Laboratório de Sistemas de Computação e Robótica
Programa de Pós-Gradução em Engenharia Elétrica
Universidade Federal de Minas Gerais (UFMG)
Av. Antônio Carlos 6627, CEP 31270-901,
Belo Horizonte, MG Brasil

DISSERTAÇÃO DE MESTRADO Nº 1012

Uma estratégia para navegação de robôs de serviço semiautônomos usando informação local e planejadores probabilísticos

Elias José de Rezende Freitas

DATA DA DEFESA: 20/10/2017

Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

UMA ESTRATÉGIA PARA NAVEGAÇÃO DE ROBÔS DE SERVIÇO SEMIAUTÔNOMOS USANDO INFORMAÇÃO LOCAL E PLANEJADORES PROBABILÍSTICOS

Elias José de Rezende Freitas

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Orientador: Prof. Guilherme Augusto Silva Pereira

Belo Horizonte - MG

Outubro de 2017

Freitas, Elias José de Rezende.
F866e Uma estratégia para navegação de robôs de serviço semiautônomos usando informação local e planejadores probabilísticos [manuscrito] / Elias José de Rezende Freitas. - 2017.
102 f., enc.: il.

Orientador: Guilherme Augusto Silva Pereira.

Dissertação (mestrado) Universidade Federal de Minas Gerais, Escola de Engenharia.

Anexos: f. 93-102.

Bibliografia: f. 85-92.

1. Engenharia elétrica - Teses. 2. Navegação de robôs móveis - Teses. 3. Desvio de obstáculos -Teses. I. Pereira, Guilherme Augusto Silva. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

"Uma Estratégia para Navegação de Robôs de Serviço Semiautônomos Usando Informação Local e Planejadores Probabilísticos"

Elias José de Rezende Freitas

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica.

Aprovada em 20 de outubro de 2017.

Por:

Prof. Dr. Guilherme Augusto Silva Pereira
DEE (UFMG) - Orientador

Prof. Dr. Vinícius Mariano Gonçalves
DEE (UFMG)

Prof. Dr. Douglas Guimarães Macharet
DCC (UFMG)

Prof. Dr. Danilo Alves de Lima
DEG (UFLA)

AGRADECIMENTOS

Esses agradecimentos são pouco em relação a tudo que recebi nesses anos. Assim, desejo que as lacunas desses agradecimentos sejam preenchidas pela minha gratidão a cada um que participa da minha vida.

Agradeço a Deus em primeiro lugar por caminhar comigo e por ter me guiado até esse momento. Nele está o sentido de tudo.

À minha esposa Michelle que me apoiou e me incentivou, enfrentando todos os desafios junto comigo, nas alegrias e nas tristezas. Aos meus pais, Adair e Maria Auxiliadora, que são uma base firme para mim e que estão sempre presentes. Em especial agradeço à minha irmã Rosana, que certamente faz parte dessa vitória, por me lembrar do Essencial. À Elizangela, Paula e João que são parte dessa família que me sustenta.

Agradeço à Comunidade Missionária de Villaregia por me formar como cristão, me ensinar a doar o pouco que tenho de forma simples à humanidade e alargar a minha família.

Ao prof. Guilherme que desde a graduação tem me orientado e incentivado nos diversos trabalhos, sem o qual não teria chegado até aqui e tantas portas não teriam sido abertas.

Aos colegas do laboratório CORO que estiveram sempre abertos a ajudar e a propor ideias.

Por fim, agradeço a todos os amigos e familiares, sem os quais toda essa aventura não teria a mesma graça.

Como dizem os moçambicanos: Khanimambo!

"Ninguém vence sozinho, nem no campo nem na vida!"
(Papa Francisco)

RESUMO

ELIAS J. R. FREITAS. **Uma estratégia para navegação de robôs de serviço semiautônomos usando informação local e planejadores probabilísticos**. 2017. 102 f. Dissertação (Mestrado Engenharia Elétrica) – Escola de Engenharia (UFMG), Belo Horizonte – MG.

Os robôs de serviço são desenvolvidos para realizarem tarefas úteis aos seres humanos, como limpar um ambiente, entregar uma encomenda ou auxiliar na mobilidade humana, especialmente, de idosos ou de pessoas portadoras de alguma necessidade especial. Existem vários tipos de robôs de serviços, dentre eles os classificados como semiautônomos, foco deste trabalho. Esses robôs são parcialmente controlados por um ser humano, ficando o robô responsável por tarefas específicas, principalmente aquelas relacionadas com a segurança da missão. Esta dissertação propõe uma estratégia para a navegação de robôs semiautônomos baseada apenas em informações locais obtidas pelos sensores instalados no robô e um planejador probabilístico que gera caminhos a serem seguidos localmente por ele, garantindo assim o desvio de obstáculos. A missão que o robô deve cumprir é definida pelo usuário por meio de uma sequência de comandos simples, como siga em frente e vire à direita, sendo esse comando codificado por campos vetoriais artificiais, que são utilizados pelo funcional de custo a ser otimizado pelo planejador. Essa estratégia de navegação é interessante já que esses robôs de serviço semiautônomos, diferentemente de outros, podem não ter um mapa do ambiente ou um sistema de localização global. Nesse caso, eles operam em ambientes desconhecidos e sua missão pode ser controlada por um operador humano. Arquiteturas puramente deliberativas, que utilizam um planejador de caminhos para desvio de obstáculos, geralmente, necessitam da localização do robô em um mapa, sendo difíceis de serem aplicadas por esse tipo de robô. Por outro lado, arquiteturas puramente reativas podem levar o robô a condições indesejadas ou fazer com que o robô execute movimentos pouco suaves. Para a avaliação experimental dessa estratégia de navegação, considerou-se um ambiente típico de prédios comerciais, hospitais e escolas composto de corredores e interseções. Nesse ambiente, a estratégia foi utilizada para controlar robôs simulados e dois robôs de serviço semiautônomos reais. Os resultados obtidos demostram que os robôs foram capazes de executar a missão definida pelo usuário, desviando eficientemente e de forma planejada de obstáculos estáticos e, para algumas situações bem definidas, dinâmicos, incluindo pessoas se locomovendo no ambiente durante a missão. A comparação com métodos clássicos baseados em SLAM mostra que, além de mais simples e eficiente, a metodologia proposta apresenta melhores resultados em ambientes dinâmicos. Ao comparar com uma estratégia reativa que utiliza campos potenciais, percebe-se que esse tipo de estratégia nem sempre fornece caminhos suaves, podendo levar a condições de mínimos locais, diferentemente do resultado obtido com a metodologia deste trabalho.

