vista que o Algoritmo 12 define a conversão entre eles. A manipulação do grafo pode ser realizada, por conseguinte, mediante a realização de operações em blocos de H e de b. Entenda-se, por conseguinte, que todos os algoritmos que recebem G como argumento tem acesso à matriz H e ao vetor b.

Algoritmo 9 Atualizar_Grafo(G, ΔP, n_1, n_2)

Se $\left(\sqrt{\Delta P_x^2 + \Delta P_y^2} > 0,5\text{m} \vee |\Delta P_\theta| > 0,5\text{rad}\right)$ **Então**
  Se $n_2 \notin$ nós G **Então**
    $G :=$ **Inserir_Nó**(G, n_2)
    $G :=$ **Inserir_Aresta**(G, n_1, n_2, ΔP)
  Fim Se
Senão
  $G :=$ **Inserir_Aresta**(G, n_1, n_2, ΔP)
Fim Se
$G :=$ **Otimizar_Grafo**(G)
Retornar G

A atualização do grafo de poses encontra-se descrita pelo Algoritmo 9. Ao receber uma pose relativa ΔP, a atualização de grafo avalia se o ΔP representa uma translação de mais que $0,5$ m ou uma rotação de mais que $0,5$ rad. Caso uma das duas condições seja satisfeita é verificado se existe um nó com identificador n_2 em G. Se não houver um nó com identificador n_2, esse é inserido em G e ligado ao nó n_1 por uma aresta que representa a restrição de movimento expressa pela diferença de poses. Caso contrário o nó n_2 já exista em G acrescenta-se uma aresta entre n_1 e n_2. É importante notar que a atualização do grafo assume, por hipótese, que n_1 exista em G. Em ambos os casos o sistema tenta otimizar G antes de retorná-lo.

Apesar de o Algoritmo 9 explicar os critérios usados para a inserção de nós e arestas, ele deixa indefinidos os processos que efetivamente realizam tais tarefas. Sob o ponto de vista de que o grafo de poses pode ser representado pela matriz de informação que lhe é peculiar, o processo de inserção de nós consiste do acréscimo de uma linha e uma coluna à matriz H e de um correspondente aumento do vetor de informação b. Mais interessante é o caso do acréscimo de arestas que encontra-se especificado no Algoritmo 10.

O Algoritmo 10 especifica as etapas requeridas para que se atualize H e b para incorporar os efeitos da restrição de uma dada aresta. No Algoritmo 10, a matriz A, B e o vetor e são definidos segundo as Equações 23, 25 e 22 da Seção 3.2, respectivamente. A matriz Ω especificada como entrada no Algoritmo 10 é a matriz de informação correspondente à incerteza *a priori* entre as poses e, na implementação feita, foi assumida fixa quando se insere uma nova aresta de modo similar a (KAESS; RANGANATHAN; DELLA-

Algoritmo 10 Inserir_Aresta(G, i, j, ΔP)

Entrada Ω
1: $\begin{bmatrix} H & b \end{bmatrix}$:= **Converter_Grafo_Matrizes**(G)
2: H_{ii} += $A_{ij}^T \cdot \Omega \cdot A_{ij}$
3: H_{ij} += $A_{ij}^T \cdot \Omega \cdot B_{ij}$
4: H_{ji} += $B_{ij}^T \cdot \Omega \cdot A_{ij}$
5: H_{jj} += $B_{ij}^T \cdot \Omega \cdot B_{ij}$
6: b_i += $A_{ij}^T \cdot \Omega \cdot e_{ij}$
7: b_j += $B_{ij}^T \cdot \Omega \cdot B_{ij}$
8: G :=**Atualizar_arestas**(H)
9: **Retornar** G

ERT, 2008; KÜMMERLE et al., 2011; DELLAERT, 2012). Ao final do processo as restrições armazenadas nas arestas do grafo são atualizadas de acordo com o Algoritmo 11.

Algoritmo 11 Atualizar_arestas(H)

1: **Para Todo** $\langle e_{ij}(\cdot),\Omega_{ij} \rangle \in C$ **Faça**
2: **Atualiza** e_{ij} aplicando a Equação 19 ao bloco H_{ij} da matriz H.
3: $\Omega_{ij} := H_{ij}$.
4: **Fim Para**
5: **Retornar** G

O Algoritmo 12 define de que modo é feita a construção das matrizes H e b dado o grafo G. Seja C o conjunto de todas as restrições presentes nas arestas de G, o Algoritmo 12 diz que se forem considerados todos os elementos do conjunto de restrições e_{ij} e suas matrizes de covariância Ω_{ij} é possível construir a matriz H e o vetor b a partir da inclusão sequencial dos efeitos de cada restrição. A matriz H e o vetor b assim obtidos refletem o estado atual de G e o Algoritmo 12 opera sobre G sem alterá-lo.

De modo análogo, de posse da matriz H e do vetor b podemos atualizar as informações contidas na estrutura de dados do grafo. Mais especificamente, o Algoritmo 11 especifica como são atualizadas as restrições de G dado H. Similarmente ao que foi definido para o Algoritmo 12, C é definido como o conjunto de arestas presentes em G, no Algoritmo 11.

O método empregado para a otimização do grafo já foi descrito no Algoritmo 1 na Seção 3.2. Faremos aqui apenas uma especialização do método descrito. As diferenças mais evidentes entre o Algoritmo 1 e o Algoritmo 13 são a lista de argumentos e o valor de retorno. Deveras, no Algoritmo 13 o único valor de entrada é G, que na implementação feita consiste de uma estrutura de dados da qual se pode extrair a lista de poses $\check{x} = x_{1:T}^{\check{}}$ e a lista de arestas C que consiste de uma lista de pares ordenados contendo as restrições

Algoritmo 12 Converter_Grafo_Matrizes(G)

1: $b := 0$
2: $H := 0$
3: **Para Todo** $\langle e_{ij}(\cdot),\Omega_{ij}\rangle \in C$ **Faça**
4: $\quad A_{ij} := \frac{\partial e_{ij}}{\partial x_i}$
5: $\quad B_{ij} := \frac{\partial e_{ij}}{\partial x_i}$
6: $\quad H_{ii} \mathrel{+}= A_{ij}^T \cdot \Omega_{ij} \cdot A_{ij}$
7: $\quad H_{ij} \mathrel{+}= A_{ij}^T \cdot \Omega_{ij} \cdot B_{ij}$
8: $\quad H_{ji} \mathrel{+}= B_{ij}^T \cdot \Omega_{ij} \cdot A_{ij}$
9: $\quad H_{jj} \mathrel{+}= B_{ij}^T \cdot \Omega_{ij} \cdot B_{ij}$
10: $\quad b_i \mathrel{+}= A_{ij}^T \cdot \Omega_{ij} \cdot e_{ij}$
11: $\quad b_j \mathrel{+}= B_{ij}^T \cdot \Omega_{ij} \cdot B_{ij}$
12: **Fim Para**
13: $x^\star := \check{x}$
14: $H^\star := H$
15: **Retornar** H, b

$e_{ij}(\cdot)$ e os blocos da matriz de informação de cada par Ω_{ij}. Vale ressaltar que fica implícito que ao terminar a atualização cada nó tem a sua pose correspondente atualizada para o valor estimado pelo processo de otimização.

