GUILHERME CARLOS RODRIGUES DE OLIVEIRA

UMA ABORDAGEM HÍBRIDA PARA O PLANEJAMENTO DE CAMINHOS PARA ROBÔS MÓVEIS EM AMBIENTES SEMI-ESTRUTURADOS FECHADOS E PARCIALMENTE OBSERVÁVEIS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS - BRASIL
2018

Ficha catalográfica preparada pela Biblioteca Central da Universidade Federal de Viçosa - Câmpus Viçosa

T
O48a
2018

Oliveira, Guilherme Carlos Rodrigues de, 1989-
Uma abordagem híbrida para o planejamento de caminhos para robôs móveis em ambientes semi-estruturados fechados e parcialmente observáveis / Guilherme Carlos Rodrigues de Oliveira. – Viçosa, MG, 2018.
xiii, 64 f. : il. (algumas color.) ; 29 cm.

Inclui anexo.
Orientador: Alexandre Santos Brandão.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f. 61-64.

1. Algorítmos computacionais. 2. Robôs - Programação. 3. Robótica. 4. Kinect (Controlador programável). 5. Detectores. I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. II. Título.

CDD 22. ed. 518.1

GUILHERME CARLOS RODRIGUES DE OLIVEIRA

UMA ABORDAGEM HÍBRIDA PARA O PLANEJAMENTO DE CAMINHOS PARA ROBÔS MÓVEIS EM AMBIENTES SEMI-ESTRUTURADOS FECHADOS E PARCIALMENTE OBSERVÁVEIS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae.*

APROVADA: 29 de junho de 2018.

Ricardo dos Santos Ferreira

Marcos Henrique Fonseca Ribeiro

Mario Sarcinelli Filho

Alexandre Santos Brandao
(Orientador)

Dedico essa dissertação à minha família, pelo apoio e amor incondicional.

"The good thing about science is that it's true whether or not you believe in it."
(Neil deGrasse Tyson)

Agradecimentos

Ao Prof. Dr. Alexandre Santos Brandão, meu orientador, dedico minha total gratidão por ter me aceitado e acreditado na minha capacidade. Pela paciência e excelência ao me ensinar e guiar sobre os estudos em robótica, sempre me manteve motivado e apaixonado pelo estudos nessa área, até mesmo a "parte matemática da coisa". Espero continuar aprendendo e levando seus conhecimentos independentemente dos rumos que a vida nos levar.

À minha família, por me amar e confortar nos momentos difíceis, assim como apoiar e incentivar sempre que necessitado, palavras não seriam suficientes para demonstrar meus mais sinceros agradecimentos.

Aos companheiros do NERO (Núcleo de especialização em robótica), agradeço o suporte, sempre prestativos e atenciosos para solucionar problemas. Ao Kevin, colega de trabalho, deixo meu agradecimento especial pela forma sempre prestativa e capaz com que me auxiliou na solução de problemas, escrita de artigos e filmagem de experimentos.

Aos amigos, colegas e chefia da Diretoria de Tecnologia da Informação (DTI-UFV) que me apoiaram nessa jornada. À Universidade Federal de Viçosa, por proporcionar a oportunidade de ingressar e cursar o programa de Pós-Graduação simultaneamente às minhas atividades como servidor.

Aos professores e funcionários do departamento de informática da Universidade Federal de Viçosa, em especial aos que ministram as disciplinas que cursei durante o programa, sempre muito dedicados e capazes.

À minha mãe, Neuza Maria Rodrigues, por todos esses anos exercendo função de pai e mãe, por ser a pedra fundamental da minha formação moral, sempre me incentivando a crescer profissionalmente e academicamente, meu mais puro agradecimento e gratidão eterna.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

OLIVEIRA, Guilherme Carlos Rodrigues de, M.Sc., Universidade Federal de Viçosa, junho de 2018. **Uma abordagem híbrida para o planejamento de caminhos para robôs móveis em ambientes semi-estruturados fechados e parcialmente observáveis.** Orientador: Alexandre Santos Brandão.

A robótica móvel aborda problemas relacionados a plataformas móveis, capazes de mudar sua localização no ambiente onde se encontram. Dentre as estratégias de navegação, encontra-se o planejamento de caminhos, cujo objetivo é encontrar rotas, ou um conjunto de ações, que levem um robô do local onde se encontra até um local de destino de forma eficiente. Para solucionar tal problema, deve-se levar em consideração algumas restrições, tais como capacidade de sensoriamento e tipo de ambiente. As soluções são divididas em duas formas distintas, o planejamento global e planejamento local. A primeira delas procura por caminhos completos que conectam o ponto de partida até o destino, evitando colisões com obstáculos conhecidos. Tal abordagem exige conhecimento prévio do ambiente e algum mecanismo de replanejamento para lidar com mudanças durante a navegação. Em contrapartida, a segunda lida com situações baseadas nas informações obtidas pelos sensores em tempo real, relacionadas ao desvio de obstáculos e limitadas à uma parte do ambiente onde o robô se move. Neste contexto, este trabalho aborda o planejamento de caminhos com uma solução híbrida, que mescla uma estratégia global e local. Para tal, utiliza-se uma navegação reativa (local) quando não existe informações prévias suficientes para traçar um caminho global. Enquanto o robô se move, as informações captadas pelos sensores modelam, de forma incremental, o ambiente. Uma vez que as informações contidas no mapa sejam suficientes, utiliza-se a estratégia global para traçar um caminho. Durante o seguimento do caminho obtido pela estratégia global, o planejamento local é novamente utilizado para evitar colisões com obstáculos que não eram previamente conhecidos ou que mudaram de localização

durante a navegação, dando, portanto, o caráter híbrido da solução. O algoritmo desenvolvido foi implementado, testado e validado em experimentos utilizando a plataforma robótica Pioneer P3-DX e o sensor de Xbox 360 Kinect. Tanto o robô quanto o sensor utilizados caracterizam plataformas de baixo custo, pois possuem capacidades limitadas de sensoriamento, característica que é utilizada para validar a eficiência do algoritmo.

Abstract

OLIVEIRA, Guilherme Carlos Rodrigues de, M.Sc., Universidade Federal de Viçosa, June, 2018. **A hybrid approach for the path planning problem of mobile robots at indoor semi-structured partially observable environments.** Adviser: Alexandre Santos Brandão.

Mobile robotics addresses problems associated with robotic platforms that are capable of changing their location in the environment. Among the navigation strategies one has the path planning, whose goal is to find routes, or a set of actions, that will lead a robot from its current place to a destination place in an efficient way. To solve this problem, some constraints must be taken into consideration, such as sensorial capabilities and the type of environment. The solutions are divided into two distinct ways, the global and local planning. The first one searches for complete paths that connect the starting point to the destination, avoiding collisions with previously known obstacles. Such approach requires a priori environment knowledge and some replanning mechanism to deal with changes during the navigation. On the other hand, the second solution deals with situations based on the sensor information obtained in real-time, related to the obstacle avoidance and limited to a part of the environment where the robot moves. In this context, this work proposes hybrid solution to the path planning problem, which merges a local and a global strategies. Thus, a reactive (local) navigation is used when there is no previous information to obtain a global path. While the robot moves, the information obtained by the sensors is used to incrementally model the environment. Once the information contained in the map is enough, the global strategy is used to find a path. While following the globally planned path, the local strategy is used to avoid collisions with obstacles not previously known or that changed their location after the navigation started, thus characterizing the hybrid solution. The proposed algorithm was implemented, tested and validated in experiments using the robotic platform

Pionner P3-DX and the sensor for Xbox 360 Kinect. Both the robot and sensor used are low-cost platforms, given the limited sensor capabilities, whose objective was to validate efficiency of the algorithm.

Capítulo 1

INTRODUÇÃO

Robótica é a ciência que estuda a percepção e manipulação do mundo real através de dispositivos computacionais [Thrun, 2002, p. 1, tradução]. Suas aplicações vão desde braços robóticos que auxiliam cirurgiões e robôs de limpeza doméstica até veículos de exploração interplanetária. A evolução da robótica, desde meados do século XX até a atualidade, deixa de ser predominante no cenário industrial da produção em larga escala para fazer parte do ambiente doméstico, auxiliando em tarefas cotidianas e estreitando o relacionamento homem-máquina [Ray et al., 2008].

1.1 Robótica móvel

Um robô, ou agente, é uma entidade que tem como propósito modificar o estado atual do mundo, ou ambiente, para um novo desejado, a fim de realizar uma tarefa. Para isso, ele conta com sensores e atuadores, responsáveis por perceber o ambiente e realizar as transformações necessárias, respectivamente. Câmeras, sonares e microfones são alguns exemplos de sensores, enquanto rodas, braços e asas são exemplos de atuadores.

A ideia básica do funcionamento de um agente é a de que se fornece como entrada as informações do ambiente, para que sejam mapeadas através de uma "função agente" em ações realizadas pelos atuadores [Russell & Norvig, 2009]. O fluxo básico de um agente é ilustrado pela Figura 1.1.

É possível distinguir três categorias de raciocínio dos agentes, dada a forma como eles tomam decisões. Os agentes podem ser deliberativos, reativos ou híbridos [Wooldridge, 2009].

Agentes deliberativos ponderam o estado atual do ambiente conhecido, seus objetivos e ações possíveis antes da tomada de decisão. Tal comportamento pode

ser descrito como planejamento, onde um algoritmo fornece um plano a ser seguido para atingir um objetivo. A Figura 1.2 ilustra o conceito de agente deliberativo.

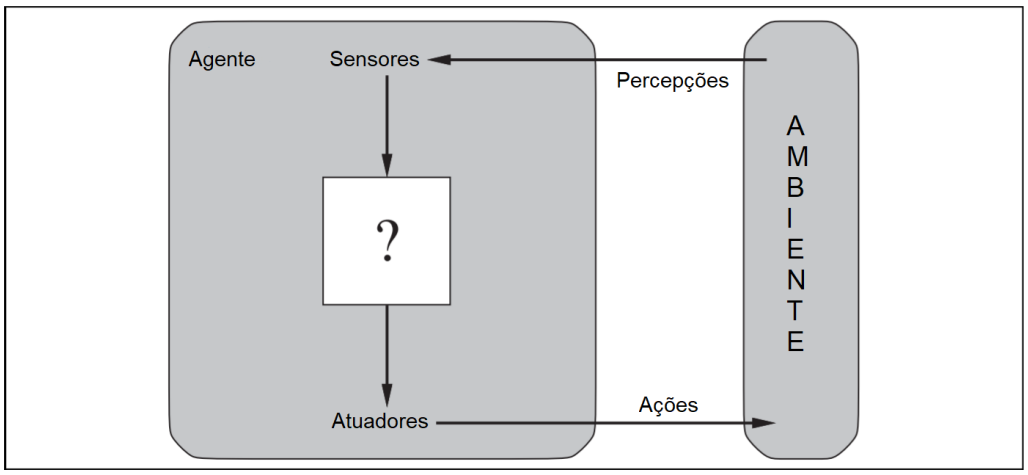

Figura 1.1. A ideia básica de um agente

Fonte: Adaptado de [Russell & Norvig, 2009, p. 35]

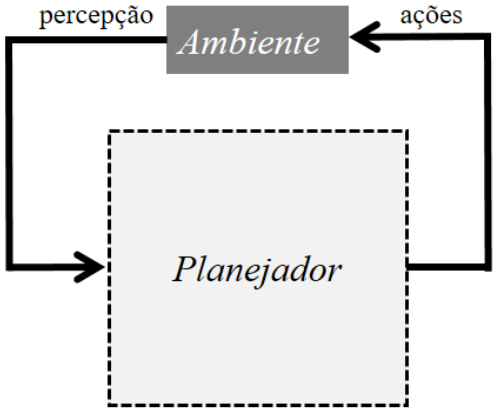

Figura 1.2. Agente deliberativo

Fonte: (Própria, 2018)

Agentes reativos são caracterizados por uma estrutura estimulo-reação, onde para cada informação de entrada captada pelos sensores existe uma ação mapeada, sem que haja conflito de ações. Apesar de não existir um plano definido, as ações normalmente são tomadas com base em tarefa, de forma à atingir um objetivo geral. A Figura 1.3 ilustra o conceito de agente reativo.

Agentes híbridos, de uma forma geral, são aqueles que possuem ao menos duas camadas, uma para lidar com o comportamento deliberativo e outra para lidar com

o reativo, antes da tomada de decisão em forma de ação. A Figura 1.4 ilustra o conceito de agente híbrido.

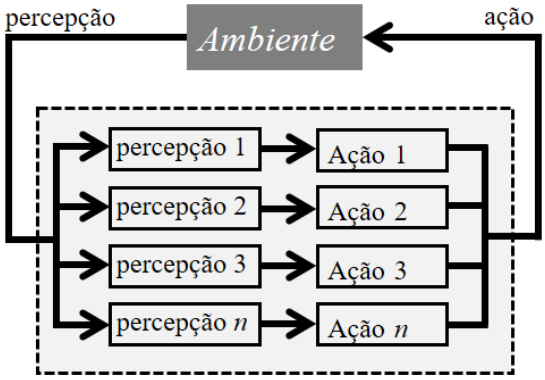

Figura 1.3. Agente reativo

Fonte: (Própria, 2018)

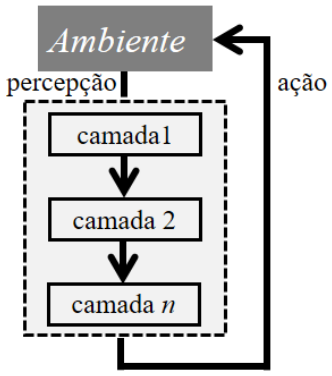

Figura 1.4. Agente híbrido

Fonte: (Própria, 2018)

A robótica móvel aborda problemas relacionados a robôs móveis, equipados com atuadores capazes de mudar sua localização no ambiente. A pesquisa nesta área tem como objetivo a construção de sistemas autônomos, capazes de se mover e tomar decisões inteligentes sem qualquer interferência humana. O uso de robôs móveis possibilita o exercício de atividades que seriam muito perigosas, exaustivas ou até mesmo impossíveis para humanos. Pode-se citar como exemplo a exploração de áreas radioativas, busca por sobreviventes em desastres e explorações no fundo do oceano ou no espaço.

Por ser tratar de uma área multidisciplinar, a robótica móvel pode ser abordada sob diferentes aspectos do desenvolvimento de plataformas móveis, desde a

Engenharia Mecânica, com o design de mecanismos de locomoção, a Engenharia Elétrica, com a integração dos componentes físicos, a Ciência da Computação com a representação e algoritmos de planejamento, até a Psicologia e Neurociência, traçando paralelos com organismos biológicos e interação homem-máquina [Dudek & Jenkin, 2010].

A estrutura de navegação autônoma de um robô é composta essencialmente por quatro componentes: percepção, localização, planejamento e controle de movimento [Mac et al., 2016], cada uma com problemas próprios. Percepção e localização normalmente estão juntas e se referem a extrair do ambiente informações uteis para navegar e se localizar neste mesmo ambiente, respectivamente. Planejamento se refere a encontrar uma sequência de ações para se navegar entre locais do ambiente. Controle de movimento são as ações necessárias para realizar de fato a navegação no ambiente. A Figura 1.5 ilustra os componentes da navegação.

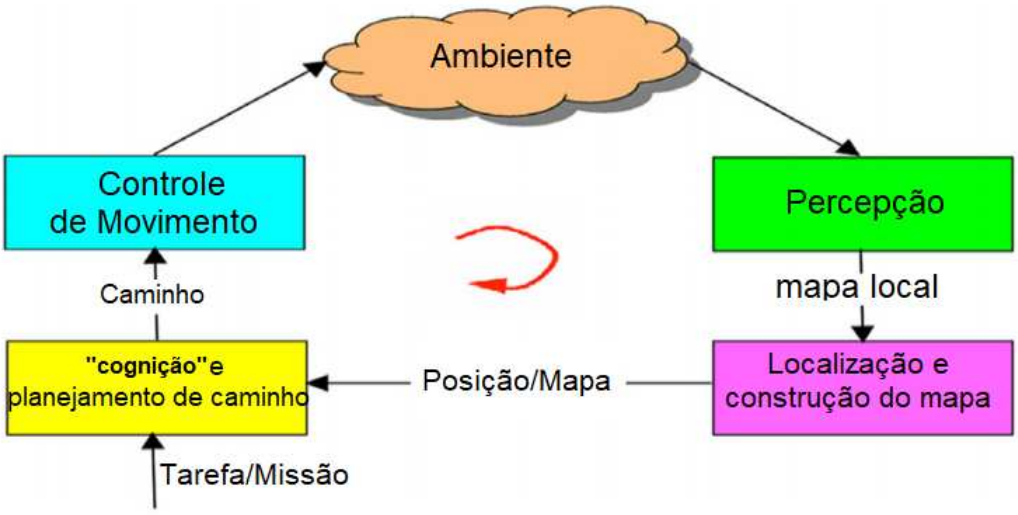

Figura 1.5. Componentes da navegação

Fonte: Adaptado de [Mac et al., 2016, p. 14]

As características do ambiente e sensores impactam diretamente na tomada de decisão. Russell & Norvig [2009] definem alguns conceitos importantes sobre o ambiente onde se pretende realizar uma tarefa. Um ambiente é dito totalmente observável quando se tem total acesso a todas informações relevantes para tomada de decisão, em qualquer instante no tempo. Caso contrário, quando não é possível obter alguma informação em algum instante no tempo ou quando as informações captadas pelos sensores não são completas, o ambiente é dito parcialmente observável. Neste caso, pode-se dizer que os sensores do robô possuem alcance limitado, não sendo

possível cobrir todo o ambiente em todos os momentos.

Quando os elementos do ambiente, sejam objetos ou outros agentes, podem mudar de localização enquanto o agente principal está tomando uma decisão, o ambiente é dito dinâmico. Quando essa situação não ocorre, o ambiente é chamado de estático, e é possível planejar ações sabendo que tudo já conhecido do ambiente não irá se alterar com o passar do tempo.

Além disso, é possível distinguir um ambiente em três categorias, baseado no grau de incerteza em relação a sua estrutura e elementos presentes. O ambiente pode ser estruturado, semi-estruturado ou não-estruturado. Em um ambiente estruturado, tanto sua estrutura topológica quanto os elementos são estáticos. A situação inversa ocorre em um ambiente não-estruturado, onde tanto a estrutura topológica quanto os elementos são dinâmicos. Já em um ambiente semi-estruturado, a estrutura é bem definida, mas existe incerteza sobre a localização dos elementos. Um exemplo clássico desse tipo de ambiente é um estacionamento, onde as vagas são fixas mas não se sabe quais estão ocupadas por veículos.

1.2 Mapeamento e modelagem do ambiente

O problema de mapeamento, na busca pela navegação autônoma, diz respeito à utilização de informações captadas por sensores, tais como câmeras, sensores laser de varredura, sonares e GPS, para representar o ambiente em estruturas computacionais, de forma a identificar objetos (também chamados de obstáculos) e áreas livres, melhorando a tomada de decisão. Na maioria das situações, o ambiente é parcialmente observável, oque força o robô a se locomover e explorar o local onde se encontra [Thrun, 2002].

Ao se mover pelo ambiente para efetuar o mapeamento, surge um segundo problema relacionado à localização do robô no mapa que está sendo construído. A junção desse problema com o mapeamento é definido como SLAM (do inglês, *Simultaneous Localization and Mapping*), o problema de simultaneamente utilizar a navegação do robô para modelar o ambiente em uma estrutura computacional e se localizar dentro dessa mesma estrutura [Thrun, 2002]. Dada a natureza imperfeita do mundo, onde as características estruturais de objetos e formas de medição são suscetíveis a erros e ruídos, a maioria dos algoritmos de mapeamento aborda o problema de forma probabilística [Thrun et al., 2002].

Dentre as estruturas utilizadas na modelagem, pode-se dividi-las em duas grandes categorias: estruturas de decomposição em células e estruturas de caminhos

conectando locais no ambiente [Souissi et al., 2013]. Destas, destaca-se o uso de *grids*, definida como uma estrutura de decomposição em células, onde cada célula do *grid* representa uma porção do mundo. Esse tipo de modelagem é simples e ao mesmo tempo muito eficiente, que facilita tanto a representação do mundo quanto na execução de algoritmos.

Considerando um ambiente modelado em um plano bidimensional, as três figuras geométricas mais utilizadas para compor um *grid* são quadrados (Figura 1.6), triângulos (Figura 1.7) e hexágonos (Figura 1.8). A desvantagem na utilização de estruturas geométricas na decomposição de um *grid* se dá na perda da precisão de representação de objetos, assim como a relação direta do aumento no consumo de memória com maior resolução do *grid* [Souissi et al., 2013].

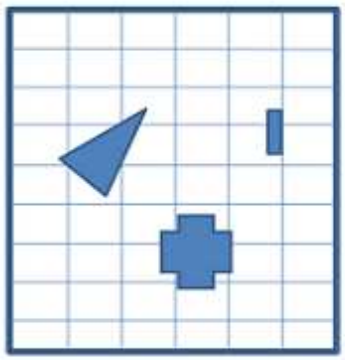

Figura 1.6. *Grid* com células quadradas.