Palavras-chave: robôs de serviço, navegação, desvio de obstáculos, planejadores probabilísticos.

ABSTRACT

ELIAS J. R. FREITAS. **Uma estratégia para navegação de robôs de serviço semiautônomos usando informação local e planejadores probabilísticos**. 2017. 102 f. Dissertação (Mestrado Engenharia Elétrica) – Escola de Engenharia (UFMG), Belo Horizonte – MG.

Service robots are designed to perform useful tasks for humans, such as cleaning environments, delivering an order or assisting human mobility, especially, assisting elderly or people with disabilities. Among several types of service robots, this work is focused in the robots classified as semi-autonomous service robots. These robots are partially controlled by a person, being the robot responsible for specific tasks, especially those related to mission safety. This master thesis proposes a strategy for the navigation of semi-autonomous service robots based on local information obtained by the sensors installed on the robot and an probabilistic planner, which calculates paths to be followed locally by the robot, thus ensuring obstacle avoidance. The robot's mission is defined by a user through a sequence of simple commands, such as go ahead and turn right. These commands are encoded by artificial vector fields, which are used by the functional cost to be optimized by the planner. This navigation strategy is interesting because semi-autonomous service robots, unlike others, may not have a map of the environment or a global localization system, since they may be navigating in unknown environments under close supervision of a human. Purely deliberative architectures use path planners to avoid obstacles and usually require the localization of the robot on a map, being difficult to be applied to this type of robot. On the other hand, purely reactive architectures may take the robot to undesired conditions or may generate non-smooth paths. For the experimental evaluation of the proposed navigation strategy, we considered a typical environment of office buildings, hospitals, and schools that is consisted of corridors and intersections. In this environment, the strategy was used to control a simulated robot and two actual semi-autonomous service robots. The results show that the robots were able to perform the user-defined mission efficiently, avoiding static obstacles and, for some well-defined situations, also dynamic obstacles, including people moving in the environment during the mission. The comparison with classical methods based on SLAM shows that, besides simpler and more computationally efficient, the proposed methodology presents better results in dynamic environments. When comparing with a reactive strategy that uses potential fields, it is noticed that this strategy does not always provide smooth paths, and can lead to local minimum conditions, unlike the result obtained with the our methodology.

Keywords: service robots, navigation, obstacle avoidance, probabilistic planners.

LISTA DE FIGURAS

LISTA DE ALGORITMOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CORO	Laboratório de Sistemas de Computação e Robótica
IEEE	*Institute of Electrical and Electronic Engineers*
IFR	*International Federation of Robotics*
ISO	*International Organization for Standardization*
MARIA	*Manipulator Robot for Interaction and Assistance*
OMPL	*Open Motion Planning Library*
ROS	*Robot Operating System*
RRT*	*Optimal Rapidly-exploring Random Trees*
SLAM	*Simultaneous Localization and Mapping*
UFMG	Universidade Federal de Minas Gerais

SUMÁRIO

CAPÍTULO

1

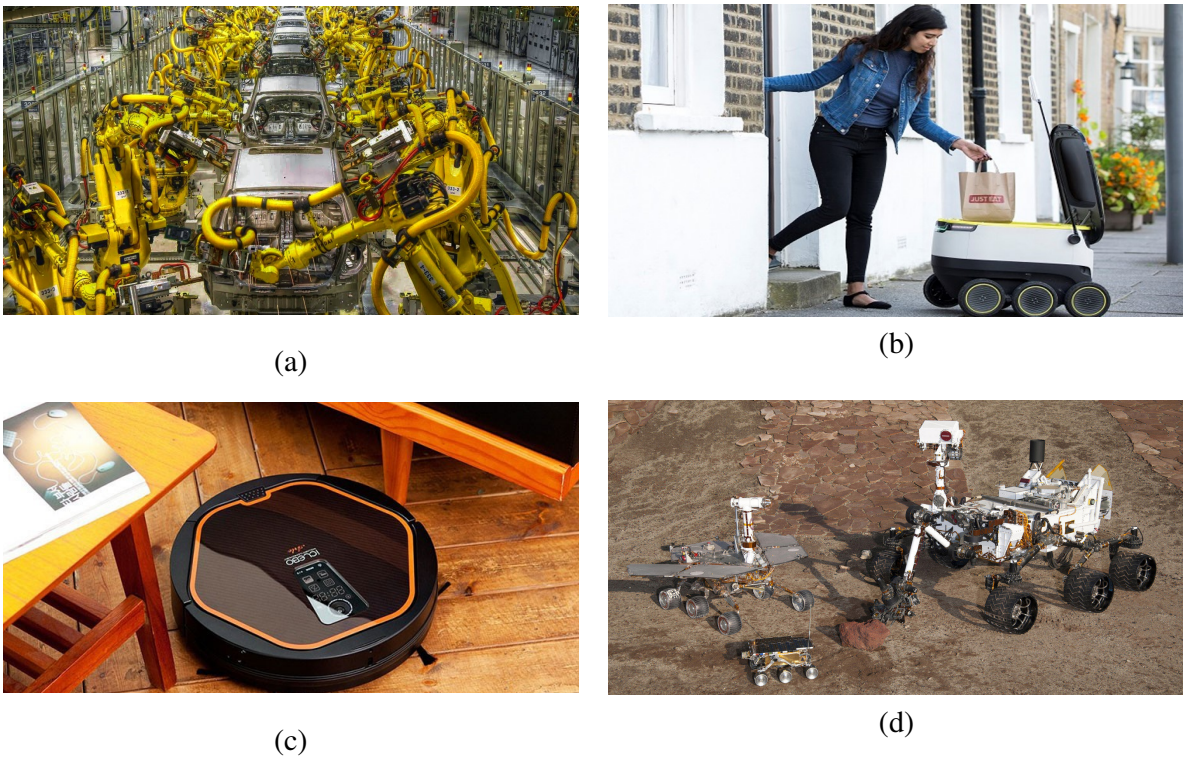
INTRODUÇÃO

> Não importa quão limitado possa parecer o começo: aquilo que é feito uma vez está feito para sempre.
>
> Henry David Thoreau, 1817-1862

Numa sociedade cada vez mais tecnológica, o desenvolvimento e a utilização de robôs têm despontado no mercado mundial. Eles já fazem parte da sociedade, estão presentes nas indústrias, nas ruas, nas casas e até mesmo no espaço, como exemplificado na Figura 1.1. De acordo com estatística fornecida pela IFR (*International Federation of Robotics*) de 2016, houve um crescimento de 15% nas vendas de robôs industriais em relação à 2014, sendo previsto que entre 2016 e 2019 mais de 1,4 milhões de novos robôs industriais serão instalados em fábricas ao redor do mundo [IFR 2016]. Em uma reportagem realizada em 2017 pela BBC [Mardell 2017], é comentada a possibilidade de 2017 marcar o início da era dos robôs, tal é a influência e possibilidades robóticas que se tem à disposição dos consumidores.