Algoritmo 13 Otimizar_Grafo(G)

1: **Enquanto** não houver convergência **Faça**
2: $\quad b := 0$
3: $\quad H := 0$
4: $\quad$ **Para Todo** $\langle e_{ij}(\cdot),\Omega_{ij}\rangle \in C$ **Faça**
5: $\quad\quad A_{ij} := \frac{\partial e_{ij}}{\partial x_i}$
6: $\quad\quad B_{ij} := \frac{\partial e_{ij}}{\partial x_i}$
7: $\quad\quad H_{ii} \mathrel{+}= A_{ij}^T \cdot \Omega_{ij} \cdot A_{ij}$
8: $\quad\quad H_{ij} \mathrel{+}= A_{ij}^T \cdot \Omega_{ij} \cdot B_{ij}$
9: $\quad\quad H_{ji} \mathrel{+}= B_{ij}^T \cdot \Omega_{ij} \cdot A_{ij}$
10: $\quad\quad H_{jj} \mathrel{+}= B_{ij}^T \cdot \Omega_{ij} \cdot B_{ij}$
11: $\quad\quad b_i \mathrel{+}= A_{ij}^T \cdot \Omega_{ij} \cdot e_{ij}$
12: $\quad\quad b_j \mathrel{+}= B_{ij}^T \cdot \Omega_{ij} \cdot B_{ij}$
13: $\quad$ **Fim Para**
14: $\quad H_{11} \mathrel{+}= I$
15: $\quad \Delta x :=$ **Resolva** $(H \cdot \Delta x = -b)$
16: $\quad \check{x} \mathrel{+}= \Delta x$
17: **Fim Enquanto**
18: $x^\star := \check{x}$
19: $H^\star := H$
20: $H_{11}^\star \mathrel{-}= I$
21: **Retornar** G

No Algoritmo 13 é realizado um processo de otimização baseado em um processo iterativo do tipo Gauss-Newton. Os valores de e_{ij}, como explicado na Seção 3.2, representam a diferença entre a observação esperada z_{ij} e a observação real $\hat{z_{ij}}$. O conceito fica mais claramente ilustrado pela Figura 21, em que estão em evidencia z_{ij}, que representa a medição virtual esperada de x_j em relação a x_i, e a elipse de covariância em torno da posição onde se esperava que x_j estivesse em relação a x_i, a qual é determinada por Ω_{ij}. Denomina-se o processo de otimizar pois ele minimiza a probabilidade negativa logarítmica de uma dada configuração de nós, *i.e.* dada uma configuração de nós e um conjunto de restrições entre eles, e assumindo que a posição dos nós é aproximadamente normalmente distribuída, existe uma função densidade de probabilidade da forma $\eta \exp(-0.5 \cdot e_{ij} \cdot \Omega_{ij} \cdot e_{ij}^T)$ para cada par de poses x_i e x_j entre os quais exista uma medição z_{ij}, em que η é uma constante de normalização. Desse fato, segue que, se for assumido que as posições dos nós são independentemente distribuídas, a probabilidade de uma configuração de nós pode ser expressa como um produtório de todos os termos exponenciais da forma enunciada. Essa conclusão segue imediatamente da premissa da independência das variáveis e da lei da probabilidade total (KOKOSKA; ZWILLINGER, 1999). O problema de maximizar a probabilidade de uma configuração pode ser formulado de maneira computacionalmente mais eficiente se for adotada a forma chamada *log-likelihood* (probabilidade logarítmica). Usando a probabilidade logarítmica negativa a tarefa de maximizar a probabilidade de uma configuração de nós se expressa como uma minimização de uma soma ao invés da maximização de um produto. A forma explícita da função objetivo não linear que deve ser minimizada para encontrar a configuração ótima dos nós é a da Equação 45:

$$F(x_{1:t}) = \sum_{\langle i,j \rangle \in C} e_{ij} \cdot \Omega_{ij} \cdot e_{ij}^T, \tag{45}$$

onde C é o conjunto de todas as arestas de G e $x_{1:t}$ denota o conjunto de todas as poses coletadas do instante de tempo 1 até o instante de tempo t. No Algoritmo 13 foi usada a linearização da Equação 45 para a aplicação do método de Gauss-Newton. O processo de derivação da forma linearizada é bastante usual e pode ser encontrado em (THRUN; MONTEMERLO, 2006; GRISETTI et al., 2010; KÜMMERLE et al., 2011). Na linha 15 do Algoritmo 13, o passo **Resolva** ($H \cdot \Delta x = -b$) foi implementado por uma fatoração QR para a solução do sistema linear. A implementação da fatoração QR utilizada foi a de (GUENNEBAUD; JACOB et al., 2010).

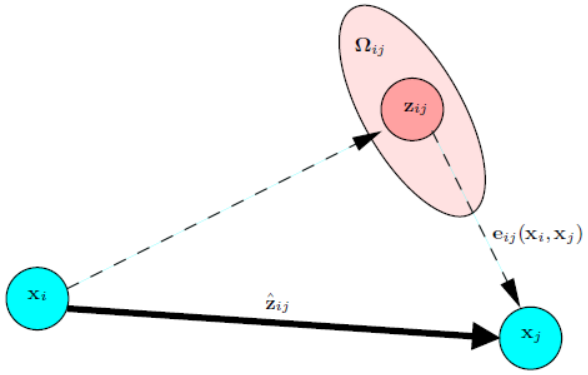


Figura 21: Ilustração de uma aresta conectando o nó x_i ao nó x_j. Imagem retirada de (GRISETTI et al., 2010).

6.4 Busca e Detecção de Retorno

O Algoritmo 14 define o funcionamento do sistema de busca e detecção de retornos a locais já visitados. O sistema de busca de retorno é invocado periodicamente com o intuito de manter a consistência global do mapa e trajetória construídos. Antes da explicação em maiores detalhes, é interessante notar que o processo de detecção de retorno funciona, em uma visão bastante sucinta, pela sucessiva eliminação de possíveis candidatos.

Algoritmo 14 Buscar_Retorno(G, l_v)

1: $\begin{bmatrix} l_c & l'_v \end{bmatrix}^T$:= **Selecionar_Candidatos**(G)
2: $\begin{bmatrix} l'_c & l''_v \end{bmatrix}^T$:= **Eliminar_*Outliers***(G, l_c, l'_v)
3: **Para** $i = 1, \cdots, N_c$ **Faça**
4: $\quad \begin{bmatrix} \Delta P & \lambda \end{bmatrix}^T$:= **Alinhar_Varreduras**(v, v_r)
5: $\quad E_{\text{antes}}$:= **Calcular_Erro**(G)
6: $\quad G$:= **Inserir_Aresta**(G, t, l'_c, ΔP)
7: $\quad G$:= **Otimizar_Grafo**(G)
8: $\quad E_{\text{depois}}$:= **Calcular_Erro**(G)
9: $\quad$ **Se** $E_{\text{antes}} < E_{\text{depois}}$ **Então**
10: $\quad\quad G$:= **Remover_Aresta**(G, t, l'_c, ΔP)
11: $\quad\quad G$:= **Otimizar_Grafo**(G)
12: $\quad$ **Fim Se**
13: **Fim Para**
14: **Retornar** G