Fonte: Retirado de [Souissi et al., 2013, p. 2]

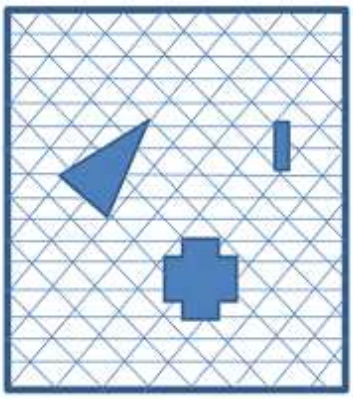

Figura 1.7. *Grid* com células triangulares.

Fonte: Retirado de [Souissi et al., 2013, p. 2]

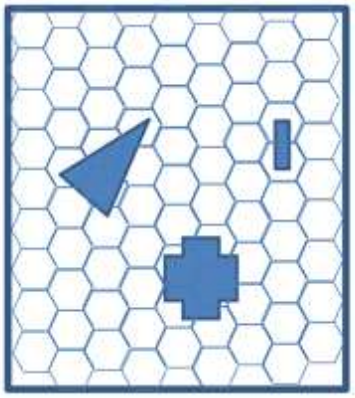

Figura 1.8. *Grid* com células hexágonais.

Fonte: Retirado de [Souissi et al., 2013, p. 2]

Figura 1.9. Planta de um prédio.

Fonte: Adaptado de [Thrun et al., 2002, p. 17]

O algoritmo probabilístico de mapeamento Occupancy Grid [Elfes, 1989] constrói incrementalmente um mapa, estruturado em um *grid*, por onde navega o robô. A estratégia tem como base o cálculo da probabilidade, dado um conjunto de medições em um instante de tempo, das células referentes à área mapeada estarem ocupadas ou vazias. Desta forma, todas as células do *grid* possuem um valor binário, que indica se a região representada do mundo está ocupada ou vazia.

As vantagens de Occupancy Grid são sua facilidade de implementação e robustez, normalmente sendo utilizado em problemas de duas dimensões, ainda que seja possível lidar com problemas em três dimensões [Thrun et al., 2002]. Além disso,

o *grid* binário obtido a partir do algoritmo simplifica a utilização de algoritmos de planejamento de caminho. As Figuras 1.9 e 1.10 ilustram o mapeamento utilizando Occupancy Grid.

Figura 1.10. Mapa da Figura 1.9 obtido através de Occupancy Grid.

Fonte: Adaptado de [Thrun et al., 2002, p. 17]

1.3 Planejamento

O Planejamento, também descrito como planejamento de caminhos na robótica móvel, pode ser visto como o problema de fornecer ao robô uma forma de se mover a fim de se sair de um local de partida até chegar em um local de destino dentro do ambiente. Uma definição clássica de planejamento é o chamado Problema do Piano [LaValle, 2006], que consiste em, considerando que se conheça a planta baixa de uma casa, mover um piano de um cômodo para outro, sem bater nos outros móveis da casa.

As soluções para o problema podem se apresentar sob diferentes níveis de abstração [LaValle, 2006]. Em teoria de controle, por exemplo, algoritmos fornecem como saída sinais de controle, tais como velocidade linear e angular, de forma continua, com realimentação sensorial. Já na inteligencia artificial o problema pode ser visto de forma discreta, onde se espera encontrar uma sequência de ações que levem o robô por um conjunto de estados, a partir do atual até o objetivo. Alguns

conceitos importantes, como estados, ações, plano e critério, descritos em [LaValle, 2006], são resumidos a seguir:

- Estado: Define uma situação atual de um sistema. Na robótica móvel, um estado pode ser descrito como posição e orientação do robô no plano. Um espaço de estados envolve todas as combinações de estados possíveis, ainda que na maioria das vezes não seja viável representá-lo explicitamente.

- Ação: Também descritos como operadores ou controles, são responsáveis por realizar a transição de um estado para outro. Essa transição pode ser definida a partir de funções estado-valor no tempo discreto ou equações diferenciais ordinárias no tempo contínuo.

- Plano: É a sequência de ações, ou estratégia, para se atingir o objetivo.

- Critério: É o resultado esperado ao seguir um plano. Os dois tipos de critério geralmente utilizados são o da alcançabilidade e otimalidade. O primeiro refere-se a alcançar o objetivo da navegação, enquanto o segundo diz respeito à eficiência, como tempo de processamento, tempo de navegação, distância percorrida e consumo de energia.

Um ponto essencial no desenvolvimento de um algoritmo de planejamento é o critério da alcançabilidade, pois espera-se que o robô chegue ao seu objetivo em um tempo finito, sem colidir com obstáculos. Nesse contexto, o volume de informações a respeito do ambiente tem um papel fundamental na tomada de decisão. Baseado em tal condição, é possível separar as soluções em duas categorias: planejamento local e planejamento global.

Algoritmos de planejamento local, de uma forma geral, são soluções que se limitam à tarefa de desvio de obstáculos em tempo real durante a navegação. Normalmente possuem baixo custo computacional e são de fácil implementação. Já para o planejamento global, algoritmos procuram encontrar um caminho completo que liga um ponto de partida até um ponto de destino no ambiente, reduzindo a distância percorrida e evitando obstáculos conhecidos, mas podem demandar mais tempo e processamento para encontrar uma solução eficiente. Os dois tipos de planejamento de caminho serão discutidos mais detalhadamente no próximo capítulo.

O planejamento híbrido, como o nome sugere, utiliza camadas de planejamento global e local. Ao se abordar o problema de forma híbrida, espera-se encontrar uma solução que seja eficiente em termos de caminho percorrido e custo computacional para navegação com desvio de obstáculos, absorvendo as vantagens de ambas as

soluções e procurando eliminar as desvantagens, ainda que nem sempre de forma integral.

1.4 O robô móvel Pioneer P3-DX

Sendo um robô do tipo uniciclo [CONTE et al., 1996], o Pioneer P3-DX (Figura 1.11) possui dois motores e duas rodas sobre o mesmo eixo, capazes de operar de forma independente, configurando um sistema de tração diferencial. Essa característica permite ao robô realizar manobras como girar sobre o próprio eixo, mas o impede de realizar manobras como andar lateralmente, por exemplo.

Por possuir restrições de movimento, considerando a navegação no plano bidimensional, o P3-DX é classificado como um robô não-holonômico [Laumond et al., 1998]. Tal classificação também se aplica à todos robôs do tipo uniciclo. No caso contrário, quando não existe nenhuma restrição de movimento no plano por onde navega, o robô é classificado como holonômico.

Figura 1.11. O robô do tipo uniciclo Pioneer P3-DX.

Fonte: Adaptado de [Adept Technology, 2011]

O Pioneer P3-DX é equipado de fábrica com 8 sensores de distância do tipo sonar, fixados na parte frontal de sua plataforma. Os sensores possuem faixa de atuação entre 0 e 5 metros, com medidas entre os ângulos -90° ,-50° ,-30° ,-10° ,+10° ,+30° ,+50° e +90°, usando como referência a frente do robô. Desconsiderando o alcance dos sensores e considerando uma visão frontal de 180° com resolução de um grau, o P3-DX consegue detectar apenas 4,41% da região a sua frente.

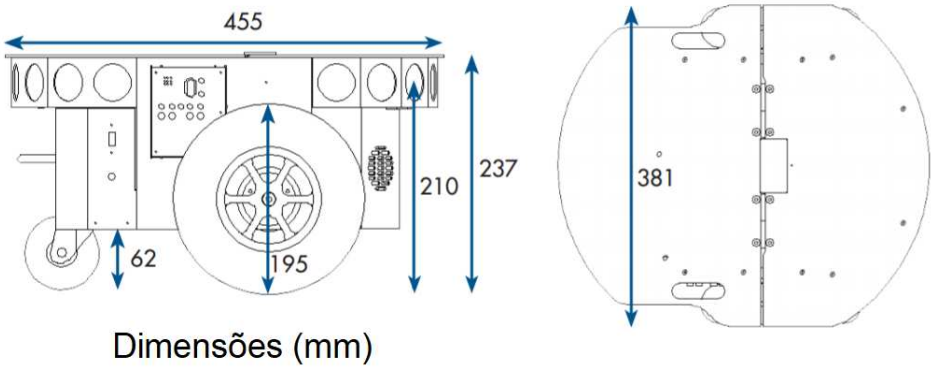

Figura 1.12. Dimensões do P3-DX.

Fonte: Adaptado de [Adept Technology, 2011]

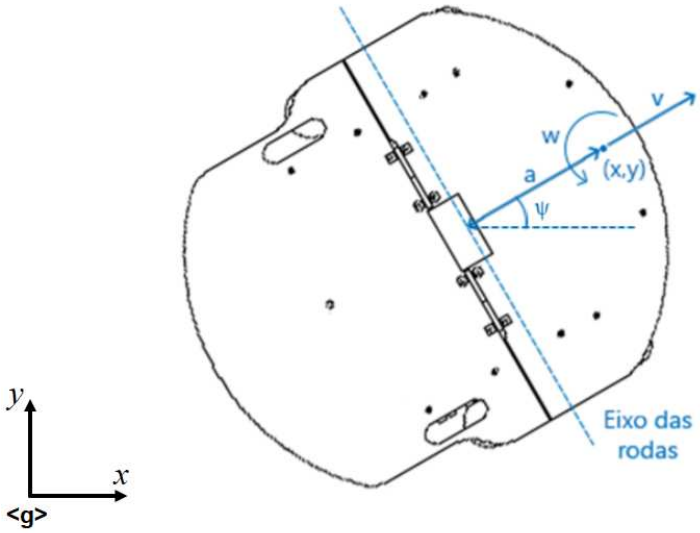

Figura 1.13. Modelo cinemático do P3-DX.

Fonte: Adaptado de [Fernandes et al., 2017, p. 1]

Para à utilização de controladores de movimento é comum definir um modelo cinemático para a plataforma móvel que executará a navegação. Considerando robôs do tipo unicilo, um modelo muito abordado na literatura, segundo [Fernandes et al., 2017], é dado por:

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -a\sin(\psi) \\ \sin(\psi) & a\cos(\psi) \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{1.1}$$

com $\dot{\psi} = \omega$, onde (x, y) é a posição do robô no referêncial $< g >$, ψ é a sua orientação em relação ao eixo x e a é a distância entre o eixo de tração virtual do robô e o ponto de controle, como ilustra a Figura 1.13.

Para a sequência dessa dissertação, o Pioneer P3-DX e o modelo cinemático descrito nessa seção serão utilizados como referência para algoritmos, controladores, simulações e experimentos realizados. A escolha se dá pelo fato da plataforma ser popular como objeto de estudo e pesquisa acadêmica. Além disso, o fato de o robô possuir um sistema de sensoriamento com medição em apenas 8 graus o caracteriza como uma plataforma de baixa capacidade de sensoriamento, ideal para avaliar o planejamento híbrido no contexto proposto.

1.5 O sensor Kinect

O sensor Kinect para Xbox 360 tem disponível duas câmeras, uma RGB e outra de profundidade, como ferramentas de sensoriamento. A câmera de profundidade possui alcance mínimo de 0,8 metros e máximo de 4 metros. Além disso, possui abertura de 57° na horizontal e 34° na vertical. Produzido com propósito de entretenimento em jogos virtuais, o Kinect também se mostra bastante popular em pesquisas acadêmicas, pelo fácil acesso em termos de valor para compra, sobretudo quando comparado a sensores de varredura à laser. Assim, a utilização desse sensor é uma alternativa de baixo custo para aplicações em robótica em ambientes fechados, especialmente para executar tarefas de mapeamento, como descrevem Khoshelham & Elberink [2012].

Nos experimentos realizados, o Kinect é utilizado juntamente com o sonar do Pioneer P3-DX (Figura 1.14) para efetuar o desvio de obstáculo. Além do desvio, o sensor também é utilizado para realizar o mapeamento do ambiente através do algoritmo *Occupancy Grid*, tendo em vista que é mais preciso e pode varrer uma área maior que os sonares do P3-DX, acelerando o processo de mapeamento. O uso do sensor e as razões de sua utilização são discutidos no Capítulo 4.

Figura 1.14. Kinect para Xbox 360 acoplado ao P3-DX.

Fonte: (Própria, 2018)

1.6 Descrição do problema

Nesta dissertação, limita-se o planejamento de caminhos para robôs terrestres com navegação no plano bidimensional, em ambientes parcialmente observáveis, semi-estruturados e sem qualquer informação *a priori*. O trabalho têm como foco plataformas de baixo custo e com capacidade de sensoriamento e processamento limitadas, para as quais algoritmos desenvolvidos devem ser eficientes computacionalmente.

Dentre possíveis tarefas realizadas em ambientes com características similares as restrições do problema, destaca-se o ambiente fabril, onde um robô deve fazer o carregamento e descarregamento de cargas em um ambiente com um certo grau de incerteza (pequenos objetos e outras cargas sobre o trajeto a ser percorrido, por exemplo), percorrendo um mesmo caminho, em um trajeto de ida e volta, diversas vezes. Esse cenário também é encontrado em hospitais, onde robôs auxiliam pacientes incapacitados e em bibliotecas, com o transporte de livros e materiais.

Nesse contexto, o algoritmo de planejamento recebe como entrada um conjunto de um ou mais pontos (coordenadas no plano bidimensional) a serem alcançados pelo robô durante a navegação. O primeiro ponto do conjunto é definido inicialmente como objetivo, até que o robô o alcance, ou seja, a posição do robô no ambiente seja igual a posição de destino. Quando isso ocorre, o próximo ponto se torna o novo destino e o processo se repete até que se chegue ao ultimo ponto do conjunto de entrada.

Com o robô parado, antes do inicio de cada movimento em direção aos pontos

fornecidos, a estratégia global é utilizada para procurar um caminho completo até o objetivo. Caso não exista, navega-se de forma reativa desviando de obstáculos em tempo real com o planejamento local. No caso contrário, o robô deve seguir o caminho obtido até alcançar o destino. Para evitar colisões com obstáculos desconhecidos durante o seguimento do caminho, o planejamento local é novamente utilizado, com o robô retornando a trajetória traçada após a evasão. Durante toda a navegação, as informações dos sensores são utilizadas para a modelagem incremental do ambiente.

1.7 Objetivos

O objetivo geral desse trabalho é desenvolver uma estratégia de navegação para robôs terrestres com base em um algoritmo híbrido de planejamento de caminho, no plano bidimensional e em ambientes parcialmente observáveis, semi-estruturados e não mapeados *a priori*. O algoritmo deve promover uma navegação continua e o desvio de obstáculos, conhecidos ou desconhecidos. Para isso, alguns objetivos específicos são listados a seguir:

- Mapear de forma incremental o ambiente utilizando apenas as informações dos sensores acoplados ao robô.
- Encontrar, quando as informações no mapa forem suficientes, caminhos completos e livres de colisões com obstáculos conhecidos, utilizando a estratégia global.
- Enquanto se segue um caminho planejado na estratégia global, desviar de obstáculos desconhecidos e retomar o caminho inicialmente planejado.

1.8 Estrutura do trabalho

Este trabalho está divido da seguinte forma: o Capítulo 1 introduziu conceitos relevantes dentro da robótica móvel. A ideia de sensores, atuadores e tipos de arquiteturas são elucidadas. Além disso, são discutidos brevemente alguns dos principais problemas relacionados à robótica móvel. A subdivisão das soluções em planejamento local e global é descrita de forma simplificada, pontuando algumas vantagens e desvantagens de cada abordagem. Por fim, o objetivo principal do trabalho assim como os específicos são listados.

O Capítulo 2 discute o problema de planejamento de caminho de forma mais aprofundada, citando algoritmos que compõem o estado-da-arte. Para a estratégia

local destaca-se o algoritmo Desvio Tangencial [Brandão et al., 2013]. Para soluções globais é destacado o algoritmo de busca em grafo A* [Hart et al., 1968], com suas vantagens e desvantagens. Por fim, discute-se a escolha dos algoritmos para compor a solução híbrida.

O Capítulo 3 apresenta a solução proposta nessa dissertação. Discutem-se os objetivos do planejamento proposto e alguns pontos chaves para a navegação híbrida. O pseudocódigo da navegação com o planejamento híbrido é apresentado e explicado.

O Capítulo 4 traz os resultados dos experimentos realizados. Dois ambientes distintos, um de simulação e outro no mundo real, são utilizados como referência. Para ambos os casos, dados da navegação são apresentados e discutidos.

O Capítulo 5 apresenta a conclusão do trabalho realizado. Discutem-se os resultados obtidos e apresentam-se algumas propostas para trabalhos futuros.

Capítulo 2

REFERENCIAL TEÓRICO

Neste capitulo serão abordados alguns conceitos e algoritmos de planejamento de caminho global e local, relevantes no contexto da robótica móvel e para a discussão da solução híbrida que é proposta neste trabalho.

2.1 Planejamento local e o de Desvio Tangencial

O planejamento de caminho local parte da premissa que o robô necessita se movimentar pelo ambiente para responder à eventos em tempo real relacionados ao desvio de obstáculos [Buniyamin et al., 2011]. Como o ambiente normalmente é desconhecido *a priori*, o menor caminho até o destino, uma linha reta, é percorrido até que o sensores detectem um obstáculo. Nesse momento, os algoritmos fornecem as ações necessárias para evitar a colisão. Após o desvio, o robô segue um novo caminho em linha a partir de sua posição. Essa situação se repete até que se alcance o destino [Buniyamin et al., 2011]. A Figura 2.1 ilustra a ideia geral do planejamento local.

A principal vantagem de se utilizar o planejamento local é que se tem uma solução de baixo custo computacional e eficiente na tarefa de desvio de obstáculos, tendo em vista que o foco desse tipo de estratégia é apenas evitar a colisão com obstáculos em tempo real, baseado nas informações de sensores. Entretanto, como não se utiliza ou não se tem a informação sobre o ambiente além da região por onde o robô se move, o caminho tomado pode não ser o melhor, para se chegar ao objetivo, além do risco de cair em mínimos locais.

O algoritmo de campos potenciais artificiais (APF, do inglês *Artificial Potential Field*) [Khatib, 1986] é uma das soluções mais populares na literatura. A ideia geral é a de que o robô se move guiado pela resultante de forças artificiais, geradas

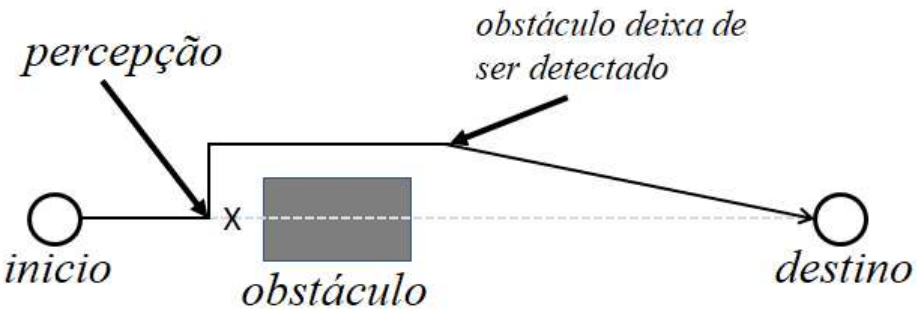

Figura 2.1. Navegação com um algoritmo de planejamento local básico

Fonte: (Própria, 2018)

por obstáculos e o ponto de destino. Uma força de atração move o robô em direção ao seu objetivo enquanto forças de repulsão geradas por obstáculos promovem o desvio de obstáculo. A simplicidade e elegância da estratégia acompanham diversas situações de mínimo local, problema que é, até hoje, tópico de pesquisa envolvendo APF [Souissi et al., 2013].

Outras soluções de sucesso na literatura, que abordam o problema de planejamento localmente com o foco em desvio de obstáculos, são o *Bug Algorithm* (*Bug1 and Bug2*) [Lumelsky & Stepanov, 1986] Campo de Força Virtual (VFF) [Borenstein & Koren, 1989], Histograma de Campo Vetorial (VFH) [Borenstein & Koren, 1991]. Alguns exemplos de soluções do planejamento local mais recentes são o *Tangential Gap Flow* (TGF) [Mujahed et al., 2016] e o *Follow The Gap Method* (FGM) [Demir & Sezer, 2017].

O algoritmo Desvio Tangencial [Brandão et al., 2013] promove o desvio de obstáculos fornecendo ao robô um alvo temporário a ser alcançado, de forma que ele siga um caminho tangente ao obstáculo. A tarefa é divida em dois passos. No primeiro, o robô deve detectar, através de sensores, o obstáculo no ambiente. A menor distância até o obstáculo e o ângulo entre o sensor que fornece essa medida e a direção atual do robô devem ser obtidos nessa etapa. No segundo passo deve-se efetuar o cálculo do alvo virtual temporário. Para isso, rotaciona-se o objetivo real do robô em um ângulo γ, obtido através da seguinte expressão:

$$\gamma = \begin{cases} -\frac{\pi}{2} - \alpha + \beta, & \text{se } \beta < 0 \\ +\frac{\pi}{2} - \alpha + \beta, & \text{caso contrário} \end{cases}, \tag{2.1}$$

onde α é definido como sendo o ângulo entre a distância do robô até o objetivo real e a direção atual do robô, e β é o ângulo entre a menor distância do robô até

o obstáculo e a direção (vetor de velocidade linear) atual do robô. A Figura 2.2 ilustra o Desvio Tangencial.