Pela ISO 8373 [ISO 2012], um robô pode ser definido como um mecanismo programável com dois ou mais eixos, movendo-se dentro de um ambiente e executando uma missão específica com um determinado grau de autonomia que depende da sua capacidade de tomar decisões para executar essa missão. Um robô totalmente autônomo realiza sua missão baseando-se, unicamente, nos dados de seus sensores e no que foi previamente programado/ensinado para ser executado. Já um robô parcialmente autônomo (semiautônomo) depende ou interage de alguma maneira com o ser humano.

Os robôs podem ser classificados ainda de acordo com sua funcionalidade como industrial, de serviço, médico, bélico ou espacial. Aqueles que serão abordados nesta dissertação são robôs de serviço, que, pela Comissão Econômica das Nações Unidas para a Europa (UNECE)



(a) (b)

(c) (d)

Figura 1.1 – Exemplos de robôs: (a) robôs encontrados na indústria automotiva, fonte: <http://robohub.org/wp-content/uploads/2014/08/Industrial-Robots-Hyundai-Heavy.jpg>, (b) robô entregador de encomendas, fonte: <https://www.fcsi.org/wp-content/uploads/2016/07/dronesjusteat-main-20160706063858504.jpg>, (c) robô iClebo aspirador de pó para uso doméstico, fonte: <https://iclebous.com/> e (d) robôs usados na exploração espacial (*Sojourner*, *Spirit* e *Curiosity*), fonte: <https://www.jpl.nasa.gov/spaceimages/details.php?id=PIA15277>.

e a IFR, são definidos, simplesmente, como sendo robôs que realizam tarefas úteis para seres humanos, relacionadas, principalmente, a trabalhos perigosos ou repetitivos ou difícil acesso ou ainda com o objetivo de auxiliar o ser humano, incluindo tarefas domésticas e excluindo as aplicações na área de automação industrial. Esse tipo de robô pode ser categorizado como: (i) robô de serviço profissional, se a finalidade for para o uso comercial, como os limpadores de ambientes de espaço públicos e os robôs entregadores de encomenda (Figura 1.1b); ou (ii) robô de serviço pessoal, que realiza uma tarefa doméstica, como o robô aspirador de pó (Figura 1.1c), ou sem fins comerciais, como uma cadeira de rodas inteligente (Figura 1.2) e robôs auxiliares de mobilidade pessoal.

Mais especificamente, são o foco desta dissertação alguns desses robôs de serviço que podem ser diretamente comandados por usuários, ou seja, onde o controle da missão é de responsabilidade do ser humano, estando o robô responsável por tarefas específicas, em especial, a segurança da missão, como é o caso das cadeiras de rodas inteligentes [Simpson, LoPresti e Cooper 2008], dos robôs que devem seguir as pessoas pelo ambiente [Müller *et al.* 2008] ou são tele-operados [Borenstein e Koren 1990] ou ainda daqueles usados na exploração espacial [Goldberg, Maimone e Matthies 2002]. Esses robôs de serviço semiautônomos se diferem dos demais robôs de serviço em pelo menos três aspectos: (i) eles não precisam conhecer

Figura 1.2 – Exemplo de cadeira de rodas inteligente, fonte: <https://smart.mit.edu/news-events/smart-fm-trials-self-driving-wheelchair>.

um mapa global do ambiente em que se deslocam; (ii) eles não precisam conhecer o objetivo ou alvo final; e (iii) eles não precisam se localizar no ambiente.

É intuito desta dissertação propor uma estratégia para a navegação desses robôs de serviço semiautônomos baseada apenas em informações locais obtidas pelos seus sensores e um planejador probabilístico que gera caminhos a serem seguidos localmente. A definição e o detalhamento do problema a ser resolvido nesta dissertação serão apresentados a seguir.

1.1 Definição do problema

O problema específico que esta dissertação se propõe solucionar pode ser formulado da seguinte forma:

Problema específico. *Possibilitar que um robô de serviço semiautônomo atuando em um ambiente desconhecido, dinâmico e interno siga um comando fornecido por um usuário de forma segura até que um novo comando seja recebido ou até que uma pré-determinada situação ou evento aconteça.*

O problema considera os robôs de serviço semiautônomos que tem como objetivo realizar tarefas não-industriais úteis aos seres humanos. Esses robôs possuem um determinado grau de autonomia, mas ainda continuam dependentes do ser humano. Neste trabalho, será considerada que a dependência desses robôs está, principalmente, ligada ao fato de que a missão a ser realizada é determinada pelo usuário. Nesse caso, o robô aguarda por comandos para a realização

de sub-tarefas pré-especificadas, como seguir em frente, virar à direita, virar à esquerda e parar. Cada tarefa definida pelo usuário tem uma validade de duração pré-fixada. Assim, o comando pode ser válido durante um determinado intervalo de tempo ou até que outro comando seja fornecido ou, ainda, válido até que seja encontrada uma situação pré-determinada, como, por exemplo, o fim de um corredor. Durante a validade do comando, o robô navega autonomamente, conforme pré-estabelecido pelo usuário.

Um segundo elemento importante do problema é garantir uma navegação segura, ou seja, após ser definida a missão do robô, ele não deve colidir com nenhum obstáculo. Nessa etapa, ele está completamente autônomo, sem a intervenção do usuário.