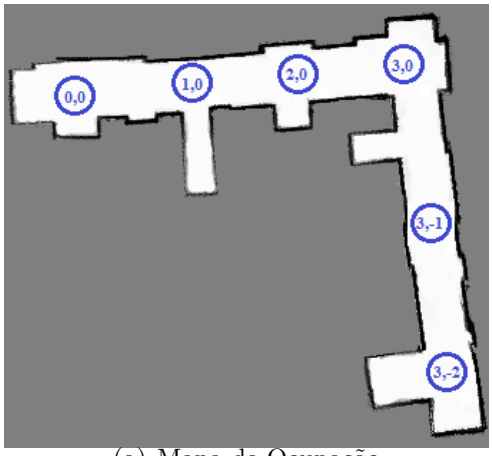
O sistema de busca de retorno recebe como argumentos: G e l_v. Ressaltamos que partir da estrutura de dados que representa G, o sistema é capaz de recuperar a estimativa da pose atual P bem como a lista de todas as poses. Tal coisa é possível pois a obtenção da lista de todas as poses consideradas requer apenas que se percorra todos os nós do grafo

G. De igual simplicidade é a obtenção da matriz de informação H, que fica armazenada na estrutura de dados que contem G e só é atualizada quando há uma das manipulações do grafo descritas na Seção 6.3. A primeira tarefa realizada é a seleção de nós com poses que sejam candidatas a um fechamento de laço l_c.

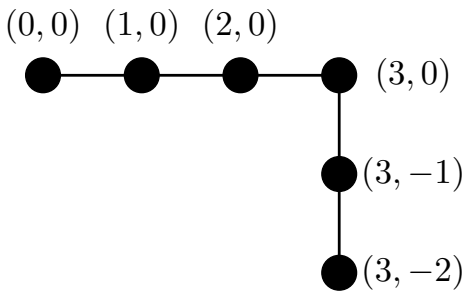
O processo de seleção de nós que sejam passíveis de ser candidatos a fechamentos é realizado pelo método **Selecionar_Candidatos**. A seleção é feita mediante a busca dos vizinhos mais próximos do nó atual de G, o que é feito com o auxílio de uma KD-Tree (*K-Dimensional Tree* – KD-Tree). O modo como são construídas as árvores KD já foi explicado na Seção 4.1.2.

Na Figura 22 exibe-se um mapa simples, que será usado em um exemplo, em que foram marcados os nós e denotadas as coordenadas translacionais dos nós, explicita-se o fato de que os nós foram inseridos no grafo na seguinte ordem: $\begin{bmatrix}0 & 0 & 0,087\end{bmatrix}$, $\begin{bmatrix}1 & 0 & 0,087\end{bmatrix}$, $\begin{bmatrix}2 & 0 & 0,087\end{bmatrix}$, $\begin{bmatrix}3 & 0 & 0.087\end{bmatrix}$, $\begin{bmatrix}3 & -1 & -1.4835\end{bmatrix}$, $\begin{bmatrix}3 & -2 & -1.4835\end{bmatrix}$. A Figura 22(a) exibe o mapa de ocupação, sobre os quais foram superpostos os nós percorridos. A Figura 22(b) mostra o grafo de poses otimizado. A Figura 22(c) exibe a árvore KD correspondente ao grafo de poses. A ordem de inserção foi imposta para simular a ordem cronológica que seria inerente ao processamento processamento de um conjunto de dados real *online*. Na Figura 22(c) os nós em verde representam partições no eixo x e os nós em vermelho representam partições em y. A razão por detrás da manutenção da estrutura de árvore KD indexando as poses, além do grafo de poses, é que ela permite uma rápida busca pelos vizinhos mais próximos, de acordo com o que foi explicado na Seção 4.1.2, e em consonância com os autores de (FRIEDMAN; BENTLEY; FINKEL, 1977; BENTLEY; FRIEDMAN, 1979; BENTLEY, 1980). Dada a convenção de que na Figura 22(c) a sub-árvore da esquerda sempre contenha os nós correspondentes a pontos inseridos cujas distâncias ao nó raiz da sub-árvore sendo analisada sejam menores que os armazenados na sub-árvore filha da direita, é fácil verificar que, por exemplo, o nó $(2, 0)$ é mais promissor numa busca por fechamentos de laço para o nó $(3, 0)$ do que o nó $(1, 0)$. Uma tarefa que se aproxima do caso real do método **Selecionar_Candidatos**, em que a distância máxima seria dada pela incerteza da posição de cada nó, é o de uma busca com uma distância máxima aceitável. Se procurarmos na Árvore KD exibida na Figura 22(c), por exemplo, o nó mais próximo de $(3, 0)$ que esteja a menos que 1 unidade de distância, usando a métrica Euclidiana, uma rápida inspeção basta para eliminar a sub-árvore com raiz em

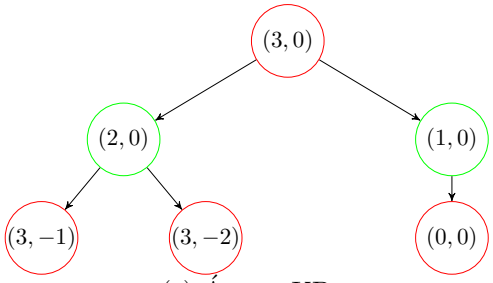
$(1, 0)$. Se supusermos que fosse desejado verificar se algum nó do grafo representa um retorno ao nó $(3, 0)$, com a condição de que a incerteza é de uma unidade de distância, apenas teríamos que verificar a varredura obtida a partir de $(2, 0)$. Isso significa que só seria necessário um único alinhamento de varreduras ao invés dos cinco que uma busca exaustiva exigiria. É essa mesma lógica que é aplicada a todos os casos em que se invoca o método **Selecionar_Candidatos**, e é ela que permite eliminar rapidamente um grande número de candidatos. Esse ganho de desempenho se deve ao fato de a complexidade esperada para a busca de um vizinho de uma pose da árvore é logarítmica em relação ao número total de nós, conforme os resultados dos autores de (FRIEDMAN; BENTLEY; FINKEL, 1977; BENTLEY, 1980).



(a) Mapa de Ocupação



(b) Grafo de Poses



(c) Árvore KD

Figura 22: Exemplo da relação entre o grafo de poses e a árvore KD que o representa.

A rotina **Eliminar_*Outliers*** funciona com base no processo de determinação de compatibilidade explicado no final do Capítulo 4 e recebe por argumentos l_c, l'_v. A rotina **Eliminar_*Outliers*** elimina de l_c, aquelas poses que estiverem fora da elipse de covariâncias na qual P se encontra e elimina também de l'_v todas as varreduras correspondentes às poses eliminadas. A elipse de covariâncias foi definida como a elipse tal que a proba-

bilidade de que a verdadeira pose atual se encontre dentro de uma distribuição normal bivariada, de cuja média é P e cuja matriz de informação é o bloco de H correspondente à pose atual é de noventa e cinco por cento. Usamos o critério de uma distribuição bivariada pois para essa etapa consideram-se apenas as variáveis que definem as componentes translacionais da pose. Desse processo de eliminação de *outliers* originam-se as listas l'_c e l''_v que contém, respectivamente, as poses candidatas a fechamentos e as varreduras obtidas a partir delas que não foram eliminadas. As poses restantes em l'_c são então testadas, uma a uma, com o intuito de encontrar uma que sirva como uma relação extemporânea, que se costuma denominar medição virtual nos sistemas de SLAM baseados em grafos de poses (GRISETTI et al., 2010). O processo de teste é baseado no alinhamento sucessivo entre a varredura de referência v_r, obtida a partir da pose atual, e todas as varreduras pertencentes à lista de varreduras l''_v que foram obtidas a partir das poses em l'_c.