A estratégia é puramente reativa, dependendo apenas das informações sensoriais. A condição de ativação do algoritmo é a existência de alguma medida de distância captada pelos sensores que seja menor do que uma distância de referência d_{obs}. Uma vez que esta condição é satisfeita por um ou mais sensores, seleciona-se a menor distância, d_{min} e, o ângulo β referente a tal distância.

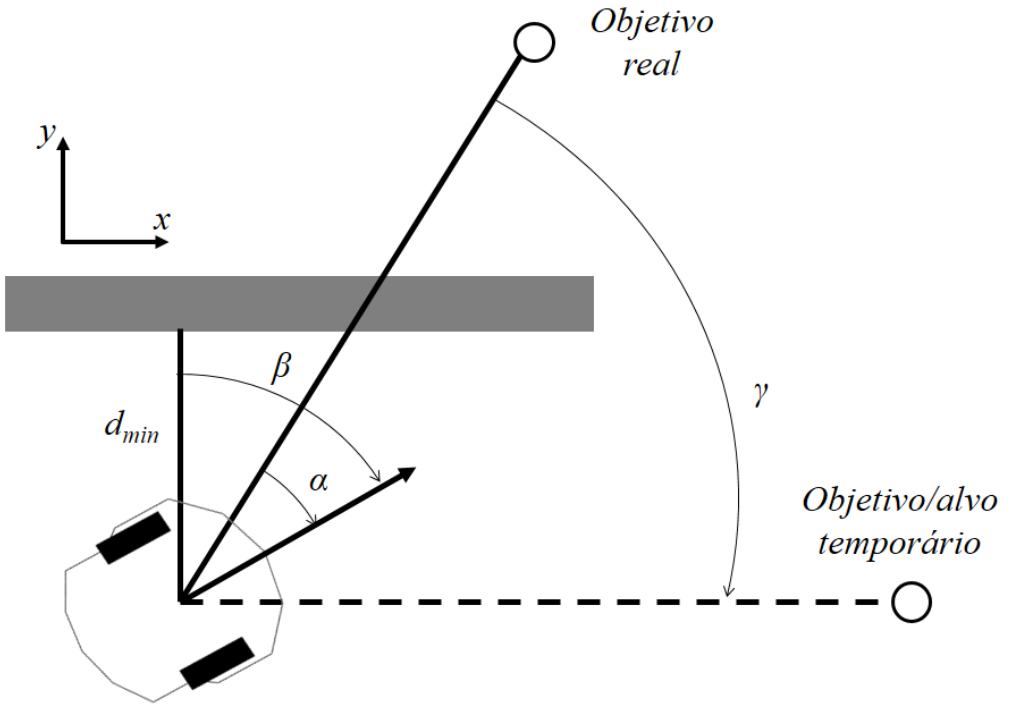

Figura 2.2. Desvio Tangencial

Fonte: (Própria, 2018)

Considerando a navegação no plano XY, o alvo temporário para se obter a evasão do obstáculo é obtido pela rotação do ponto objetivo de acordo com o ângulo γ, dada por

$$X_v = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) \\ \sin(\gamma) & \cos(\gamma) \end{bmatrix} X_d \ , \tag{2.2}$$

,

onde X_v é o alvo virtual temporário e X_d é o objetivo real de destino.

Uma situação importante, do ponto de vista prático da utilização do algoritmo, é quando o formato do obstáculo leva o robô a contorná-lo, sem de fato terminar a evasão (Figura 2.3). Para superar essa limitação, utiliza-se um fator de esquecimento

f_e ($f_e \in [0, 1]$), de tal forma que as rotações do alvo virtual sejam suavizadas, permitindo que o robô se afaste ligeiramente do alvo, fora da distância d_{min} definida. Assim, o novo cálculo de γ é descrito por:

$$\gamma[k] = (1 - f_e)\gamma[k-1] + \gamma[k]f_e, \tag{2.3}$$

onde $\gamma[k]$ é o atual valor, obtido por (2.2) e $\gamma[k-1]$ é o seu valor no instante de amostragem anterior. O pseudocódigo do Desvio Tangencial é exibido no algoritmo 1.

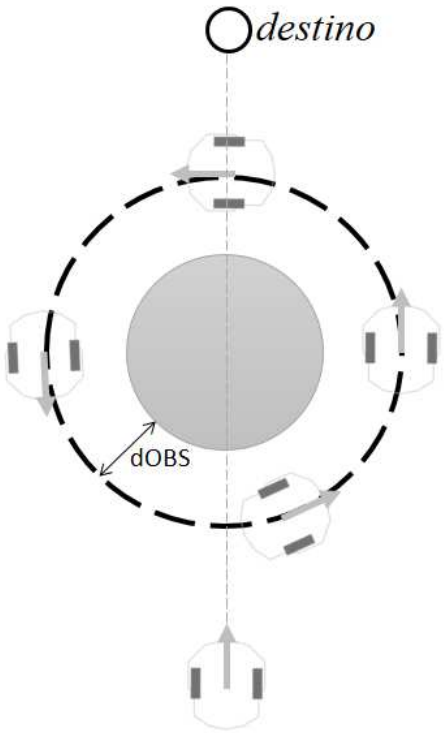

Figura 2.3. Contorno do obstáculo durante a evasão

Fonte: (Própria, 2018)

Inicialmente, o alvo virtual é definido como sendo o ponto objetivo (linha 2). Caso a menor distância (d_{min}) captada por sensores seja menor que a distância observável d_{obs} (linha 3), calcula-se os ângulos necessários para a rotação, β , θ e α (linhas 4-6). Se β for negativo (linha 7), significa que o alvo está a direita do robô e utiliza-se a expressão da linha 8 para se obter γ. Caso contrário (linha 9), utiliza-se a expressão da linha 10. A norma entre a posição atual do robô e o ponto objetivo (linha 12) é utilizada para obtenção do alvo virtual, obtido na linha 13. Caso a rotação não tenha ocorrido, o valor de γ é suavizado conforme o fator de

esquecimento (linha 15). Por fim, o valor de X_v, rotacionado ou não, é fornecido como saída do algoritmo (linha 18).

Algoritmo 1: DESVIO TANGENCIAL

Entrada: γ, f_e, d_{min}, d_{obs}
Saída: Alvo virtual X_v

1 **início**
2 $\quad X_v \leftarrow X_d$
3 $\quad$ **se** $d_{min} < d_{obs}$ **então**
4 $\quad\quad \beta \leftarrow AnguloDaMedidaMaisProxima$
5 $\quad\quad \theta \leftarrow \arctan(y_d/x_d)$
6 $\quad\quad \alpha \leftarrow \theta - \psi$
7 $\quad\quad$ **se** $\beta < 0$ **então**
8 $\quad\quad\quad \gamma \leftarrow -\pi/2 - \alpha + \beta$
9 $\quad\quad$ **senão**
10 $\quad\quad\quad \gamma \leftarrow +\pi/2 - \alpha + \beta$
11 $\quad\quad$ **fim**
12 $\quad\quad d \leftarrow ||(X_v - X)||$
13 $\quad\quad X_v \leftarrow X + d\begin{bmatrix}\cos(\theta-\gamma) & \sin(\theta-\gamma)\end{bmatrix}^T$
14 $\quad$ **senão**
15 $\quad\quad \gamma \leftarrow \gamma(1 - f_e)$
16 $\quad$ **fim**
17 **fim**
18 **retorna** X_v

O Desvio Tangencial, apesar de assumir que sejam conhecidos a menor distância e ângulo em relação ao obstáculo, pode ser utilizado em situações onde se conhece apenas algumas medidas e ângulos referentes, como ilustra a Figura 2.4. Neste caso, o algoritmo não promove o desvio em uma trajetória tangente ao obstáculo, podendo levar o robô em uma trajetória que se afasta do obstáculo ou se aproxima levemente do mesmo. Essa situação restringe a evasão para obstáculos muito longos (horizontal ou verticalmente), ainda que do ponto de vista prático sejam incomuns de existir em um ambiente fechado.

Outras limitações do algoritmo de Desvio Tangencial são situações com obstáculos em formato de L, V e U, onde apenas uma rotação do alvo real não é suficiente para efetuar a evasão. Nesses casos pode ser necessária uma maior rotação do alvo temporário e, ainda assim, existe o risco de colisão. Tais situações e técnicas para evitar colisões podem ser encontradas em [Brandão et al., 2013].

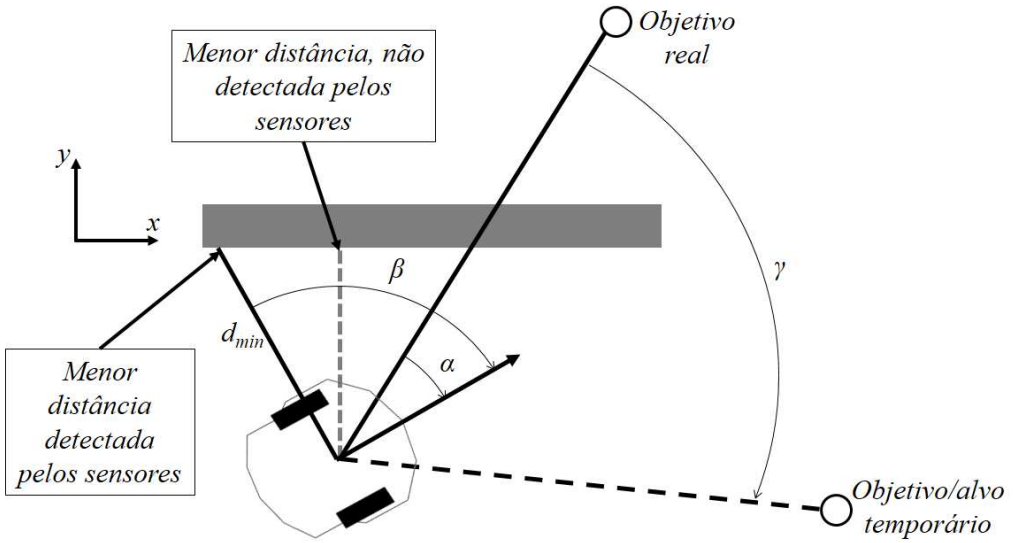

Figura 2.4. Desvio Tangencial sem a informação da menor distância real do robô para o obstáculo.

Fonte: (Própria, 2018)

2.2 Planejamento global e o algoritmo A*

Para traçar caminhos completos é necessário alguma modelagem do ambiente por onde o robô se move, ainda que não se tenha informações completas *a priori*. Neste caso, pode-se assumir algumas premissas para completar o planejamento (como, por exemplo, tratar áreas desconhecidas como zona livre para navegação). Entretanto, ao projetar um algoritmo de planejamento global é necessário que o mesmo consiga lidar com novas informações e mudanças no ambiente, tais como novas áreas livres para navegação e obstáculos sobre o caminho traçado, especialmente durante a navegação.

Replanejar um caminho completo, sobretudo para lidar com o desvio de obstáculos em tempo real, é o grande desafio no desenvolvimento de uma solução de planejamento global. De um ponto de vista prático, espera-se que o algoritmo forneça uma navegação estável e progressiva em direção ao local de destino, de modo a evitar que o robô, caso ocorra uma situação imprevista, fique parado processando buscas por novos caminhos.

Alguns autores abordam o problema com técnicas baseadas em amostragem. A ideia geral é espalhar pontos aleatórios sobre o mapa e conectá-los, até que se tenha um caminho completo até o objetivo. Segundo Elbanhawi & Simic [2014], os dois algoritmos mais famosos utilizados na literatura são o *Rapidly-Exploring*

Random Trees (RRT) [LaValle, 1998] e *Proabilistic Roadmaps* (PRM) [Amato & Wu, 1996]. Os métodos, apesar de intuitivos e de fácil implementação, demoram a convergir para caminhos ótimos, o que continua sendo tópico de pesquisa para ambas as soluções [Elbanhawi & Simic, 2014] [Karaman & Frazzoli, 2013].

Algumas heurísticas também são escopo de pesquisa em planejamento global, especialmente para ambientes com alto grau de incerteza [Mac et al., 2016]. Tal abordagem é o caso de Colônia de formigas (ACO) [Garcia et al., 2009], Otimização por enxame de partículas (PSO) [Zhang et al., 2013] e Algoritmos genéticos (GA) [Hu & Yang, 2004]. Apesar dos bons resultados, heurísticas possuem alto custo computacional para convergir em uma solução ótima em tempo aceitável.

Uma família de algoritmos determinísticos, que podem ser descritos como algoritmos de inteligência artificial clássica, reduz o problema de planejamento à uma busca em grafo [Ferguson et al., 2005], em contraste com as abordagens heurísticas. O ambiente é representado como um grafo $G = (V, E)$, onde V é o conjunto de vértices, ou nós, que representa possíveis locais no ambiente (coordenadas (x, y) em um ambiente bidimensional, por exemplo). E é o conjunto de arestas, onde o peso de cada aresta pode representar o custo (ou distância) de se mover de um local até outro. A Figura 2.5 ilustra um ambiente representado como grafo.

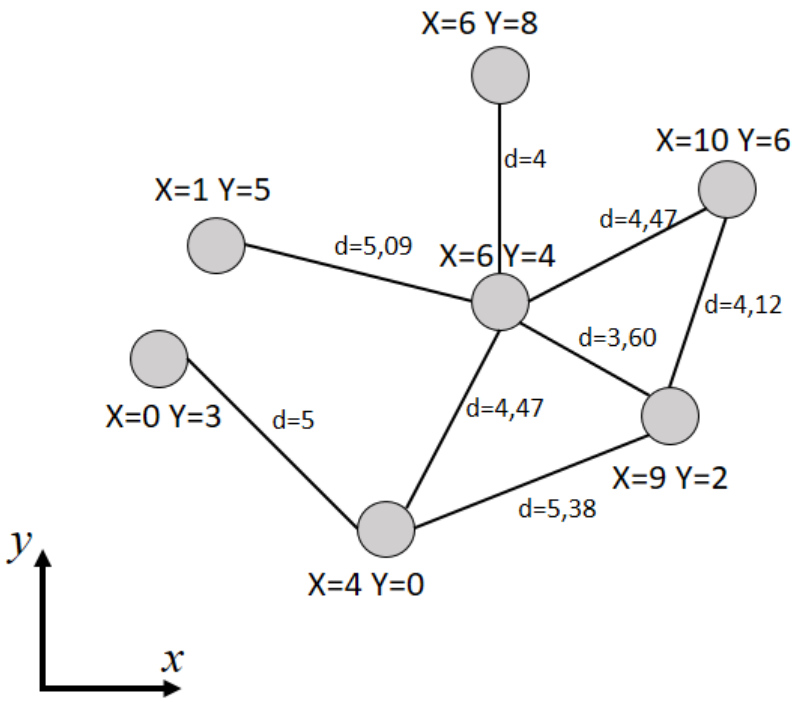

Figura 2.5. Ambiente representado em um grafo, com pares ordenados (x, y).

Fonte: (Própria, 2018)

Um fator importante, que ajuda a explicar a popularidade das buscas em grafo, é o fato de *grids* (Figura 2.6), uma das formas mais comuns de representação do ambiente, ser implicitamente um grafo. Cada célula do *grid* equivale a um vértice e existem arestas conectando células adjacentes, na vertical, horizontal e diagonal. O peso, ou custo, de se mover de uma célula até outra normalmente é definido como d para células adjacentes na horizontal e vertical e $d\sqrt{2}$ para células adjacentes na diagonal.

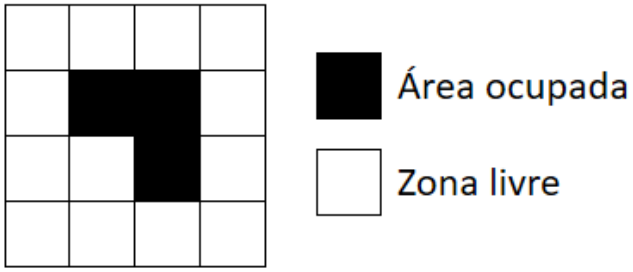

Figura 2.6. *Occupancy Grid* simples.

Fonte: (Própria, 2018)

Os pioneiros nessa família segundo Souissi et al. [2013], os algoritmos A* [Hart et al., 1968] e Algoritmo de Dijkstra [Dijkstra, 1959] obtêm êxito em encontrar o caminho ótimo (de menor custo) entre dois vértices de um grafo. A diferença básica entre os algoritmos A* e de Dijkstra é que o primeiro utiliza uma função heurística para guiar a busca e reduzir o tempo de processamento. Devido à grande importância de A* no contexto do problema de planejamento e para essa dissertação, a solução será discutida mais detalhadamente a seguir.

Considerando um grafo $G = (V, E)$ e $V = n_1, n_2, n_3, ..., n_i$ e E o conjunto de arestas de G, uma aresta e_{ij} representa uma conexão de n_i até n_j. Pode-se atribuir um custo, também chamado de peso, para cada aresta do grafo. O custo c_i de e_{ij} pode representar a distância em linha reta de se mover de um local n_i até n_j, por exemplo.

Um caminho entre um nó de partida n_s até um nó de destino n_g é um conjunto de nós ordenados (n_s, n_i, n_{i+1}, ... , n_g), um subgrafo de G, onde n_{i+1} é o sucessor de n_i e existe uma aresta conectando n_{i+1} a n_i [Hart et al., 1968]. O Custo do caminho é definido com a soma dos pesos de todas as arestas desse subgrafo. Um caminho ótimo entre n_s e n_g é o subgrafo cujo custo total é o menor possível para o grafo G.

O Algoritmo proposto por Hart et al. [1968] utiliza uma função heurística para guiar a busca por caminho ótimo em um grafo, com o objetivo de se expandir o menor número de nós durante a busca. Duas listas são criadas, a lista Aberta (LA) e a Fechada (LF). A primeira contém os nós expandidos à serem avaliados, enquanto a segunda armazena os nós que já foram avaliados. Um nó $n_i \in LA$ é escolhido para se avaliar, adicionando-se os nós sucessores (conectados diretamente por uma aresta) à LA. A escolha de n_i é realizada pegando o nó que possui o menor valor da função f, definida como

$$f(n) = g(n) + h(n), \tag{2.4}$$

onde $g(n)$ é o custo do caminho atual até n_i e $h(n)$ é a função heurística, que estima o custo de n_i até o nó de destino. O algoritmo A* é ilustrado pela Figura 2.7 e seu pseudocódigo é dado pelo Algoritmo 2.

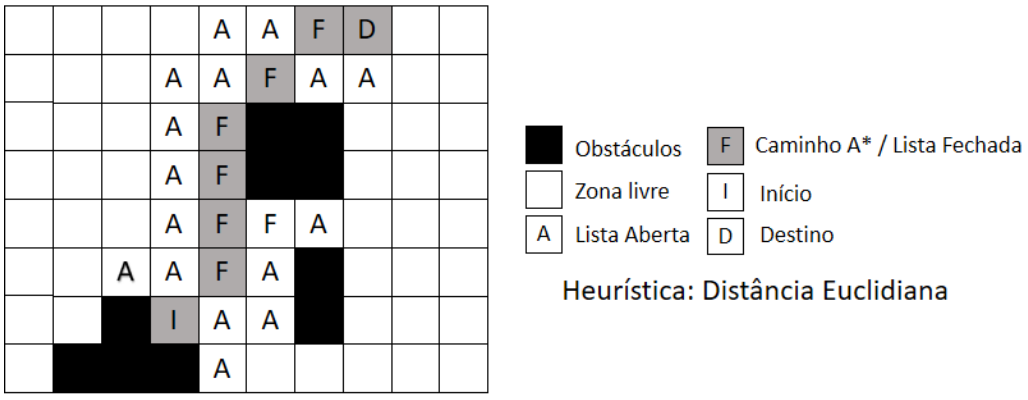

Figura 2.7. Algoritmo A*.

Fonte: (Própria, 2018)

Ao se expandir um nó n para obter seu sucessores, algumas condições são avaliadas. Se algum sucessor de n_i que já tenha sido avaliado e esteja na lista Fechada possuir o valor de $g(n_i)$ a partir de n menor que o previamente calculado este volta a ser adicionado à lista Aberta, desta vez tendo como antecessor o nó n. Isso acontece porque o caminho através de n é melhor que o caminho pelo nó que expandiu n_i previamente.