Um terceiro ponto relevante é a complexidade do ambiente a ser considerado, sendo um ambiente interno, desconhecido e dinâmico. Para ilustrar uma situação comum, pode-se imaginar uma situação em que se entra em um prédio pela primeira vez e tem-se que ir até uma determinada sala. Nessa situação, não se tem informações *a priori* do local (ambiente desconhecido), haveria pessoas se movimentando (ambiente dinâmico) e seria um ambiente interno no qual não chegaria, por exemplo, um bom sinal para um sensor de geoposicionamento (GPS). Assim, esse problema é resolvido orientando-se pelas indicações fornecidas para chegar até a sala, como: siga o corredor até o final, vire à direita e entre na terceira porta. Da mesma forma, um robô semiautônomo sem precisar ter a sua localização global e o conhecimento do mapa do ambiente deve ser capaz de navegar por ele, seguindo uma sequência de comandos fornecidos por um usuário.

Na próxima seção será apresentada a principal motivação para o desenvolvimento deste trabalho.

1.2 Motivação

Diversos setores já se beneficiam dos robôs de serviço. Um setor cada vez mais importante é aquele que visa a inclusão social de idosos e de pessoas portadoras de alguma necessidade especial. No Brasil, segundo o IBGE [G1 2014], 24 % da população possui algum tipo de deficiência física e 13 % são idosos. Dessa forma, uma significante parcela da população poderia ter à sua disposição robôs que possibilitem melhorar a sua qualidade de vida e auxiliá-los em suas tarefas diárias, dando maior autonomia e mobilidade.

Um exemplo disso é o desenvolvimento de cadeiras de rodas inteligentes ou, em inglês, *smart wheelchair*, que são cadeiras de rodas motorizadas dotadas das funcionalidades existentes nos robôs, como localização e desvio de obstáculos. Essas cadeiras de rodas podem ser utilizadas tanto por usuários que possuem dificuldade ou são impossibilitados de operar as cadeiras de rodas tradicionais, manuais ou motorizadas. Também são utilizadas por aqueles que necessitam de frequente auxílio de terceiros ou que gastam muito tempo para vencer os desafios de um percurso, enfrentando manobras complicadas para entrar e sair de alguns lugares, desviar de obstáculos,

dentre outras dificuldades. Projetos da Panasonic e da SMART (*The Singapore-MIT Alliance for Research and Technology*), que ganharam, recentemente, destaque na mídia, demonstram como é possível contribuir na qualidade de vida dos cadeirantes durante sua locomoção dentro de um hospital ou num aeroporto [Scudellari 2017].

Mesmo que as pesquisas no desenvolvimento de cadeiras de rodas inteligentes tenham começado no final da década de 70 [Clark e Roemer 1977], ainda é um grande desafio a utilização dessas cadeiras no dia a dia das pessoas, dada a complexidade e a diversidade de situações a serem tratadas pelo seu sistema de navegação, tornando-as ainda pouco viáveis. Assim, a estratégia de navegação proposta nesta dissertação pode contribuir para o desenvolvimento dessas cadeiras, permitindo a navegação em ambientes desconhecidos de forma planejada e segura, a partir de comandos simples fornecidos pelo usuário.

1.3 Objetivos

O objetivo geral do presente trabalho é fornecer uma estratégia para a navegação de robôs de serviço semiautônomos, tendo como objetivos específicos no seu desenvolvimento:

- A aplicação de técnicas existentes de planejamento de movimento;
- A integração de técnicas de localização, mapeamento e planejamento para a composição de uma estratégia de navegação para o robô;
- A validação da estratégia de navegação para um ambiente desconhecido, interno e dinâmico, por meio de uma plataforma de simulação e com robôs reais;

1.4 Contribuições e Relevância

A principal contribuição deste trabalho é o desenvolvimento de uma estratégia de navegação aplicada, principalmente, aos robôs de serviço semiautônomos. Essa estratégia diferencia-se da maioria das estratégias já desenvolvidas até então por não requerer o mapa do ambiente nem a localização global do robô nesse mapa, baseando-se apenas nos dados de um sensor de profundidade, por exemplo, um *laser* e no sistema de odometria do robô. A partir desses dados, um planejador de movimento fornece caminhos a serem seguidos localmente, garantindo que o robô desvie de obstáculos existentes no ambiente e que cumpra a tarefa estabelecida por um usuário, de forma planejada.

Essa tarefa é codificada por meio de um campo vetorial artificial e incorporada diretamente ao planejador de movimento. Esse tipo de solução é bem atual [Pereira, Choudhury e Scherer 2016] [Ko, Kim e Park 2014] e, com o intuito de ser mais eficiente computacionalmente, este trabalho contribui com uma heurística simples aplicada ao funcional de custo de um planejador probabilístico.

O desenvolvimento desta dissertação resultou na publicação de um artigo [Freitas, Passos e Pereira 2017] , que detalha a implementação da estratégia de navegação para o desvio de obstáculos aplicada aos robôs de serviço semiautônomos.

1.5 Organização da dissertação

O Capítulo 1 apresentou a introdução deste trabalho, o problema específico que se pretende solucionar, as motivações envolvidas, os objetivos a serem alcançados e as principais contribuições. O Capítulo 2 será dedicado à revisão bibliográfica, apresentando um elenco de soluções e conceitos no desenvolvimento de robôs de serviço. Em seguida, no Capítulo 3, será apresentada a metodologia para a elaboração da estratégia de navegação proposta para robôs de serviço semiautônomos, detalhando os módulos que compõem essa estratégia e analisando-a. Exposta a metodologia, o Capítulo 4 tem como objetivo apresentar o ambiente escolhido para realizar uma série de experimentos, descrevendo também como essa metodologia foi implementada e finalizando com os resultados obtidos tanto via simulação quanto utilizando robôs reais. Por fim, no Capítulo 5, serão apresentadas as considerações finais e as propostas para trabalhos futuros.

CAPÍTULO

2

REVISÃO BIBLIOGRÁFICA

"Centenas de vezes por dia eu me lembro que minha vida interna e externa são baseadas nos trabalhos de outros homens, vivos e mortos, e que preciso me esforçar para poder dar na mesma medida em que recebi e que ainda recebo."

Albert Einstein, 1879-1955

Neste capítulo, é apresentada uma revisão de soluções e de conceitos encontrados na literatura para o desenvolvimento de robôs de serviço. Para tal, serão apresentadas as principais arquiteturas de *hardware* e *software* (Seção 2.1), os principais sensores (Seção 2.2) que fornecem informações para os módulos que compõem o sistema inteligente dos robôs de serviço: localização, mapeamento (Seção 2.3) e planejamento de movimento (Seção 2.4). Por último, na Seção 2.5, são expostas estratégias de navegação.