O processo de teste de candidatos procede de maneira metódica verificando, uma por vez, as poses que restarem na lista de possíveis retornos. Para cada pose de l'_c invoca-se o procedimento de alinhamento de varreduras fornecendo-lhe por argumentos v_r e a varredura $l_v(i)$ obtida da pose candidata.

O processo de alinhamento produz como resultado a pose relativa entre a pose atual e cada candidata e o erro incorrido por cada alinhamento, conforme especificado no Algoritmo 8. Para validar a possível relação de correspondências assume-se como critério que uma relação válida não deve fazer crescer o erro do processo de otimização do grafo, isso é uma relação de correspondência deve levar a uma redução do erro, conforme quantificado pelo Algoritmo 15, ou, ao menos, mantê-lo no mesmo nível. Essa condição equivale a assumir, em termos probabilísticos, que a inserção de uma aresta válida não deve tornar a configuração dos nós menos provável do que ela era antes dessa inserção. O processo continua inserindo as arestas, otimizando a configuração dos nós no grafo e comparando o erro incorrido dessa otimização àquele que se observou antes da inserção da aresta. Se for observado um aumento desse erro a nova aresta é removida do grafo de poses e a pose candidata a fechamento de laço deixa de ser considerada uma candidata. O processo continua até que não haja mais candidatas na lista. *Ipso facto*, a busca pode encerrar-se sem que haja a inclusão de qualquer aresta ou, adversamente, com a inclusão de várias. Ao encerrar-se a busca, é retornado o grafo de poses atualizado. O grafo G,

retornado, é obtido pela inclusão de todas as arestas válidas e a subsequente otimização de G.

O Algoritmo 15 calcula o erro cometido pelo grafo varrendo todas as arestas entre os nós do grafo e aplicando a elas a Equação 45. No Algoritmo 15 o valor de e_{ij} usado é aquele armazenado na aresta entre x_i e x_j, que foi fornecido quando se inseriu a aresta.

Algoritmo 15 Calcular_Erro(G)

1: $E := 0$
2: **Para** $i = 1, \cdots, t$ **Faça**
3: **Para** $j = 1, \cdots, t$ **Faça**
4: **Se** Existir aresta entre x_i e x_j **Então**
5: $E := E + e_{ij} \cdot \Omega_{ij} \cdot e_{ij}^T$
6: **Fim Se**
7: **Fim Para**
8: **Fim Para**
9: **Retornar** E

Concluiremos a presente discussão sobre a manipulação do grafo de poses lidando com a metodologia empregada para a remoção de arestas do grafo de poses sob a representação matricial. O Algoritmo 16 descreve o procedimento para que se remova uma aresta, e sua restrição. Com efeito, a operação delineada é inversa daquela descrita no Algoritmo 10 e os passos consistem da subtração dos termos que se adicionara à matriz H e ao vetor b. O procedimento de remoção de aresta é usado exclusivamente pelo sistema de busca de retorno descrito na Seção 6.4.

Algoritmo 16 Remover_Aresta(G, i, j, ΔP)

Entrada Ω
1: $\begin{bmatrix} H & b \end{bmatrix}$:= **Converter_Grafo_Matrizes**(G)
2: H_{ii} -= $A_{ij}^T \cdot \Omega \cdot A_{ij}$
3: H_{ij} -= $A_{ij}^T \cdot \Omega \cdot B_{ij}$
4: H_{ji} -= $B_{ij}^T \cdot \Omega \cdot A_{ij}$
5: H_{jj} -= $B_{ij}^T \cdot \Omega \cdot B_{ij}$
6: b_i -= $A_{ij}^T \cdot \Omega \cdot e_{ij}$
7: b_j -= $B_{ij}^T \cdot \Omega \cdot B_{ij}$
8: G :=**Atualizar_arestas**(H)
9: **Retornar** G

6.5 O P3DX

Nas coletas de dados que realizamos, para os experimentos que as demandaram, foi utilizado o robô P3DX da *Mobile Robots*. O P3DX é um robô com motor diferencial que:

pesa 9kg, possui corpo composto por placas de alumínio de 1.6mm, é capaz de carregar uma carga útil de até 17kg e tem pneus de borracha preenchidos por espuma. O manual do fabricante fornece esquemas que especificam as dimensões do robô, um dos quais reproduzimos na Figura 23, onde todas as medidas se encontram expressas em milímetros.

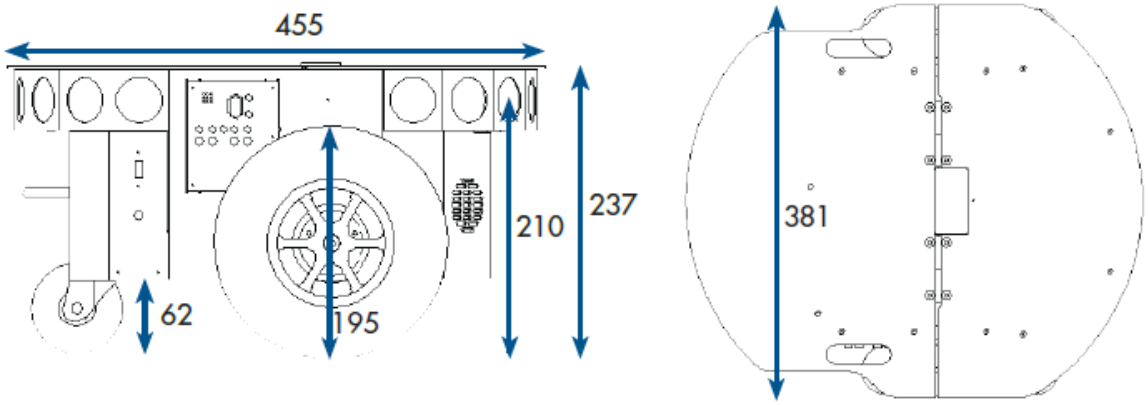


Figura 23: Diagrama esquemático do P3DX.

Segundo o fabricante, o motor do robô sobre o qual discorremos é capaz de fazê-lo alcançar velocidades translacionais de até $1,2m/s$ e velocidades angulares de até $300°/s$.O motor, juntamente com todos os componentes do robô e acessórios são alimentados por um banco de baterias. O P3DX possui capacidade para portar três baterias com tensão de 12V que são capazes de fornecer até $7,2$Ah e cujo tempo de duração é entre oito e dez horas. As baterias são recarregáveis, sendo seu tempo de recarga especificado pelo fabricante como sendo de doze horas. O robô que empregamos para a coleta de dados é dotado de um *laser range finder* SICK LMS100. O LRF(*Laser Range Finder*) empregado possui um alcance de 20m e varre um setor circular de 270° de modo tal que na configuração que foi utilizada são retornadas 541 medidas de distância em cada varredura, ou seja uma medida a cada meio grau.