Caso n_i esteja na lista aberta, o valor $g(n_i)$ é atualizado, caso seja menor que o calculado anteriormente e, novamente, define-se n como antecessor de n_i. Se n_i

Algoritmo 2: BUSCA A*

Entrada: Grafo G
Saída: Caminho Ca

início
  $S \leftarrow inicio$
  Marque S aberto e calcule $f(s)$
  Selecione o nó aberto n com menor valor de $f(n)$, resolvendo empates de forma arbitrária. Se não existir nós marcados abertos, não existe caminho e a busca termina
  se $n == destino$ **então**
    Selecione o nó antecessor de n, do passo onde n foi como marcado aberto.
    continue recursivamente até chegar à S.
    retorne o caminho percorrido como sendo o caminho A*. termine.
  senão
    Marque n fechado
    para cada *sucessor* n_s *de* n **hacer**
      calcule $g(n_s)$
      se n_s *está marcado fechado e* $g(n_s)$ *é menor que o calculado previamente* **então**
        Desmarque n_s como fechado
      fim
      se n_s *está marcado aberto e* $g(n_s)$ *é menor que o calculado previamente* **então**
        Atualize o valor de $g(n_s)$ e $f(n_s)$
        Defina o antecessor de n_s como n
      fim
      se n_s *não está marcado aberto nem fechado* **então**
        Marque n_s como aberto
        Defina o custo de de n_s como $g(n_s)$
        Calcule $f(n_s)$
        Defina o antecessor de n_s como n
      fim
    fin
    vá para **4**
  fim
fim

não está na Lista Aberta nem na Lista Fechada, ele é adicionado na Lista Aberta, adicionando o custo $g(n_i)$, o valor de $h(n_i)$ e n como antecessor.

Quando n_i é igual ao nó de destino, a busca termina e foi encontrado o caminho. Para retornar à sequência de nós que criam tal caminho selecionam-se recursivamente os nós antecessores, que foram utilizados na expansão do nó de destino até o nó inicial.

Como mencionado anteriormente, A* é o pioneiro dentro de uma família de algoritmos de planejamento de caminho a partir de busca em grafo. Ainda que a heurística acelere sua execução, A* se mostra ineficiente do ponto de vista prático onde ambientes não são estáticos, pois deve-se reexecutar a busca completa toda vez que for necessário se adaptar às mudanças no ambiente (surgimento de um obstáculo não conhecido ou uma zona livre onde previamente havia um obstáculo). Essa característica torna o planejamento lento e impraticável para a maioria das aplicações.

Para superar as limitações de A*, várias soluções na literatura, baseadas em busca em grafo, procuram fornecer ferramentas de replanejamento em tempo real para encontrar novos caminhos completos e de forma eficiente. É o caso de D* (*Dynamic A* search*) [Stentz, 1994], Focused D* [Stentz et al., 1995], LPA* (*Lifelong Planning A**) [Koenig et al., 2004] e D* Lite [Koenig & Likhachev, 2002], sendo esse último considerado o estado da arte para busca em grafo.

2.3 Planejamento híbrido

Como discutido neste capítulo, algoritmos de planejamento local são eficientes na tarefa de desvio de obstáculo e possuem baixo custo computacional, mas a tomada de decisão na evasão do obstáculo pode levar a caminhos não eficientes durante a navegação em direção ao ponto de destino. Por outro lado, algoritmos de planejamento global embasados no conhecimento *a priori* do ambiente, encontram caminhos eficientes e livres de colisão, mas demandam mais poder de processamento, sobretudo para lidar com situações em tempo real.

Visando obter as vantagens de cada tipo de planejamento e ao mesmo tempo reduzir ou eliminar as desvantagens, autores buscam soluções híbridas para o planejamento, como é o caso desse trabalho. Exemplos de soluções que também integram planejamento local e global em um único algoritmo podem ser encontradas em [Wang et al., 2002] e [Zhang et al., 2012].

O Desvio Tangencial é utilizado na camada reativa, ou local, do planejamento

proposto nesta dissertação. A escolha do algoritmo se dá por ser uma estratégia simples, de baixo custo computacional e muito eficiente na tarefa de desvio de obstáculos, onde a sua execução depende apenas das informações dos sensores em tempo real, sem a exigência de qualquer informação prévia sobre o ambiente. O tempo de processamento do cálculo do alvo virtual, em qualquer situação, se resume a uma operação matemática, sendo essa uma solução da ordem de complexidade constante O(1).

Para compor a camada deliberativa, o algoritmo A* é utilizado como planejamento global de caminho, sempre antes do robô iniciar um movimento. A Busca é realizada para encontrar caminhos livres de obstáculos dentro da área conhecida do ambiente. Para lidar com novas informações em tempo real, tendo em vista que o trabalho procura contemplar plataformas de baixo custo, o Desvio Tangencial é utilizado como ferramenta de replanejamento da navegação deliberativa. A solução híbrida proposta, envolvendo os dois algoritmos, é discutida no próximo capítulo.

Capítulo 3

NAVEGAÇÃO HÍBRIDA

Como dito anteriormente, os algoritmos A* e Desvio Tangencial compõem a camada deliberativa e reativa, respectivamente, da solução híbrida proposta nesta dissertação. Além disso, assume-se que o algoritmo de mapeamento *Occupancy Grid* é utilizado durante a navegação, para modelagem do ambiente que é a base do planejamento através da busca em grafo. Para realizar o planejamento de caminhos, supõe-se que é fornecido ao robô um conjunto de pontos (coordenadas (x, y) no plano) que devem ser alcançados para se completar a navegação. Nesse contexto, o planejamento global é sempre utilizado antes do robô iniciar um movimento, procurando um caminho completo até cada ponto objetivo. Para o desvio de obstáculos em tempo real utiliza-se o planejamento local, tendo um caminho sido obtido pelo planejamento global ou não.

Por exemplo, supondo como entrada do algoritmo os pontos $([(x_1, y_1), (x_2, y_2), (x_3, y_3)])$ e que o robô se encontra inicialmente em (x_r, y_r), o algoritmo A* é executado, antes do robô começar a se mover, para procurar um caminho completo de (x_r, y_r) até (x_1, y_1). Se não foi possível obter tal caminho, o robô navega reativamente, movendo-se em direção ao objetivo e desviando de obstáculos em tempo real com o Desvio Tangencial. Caso contrário, o robô segue o caminho obtido, promovendo o desvio de obstáculos conhecidos. Para lidar com obstáculos desconhecidos que estejam sobre o caminho planejado e que possam ocasionar colisões durante o seguimento do caminho, utiliza-se novamente o Desvio Tangencial. Ao chegar em (x_1, y_1), o planejamento global é novamente utilizado para procurar um caminho até (x_2, y_2) e o processo se repete até que se chegue no destino, (x_3, y_3).

As Figuras 3.1, 3.2, 3.3 e 3.4 ilustram a navegação com o planejamento híbrido proposto. O algoritmo A* sempre é executado antes do robô iniciar um movimento.

Como não se conta com informações *a priori* e considera-se áreas desconhecidas como obstáculos, a busca pode não encontrar um caminho completo em algumas execuções. Quando isso acontece, o robô adota um comportamento totalmente reativo (Figura 3.1) desviando de obstáculos em tempo real.

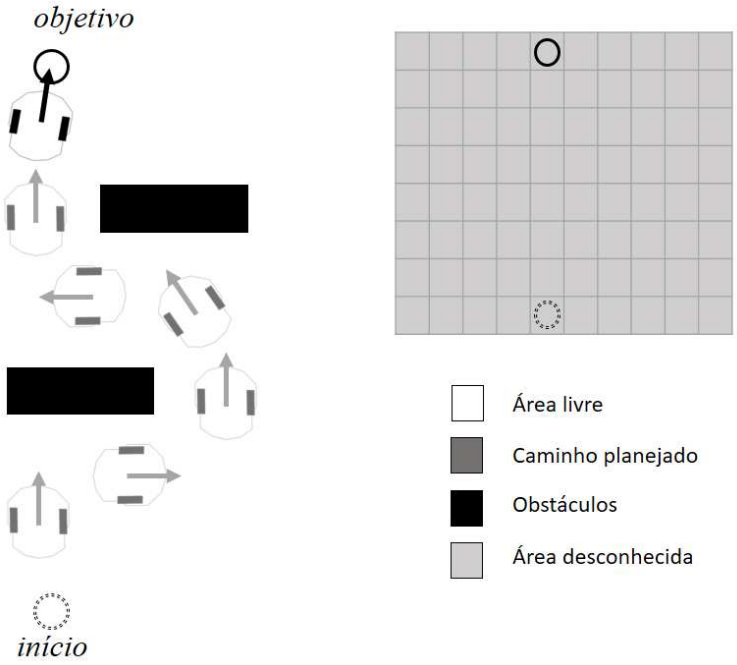

Figura 3.1. Navegação reativa.

Fonte: (Própria, 2018)

Uma vez obtido um caminho, espera-se que o robô navegue ao longo do mesmo e evite colisões com os obstáculos conhecidos (Figura 3.2). Entretanto, se algum obstáculo, não conhecido, estiver próximo ou sobre o caminho traçado pelo planejamento global, o robô utiliza novamente o Desvio Tangencial até que o obstáculo tenha sido transposto (Figura 3.3). Em seguida, retorna para o caminho planejado até completar o trajeto. Por último, de posse das novas informações sobre o ambiente, o robô navega por um novo caminho que evita colisões com obstáculos do mapa atualizado (Figura 3.4).

De um ponto de vista prático no planejamento de caminhos, deve-se adicionar uma zona de segurança em volta dos obstáculos para acomodar as dimensões físicas do robô (Figura 3.5). Tal margem de segurança é utilizada no algoritmo híbrido proposto como referência para definir o limite da distância observável d_{obs} do Desvio Tangencial (Figura 3.6). d_{obs} deve ter valor máximo igual à zona de segurança, pois caso seja superior o robô tenderá a efetuar desvios em situações onde seguir o caminho traçado ja é suficiente para evitar colisões com obstáculos conhecidos.

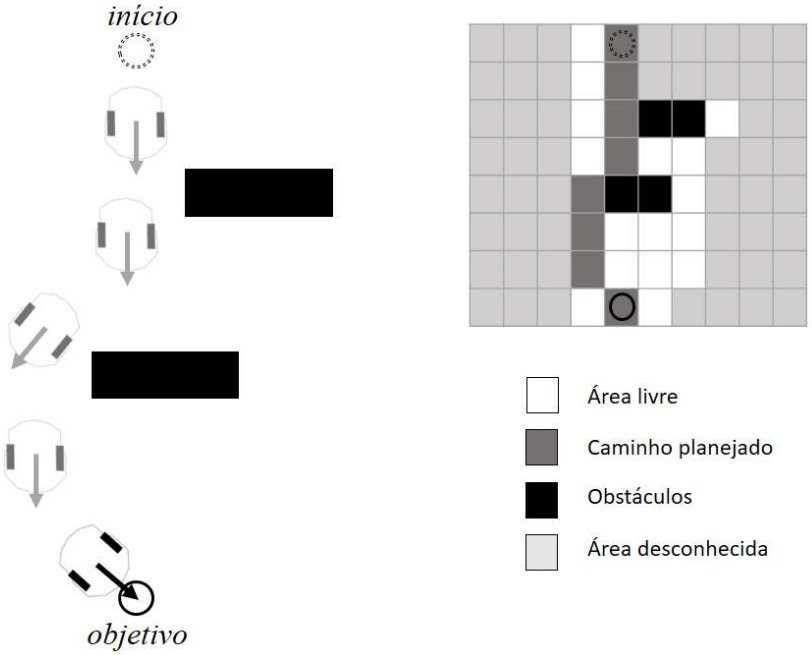

Figura 3.2. Navegação deliberativa.

Fonte: (Própria, 2018)

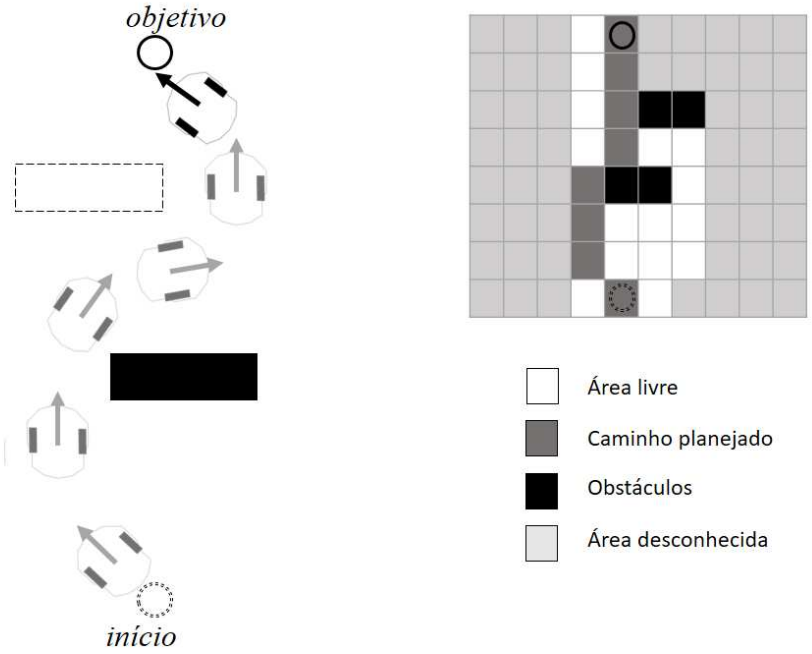

Figura 3.3. Navegação híbrida.

Fonte: (Própria, 2018)

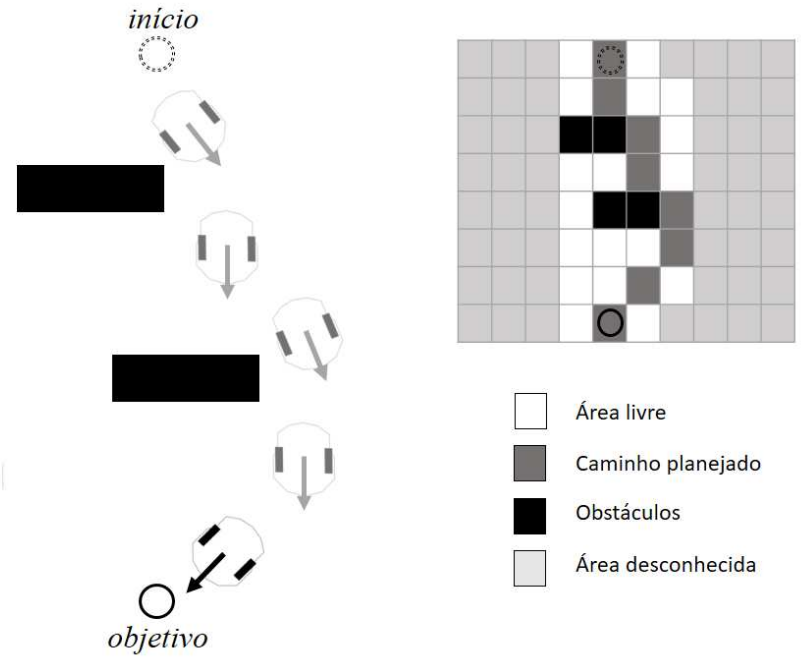

Figura 3.4. Navegação com mapa atualizado.

Fonte: (Própria, 2018)

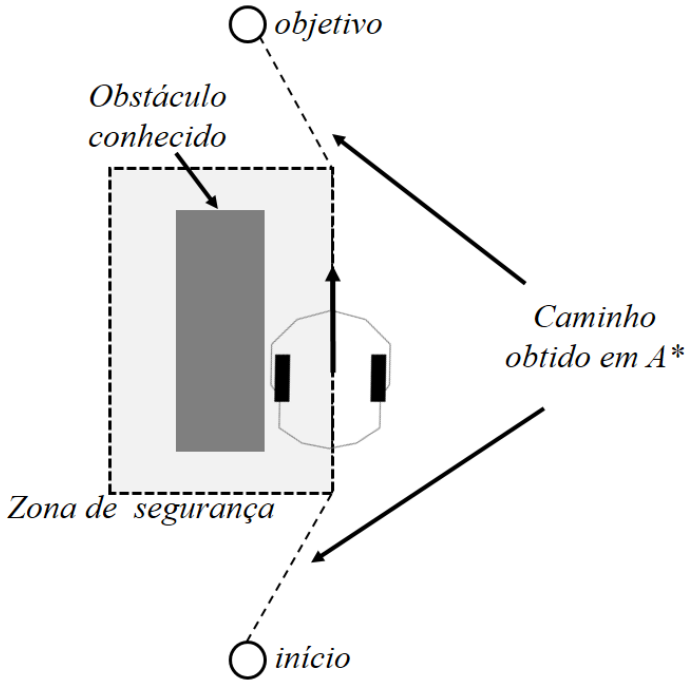

Figura 3.5. Zona de segurança mínima para acomodar as dimensões do robô.

Fonte: (Própria, 2018)

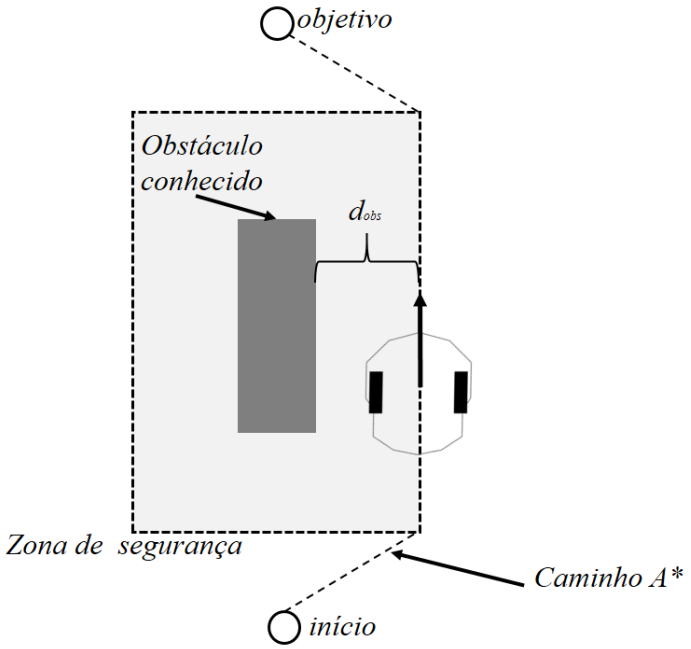

Figura 3.6. Limite para d_{obs} de forma à evitar desvios desnecessários.

Fonte: (Própria, 2018)

Quando o robô não segue um caminho planejado previamente, não existe um caminho de referência com obstáculos conhecidos. Logo, pode-se definir d_{obs} de forma arbitrária, dependendo da aplicação e capacidade de sensoriamento, para se efetuar o desvio de forma segura.

Enquanto segue um caminho, o robô pode efetuar o Desvio Tangencial se algum obstáculo estiver dentro do limite d_{obs}, como mencionado anteriormente. Entretanto deve-se definir a ação a ser tomada após o desvio, de forma à retornar ao caminho planejado. No algoritmo proposto, tal ação consiste em selecionar o ponto, sobre o caminho, que esteja mais próximo do robô e que ainda não tenha sido alcançado durante navegação, definido como ponto de retorno (Figura 3.7). Essa escolha tem como objetivo dispensar pontos próximos ao obstáculo, evitando um futuro desvio desnecessário. A navegação através do planejamento híbrido é apresentada pelo Algoritmo 3, em forma de pseudocódigo.

O Algoritmo 3 recebe como entrada o mapa, a pose atual e a desejada (coordenadas x e y no plano bidimensional e orientação). Define-se os valores para γ, f_e, d_{obsA} e d_{obsT} (linhas 2-5). Inicialmente γ possui valor nulo, pois ainda não foi realizado nenhum Desvio Tangencial. ($f_e \in [0, 1]$) é o fator de esquecimento, utilizado para a suavização na rotação do alvo temporário. Quanto menor f_e, mais suave é a rotação.

Algoritmo 3: Navegação híbrida.

```
   Entrada: Occupancy Grid OG, Pose atual X, Pose desejada X_d
 1 início
 2     γ ← 0
 3     fe ← FatorDeEsquecimento
 4     d_obsA ← DistanciaObservavelDuranteSeguimento
 5     d_obsT ← DistanciaObservavelParaDesvioTangencial
 6     rota ← BuscaAestrela(OG, X, X_d)
 7     se EncontrouRota então
 8         caminho ← GeraCaminho(rota)
 9         d_obs ← d_obsA
10     senão
11         d_obs ← dobsT
12     fim
13     repita
14         X ← OdometriaAtualDoRobo()
15         M ← LerDadosSensores()
16         AtualizaMapa(OG, M, X)
17         se EncontrouRota então
18             se Executando Desvio Tangencial ou
               Existe obstaculo a distncia menor que d_obsA então
19                 d_obs ← d_obsT
20                 Controlador ← ControladorDeErroDePosicao
21                 X_v, γ ← DesvioTangencial(X_d, min(M), d_obs, fe, γ)
22             senão
23                 d_obs ← d_obsA
24                 Controlador ← ControladorParaSeguirCaminho
25                 X_v ← PontoMaisProximo(X, caminho)
26             fim
27         senão
28             Controlador ← ControladorDeErroDePosicao
29             X_v, γ ← DesvioTangencial(X_d, min(M), d_obs, fe, γ)
30         fim
31         U ← Controlador(X, X_v)
32         EnviaSinalDeControle(U)
33     até X = X_d;
34 fim
```

A distância observável d_{obsA} é utilizada para se executar o Desvio Tangencial enquanto se segue um caminho obtido na busca A* (linha 9). d_{obsT} é a distância observável quando se navega reativamente ou quando se está executando o Desvio Tangencial, sem um caminho como referência (linha 11). As linhas 14-16 são os comandos para se obter a odometria (posição e orientação) do robô e informações

dos sensores, que serão utilizados para atualizar o mapa do ambiente.