2.1 Arquitetura de *hardware* e *software*

Arquitetura em robótica pode ser entendida, segundo Grassi e Okamoto [2014a], como uma descrição da estrutura e da organização de *hardware* e de *software* necessária para o desenvolvimento do sistema de controle inteligente do robô, devendo conter a maneira como os módulos que compõem o sistema se comunicam e se interagem. Analisando a maneira com que o robô determina suas ações, pode-se classificar as arquiteturas em (i) arquiteturas reativas, (ii) arquiteturas deliberativas e (iii) arquiteturas híbridas.

As arquiteturas reativas foram introduzidas por Brooks [1986] e buscam permitir uma resposta mais rápida às mudanças do ambiente. Dessa forma, o robô reage instantaneamente

à presença de obstáculos recém detectados sem se preocupar com as consequências de sua reação. As estratégias reativas tem como vantagens serem muito eficientes e não necessitarem de localização global nem de modelos complexos para os obstáculos e para o ambiente [Arkin 1998]. Estes benefícios facilitam a sua utilização em robôs semiautônomos [Chik *et al.* 2016] e [Baklouti, Amor e Jallouli 2017]. No entanto, uma desvantagem é que essas arquiteturas não levam em consideração um estado futuro, podendo levar o robô a condições indesejadas, fazendo-o parar constantemente ou seguir trajetórias muito longas. Outra desvantagem está no fato que o desenvolvimento de um sistema reativo torna-se mais complicado à medida que aumenta a complexidade da missão a ser executada pelo robô e a diversidade de situações que ele deve enfrentar no ambiente para executá-la.

Ao contrário, as arquiteturas deliberativas, também presentes desde o início da robótica [Khatib 1986], são aquelas que se baseiam em um modelo do mundo, ou seja, há o conhecimento *a priori* do ambiente, por meio de mapas, e com base na localização do robô determina-se como a missão será realizada. Dessa forma, os obstáculos estáticos considerados no modelo serão evitados no cumprimento de sua missão. Nesse tipo de arquitetura, é possível evitar obstáculos dinâmicos que podem aparecer no ambiente por meio de um replanejamento da missão ao longo de sua execução. Esse replanejamento da missão é baseado na atualização do modelo do mundo anteriormente fornecido, podendo gerar trajetórias ótimas e seguras para o robô. Um trabalho recente e interessante na área é [Yang, Lim e Yoon 2016], onde o planejamento da trajetória do robô em um ambiente com pessoas considera explicitamente as incertezas associadas à velocidade e à posição dessas pessoas. Em síntese, nesse tipo de arquitetura, para o planejamento do caminho e para o seu seguimento, há a necessidade de se ter o mapa do ambiente antecipadamente e, principalmente, da localização do robô nesse mapa [Choset *et al.* 2005]. Esses pré-requisitos nem sempre são simples de serem obtidos por alguns robôs semiautônomos, como as cadeiras de rodas se locomovendo em ambientes internos desconhecidos [Baklouti, Amor e Jallouli 2017].

As arquiteturas hibridas, em contrapartida, combinam características tanto de arquiteturas reativas quanto de arquiteturas deliberativas, ou seja, o grau de deliberação e reação encontrados nessas arquiteturas variam, como ilustrado na Figura 2.1. Normalmente, utilizam a deliberação para o planejamento das ações do robô com base em um mapa e a reatividade para a execução da ação planejada, levando em consideração as mudanças do ambiente [Grassi e Okamoto 2014].

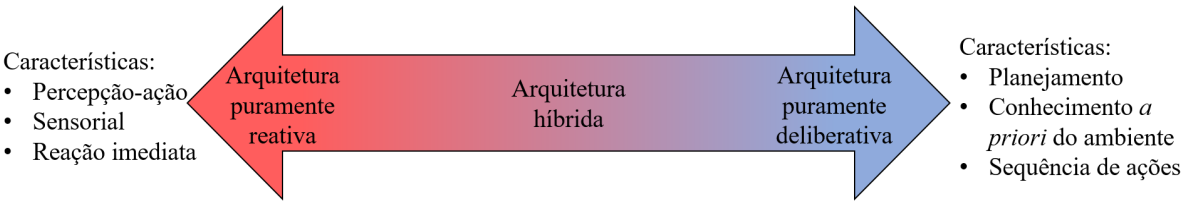


Figura 2.1 – Classificação quanto à tomada de decisão, adaptada de [Grassi e Okamoto 2014].

Um exemplo de arquitetura híbrida pode ser encontrado em [Goller *et al.* 2010]. Esse artigo apresenta um robô de serviço que tem como objetivo locomover-se em um supermercado, desviando dos obstáculos que podem surgir nesse ambiente. A arquitetura apresentada está dividida em três camadas hierárquicas. A primeira é responsável pela estratégia (camada deliberativa): um mapa prévio é conhecido e com base nele é feito um planejamento global utilizando o algoritmo A*. A segunda camada é responsável por evitar obstáculos dinâmicos de forma preditiva, com base no movimento dos obstáculos. A última camada é reativa, fornecendo comandos para repelir o robô de um obstáculo que não foi desviado pelo planejador. Outros exemplos similares de arquitetura híbrida podem ser vistos ainda em [Araujo, Caminhas e Pereira 2015], [Nakhaeinia *et al.* 2015] e [Lopes *et al.* 2016].

Para a integração via *software* de todos os módulos que constituem a arquitetura escolhida para o desenvolvimento de um robô de serviço, conforme é mencionado em [Renawi, Jaradat e Abdel-Hafez 2017], a principal plataforma que tem sido utilizada é o ROS[1] (*Robot Operating System*) [Quigley *et al.* 2009]. Essa plataforma permite a integração e comunicação com as várias partes do sistema e possibilita uma abstração entre o *hardware* físico e o virtual, possibilitando que os módulos desenvolvidos sejam testados tanto na simulação quanto no ambiente real.

Neste trabalho, como apresentado no Capítulo 3, será proposto a utilização de uma arquitetura híbrida e no Capítulo 4 é detalhado o desenvolvimento do *software* que foi realizado utilizando o ROS.