6.6 Considerações Finais do Capítulo

Nessa seção foram discutidos: o sistema de alinhamento de varreduras, o sistema de detecção de retorno a locais que se visitara e o sistema de manipulação e otimização de grafos de poses. No próximo Capítulo falaremos sobre: os testes que conduzimos com conjuntos de dados de domínio público, que foram obtidos de (HOWARD; ROY, 2003) e os

experimentos em que foi empregado o P3DX. O uso de conjuntos de dados de domínio público e com grande notoriedade como os de (HOWARD; ROY, 2003) permite, pela sua grande disseminação, que se realize comparações com outras soluções para o problema de SLAM. O próximo capítulo possui por objetivo dar ao leitor uma avaliação do desempenho do sistema proposto além de fornecer uma comparação de como o desempenho do sistema que apresentamos até aqui se compara com os resultados apresentados por outrem.

Capítulo 7

AVALIAÇÃO DO MÉTODO PROPOSTO

O P resente capítulo descreve os detalhes dos sistemas utilizados na implementação, discorre sobre os resultados obtidos nos testes de *benchmarking* realizados para que pudéssemos comparar a acurácia do sistema que desenvolvido e sobre os experimentos que realizamos empregando o robô P3DX. Na Seção 7.1 descrevemos a métrica usada para a comparação de acurácia supramencionada. Na Seção 7.2 são descritos os ambientes e percursos que se verifica nos conjuntos de dados de domínio público que usamos. Na Seção 7.3 são exibidos os mapas e as trajetórias resultantes. Na Seção 7.4 são discutidos os resultados, encerrando o capítulo.

A implementação dos sistema foi realizada usando a linguagem de programação C++. Foram utilizadas: a biblioteca de álgebra linear Eigen versão 3.2.9, a biblioteca nanoflann versão 1.2.2 (para a implementação da busca de vizinhos mais próximos em árvores-KD) e a biblioteca OpenCV versão 3.2.0 (para produzir os arquivos de imagem com os mapas de ocupação). Os testes com os corpos de dados de conjuntos de dados de domínio público foram realizados em um PC com Intel Core $i7$ 870 $2,93$GHz, 8Gb RAM rodando Debian 8.6.

7.1 Metodologia de Avaliação

A dificuldade na condução de experimentos para a avaliação de sistemas cujo objetivo seja a solução do problema de mapeamento e localização simultâneos advém, em parte, do fato de que não há um padrão estabelecido para a avaliação desse tipo de sistema na comunidade de pesquisadores da área. Em (BONARINI et al., 2006; CERIANI GIULIO FONTANA et al., 2009), os autores descrevem diversas métricas para a avaliação da acurácia

de sistemas de SLAM. O método proposto por Kümerle et al. em (KÜMMERLE et al., 2009) para a comparação da acurácia de soluções para o SLAM se baseia na avaliação das relações entre as poses do robô durante o processo de aquisição de dados. Agindo desse modo, e não comparando diretamente os mapas, obtém-se dois ganhos principais: a capacidade de avaliar algoritmos que produzam resultados com representações diferentes e a invariância ao tipo de sensor utilizado. Para o presente trabalho a primeira qualidade dessa metodologia é de maior importância. Com efeito, o método descrito exige apenas que a trajetória estimada pelo algoritmo, mediante o processamento de qualquer tipo de medição que se faça, se encontre disponível. O método de avaliação nos pareceu razoável e foi adotado, sendo a seguir descrito o modo pelo qual foi aplicado ao nosso trabalho. Deveras, a avaliação da acurácia, por esse método se baseia fundamentalmente na aplicação das Equações 46 e 47. A Equação 46 descreve o Erro absoluto translacional e é definida por:

$$\lambda_{translacional} = \frac{1}{N}\sum_{i,j}^{N} |T(\delta_{i,j} - \delta^*_{i,j})|, \tag{46}$$

onde N é o número de relações consideradas para a avaliação, $\delta_{i,j}$ é a diferença entre as poses i e j, definida pela Equação 16, $\delta^*_{i,j}$ é a relação de referência e T$(\cdot)$ separa as componentes translacionais de uma pose relativa. A Equação 47 é definida por:

$$\lambda_{rotacional} = \frac{1}{N}\sum_{i,j}^{N} |R(\delta_{i,j} - \delta^*_{i,j})|, \tag{47}$$

onde N é o número de relações consideradas para a avaliação, $\delta_{i,j}$ é a diferença entre as poses i e j, definida pela Equação 16, $\delta^*_{i,j}$ é a relação de referência e R$(\cdot)$ separa a componente rotacional de uma pose relativa. Em (KÜMMERLE et al., 2009) os autores não especificam critério para que se selecione as relações a incluir na avaliação das técnicas, e escrevem que tal seleção pode ser feita com o intuito de caracterizar as diferenças entre os algoritmos sob comparação.

7.2 Conjuntos de Dados

Nessa seção descreveremos os oito conjuntos de dados de domínio público que utilizamos para a realização dos experimentos. Os corpos de dados foram selecionados com base em dois critérios: a diversidade e a notoriedade. Por diversidade queremos dizer que buscamos fazer um apanhado de ambientes a mapear que apresentassem características distintas,

como pode-se observar tanto das descrições que fizemos de cada um deles quanto pelos mapas de referência que acompanham cada uma das descrições. A notoriedade de que falamos, por sua vez, refere-se ao uso disseminado de determinada fonte de dados pela comunidade acadêmica, conforme representada pela aparição desses conjuntos de dados em artigos da área em questão como (GRISETTI; STACHNISS; BURGARD, 2007; KAESS; RANGANATHAN; DELLAERT, 2008; OLSON, 2008; KÜMMERLE et al., 2009; GRISETTI et al., 2010; KÜMMERLE et al., 2011; HESS et al., 2016).

7.2.1 Conjunto de dados FR079

O FR079 é um conjunto de dados de domínio público, coletado por Cyrill Stachniss no Edifício de número 79 da Universidade de Freiburg no ano de 2003. Nele há 2928 varreduras coletadas ao longo de um percurso que a odometria estima ser de $390,88$ metros. Esse conjunto de dados representa um ambiente composto por salas dispostas paralelamente a um corredor retilíneo. Ademais, há diversos instantes em que o robô que realizou a coleta retorna a pontos que visitara. O ambiente encontra-se ilustrado pela Figura 25.

Figura 24: Mapa de referência do conjunto FR079.

7.2.2 Conjunto de Dados INTEL

O conjunto de dados INTEL (*INTEL Dataset*) é um conjunto de dados de domínio público, compilado por Dirk Hähnel (HOWARD; ROY, 2003). Nele, encontram-se os dados métricos obtidos em uma seção exploratória realizada no laboratório de pesquisa da Intel em Seattle no ano de 2003. Nesse corpo de dados há 5452 varreduras realizadas usando um *Laser Range Finder* ao longo de um percurso que, pela estimativa odométrica, se estende por $505,99$ metros. Este *benchmark* em particular foi escolhido por ser de uso bastante difundido entre os pesquisadores que trabalham com localização, mapeamento

e SLAM, como pode-se observar pelo número de artigos em que ele figura. Morfologicamente, a estrutura de onde os dados foram obtidos para a construção do conjunto de dados é de interesse considerável, por conter diversas regiões similares. A existência de ambiguidades do gênero que mencionamos torna difícil a tarefa de identificar locais já visitados com algum grau de confiança. A construção em questão consiste de um corredor curvilíneo com diversos cômodos que lhe são adjacentes. Durante a aquisição de dados, o robô visitou as salas que cercam o corredor principal. O trajeto percorrido durante a exploração do ambiente apresenta diversos instantes em que o robô retornou a lugares onde estivera, o que permite que se avalie a capacidade do sistema de SLAM de operar em ambientes contendo laços (*loops*). Na Figura 25 observa-se o ambiente descrito.