O algoritmo supõe a utilização do resultado obtido na busca A* para gerar um caminho a ser seguido (linha 8), fato necessário para utilizar um controlador de seguimento de caminho (linha 24). Na pior das hipóteses, pode-se utilizar um controlador *Carrot Chasing*, onde se fornece sequencialmente os pontos, referentes às posições do *grid* obtidas em A*, como objetivos a serem alcançados até chegar ao destino final.

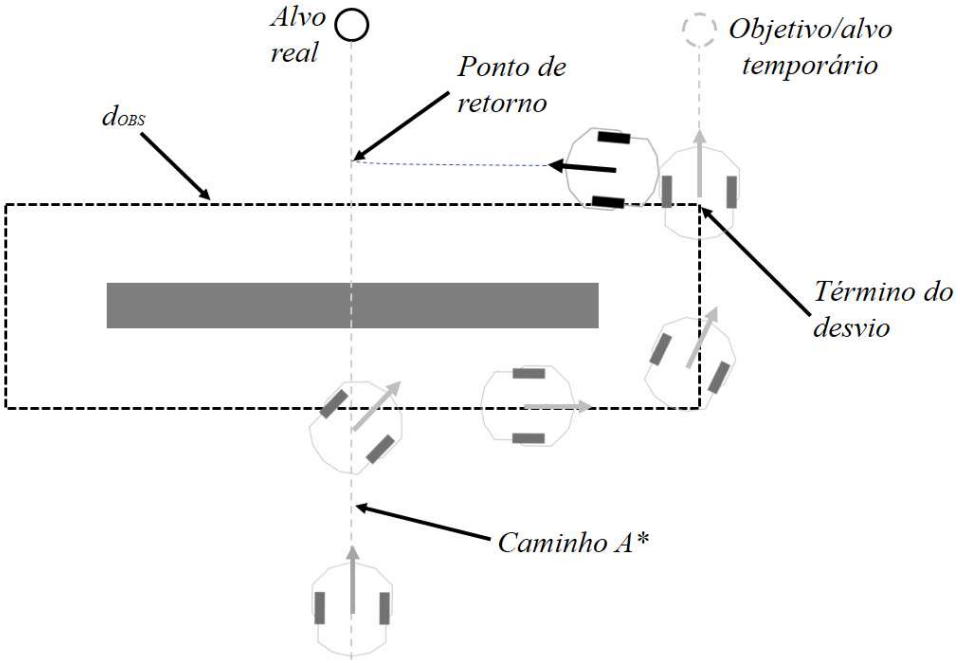

Figura 3.7. Retorno para o caminho A* após o desvio de obstáculos desconhecidos.

Fonte: (Própria, 2018)

Quando não há um caminho para a ser seguido ou se está executando o Desvio Tangencial, assume-se a utilização de um controlador de erro de posição (linhas 20 e 28). A menção ao uso de controladores de movimento é feita em um nível de abstração mais alto nessa etapa, tendo em vista que pode variar de acordo com o tipo de robô e aplicação.

Algumas condições são avaliadas quando se está navegando deliberadamente (linha 17). As duas primeiras (linha 18), se é detectado um obstáculo a uma distância dentro limite observável d_{obsA} e o robô não está executando o Desvio Tangencial ou se o robô está executando o Desvio Tangencial e está dentro do limite d_{obsT}. Nestes casos a distância observável é definida como d_{obsT}, pois é necessário efetuar uma evasão, logo d_{obs} passa a ser o mesmo do Desvio Tangencial (linha 19).

Caso o robô esteja seguindo um caminho e não exista um obstáculo dentro do limite d_{obsA}, mantem-se d_{obs} para A* (linha 23). A linha 25 indica que utiliza-se sempre o ponto mais próximo ao robô, que ainda não foi alcançado, como referência para o seguimento de caminho.

Se não foi possível encontrar um caminho, navega-se reativamente comparando se alguma das medidas coletadas pelos sensores é menor que d_{obs}. Caso essa condição seja satisfeita, X_d é rotacionado e executa-se o Desvio Tangencial (linhas 28-29). Caso contrário, X_d permanece o mesmo. As linhas 31-32 indicam o uso do controlador e o envio dos sinais de controle, para efetuar a navegação de fato.

A não obrigação de se ter um mapa *a priori* e o mapeamento proposto durante a navegação tem como objetivo fortalecer o comportamento autônomo do robô, fazendo com que navegação fique eficiente, em termos de caminhos percorridos, com o passar do tempo. Além disso, o uso restrito do A* na etapa antes de se inciar cada movimento reduz a carga de processamento do planejamento da navegação em tempo real. Assim, o planejamento híbrido se apresenta como uma solução simples e eficiente em termos de custo computacional, com uma navegação contínua sem interrupção para efetuar algum processamento.

A contribuição deste trabalho se dá, principalmente, em dois pontos: a forma como os algoritmos são integrados em uma única solução e a utilização do Desvio Tangencial como ferramenta de replanejamento em tempo real do algoritmo A*. Outra forma de se visualizar o planejamento aqui proposto é como sendo uma solução heurística de baixo custo computacional, tendo em vista que o Desvio Tangencial não garante a otimalidade na evasão de obstáculos.

Capítulo 4

RESULTADOS

Os resultados descritos nesse capítulo foram obtidos a partir de dois ambientes distintos. O primeiro, em um simulador de plataformas móveis e, o segundo, em um ambiente real preparado para à navegação. Em ambos os casos, o algoritmo foi implementado utilizando-se a linguagem interpretada do ambiente de desenvolvimento MATLAB, executado em uma máquina i7-7700HQ CPU @ 2.80GHz com 8GB de memória RAM.

O *Framework* ARIA[1] fornece ferramentas para ajudar no desenvolvimento de aplicações para diversas plataformas móveis, incluindo o Pioneer P3-DX. A biblioteca desenvolvida em C++ facilita a comunicação com o robô, sendo possível fazer o mesmo se locomover enviando comandos em um nível maior de abstração, como velocidade linear e angular. Também é possível extrair informações do P3-DX de forma simples, como distância captada pelos sonares e odometria.

Para fornecer os sinais de controle enviados ao robô, em todos experimentos, foi utilizado o controlador proposto em [Fernandes et al., 2017], utilizando como referência a função tangente hiperbólica e tendo como valores para as constantes $k_1 = 0.2$, $k_2 = 0.4$. Por padrão, a biblioteca ARIA utiliza a frequência de 10Hz (0,1s) para o envio de informações ao Pioneer P3-DX. Assim sendo, foi utilizado um temporizador de forma a enviar um sinal de controle somente 0,1s após o anterior.

A distância euclidiana foi utilizada como heurística para a busca A*, no primeiro experimento em simulador e nos experimentos em ambiente real. Além disso, um experimento, também em ambiente de simulação, foi realizado para comparar a eficiência entre distância Euclidiana e a distância Manhattan. Para efetuar o seguimento de caminho, foi utilizado uma adaptação, para o plano bidimensional, do algoritmo proposto em [Brandão et al., 2011]. A saturação na velocidade linear para

[1]http://robots.mobilerobots.com/wiki/ARIA

o seguimento foi definida em 0,2m/s. Os caminhos obtidos em A* foram transformados em curvas utilizando o método de interpolação *Hermite Piecewise Interpolation*, fornecido de forma nativa pelo MATLAB.

Foram coletados alguns dados da navegação, tais como distância percorrida, velocidade, tempo de navegação e IASC (Integral absoluta do sinal de controle), estimativa de gasto energético que analisa os sinais de controles efetivamente executados pelo robô durante a navegação e é calculada por $\int_0^t ||\mathbf{u}||dt$, onde u é a velocidade desenvolvida pelo robô. Também se exibe através de gráfico o erro de posição do robô, que representa a distância entre a posição do robô e a posição desejada, obtido através da norma do vetor formado entre as posições.

As características específicas da navegação em cada situação, tais como ambiente de desenvolvimento, obstáculos e sensoriamento, serão discutidos a seguir.

4.1 Experimento 1: Simulador MobileSim

O primeiro experimento teve como objetivo validar a estratégia proposta nessa dissertação, promovendo a navegação com o comportamento reativo, deliberativo e híbrido. Foi utilizado o simulador de plataformas robóticas MobileSim[2], tendo como modelo de simulação o Pioneer P3-DX com 8 sonares. O experimento proposto nessa seção tem como base o artigo produzido nas etapas iniciais dessa dissertação, que pode ser encontrado em [de Oliveira et al., 2018]. A diferença principal é a utilização de um algoritmo para seguimento de caminho no experimento aqui apresentado.

A comunicação entre o código implementado e o simulador foi realizada utilizando a biblioteca ARIA. Como a biblioteca é desenvolvida em C++, uma linguagem compilada, enquanto o MATLAB executa scripts de forma interpretada, foi necessário gerar arquivos MEX do ARIA. MEX são programas que permitem a utilização de subrotinas escritas em C, C++ e Fortran dentro do MATLAB como se fossem funções nativas. A ferramenta para gerar essa transformação é fornecida de forma nativa pelo MATLAB.

Considerando que o ambiente de simulação é menos suscetível à ruídos em medidas de sensores e odometria, optou-se por realizar o experimento apenas com a utilização dos sonares padrões do P3-DX, distribuídos conforme a Figura 4.11. Para o mapeamento, foi utilizado o pacote de ferramentas para robótica móvel fornecido pelo MATLAB, o *Robotics System Toolbox*[3], que conta com o algoritmo *Occupancy Grid* já implementado.

[2]http://robots.mobilerobots.com/wiki/MobileSim
[3]https://www.mathworks.com/products/robotics.html

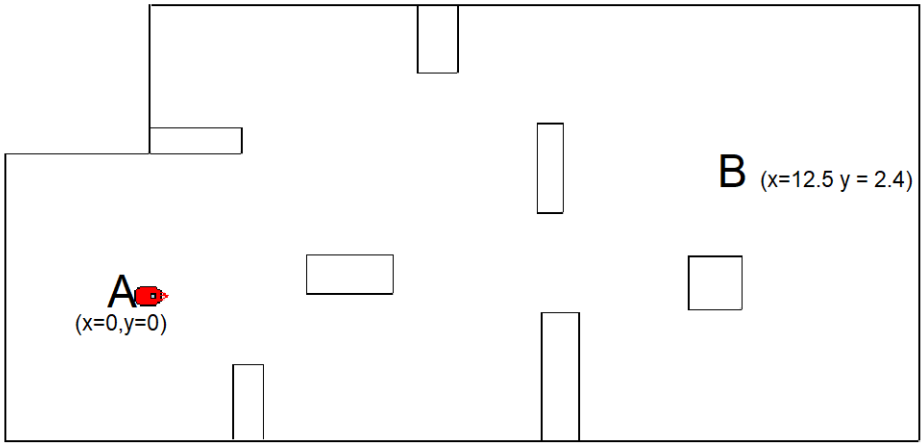

Figura 4.1. Ambiente de simulação

Fonte: (Própria, 2018)

O experimento consistiu em fazer o robô se mover entre os pontos A e B do mapa exibido na Figura 4.1, com as variáveis para à distância observável d_{obsA} = 0,4m e d_{obsT} = 1m. O fator de esquecimento foi definido como f_e = 0,75. O ciclo completo a ser percorrido foi definido como A-B-A-B, considerando o inicio da navegação em A.

Cada percurso será, a partir desse ponto, identificado da seguinte forma: trajeto S1 como primeiro A-B, trajeto S2 como primeiro B-A e trajeto S3 como segundo A-B. Para cada Figura de deslocamento, o ambiente é representado baseado no mapa obtido ao final da navegação (O ambiente representado na Figura 4.2 é aquele que foi obtido pelo mapeamento ao final do trajeto S1, por exemplo). Os Pontos pretos representam áreas desconhecidas e obstáculos, enquanto áreas em branco representam zona livre para navegação.

Inicialmente, antes de S1, como não existe um mapa e considera-se áreas desconhecidas como obstáculos para busca A*, o robô navega reativamente, efetuando o Desvio Tangencial. Nota-se pela trajetória percorrida pelo robô que o Desvio não é executado de forma totalmente tangente ao obstáculo, razão discutida no Capítulo 3. Essa limitação também faz com que o robô perca, em algumas situações, a detecção do obstáculo durante a evasão, levando-o a se dirigir novamente em direção ao destino real até que o obstáculo seja novamente detectado por um dos sensores.

Outro ponto importante é que o desvio do obstáculo **O** indicado na Figura 4.2 leva o robô em um caminho menos eficiente para à evasão. Isso se deve ao caráter reativo do planejamento local, que age baseado nas informações dos sensores disponíveis em cada momento. Em outras simulações, o robô poderia desviar do

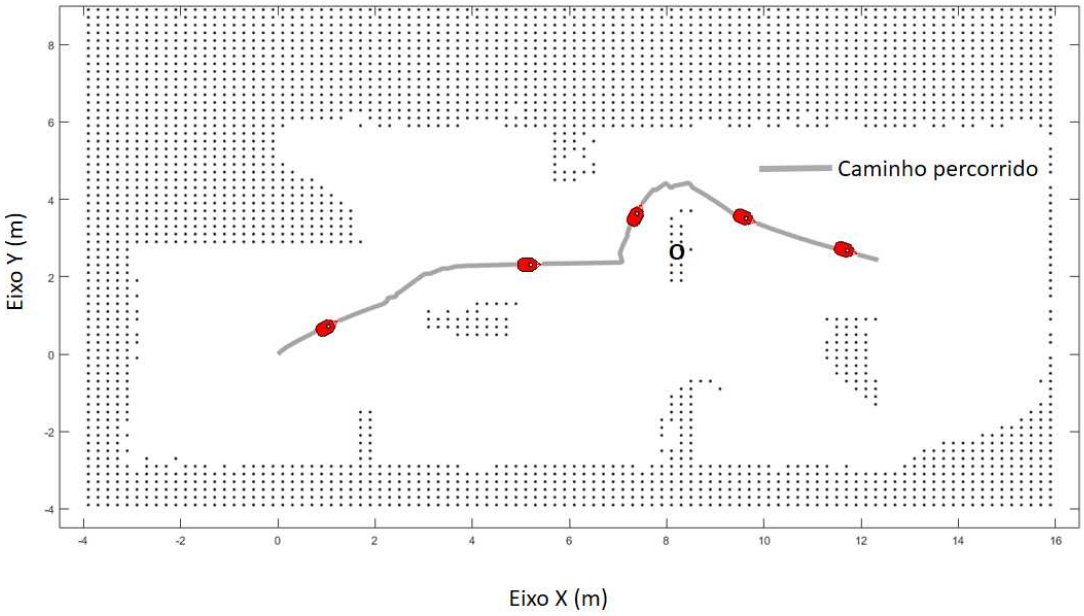

Figura 4.2. Deslocamento no ambiente com o mapa obtido ao final do trajeto S1.

Fonte: (Própria, 2018)

obstáculo e seguir por um caminho pelo centro do mapa.

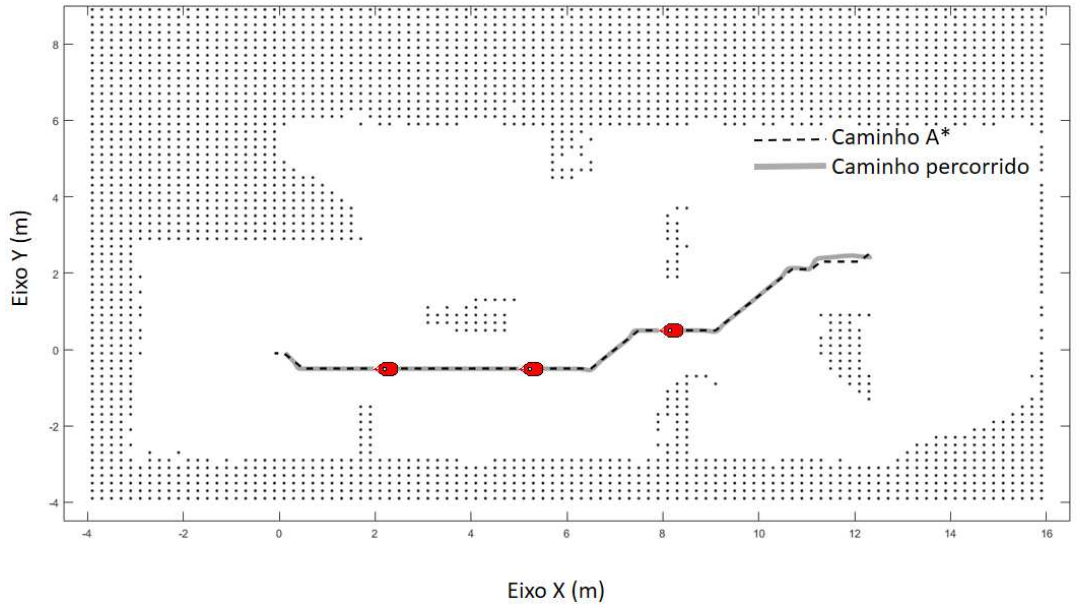

Figura 4.3. Deslocamento no ambiente com o mapa obtido ao final do trajeto S2.

Fonte: (Própria, 2018)

Em S2, o robô possui o mapeamento efetuado na navegação anterior. Com isso, foi possível obter um caminho através de A*. Quatro células, onde cada uma

representa $0,1 \times 0,1m^2$ do ambiente, foram utilizadas como zona de segurança. Pode-se notar que o caminho obtido foi suficiente para à evasão de obstáculos, com a distância observável $d_{obsA} = 0,4m$, igual à margem da zona de segurança, sem a necessidade de realizar o Desvio Tangencial. Além disso, nota-se pela Figura 4.5 que o robô leva menos de 60 segundos para chegar ao seu destino, mais eficiente quando comparado aos quase 90 segundos do trajeto S1.

Em S3, novamente existe um caminho obtido em A*, que desta é diferente do anterior (Figura 4.4). A alteração do caminho encontrado ocorre porque as informações no mapa se atualizam enquanto o robô navega e surgem novas informações a respeito de áreas ocupadas e livres. Esse fato impacta diretamente na execução da busca, ainda mais quando consideramos a zona de segurança utilizada para "aumentar"obstáculos.

Ainda em S3, quando chega à área da Figura 4.4 destacada por um retângulo pontilhado, o robô, enquanto segue o caminho, detecta um obstáculo a uma distância menor que 0,4m. Isso acontece porque o obstáculo não havia sido totalmente mapeado na navegação anterior, e o planejamento foi realizado acreditando-se que aquela região era zona livre (Figura 4.3). Após executar o Desvio Tangencial para evitar o obstáculo detectado, o robô retoma o caminho A*, seguindo até o final do trajeto.

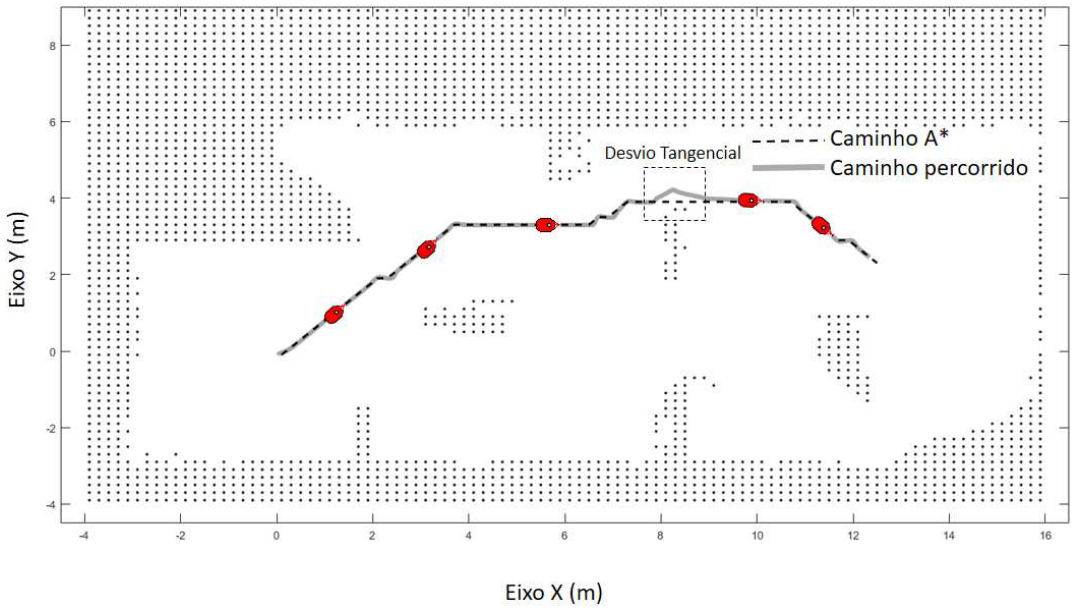

Figura 4.4. Deslocamento no ambiente com o mapa obtido ao final do trajeto S3.