2.2 Sensores

Nesta seção serão apresentados os principais sensores presentes nos robôs de serviço que tem como objetivo fornecer informações sobre si e sobre o ambiente para os módulos do sistema inteligente do robô.

Os sensores podem ser classificados quanto ao método de medição, ativo ou passivo, e quanto à origem de suas medidas, proprioceptivos ou exteroceptivos [Siegwart, Nourbakhsh e Scaramuzza 2011]. Um sensor ativo é aquele que, para realizar uma medição, emite energia e mede a reação do ambiente com relação à essa emissão, como, por exemplo, o sonar. Já um sensor passivo mede a energia do ambiente que ele recebe, como, por exemplo, uma câmera. Os sensores proprioceptivos são aqueles que realizam medidas de parâmetros do sistema, como a velocidade das rodas, a carga da bateria e a sua aceleração. Por outro lado, os exteroceptivos são aqueles que obtêm informações sobre o ambiente no qual o robô está inserido, tais como os sonares, os *lasers* e o GPS.

Na Tabela 2.1 pode-se ver uma lista de classificação dos principais sensores utilizados

[1] O ROS foi desenvolvido originalmente em 2007 pela *Stanford Artificial Intelligence Laboratory* (SAIL) com apoio da *Stanford AI Robot*. A partir de 2008, o seu desenvolvimento passou a ser feito no instituto de pesquisa em robótica *Willow Garage*, contando com a colaboração de mais de vinte instituições [Fernandez *et al.* 2015].

nos robôs de serviço e a faixa de preço para adquiri-los. Pode-se perceber que há uma variação grande entre os preços desses sensores, relacionado, principalmente, com a precisão obtida com cada um deles. A título de exemplo, as câmeras RGB têm um custo menor que os *lasers* , porém o sistema de localização que as utiliza, normalmente, são mais imprecisos [Zhiwei *et al.* 2013]. Dessa forma, no desenvolvimento de robôs de serviço, principalmente, mais acessíveis à população, é necessário levar em consideração o custo-benefício desejado.

Tabela 2.1 – Classificação dos principais sensores utilizados nos robôs de serviço e a faixa de preço para adquiri-los.

Sensor	**Classificação**	**Faixa de preço (R$)**
Câmera CCD/CMOS	Exteroceptivo Passivo	100,00 - 300,00
GPS	Exteroceptivo Ativo	50,00 - 300,00
IMU	Proprioceptivo Passivo	300,00 - 700,00
Laser LIDAR/LADAR	Exteroceptivo Ativo	2000,00 - 10.000,00
Odômetro ótico	Proprioceptivo Ativo	150,00 - 300,00
Sonar	Exteroceptivo Ativo	100,00 - 150,00
Sensor RGB-D	Exteroceptivo Ativo	300,00 - 500,00

Por sua vez, a Tabela 2.2 apresenta um resumo do conjunto de sensores presentes no módulo de percepção de vários robôs de serviço. Note que o odômetro (no inglês *wheel encoder*) está presente em todos os robôs, sendo um elemento básico para a medição de velocidade e posição[2].

Nos robôs mais recentes, observa-se a utilização do sensor RGB-D, tal como o *Kinect* [Ellaithy, Huang e Yeh 2012], capaz de obter diversas informações do ambiente, como aspectos visuais, texturas, além da distância em relação ao objeto, possibilitando, por exemplo, uma reconstrução 3D do ambiente. Pela sua atual popularidade, o valor desses sensores, como visto na Tabela 2.1, chega a ser mais de vinte vezes menor que um *laser*. Porém, apresentam desvantagens como o curto alcance, geralmente, cinco metros, e variações na medição dependendo da luminosidade do ambiente. Por isso, há uma tendência em combinar o sensor RGB-D com o *laser*.

Pelo seu baixo custo, as câmeras CDD/CMOS são encontradas em alguns robôs e são, comumente, utilizadas como interface com o usuário. Como esperado, o GPS não é utilizado em ambientes internos, como é o caso da maioria dos robôs pesquisados. Sua utilização é bastante comum quando se trata de ambientes externos, provendo uma localização absoluta do robô. Também a IMU (Unidade de Medição Inercial, do inglês *Inertial Measurement Unit*) tem sido pouco utilizada nos robôs de serviços, normalmente, a sua utilização tem como objetivo melhorar a medição da orientação do robô.

[2]O odômetro conta o número de voltas dadas por uma roda e com base nessa medição estima a posição e a velocidade do robô. Um dos grandes problemas desse sensor é que para determinar a posição ele integra a informação de velocidade, ou seja, pequenos erros nas medições, devido às incertezas ou mesmo à derrapagem ou deslizamento das rodas, se acumulam com o passar do tempo.

Tabela 2.2 – Principais sensores presentes no módulo de percepção dos robôs de serviço.

	Câmera	**GPS**	**IMU**	**Laser**	**Odômetro**	**Sensor RGB-D**	**Sonar**
Rhino [Burgard *et al.* 1999]				X	X		X
Minerva [Thrun *et al.* 1999]	X			X	X		X
NavChair [Levine *et al.* 1999]					X		X
Guidecane [Shoval, Ulrich e Borenstein 2003]					X		X
RoboX [Siegwart *et al.* 2003]	X			X	X		
PR2 Marder [Marder-Eppstein *et al.* 2010]	X		X	X	X		
GuiaBot [Dayoub, Morris e Corke 2015]	X		X	X	X	X	
NimbRo [Stückler *et al.* 2015]	X		X	X	X	X	
MARIA [Araujo, Caminhas e Pereira 2015]				X	X	X	
MOBOT [Fotinea *et al.* 2016]				X	X	X	
B21r [Bellarbi *et al.* 2016]				X	X	X	
RobChair [Lopes *et al.* 2016]				X	X		
SWS [Schwesinger *et al.* 2016]		X	X	X	X	X	
Smart Wheelchair [Baklouti, Amor e Jallouli 2017]					X	X	

2.3 Localização e Mapeamento

Admitindo-se que o robô possui sensores proprioceptivos e exteroceptivos, como visto na Seção 2.2, o robô é capaz de obter informações sobre si e do ambiente à sua volta. Por meio dessas informações, o módulo de localização permite conhecer onde o robô se encontra no espaço e o módulo de mapeamento fornece uma representação espacial do ambiente [Boal, Sánchez-Miralles e Arranz 2013]. Esses dois módulos são assuntos apresentados nesta seção.