Figura 25: Mapa de referência do conjunto INTEL.

7.2.3 Conjunto de Dados MIT CSAIL

O conjunto de dados MIT CSAIL (*Computer Science Artificial Intelligence Laboratory*) representa o terceiro andar do prédio de Ciência da Computação e Inteligência Artificial do MIT e é um conjunto de dados de domínio público coletado por Cyrill Stachniss em 2005. Ele se caracteriza pela existência de grandes ambientes e corredores retilíneos que os conectam. Sobre o trajeto percorrido, trata-se de um conjunto de ciclos fechados que atravessam toda a área visível no mapa de modo razoavelmente direta. A estrutura do ambiente com corredores estreitos que são precedidos e sucedidos por pequenos percursos fechados leva a uma situação em que o acúmulo de erros, especialmente aqueles decor-

rentes de desvios na orientação dificulta o reconhecimento de pontos em que os ciclos supramencionados se fecham, o que representa o maior desafio do conjunto de dados pelo que pudemos observar. O mapa que usamos como referência para a avaliação do sistema encontra-se na Figura 26.

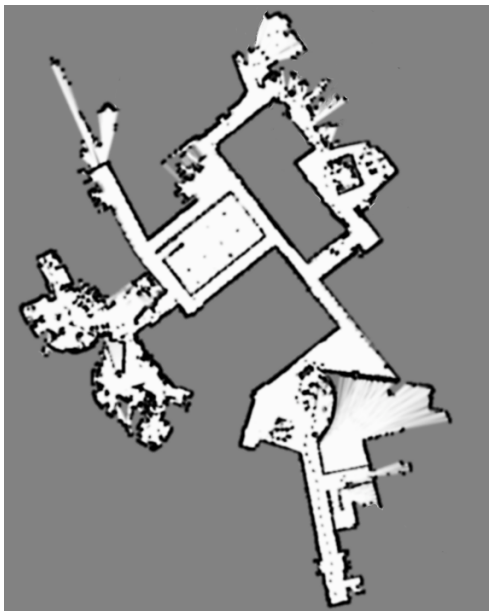


Figura 26: Mapa de referência do conjunto CSAIL.

7.2.4 Conjunto de Dados KILLIAN

O conjunto de dados originário do chamado corredor infinito, também conhecido como *Killian Court*, do MIT foi obtido por Mike Bosse e John Leonard. O ambiente descrito pelo conjunto de dados em questão é um corredor de 251m que atravessa os principais prédios do *campus* do MIT em *Massachussetts*. O ambiente consiste de três laços maiores e dois menores, totalizando cinco laços formados pelo corredor principal e suas ramificações. Os desafios principais apresentados por esse conjunto de dados são a existência de longos corredores com pequena variação de contorno e a presença de um número considerável de laços com tamanhos de pequenos a extensos. O mapa usado como referência encontra-se ilustrado na Figura 27.

7.2.5 Conjunto de Dados OREBRO

O conjunto de dados referido como OREBRO em (STACHNISS, 2018), foi coletado por Henrik Andreasson, Per Larsson e Tom Duckett. Ele consiste de um corpo de mensurações constituído por 237 varreduras, cada qual com 361 medições. O ambiente de onde foram

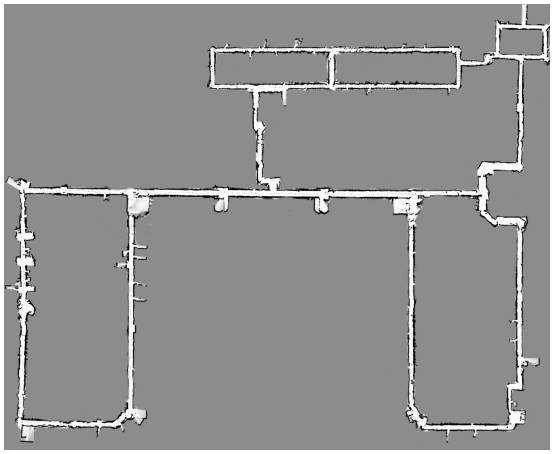


Figura 27: Mapa de referência do conjunto KILLIAN.

obtidos os dados consiste de três grandes corredores e dois laços de extensão considerável. O mapa construído com base na trajetória de referência, disponível em (KÜMMERLE et al., 2009), pode ser observado na Figura 28.

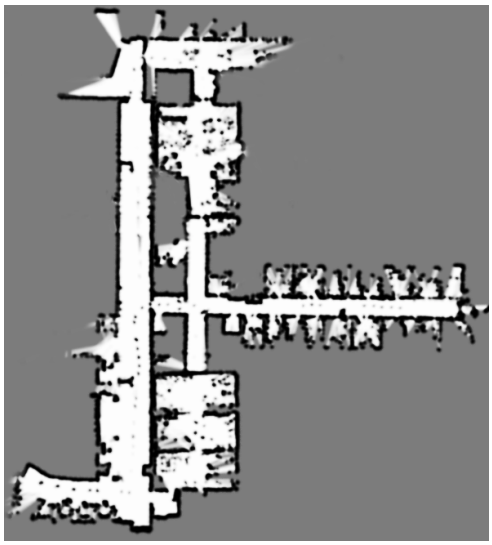


Figura 28: Mapa de referência do conjunto OREBRO.

7.2.6 Conjunto de Dados ACES

O conjunto de dados denominado ACES representa o terceiro andar do prédio homônimo no campos da Universidade do Texas (UT) em Austin. O ambiente possui uma forma que se assemelha a um quadrado cuja borda é composta por corredores, com mais dois corredores que se encontram no centro de modo cruciforme e dividem a região do espaço que é delimitada pelo quadrilátero formado pelos quatro corredores externos em quatro regiões poligonais de quatro lados de tamanho menor. O conjunto de dados foi coletado

por Patrick Beeson e encontra-se disponível em (HOWARD; ROY, 2003) e em (STACHNISS, 2018). Segundo a descrição detalhada exibida em (HOWARD; ROY, 2003), a exploração do ambiente iniciou-se pelo laço externo, e prosseguiu pela travessia do corredor interno não percorrido da primeira vez, completando assim a divisão da área da região em quatro polígonos que se aproximam de quadrados. O mapa do conjunto de dados construído com base na trajetória referência (*groundtruth*), fornecida em (KÜMMERLE et al., 2009), pode ser observado na Figura 29. Na Figura 29 evidencia-se de modo claro a similaridade de diversas regiões desse ambiente, o que apresenta um desafio para a realização do reconhecimento de locais já visitados usando-se apenas dados morfológicos.