Fonte: (Própria, 2018)

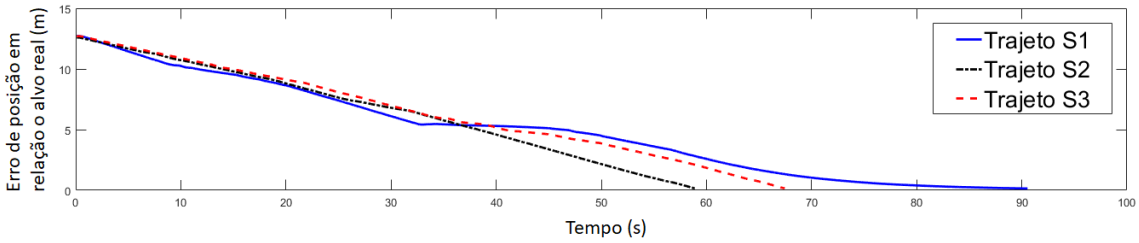

Figura 4.5. Erro de posição obtido na simulação.

Fonte: (Própria, 2018)

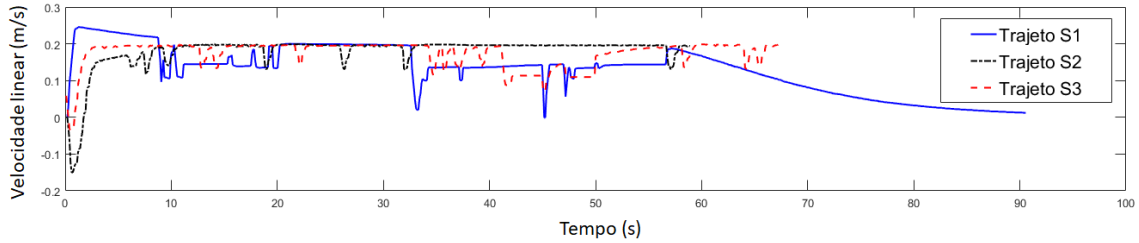

Figura 4.6. Velocidade linear obtida na simulação.

Fonte: (Própria, 2018)

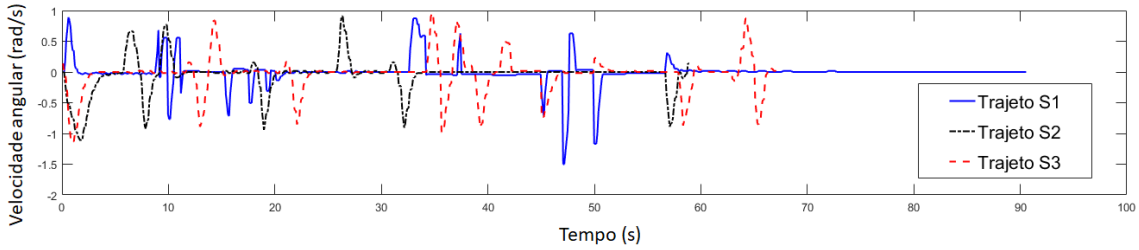

Figura 4.7. Velocidade angular obtida na simulação.

Fonte: (Própria, 2018)

Tabela 4.1. Resultados da simulação

Dados/Percurso	traj. S1	traj. S2	traj. S3
Distância percorrida (m)	15,22	13,82	14,99
Vel. linear média (m/s)	0,13 ± 0,06	0,18 ± 0,05	0,17 ± 0,04
IASC	0,15	0,15	0,17
Tempo de navegação (s)	90,55	59,18	67,88

Fonte: (Própria, 2018)

Os gráficos do erro de deslocamento indicam que os caminhos obtidos em A* são mais eficientes, em termos de distância percorrida, que aquele tomado na navegação reativa. A saturação da velocidade é a mesma quando se navega reativamente e deliberativamente, mas o robô alcança o destino em tempos menores quando navega

seguindo um caminho. A Tabela 4.1 resume os dados navegação realizada.

O video referente a este experimento pode ser encontrado em `https://www.youtube.com/watch?v=AihmDUMxU9Q`.

Para comparar a eficiência das funções Distância Euclidiana e Distância Manhattan como heurísticas no algoritmo A* e avaliar o planejamento após um tempo significativo de navegação, um segundo experimento foi realizado, onde o robô deveria realizar o percurso A-B-A da Figura 4.1, começando em A, 21 vezes em um total de 42 movimentos. Foi utilizado como resolução do *grid* para o mapeamento células que representavam $0,05 \times 0,05m^2$ do ambiente, o dobro da resolução do primeiro experimento, e quatro células como zona de segurança para execução da busca. Os dados de navegação em cada situação são exibidos na Tabela 4.2.

Tabela 4.2. Resultados comparativos entre as heurísticas

Dados/Heurística	Dist. Euclidiana	Dist. Manhattan
Distância total percorrida (km)	0,62	0,62
Distância média em cada movimento (m)	14,88	14,83
Distância mínima em um movimento (m)	13,37	13,33
IASC	9,76	9,14

Fonte: (Própria, 2018)

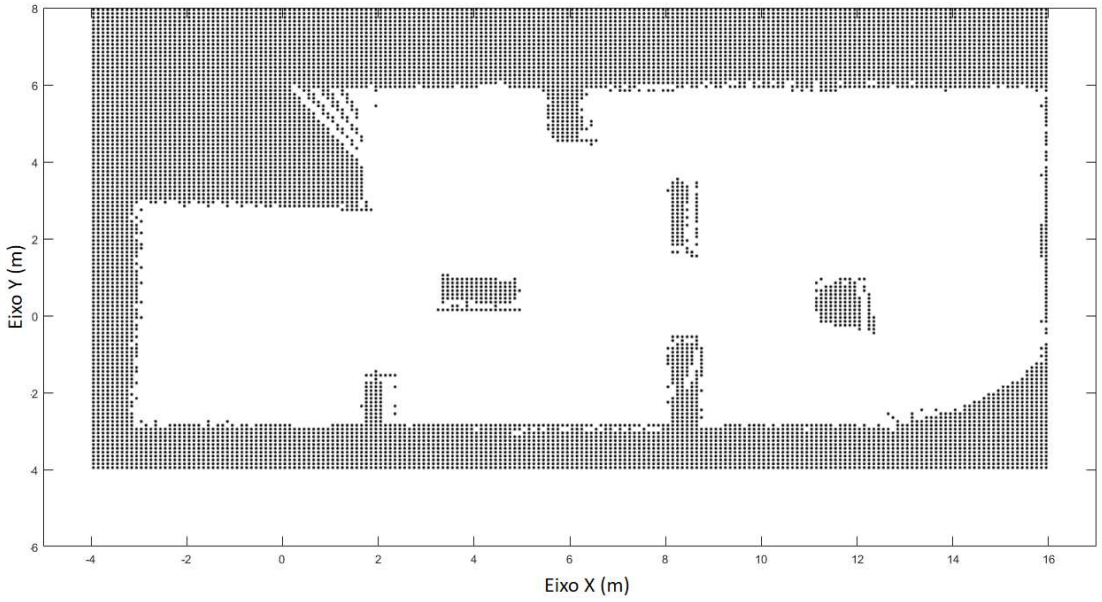

Figura 4.8. Mapa obtido ao final da navegação com a heurística Euclidiana.

Fonte: (Própria, 2018)

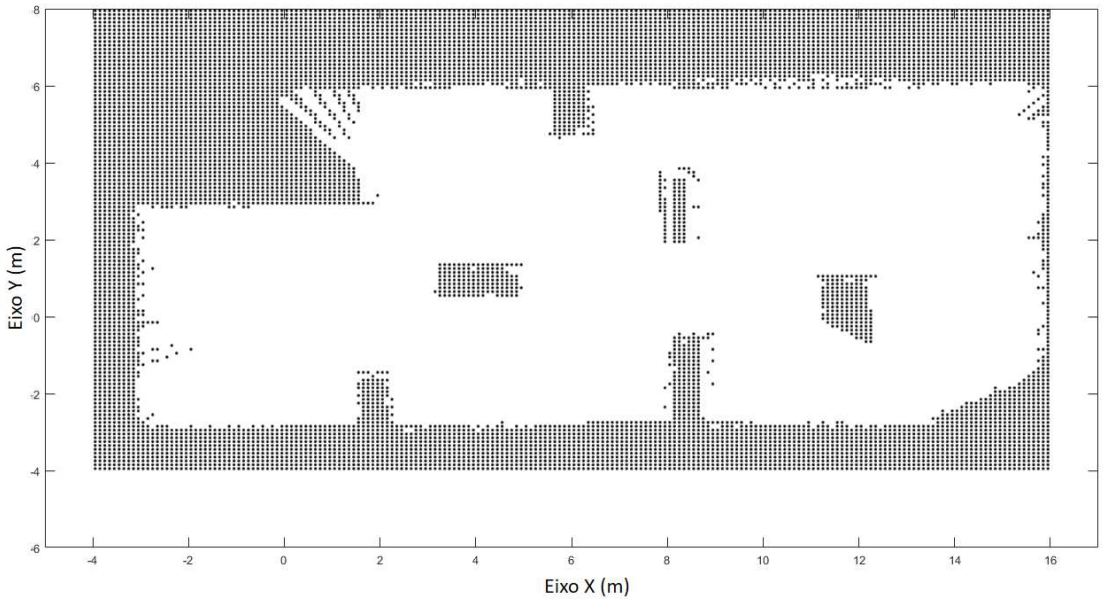

Figura 4.9. Mapa obtido ao final da navegação com a heurística Manhattan.

Fonte: (Própria, 2018)

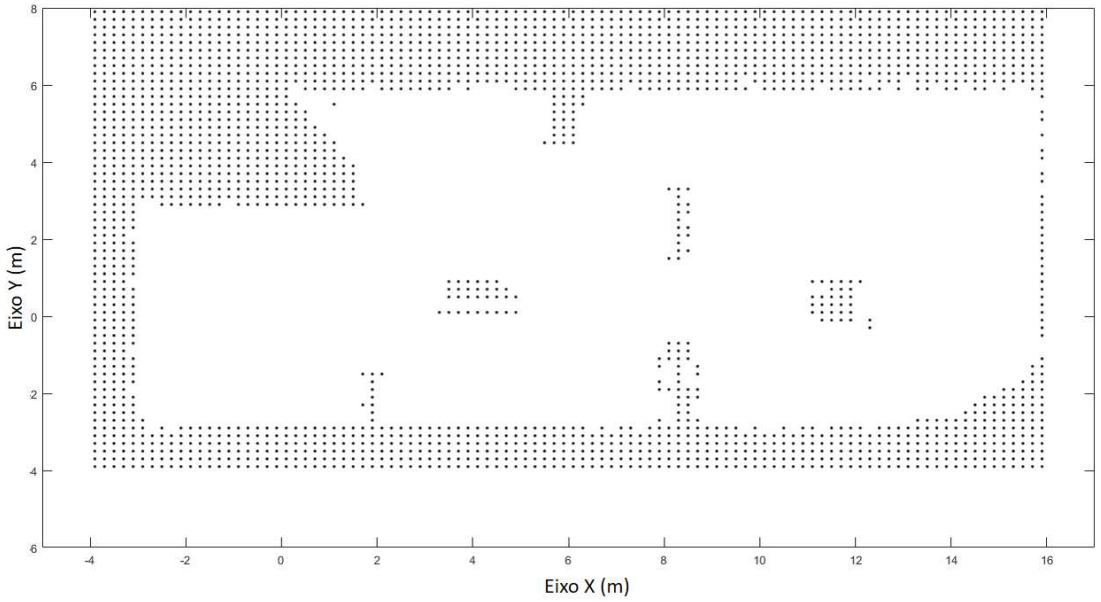

Figura 4.10. Mapa de menor resolução obtido ao final da navegação com a heurística Euclidiana.

Fonte: (Própria, 2018)

Os mapas obtidos ao final da navegação do experimento de comparação das funções heurísticas são ilustrados pelas Figuras 4.8 e 4.9, exibindo o impacto de

cada uma nos caminhos obtidos e consequentemente na modelagem do mapa. Para avaliar o efeito da resolução do *grid* no mapeamento, um segundo mapa foi obtido de forma paralela, sem a sua utilização de fato no planejamento global, durante a navegação referente a Figura 4.8. Utilizou-se uma resolução de $0,1 \times 0,1m^2$ (metade da anterior) para cada célula do *grid*, representando cada célula uma maior porção do ambiente. O mapa obtido ao final da navegação é exibido na Figura 4.10.

4.2 Experimento 2: Ambiente real

Após a validação do algoritmo em experimentos realizados no simulador, o mesmo foi utilizado no experimento em ambiente real, com a versão física do P3-DX. Os testes iniciais mostraram que, ao contrário do ambiente de simulação, os ruídos em odometria e imprecisão dos sensores tornavam a navegação inviável, produzindo mapas tortos e sem conexão com a realidade.

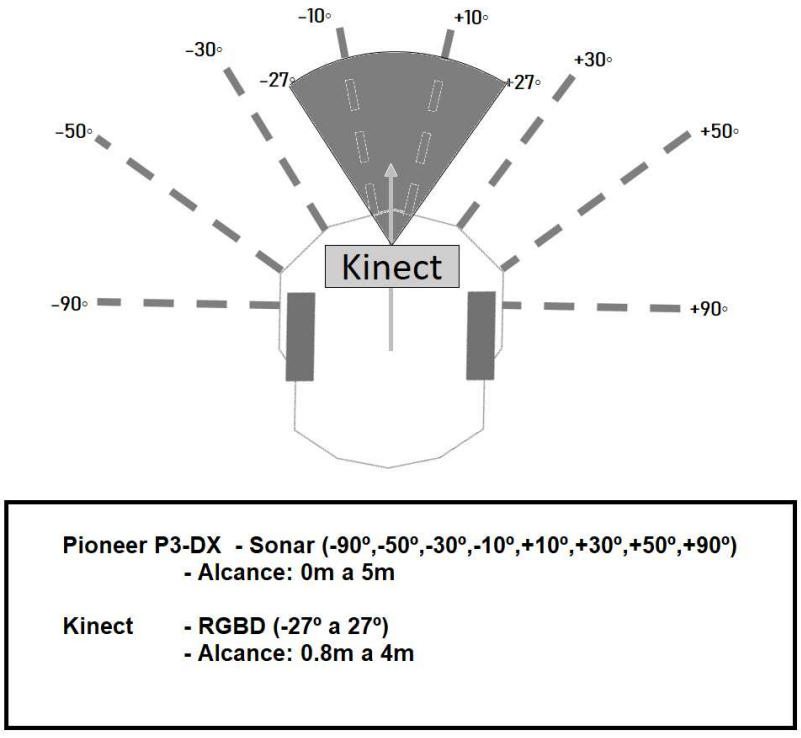

Figura 4.11. Sensoriamento combinado do P3-DX e Kinect.

Fonte: (Própria, 2018)

Para superar o problema com os sensores, adicionou-se o Kinect ao sistema de sensoriamento. O sensor foi colocado sob a plataforma do P3-DX e sua utilização

aumentou o poder de sensoriamento frontal do robô, conforme ilustra a Figura 4.11. Com a adicção do novo sensor e a imprecisão do sonar, a tarefa de mapeamento passou a ser exclusiva do Kinect. Já para o desvio de obstáculos, tendo em vista que o Kinect só capta uma área frontal limitada, os dois sensores foram combinados, aumentando o poder de detecção do robô.

Para esse experimento utilizou-se o conjunto de ferramentas e bibliotecas ROS (*Robot Operating System*) [4], versão kinetic. O algoritmo foi novamente implementado no ambiente MATLAB, mas a comunicação com o robô foi realizada através do pacote RosAria, um pacote baseado na biblioteca ARIA para o ROS. Mais informações sobre o ROS e a configuração utilizada para o experimento podem ser encontradas no anexo ao final dessa dissertação.

A execução do algoritmo de *Occupancy Grid* foi realizada por um outro pacote do ROS, o rtabmap. A ferramenta, que possui compatibilidade com o Kinect, realiza o mapeamento do ambiente e fornece correções na odometria fornecida pelo RosAria, através do mecanismo de Detecção de fechamento de Loop (*Loop Closure Detection*) Labbé & Michaud [2014].

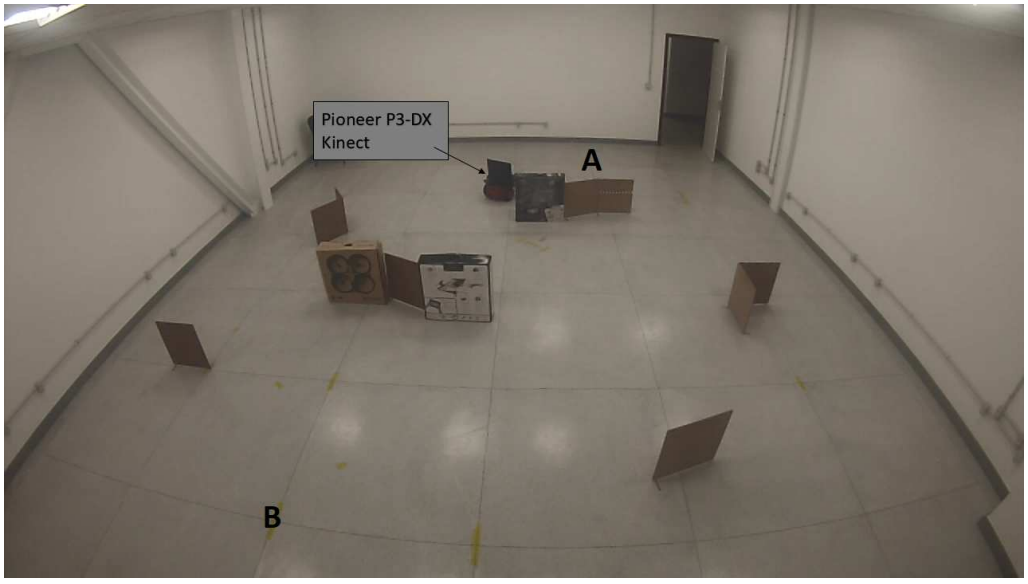

Figura 4.12. Ambiente experimental

Fonte:(Própria, 2018)

O ambiente experimental foi montado conforme a Figura 4.12, uma sala de aproximadamente $50m^2$ onde caixas retangulares foram utilizadas como obstáculos. O objetivo traçado para o robô foi de completar o ciclo A-B-A-B-A, a partir do ponto A e considerando $A(x, y = [0m, 0m])$ e $B(x, y = [6.25m, -2.4m])$. Para simplificar o

[4] http://www.ros.org/

entendimento, daqui em diante, a primeira ida de A até B será chamada de trajeto E1, o primeiro B para A de trajeto E2, o segundo A para B de trajeto E3 e, por fim, o segundo B para A de trajeto E4.

Como parâmetros para o desvio foram definidos $d_{obsT} = 0,85m$ e $d_{obsA} = 0,4m$.. A resolução do *grid* para o mapeamento foi definida de forma que cada célula do *grid* representasse $0,1 \times 0,1m^2$ do mundo real. Como fator de esquecimento, utilizou-se $f_e = 0,75$.

No trajeto E1, como esperado, o robô navega reativamente executando o Desvio Tangencial e captando informações do ambientes para o mapeamento (Figura 4.13). Como não se segue um caminho, o robô navega utilizando o controlador de erro de posição. Esse fato justifica a queda gradual e suave nas velocidades linear e angular quando o robô se aproxima do ponto de destino (Figuras 4.20 e 4.21).

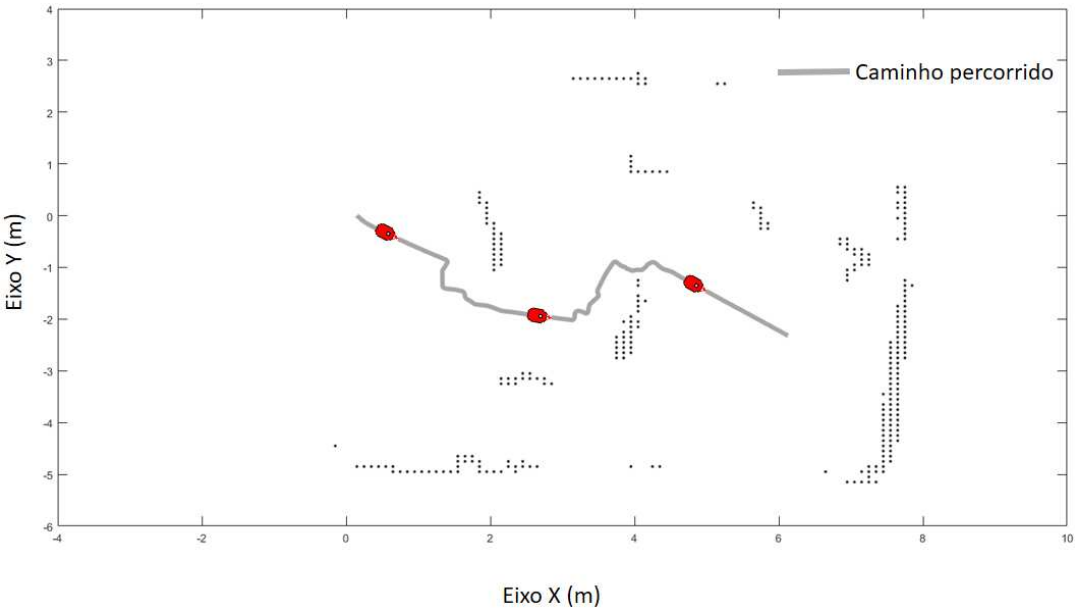

Figura 4.13. Deslocamento no ambiente com o mapa obtido ao final do trajeto E1.