Técnicas probabilísticas, baseadas no Filtro de *Bayes*, são amplamente adotadas para resolver o problema de localização de um robô enquanto ele se movimenta em um ambiente. Essas técnicas procuram representar a crença sobre a localização do robô por meio de uma função de densidade de probabilidade (PDF), $p(x_t|z_{1:t}, u_{1:t})$, baseando-se em todas ações de controle, $u_{1:t}$, e medições, $z_{1:t}$, realizadas até o instante t [Thrun, Burgard e Fox 2005]. A PDF pode ser obtida recursivamente por meio do Filtro de Bayes, apresentado no Algoritmo 1.

Algoritmo 1: Filtro de Bayes.

Entrada: $p(x_{t-1})$ = crença *a priori* sobre a localização;
u_t = comando fornecido no instante t;
z_t= medições realizadas no instante t;
Resultado: $p(x_t|z_{1:t}, u_{1:t})$ = crença sobre a localização no instante t.

1 **para todo** x_t **faça**
2 $\quad \bar{p}(x_t) = \int p(x_t|u_t, x_{t-1}) \cdot p(x_{t-1}) dx$ `// predição`
3 $\quad p(x_t|z_{1:t}, u_{1:t}) = \eta \cdot p(z_t|x_t) \cdot \bar{p}(x_t)$ `// atualização`
4 **fim**
5 **retorna** $p(x_t|z_{1:t}, u_{1:t})$

A solução para a integral do filtro de Bayes é difícil de ser obtida. Assim, soluções baseados no filtro de Kalman buscam representar a crença sobre a localização por meio de uma distribuição gaussiana [Thrun, Burgard e Fox 2005]. Pode-se dizer que essas técnicas rastreiam,

ao longo do movimento do robô, a estimativa inicial fornecida da localização do robô, conforme pode ser observado em [Marder-Eppstein *et al.* 2010] e [Siegwart *et al.* 2003]. Por sua vez, outra maneira de representar a crença sobre a localização do robô é por meio de um conjunto de amostras ponderadas, como as soluções baseadas no filtro de Partículas [Thrun 2002]. Com essa representação, consegue-se solucionar o problema de localização quando não é conhecida a posição inicial do robô ou quando o robô é sequestrado[3]. Exemplo da aplicação desse tipo de filtro em robôs de serviço pode ser visto em [Thrun *et al.* 1999] e [Cen *et al.* 2008].

No que diz respeito ao mapeamento, o foco é gerar uma representação do ambiente, na forma de um mapa. Dois tipos de mapas destacam-se nos trabalhos com robôs de serviço: (i) mapas métricos que representam o ambiente como um conjunto de coordenadas para cada obstáculo e (ii) mapas topológicos que modelam o ambiente como um grafo [4]. A Grade de Ocupação [Elfes 1989], apresentado na Figura 2.2, é um exemplo de mapa métrico que pode ser encontrado em vários trabalhos [Burgard *et al.* 1999], [Thrun *et al.* 1999], [Marder-Eppstein *et al.* 2010] e [Nakhaeinia *et al.* 2015]. Nele o ambiente é dividido em células, formando uma grade (*grid*), sendo que cada célula está associada a um número $p(i_x, i_y) \in [0, 1]$, que indica a probabilidade dela estar ocupada. Por sua vez, o mapa topológico, ilustrado na Figura 2.3, tem a vantagem de ter baixo custo computacional e não exigir a posição exata do robô, como é utilizado em [Araujo, Caminhas e Pereira 2015]. No entanto, os mapas topológicos são mais difíceis de serem construídos e mantidos para ambientes grandes e são susceptíveis a erros quando o ambiente possui lugares similares [Souza *et al.* 2014].

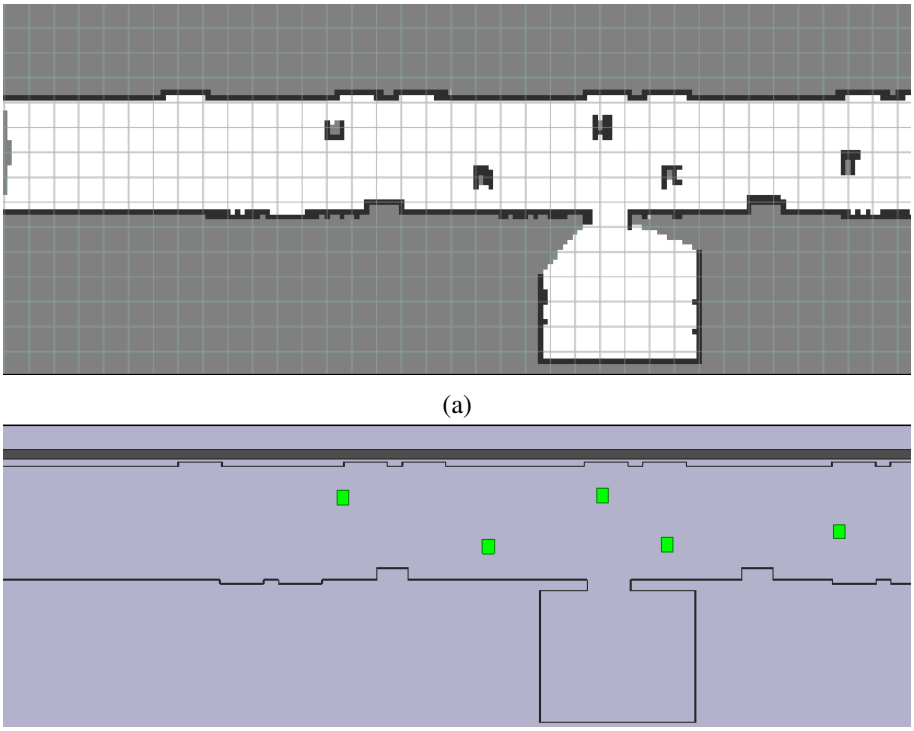
Devido a interdependência entre a obtenção da localização do robô e o mapeamento, é usual tratá-los como um problema único de localização e mapeamento simultâneos (SLAM, do inglês *Simultaneous Localization and Mapping*) [Souza *et al.* 2014]. As técnicas para resolver esse problema são constituídas de pelo menos quatro partes: (i) extração de características do ambiente, (ii) associação dos dados, (iii) estimação e atualização da localização e (iv) atualização do mapa. Diferentes soluções podem ser adotadas para cada uma dessas partes, o que possibilitou o desenvolvimento de uma variedade de algoritmos, tais como aqueles baseados no EKF-SLAM [Xie *et al.* 2017], baseados em grade de ocupação [Dib, Beaufort e Charpillet 2014] e em Filtro de Partículas [Kohlbrecher *et al.* 2011][5] e [Grisetti, Stachniss e Burgard 2007][6].