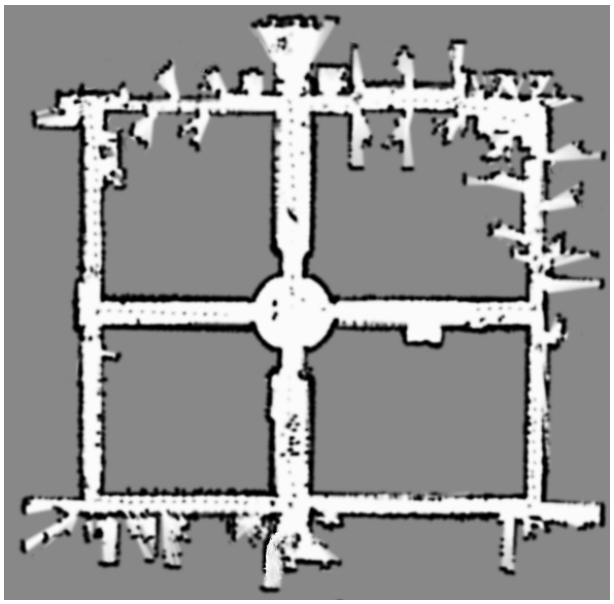


Figura 29: Mapa de referência do conjunto ACES.

7.2.7 Conjunto de dados MEXICO

O conjunto de dados denotado MEXICO possui dados coletados por Nick Roy em um centro de convenções na cidade de Acapulco no México e disponibilizados em (STACHNISS, 2018). Esse corpo de dados foi coletado em um ambiente com traços típicos de espaços artificiais dedicados ao encontro de pessoas, tais como: contornos retilíneos, pequenos obstáculos e cantos de perfis aproximadamente ortogonais. O mapa construído baseado na trajetória de referência, fornecida em (KÜMMERLE et al., 2009), pode ser observado na Figura 30.

Figura 30: Mapa de referência do conjunto MEXICO.

7.2.8 Conjunto de dados Seattle

O conjunto de dados que denominado SEATTLE, encontra-se disponível em (STACHNISS, 2018) e foi colhido por Dirk Hähnel. Apresenta-se sob a forma de um conjunto de 241 leituras acompanhadas dos dados odométricos coletados durante o processo. Trata-se de um ambiente simples composto por corredores na forma de um circuito fechado conectado a um corredor simples. Por sua simplicidade é um conjunto de dados interessante para que se avalie o comportamento do sistema na presença de ciclos fechados estando ausentes maiores complicações. O mapa construído com a trajetória de referência, fornecida em (KÜMMERLE et al., 2009), pode ser observado na Figura 31.

Figura 31: Mapa de referência do conjunto SEATTLE.

7.3 Avaliação dos Resultados

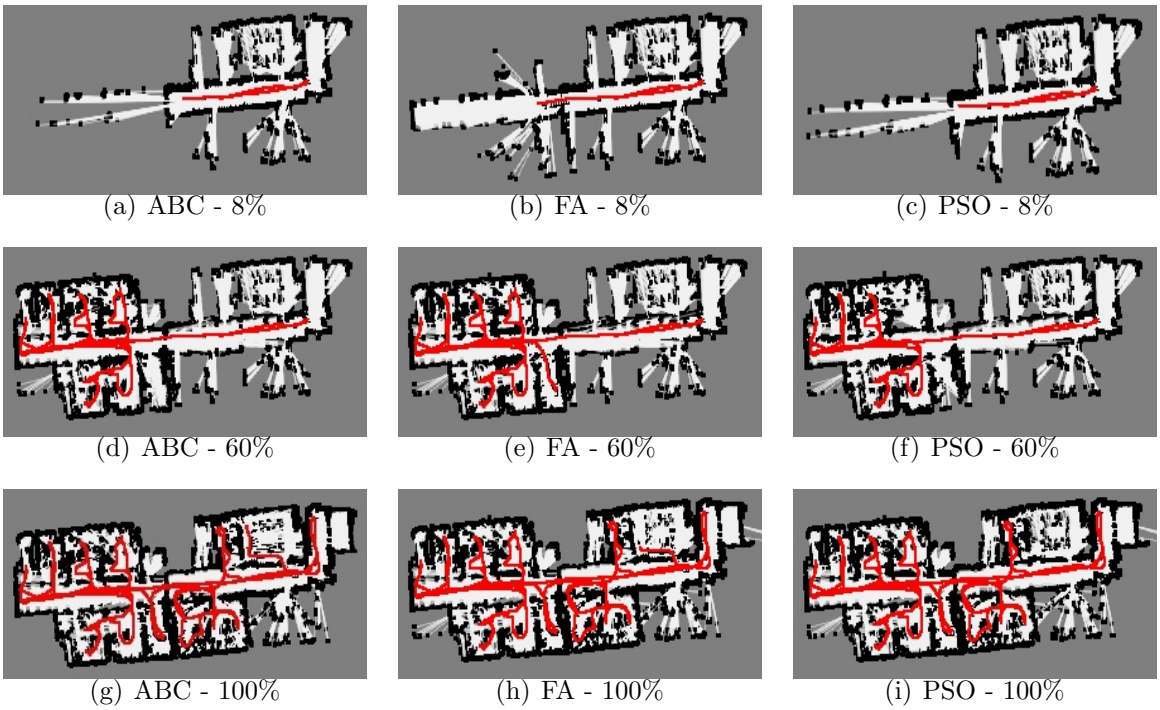
Uma avaliação objetiva de resultados experimentais exige que seja estabelecida uma métrica de comparação bem definida e com significado claro. É escassa a literatura acerca desse aspecto particular da pesquisa sobre SLAM. Em (KÜMMERLE et al., 2009), os autores advogam o uso de uma métrica que considere a energia de deformação necessária para transformar a estimativa fornecida por uma determinada técnica de localização e mapeamento simultâneos e a que seria considerada no resultado ideal, se possível o *groundtruth*.

Eles alegam também, e de modo bastante convincente, que isso pode ser feito pela aplicação de ideias similares às propostas por Lu e Milios em (LU; MILIOS, 1997). Kümmerle et al. descrevem então a aplicação da supracitada métrica a diversos conjuntos de dados, a maioria dos quais foram também utilizados para examinar as características do sistema de SLAM proposto nessa dissertação. Não só isso, mas também à uma vantagem inerente e irrefutável ao uso dessa técnica, em virtude de ter ela sido utilizada na elaboração de trabalhos recentes como (KÜMMERLE et al., 2009; TIPALDI; BRAUN; ARRAS, 2014; HESS et al., 2016) deve-se a opção feita por acrescer ao nosso trabalho o uso dessa métrica como um dos modos de compará-la às tantas outras descritas na literatura acadêmica. Aplicá-la-emos aos casos em que isso permita a comparação dos resultados obtidos com aqueles em (KÜMMERLE et al., 2009; TIPALDI; BRAUN; ARRAS, 2014; HESS et al., 2016) os quais citaremos *ipsis litteris*.

7.3.1 Conjunto de dados FR079

Após a condução dos experimentos com esse mapa calculamos o Erro absoluto translacional e o Erro absoluto rotacional das soluções obtidas com cada técnica de otimização de enxame e os erros absolutos translacionais e rotacionais podem ser observados nas Figuras 48 e 49, respectivamente.