Fonte: (Própria, 2018)

No trajeto E2, o mapeamento se mostrou insuficiente para encontrar um caminho com a busca A*. Isso aconteceu devido à zona de segurança, definida em quatro células, que impossibilitou a utilização do algoritmo para encontrar um caminho na região percorrida reativamente na navegação anterior. Esse fato é ilustrado pelas Figuras 4.14 e 4.15, que mostram o mapa obtido com e sem a zona de segurança.

Já em E3, a navegação no trajeto E2 se mostrou suficiente para encontrar um caminho com A*. Além disso, pode-se perceber a ação da correção de odometria,

que acontece não só para o posicionamento do robô mas para o mapa de referência. O mapa obtido ao final da navegação anterior (Figura 4.16) é muito menos realista que o mapa obtido ao final da navegação no trajeto E3. Percebe-se que o robô chega ao destino antes de completar 45 segundos de navegação, muito mais eficiente que os mais de 70 segundos do trajeto E1 e trajeto E2.

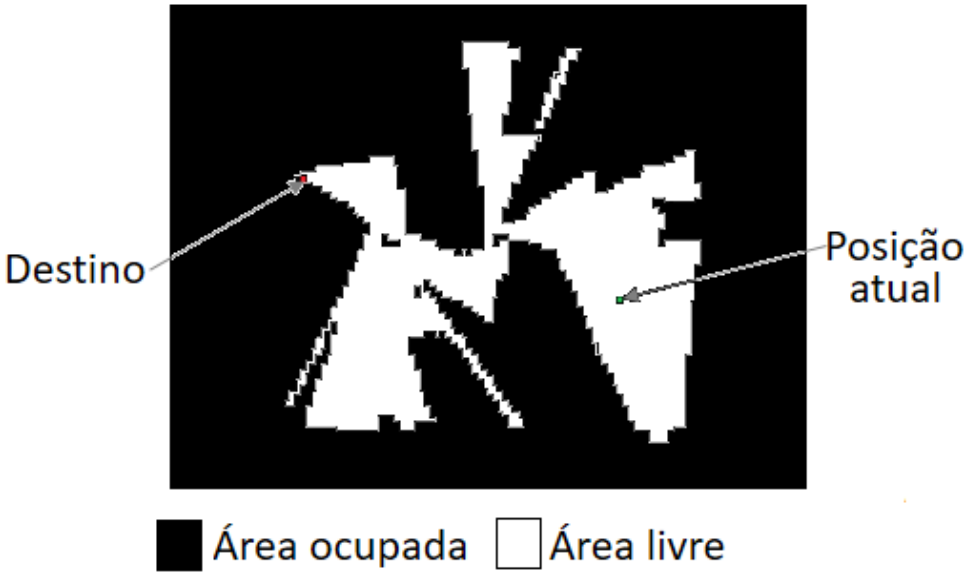

Figura 4.14. Antes de E2: Sem zona de segurança.

Fonte: (Própria, 2018)

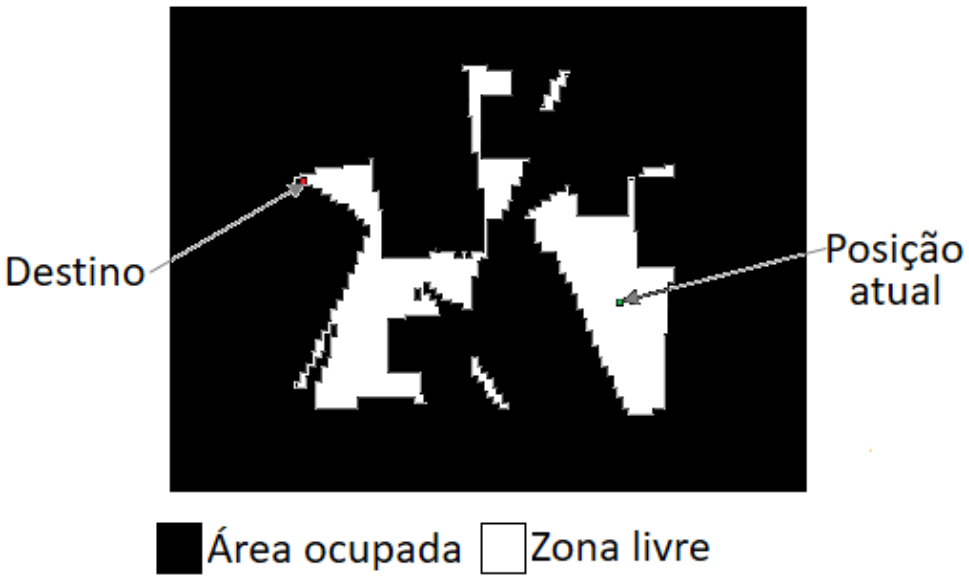

Figura 4.15. Antes de E2: Com zona de segurança de 4 células.

Fonte: (Própria, 2018)

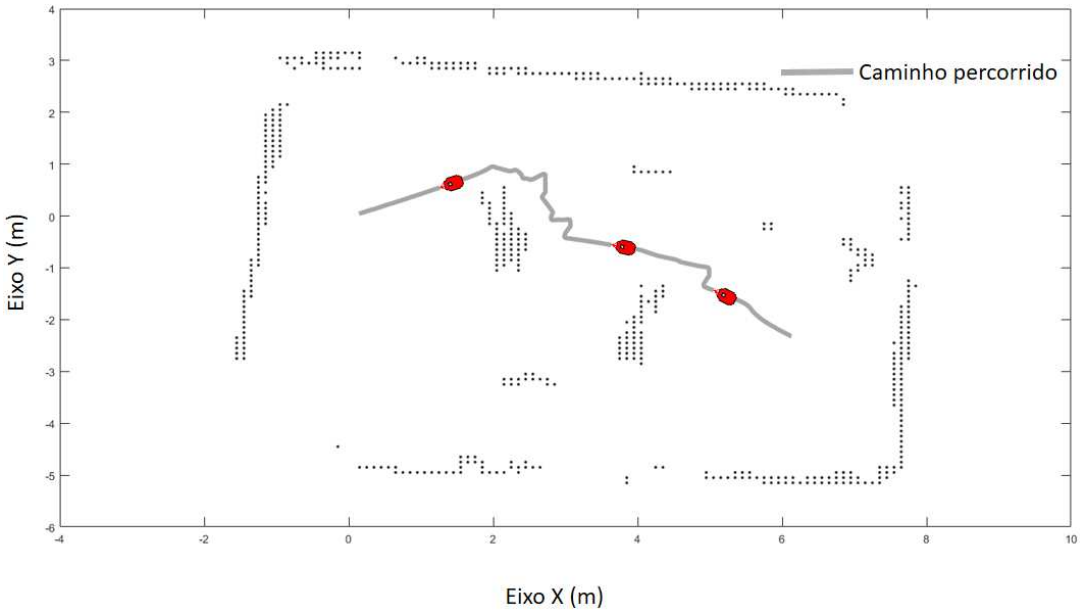

Figura 4.16. Deslocamento no ambiente com o mapa obtido ao final do trajeto E2.

Fonte: (Própria, 2018)

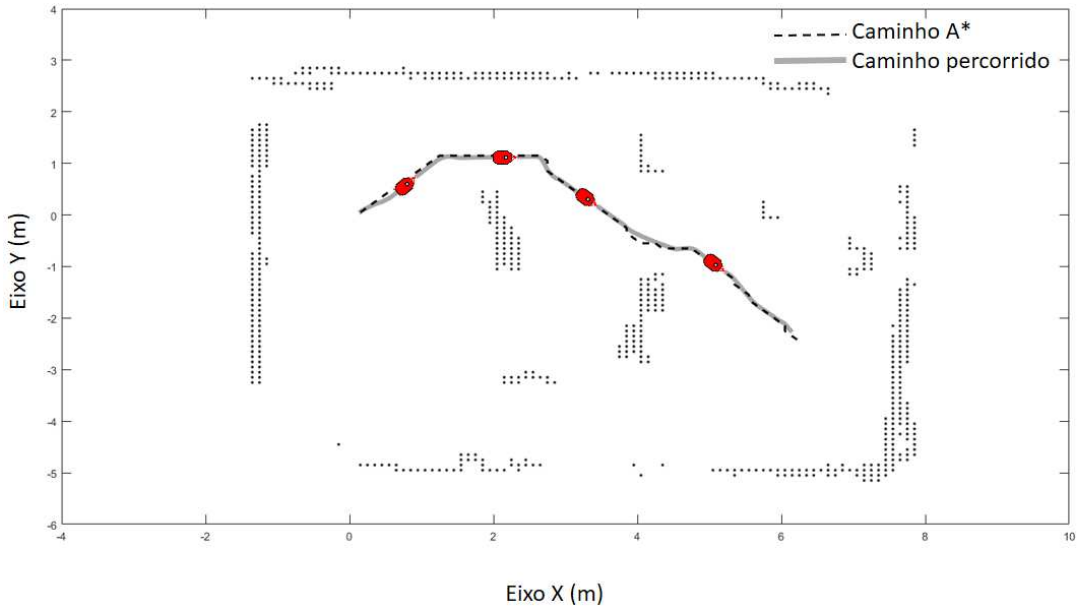

Figura 4.17. Deslocamento no ambiente com o mapa obtido ao final do trajeto E3.

Fonte: (Própria, 2018)

No trajeto E4 um caminho muito parecido com o anterior é encontrado. Considerando a navegação realizada nos trajetos anteriores, um possível caminho diferente já mapeado seria passando pelo centro do mapa, mas isso não acontece devido

à zona de segurança. Pode-se observar que alguns obstáculos presentes ao final do trajeto E3 aparecem de forma menos realista ao final de E4. Isso acontece devido a imprecisão dos sensores, erros de odometria e o modelo probabilístico de *Occupancy Grid*, fato que também foi percebido durante as simulações. A navegação é continua com valores de velocidade linear no limite definido em quase todo percurso, sem a necessidade de desviar de obstáculos. A tabela 4.3 traz os dados obtidos no experimento realizado.

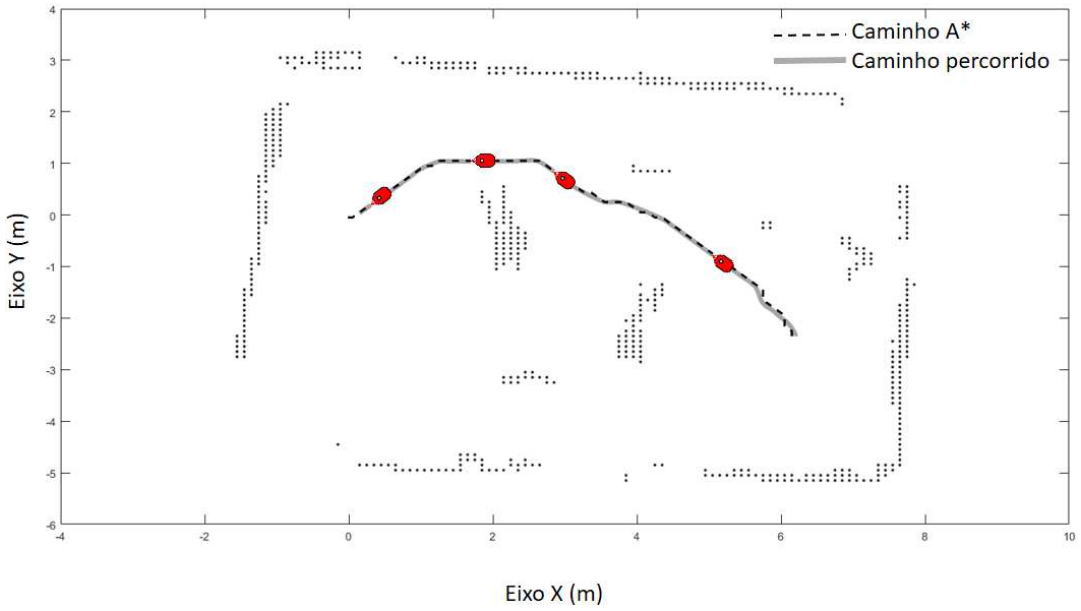

Figura 4.18. Deslocamento no ambiente com o mapa obtido ao final do trajeto E4.

Fonte: (Própria, 2018)

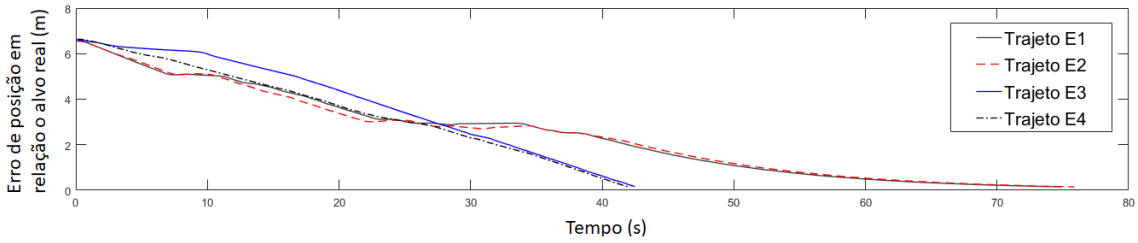

Figura 4.19. Erro de posição no experimento em ambiente real.

Fonte: (Própria, 2018)

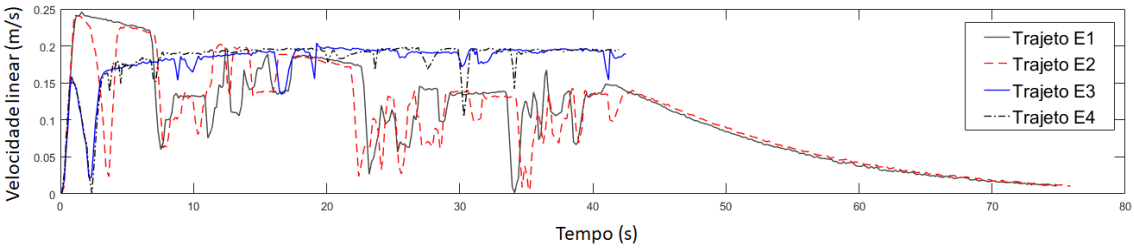

Figura 4.20. Velocidade linear no experimento em ambiente real.

Fonte: (Própria, 2018)

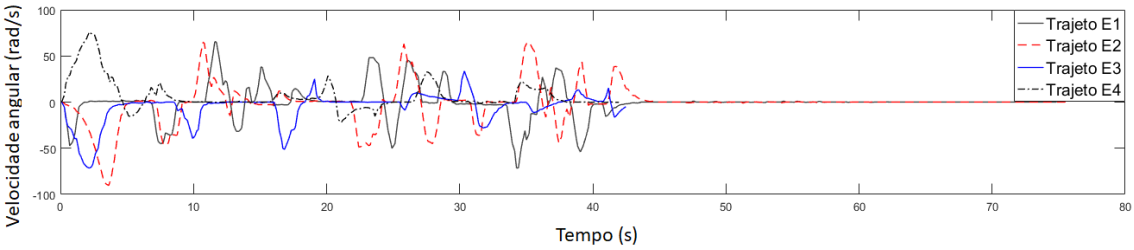

Figura 4.21. Velocidade angular no experimento em ambiente real.

Fonte: (Própria, 2018)

Tabela 4.3. Resultados do experimento em ambiente real

Dados/Percurso	traj. E1	traj. E2	traj. E3	traj. E4
Distância percorrida (m)	8,52	8,53	7,97	7,97
Vel. linear média (m/s)	0,10 ± 0,06	0,10 ± 0,06	0,18 ± 0,03	0,18 ± 0,03
IASC	1,00	0,48	0,22	0,28
Tempo de navegação (s)	74,86	75,88	42,48	41,96

Fonte: (Própria, 2018)

4.3 Experimento 3: Controle híbrido

Para avaliar o controle híbrido com uma utilização mais significativa do Desvio Tangencial enquanto se segue um caminho, foi realizado um novo experimento. O robô foi colocado em frente à um obstáculo, tendo como objetivo de navegação um ponto atrás do obstáculo. Para simular a situação proposta, forneceu-se um mapa onde todas as células estavam definidas como área livre e o executou-se o algoritmo A*, que como esperado, encontrou um caminho em linha reta até o destino. Assim, esperava-se que o robô seguisse em linha reta até detectar o obstáculo, efetuasse a evasão e retomasse o caminho planejado.

Duas situações de obstáculos foram definidas para o experimento. A primeira (Figura 4.22), onde um obstáculo colocado de forma ortogonal ao caminho traçado. Já a segunda (Figura 4.24), além do obstáculo ortogonal foram adicionados outros

dois de forma inclinada. O objetivo foi avaliar como o robô se comporta quando existe uma menor capacidade de detecção do obstáculo (primeira situação) e quando existe uma referência maior do obstáculo para a evasão (segunda situação).

Figura 4.22. Situação 1: Obstáculo mais fácil de se detectar.

Fonte: (Própria, 2018)

Em ambas situações, diferentes valores para d_{obsA} e d_{obsT} foram testados. As Figuras 4.23 e 4.25 mostram o caminho percorrido pelo robô em cada situação. Pode-se notar que a baixa capacidade de sensoriamento do robô causa a oscilação na navegação, tendo em vista que o robô “perde de vista"o obstáculo enquanto está efetuando o desvio, o levando à tentar retomar o caminho planejado pelo A*. Notou-se também que para valores de $d_{obsA} = 0.55$ e $d_{obsT} = 0.85$ o robô não conseguiu completar à evasão com segurança para o obstáculo totalmente ortogonal (Figura 4.23), levando-o a colidir com o obstáculo.

Os videos dos experimentos em ambiente real podem ser encontrados em `https://www.youtube.com/watch?v=1mzWuNmF2Sg` e `https://www.youtube.com/watch?v=LJaxxm4_sII`.

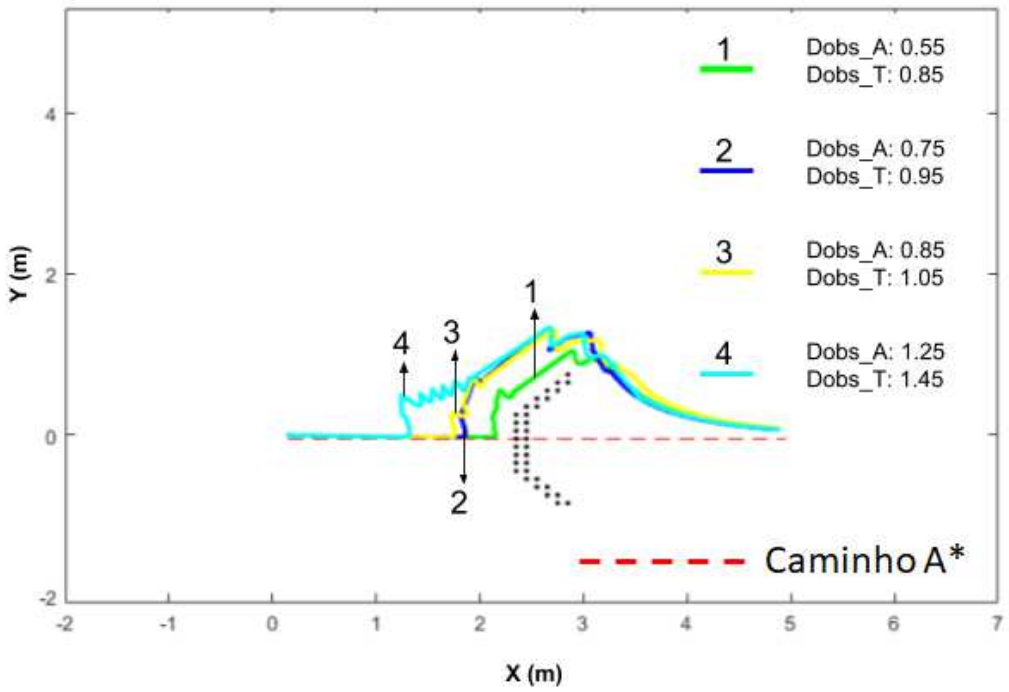

Figura 4.23. Situação 1: Desvio sobre o obstáculo mais fácil de se detectar.

Fonte: (Própria, 2018)

Figura 4.24. Situação 2: Obstáculo mais difícil de se detectar.