Na Figura 32, foram marcadas quatro regiões de interesse nas quais podem ser observadas discrepâncias entre as soluções ao término do processo. Na Figura 33(a) observa-se que a região 1, diferentemente do que acontece na Figura 33(b) e na Figura 33(c), não apresenta feixes que atravessem o obstáculo imposto pela parede. Ao comparar essa característica à região 2 da Figura 33(e) e da Figura 33(f), veremos que há indícios de que durante o processo de mapeamento o sistema falhou em reconhecer que o feixe atravessava um obstáculo, o que indica que em alguma das poses de onde se observa essa região houve discrepância entre as estimativas o que levou o sistema a deixar, momentaneamente algumas das células do mapa nas cercanias da parede com um valor de ocupação que represente uma probabilidade inferior a 50% de que a célula em questão esteja preenchida. O mesmo vale para a comparação da região 3 da Figura 32(h) e da Figura 32(i) que apresentam essa falha, embora a Figura 33(g) não a possua. Não há diferenças notáveis na região 4 entre a Figura 32(j), a Figura 32(k) e a Figura 32(l).



(a) ABC - 8% (b) FA - 8% (c) PSO - 8%

(d) ABC - 60% (e) FA - 60% (f) PSO - 60%

(g) ABC - 100% (h) FA - 100% (i) PSO - 100%

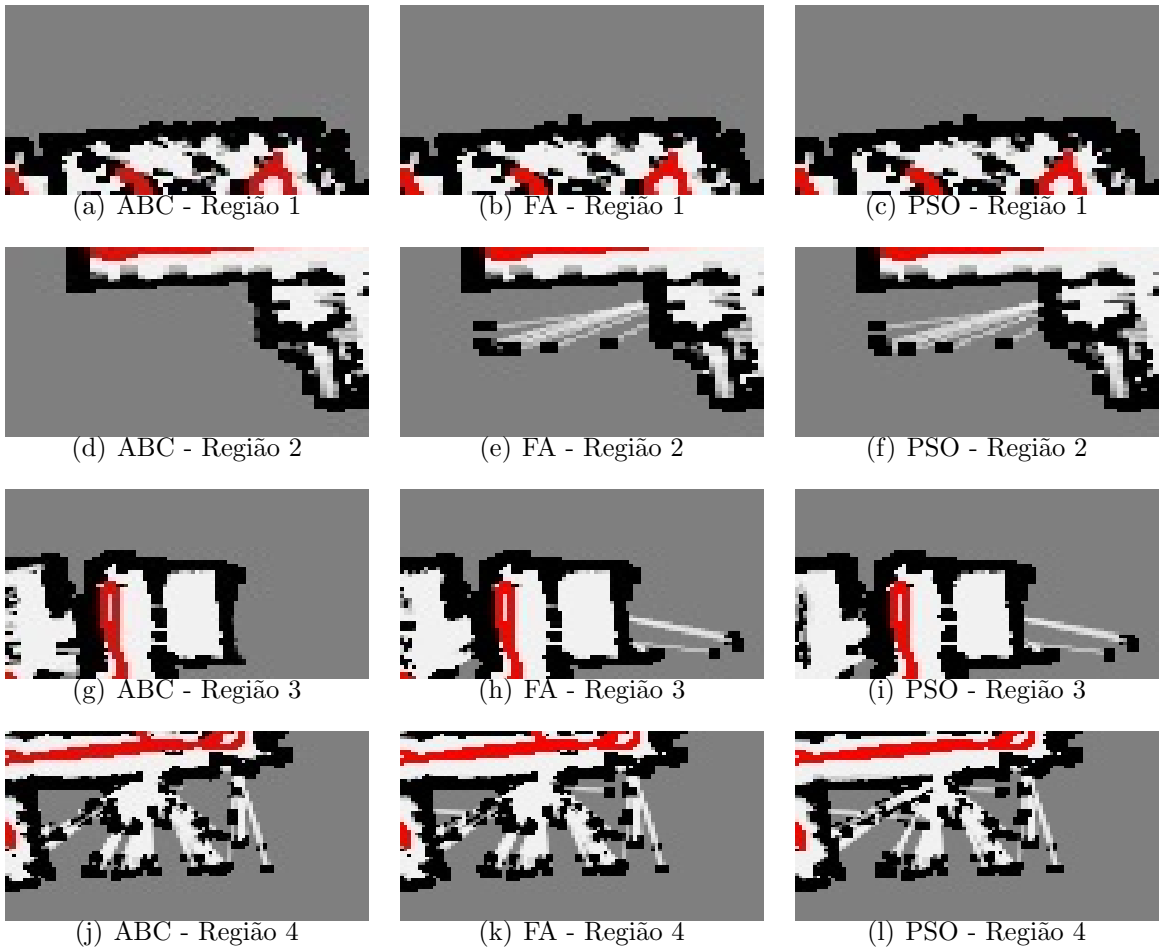
Figura 32: Mapa do prédio 79 da universidade de Freiburg gerado com as três meta-heurísticas sob análise.

Em todas as instancias o ABC demonstrou um resultado superior ao final do processo. Isso é indício de que a qualidade das estimativas de transformações relativas fornecidas pelo ABC foi superior: às do FA e às do PSO.

7.3.2 Corredor Infinito do MIT

Os resultados obtidos nesse experimento mostram uma maior discrepância em relação à referência (*groundtruth*). Isso já era esperado em virtude das características que mencionamos ao descrever esse conjunto de dados. Os enormes corredores com pouca variação morfológica dos contornos observáveis e os grandes laços contribuíram para um erro médio maior para o conjunto de restrições do grafo após o término do SLAM. Uma análise dos resultados apresentados na Figura 34 revela, que todas as meta-heurísticas mostraram uma baixa acurácia.

Deveras, a supramencionada discrepância pode ser observada por meio de uma análise do diagrama de barras presente na Figura 48. A existência de longos corredores aliada à prensença de diversos, e longos laços, exacerba a tendência do sistema de cometer erros na componente angular das poses estimadas, conforme pode ser observado na Figura 49. O erro na determinação da orientação do sistema pode levar à ocorrên-



(a) ABC - Região 1 (b) FA - Região 1 (c) PSO - Região 1

(d) ABC - Região 2 (e) FA - Região 2 (f) PSO - Região 2

(g) ABC - Região 3 (h) FA - Região 3 (i) PSO - Região 3

(j) ABC - Região 4 (k) FA - Região 4 (l) PSO - Região 4

Figura 33: Comparação das quatro regiões de interesse do prédio 79 da universidade de Freiburg selecionadas para análise, geradas com as três meta-heurísticas sob investigação.

cia do fenômeno conhecido como *drift* na ausência de medidas exteroceptivas confiáveis que permitam ancorar alguma das poses a determinada região do espaço. A natureza incremental do sistema descrito no presente trabalho faz com que esse tipo de erro, na falta de forte evidência que o exponha, se perpetue e, desse modo, corrompa todas as inferências realizadas a partir do instante em que fôra cometido. A contramedida adotada para mitigar esse problema se baseia no uso da busca por locais já visitados descrita na Seção 4.3.1. É digna de nota, entretanto, a influência da grande similaridade entre as varreduras coletadas ao longo do percurso realizado na coleta desse conjunto de dados sobre o reconhecimento de fechamentos de laço mediante uma análise morfológica do ambiente por elas descrito. A supramencionada similaridade leva à um grande número de possíveis correspondências para cada varredura, e essa ambiguidade reduz a eficácia do processo de criação de restrições extemporâneas. A falha, ou mesmo a baixa eficiência, na criação de restrições entre referenciais correspondentes a varreduras coletadas em instantes de