Fonte: (Própria, 2018)

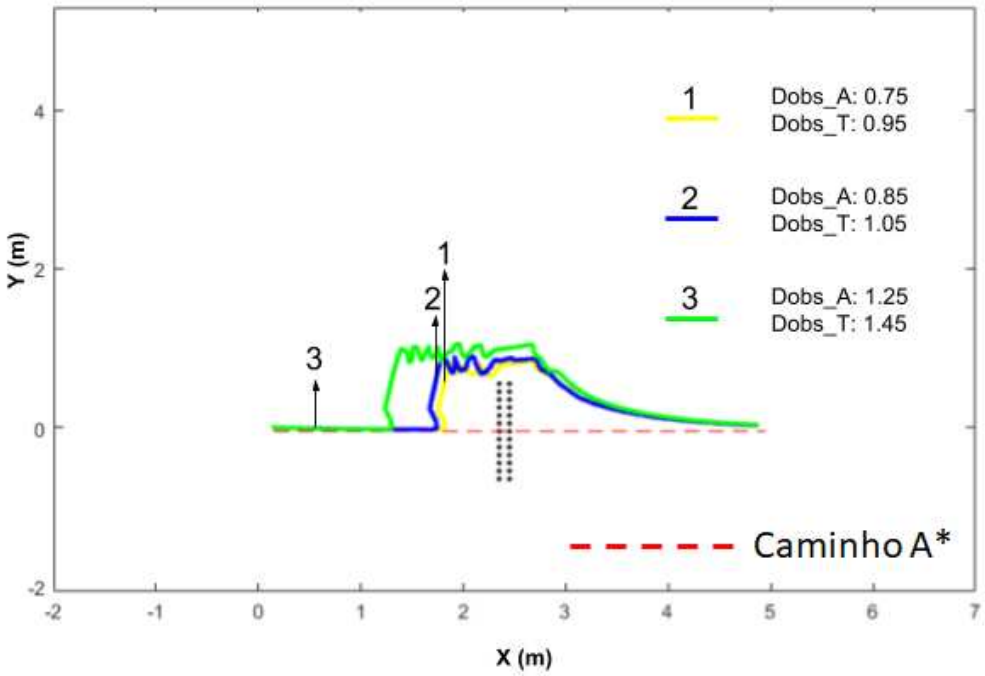

Figura 4.25. Situação 2: Desvio sobre o obstáculo mais difícil de se detectar.

Fonte: (Própria, 2018)

Capítulo 5

CONCLUSÕES

Este trabalho traz uma estratégia de navegação para robôs móveis com base em um algoritmo de planejamento híbrido. A solução obteve sucesso em experimentos através de simuladores e em ambiente real, promovendo uma navegação totalmente autônoma, sem interferência humana e ajuda de sensores externos.

Os resultados da simulação mostram que a navegação é muito mais eficiente quando se segue um caminho obtido pelo planejamento global. O experimento com a navegação prolongada por mais tempo mostra que as heurísticas distância Euclidiana e distância Manhattan produzem resultados similares na eficiência da navegação. Além disso, foi possível observar que o mapeamento não converge para um mapa estável. Isso se deve a capacidade limitada de varredura dos sensores, que fazem com células relacionadas a áreas com obstáculos alternem entre os valores ocupado e vazio constantemente. Também é possível observar que a diminuição na resolução do *grid* não produz efeito positivo na convergência de representação do ambiente.

Apesar da validação através do simulador, o sonar do Pioneer P3-DX se mostrou inviável no experimento real, com grande imprecisão de sensoriamento. Outro fator importante, a odometria fornecida pelo robô, também ruidosa, acaba afetando diretamente o mapeamento, gerando mapas sem conexão com a realidade. A adição do sensor Kinect juntamente com a ferramenta Rtabmap acabou se mostrando muito efetiva na correção da odometria, sem descaracterizar o conjunto robô e sensor como plataforma de baixo custo e capacidade de sensoriamento. Assim, foi possível realizar o mapeamento do ambiente enquanto o robô navegava pelo ambiente, obtendo uma representação muito próxima da realidade.

Apesar de não conseguir varrer totalmente a parte frontal do robô, os sensores se mostraram capazes de realizar a manobra de evasão de obstáculos com relativa

segurança, ainda que o Kinect não detecte obstáculos a distâncias inferiores a 0,8m (o sonar continua detectando o obstáculo), sendo vantajoso utilizar valores maiores que 0,8m para d_{obs}, especialmente para a navegação reativa.

O mapeamento do ambiente, enquanto se navega, acaba gerando mapas incompletos com ausência de informação sobre obstáculos, ou parte deles. Esse fato, somado às dimensões físicas do robô, fazem com que seja essencial a utilização da zona de segurança para efetuar a busca A*. Nos experimentos, a busca foi capaz de encontrar um caminho livre de colisões com obstáculos conhecidos, na área mapeada, sempre que a zona de segurança não desconectasse os pontos de partida e destino, como ilustrado pelas Figuras 4.14 e 4.15. Além disso, os resultados mostram que o robô consegue efetuar o controle híbrido retornando para o caminho planejado mesmo em total ausência de conhecimento do obstáculo. As limitações, nesse caso, se dão por causa da capacidade de sensoriamento.

O sucesso em promover uma navegação continua sem que o robô pare para efetuar algum processamento de planejamento, principalmente quando de forma deliberativa com o desvio de obstáculos desconhecidos, pode ser bastante útil em tarefas relacionados à robôs de baixo custo, como os de uso doméstico. A baixa exigência computacional faz com que o algoritmo possa ser implementado em aplicações onde a navegação autônoma é uma tarefa secundária, sem acarretar peso de processamento para a tarefa principal.

Algumas aplicações práticas para a navegação proposta nesse trabalho podem ser robôs que auxiliam pessoas à mover objetos entre locais de um ambiente de trabalho, como por exemplo livros em bibliotecas, peças em linhas de produção e materiais em um armazém. Esses tipos de ambientes normalmente são parcialmente observáveis e deve-se realizar um percurso de ida e volta pré-definido várias vezes. Cadeiras de rodas autônomas podem se beneficiar com uma solução computacional de baixo custo, diminuindo o custo de produção. O Robôs de limpeza domésticos podem ser mais eficientes se o ambiente passa a ser conhecido com o tempo, melhorando a navegação, diminuindo distância percorrida e energia consumida para realizar suas tarefas cotidianas.

Como trabalhos futuros, sugere-se experimentos com a utilização de sensores capazes de varrer uma área maior que o Kinect, sem a limitação de detectar obstáculos muito próximos. Do ponto de vista do algoritmo, pode-se utilizar informações do mapeamento para tomar decisões mais inteligentes ao efetuar o Desvio Tangencial (avaliar se o desvio para um lado é mais vantajoso que outro por exemplo), sendo esse um ponto que pode ser melhorado.

Também pode-se estudar diferentes formas de fazer o robô retomar o caminho

planejado após efetuar um desvio não planejado, tendo em vista que a forma com que é proposta nessa dissertação (escolher o ponto mais próximo não alcançado) é relativamente simples.

Apêndice A

Componentes do experimento

Para realização do experimento em ambiente real descrito no Capitulo 4 foram utilizados a ferramenta ROS (Robot Operating System), o ambiente de desenvolvimento Matlab (Matrix Laboratory), o sensor RGB-D Kinect e a plataforma robótica Pioneer P3-DX. A integração desses elementos se deu com o ROS provendo os drivers e bibliotecas para realizar o envio e recebimento de informações entre todos os componentes do experimento, através do esquema de rede baseada em grafo em que é desenvolvido o ROS. A forma como essa integração é discutida a seguir.

A.1 ROS

O ROS é um framework *open source* que funciona através de uma rede distribuída ponto-a-ponto baseada em grafos. De uma forma geral, um nó no ROS é uma entidade que publica e recebe mensagens dentro da rede. Como é voltado para a robótica, o *framework*, assim como a comunidade que o utiliza, oferecem já implementados diversos drivers e bibliotecas, encapsulados como nós, para facilitar o desenvolvimento de soluções. O ROS pode ser instalado em qualquer sistema operacional linux e permite o desenvolvimento de aplicações próprias para serem utilizadas dentro da rede.

Os pontos chaves do funcionamento dos nós são os parâmetros, os tópicos inscritos e as mensagens publicadas. Parâmetros são as configurações do nó no estado atual dentro da rede. A calibração de um laser pode variar dependendo dos valores parâmetros quando o nó referente a este laser é inserido na rede, por exemplo. Tópicos inscritos é a lista dos tópicos dos quais um nó receberá mensagens dentro da rede. Esta informação é importante para um outro nó enviar um comando para outro através de uma mensagem, por exemplo. Quando um nó publica uma

mensagem, associada à um tópico, todos os nós dentro da rede que tenham se inscrito para receber mensagens daquele tópico receberam essa mensagem. Uma mensagem pode ser as leituras captadas por uma varredura a laser, por exemplo.

As bibliotecas fornecidas pelo ROS e pela comunidade, que foram utilizadas nesse trabalho, serão discutidas a seguir.

A.1.1 RosAria

O pacote RosAria é uma interface para o ROS baseada na biblioteca ARIA e compatível com a maioria dos robôs produzidos por Adept MobileRobots, MobileRobots Inc. e ActivMedia mobile, dentre eles o Pioneer P3-DX. RosAria facilita a conexão com as plataformas, publicando mensagens referentes a odometria e velocidade linear e angular por exemplo. O nó RosAria também aceita o envio de mensagens para mover o robô e a definição de parâmetros para calibração da odometria. Os tópicos relevantes para a navegação realizada no experimento são listados a seguir:

- cmd_vel : recebe mensagens com os parametros da velocidade linear e angular desejadas para mover o robô
- pose : publica mensagens com as informações da odometria do robô, posição X,Y,Z e orientação em X,Y,Z
- sonar : publica as leituras do sonar, com as posições X e Y, tendo como referência o centro do robô.

A.1.2 Rtabmap_ros

O Pacote Rtabmap_ros é um compilado da ferramenta rtabmap [1], uma abordagem SLAM baseada em câmeras RGB-D e a estratégia de Loop Closure Detection. O nó rtabmap é capaz de fornecer para rede o mapeamento em forma de *Occupancy Grid* sob o tópico grid_map, quando configurado conforme a Figura A.1. Para isso, utiliza-se o Kinect como fonte RGB-D, a odometria do P3-DX fornecida pelo RosAria e a varredura a laser, que neste caso é simulada a partir do Kinect, utilizando o pacote depthimage_to_laserscan. As informações do Kinect, que são utilizadas tanto pelo Rtabmap quanto o Depthimage_to_laserscan, são fornecidas pelo pacote Freenect_stack, o driver do sensor compilado como nó dentro da rede.

[1] http://introlab.github.io/rtabmap/

O arquivo de configuração, chamado "experimento.launch", utilizado para inicializar todos os nós, com os devidos parâmetros de configuração, utilizados no experimento dentro da rede é dado a seguir:

```
<launch>
<!-- ROSARIA -->
<node pkg="rosaria" type="RosAria" name="RosAria"></node>
<!-- KINECT DRIVER -->
<include file="$(find freenect_launch)/launch/examples/freenect-registered-xyzrgb.
    launch"></include>
<!-- KINECT PARA LASERSCAN -->
<node pkg="depthimage_to_laserscan" type="depthimage_to_laserscan" name="
    depthimage_to_laserscan">
<remap from="image"          to="/camera/depth_registered/image_raw"/>
<remap from="camera_info"    to="/camera/depth_registered/camera_info"/>
<remap from="scan"           to="/kinect_scan"/>
<param name="range_max"      type="double" value="3.9"/>
<param name="range_min"      type="double" value="0.6"/>
<param name="scan_height"    type="double" value="1"/>
</node>
<!-- SLAM -->
<group ns="rtabmap">
<node name="rtabmap" pkg="rtabmap_ros" type="rtabmap" output="screen" args="--
    delete_db_on_start">
<param name="frame_id"           type="string" value="camera_link"/>
<param name="subscribe_depth"    type="bool" value="true"/>
<param name="subscribe_scan"     type="bool" value="true"/>
<param name="queue_size"         type="int" value="10"/>
<remap from="odom"               to="/RosAria/pose"/>
<remap from="scan"               to="/kinect_scan"/>
<remap from="rgb/image"          to="/camera/rgb/image_rect_color"/>
<remap from="depth/image"        to="/camera/depth_registered/image_raw"/>
<remap from="rgb/camera_info"    to="/camera/rgb/camera_info"/>
<!-- PARAMETROS RTAB-Map -->
<param name="grid_size"                  type="double" value="12"/>
<param name="grid_eroded"                type="bool"   value="true"/>
<param name="grid_cell_size"             type="double" value="0.10"/>
<param name="grid_unknown_space_filled" type="bool"   value="true"/>
<param name="RGBD/ProximityBySpace"      type="string" value="false"/>
<param name="RGBD/AngularUpdate"         type="string" value="0.01"/>
<param name="RGBD/LinearUpdate"          type="string" value="0.01"/>
<param name="RGBD/OptimizeFromGraphEnd" type="string" value="false"/>
<param name="Optimizer/Slam2D"           type="string" value="true"/>
<param name="Reg/Strategy"               type="string" value="1"/>
<param name="Reg/Force3DoF"              type="string" value="true"/>
<param name="Vis/MaxDepth"               type="string" value="4.0"/>
<param name="Vis/MinInliers"             type="string" value="5"/>
<param name="Vis/InlierDistance"         type="string" value="0.05"/>
<param name="Rtabmap/TimeThr"            type="string" value="700"/>
<param name="Mem/RehearsalSimilarity"    type="string" value="0.45"/>
<param name="Icp/CorrespondenceRatio"    type="string" value="0.5"/>
</node>
</group>
</launch>
```

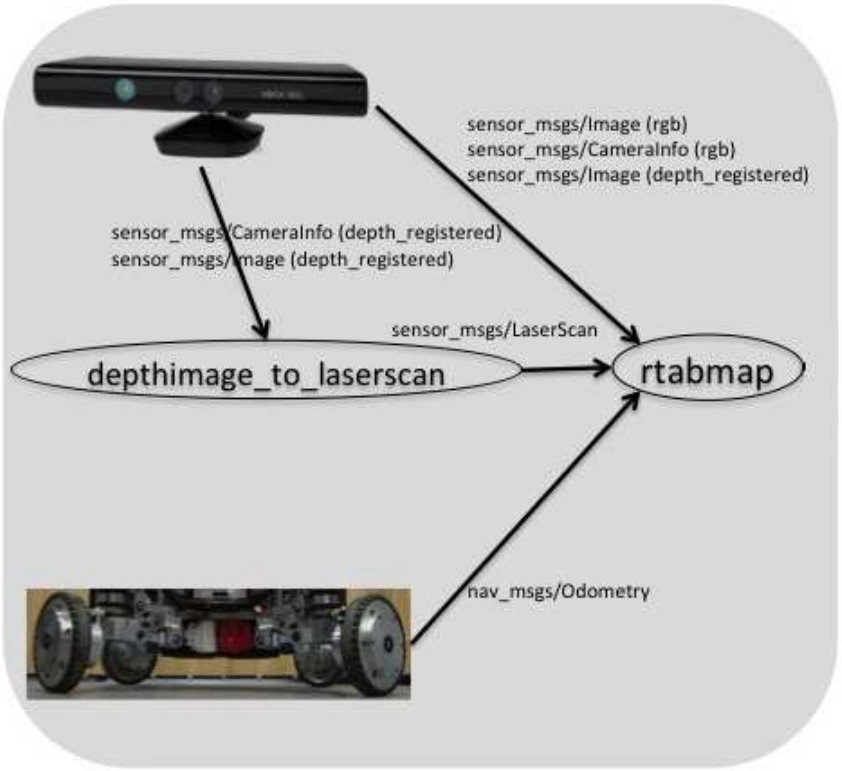

Figura A.1. Configuração do Rtabmap_ros

Fonte: Adaptado de [ROS.org, 2016]

Referências Bibliográficas

Adept Technology, I. (2011). Pioner 3 - dx. Online.

Amato, N. M. & Wu, Y. (1996). A randomized roadmap method for path and manipulation planning. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 1, pp. 113--120. IEEE.

Borenstein, J. & Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on systems, Man, and Cybernetics*, 19(5):1179--1187.

Borenstein, J. & Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation*, 7(3):278--288.

Brandão, A. S.; Andaluz, V. H.; Sarcinelli-Filho, M. & Carelli, R. (2011). 3-d path-following with a miniature helicopter using a high-level nonlinear underactuated controller. In *2011 9th IEEE International Conference on Control and Automation (ICCA)*, pp. 434–439.

Brandão, A. S.; Sarcinelli-Filho, M. & Carelli, R. (2013). An analytical approach to avoid obstacles in mobile robot navigation. *International Journal of Advanced Robotic Systems*, 10(6):278.

Buniyamin, N.; Ngah, W. W.; Sariff, N. & Mohamad, Z. (2011). A simple local path planning algorithm for autonomous mobile robots. *International journal of systems applications, Engineering & development*, 5(2):151--159.

CONTE, G.; LONGHI, S. & ZULLI, R. (1996). Motion planning for unicycle and car-like robots. *International Journal of Systems Science*, 27(8):791–798.

de Oliveira, G. C. R.; de Carvalho, K. B. & Brandão, A. S. (2018). A hybrid strategy for robot navigation in semi-structured environments. In *2018 IEEE International Conference on Industrial Technology (ICIT)*, pp. 23–28.

Demir, M. & Sezer, V. (2017). Improved follow the gap method for obstacle avoidance. In *Advanced Intelligent Mechatronics (AIM), 2017 IEEE International Conference on*, pp. 1435--1440. IEEE.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269--271.

Dudek, G. & Jenkin, M. (2010). *Computational principles of mobile robotics*. Cambridge university press, New York, NY, USA, 2nd edição.

Elbanhawi, M. & Simic, M. (2014). Sampling-based robot motion planning: A review. *Ieee access*, 2:56--77.

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46--57.

Ferguson, D.; Likhachev, M. & Stentz, A. (2005). A guide to heuristic-based path planning. In *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS)*, pp. 9--18.

Fernandes, R.; Bessa, M. M. & Brandão, A. S. (2017). Performance analysis of positioning controller for mobile robots. In *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)*, pp. 1–5.

Garcia, M. P.; Montiel, O.; Castillo, O.; Sepúlveda, R. & Melin, P. (2009). Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. *Applied Soft Computing*, 9(3):1102--1110.

Hart, P. E.; Nilsson, N. J. & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100--107.

Hu, Y. & Yang, S. X. (2004). A knowledge based genetic algorithm for path planning of a mobile robot. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 5, pp. 4350--4355. IEEE.

Karaman, S. & Frazzoli, E. (2013). Sampling-based optimal motion planning for non-holonomic dynamical systems. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 5041--5047. IEEE.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pp. 396--404. Springer.

Khoshelham, K. & Elberink, S. O. (2012). Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437--1454.

Koenig, S. & Likhachev, M. (2002). D* lite. *AAAI National Conference on Artificial Intelligence*, pp. 476--483.

Koenig, S.; Likhachev, M. & Furcy, D. (2004). Lifelong planning a*. *Artificial Intelligence*, 155(1-2):93--146.

Labbé, M. & Michaud, F. (2014). Online global loop closure detection for large-scale multi-session graph-based slam. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 2661--2666. IEEE.

Laumond, J.-P.; Sekhavat, S. & Lamiraux, F. (1998). Guidelines in nonholonomic motion planning for mobile robots. In *Robot motion planning and control*, pp. 1--53. Springer.

LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. *Tech. Rep. TR 98-11*.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.

Lumelsky, V. & Stepanov, A. (1986). Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE transactions on Automatic control*, 31(11):1058--1063.

Mac, T. T.; Copot, C.; Tran, D. T. & De Keyser, R. (2016). Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems*, 86:13--28.

Mujahed, M.; Fischer, D. & Mertsching, B. (2016). Tangential gap flow (tgf) navigation: A new reactive obstacle avoidance approach for highly cluttered environments. *Robotics and Autonomous Systems*, 84:15--30.

Ray, C.; Mondada, F. & Siegwart, R. (2008). What do people expect from robots? In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 3816--3821. IEEE.

ROS.org (2016). Kinect + odometry + fake 2d laser from kinect. Online.

Russell, S. & Norvig, P. (2009). Artificial intelligence: a modern approach. 3rd. *Prentice Hall Press, Upper Saddle River, NJ, USA. isbn*, 136042597:9780136042594.

Souissi, O.; Benatitallah, R.; Duvivier, D.; Artiba, A.; Belanger, N. & Feyzeau, P. (2013). Path planning: A 2013 survey. In *Industrial Engineering and Systems Management (IESM), Proceedings of 2013 International Conference on*, pp. 1--8. IEEE.

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 3310--3317. IEEE.

Stentz, A. et al. (1995). The focussed d* algorithm for real-time replanning. In *IJCAI*, volume 95, pp. 1652--1659.

Thrun, S. (2002). Probabilistic robotics. *Communications of the ACM*, 45(3):52--57.

Thrun, S. et al. (2002). Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1:1--35.

Wang, L. C.; Yong, L. S. & Ang, M. H. (2002). Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment. In *Intelligent Control, 2002. Proceedings of the 2002 IEEE International Symposium on*, pp. 821--826. IEEE.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.

Zhang, H.; Butzke, J. & Likhachev, M. (2012). Combining global and local planning with guarantees on completeness. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 4500--4506. IEEE.

Zhang, Y.; Gong, D.-w. & Zhang, J.-h. (2013). Robot path planning in uncertain environment using multi-objective particle swarm optimization. *Neurocomputing*, 103:172--185.