[3] O problema relacionado quando uma força externa transporta o robô de uma posição para outra posição desconhecida é chamado de problema do robô sequestrado.

[4] Pode-se entender um grafo como um par de conjuntos: um conjunto de vértices e um conjunto arestas. Cada aresta é um par ordenado de vértices, sendo que o primeiro vértice do par é a ponta inicial da aresta e o segundo é a ponta final.

[5] Pacote distribuído no ROS como hector_slam [ROS 2017].

[6] Pacote distribuído no ROS como GMapping [ROS 2017].



(a)

(b)

Figura 2.2 – (a) Exemplo de grade de ocupação gerada a partir dos dados de um *laser* de um robô em um corredor com obstáculos, mostrado em (b). As células em cor branca são os espaços considerados livres, em preto os espaços ocupados pelos obstáculos e em cinza estão representadas as células que não foram vistas pelo *laser*.

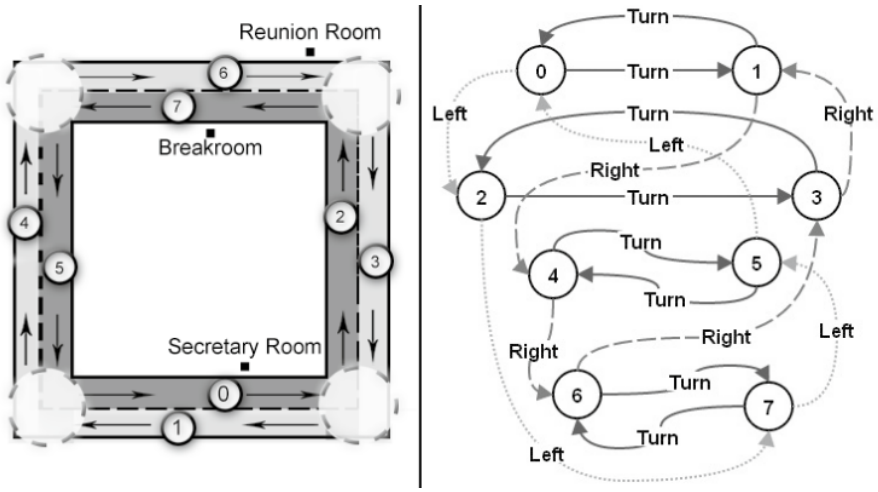


Figura 2.3 – Exemplo de mapa topológico utilizado em [Araujo, Caminhas e Pereira 2015].

Nesta última década, vê-se o esforço de vários trabalhos em possibilitar que o robô se localize e faça o mapeamento do ambiente simultaneamente, utilizando apenas sensores RGB-D ou câmeras CCD/CMOS em substituição aos *lasers* [Fuentes-Pacheco, Ruiz-Ascencio e Rendón-Mancha 2012], tendo em vista que esses sensores tem um custo menor que o do *laser* e o número de informações que se pode extrair é maior, como mencionado na Seção 2.2.

2.4 Planejamento de movimento

Como apresentado na Seção 2.3, a localização e o mapeamento são módulos essenciais ao robô. Nesta seção, será apresentado o terceiro módulo imprescindível: o planejamento de movimento. Esse módulo tem, tipicamente, o objetivo de resolver o seguinte problema:

Problema 1. *Encontre um caminho livre de colisões* $\tau(s)$, $s \in [0,1]$, *onde* $\tau(0)$ *representa a configuração inicial do robô e* $\tau(1)$ *representa o alvo a ser alcançado [Choset et al. 2005].*

Repare que a solução desse problema, fornecida por um planejador, é um caminho, ou seja, é um conjunto de pontos no espaço que garante que o robô ao locomover-se não irá colidir com nenhum obstáculo. Diferentemente, outra abordagem, conhecida como planejamento de trajetória, fornece como solução um conjunto de ações de controle que satisfazem, além da ausência de colisões, restrições temporais pré-especificadas.

Conforme Lynch e Park [2017], pode-se elencar ao menos cinco propriedades dos planejadores que são úteis para distingui-los entre os vários existentes. São elas:

1. Completude: o planejador pode ser dito completo se assegurar encontrar uma solução caso ela exista; incompleto se não garantir uma solução; e probabilisticamente completo, significando que, se existe uma solução, a probabilidade de encontrá-la converge para 1 (um) quando o número de amostras em um planejador probabilístico tende para infinito.

2. Optimalidade: Dado um determinado custo, um planejador é ótimo quando encontra uma solução que minimiza esse custo.

3. Temporalidade: o planejador pode ser dito *Anytime* quando esse procura por soluções melhores após uma primeira solução encontrada, podendo ser interrompido a qualquer momento, como por exemplo, após um limite de tempo especificado. Dessa forma, esse tipo de planejador retorna a melhor solução até aquele instante.

4. Complexidade Computacional: a complexidade computacional está relacionada com o tempo de processamento gasto pelo planejador. Um menor custo computacional é interessante para os robôs de serviço, principalmente, para viabilizar um planejamento em tempo real.

5. Globalidade: o planejador é dito global quando utiliza a informação completa do espaço de tarefa[7] para retornar uma solução e dito planejador local quando ele utiliza apenas um subespaço. É comum encontrar soluções em que há um planejador global que utiliza um mapa global estático combinado com um planejador local que lida com o ambiente dinâmico.

[7] O espaço de tarefa pode ser definido como o ambiente no qual o robô é capaz de atuar [Lynch e Park 2017], no caso deste trabalho, o espaço $\mathbb{R}^2$.

Há um grande número de soluções para o problema de planejamento de movimento, que podem ser agrupadas com base nos métodos mais aplicados aos robôs móveis da seguinte forma: mapa de rotas (do inglês *Roadmap*), métodos baseados em campos potenciais, métodos baseados em desvio de obstáculo e métodos probabilísticos.

Os métodos *Roadmap* procuram criar rotas unidimensionais no espaço livre, que podem ser simplesmente retas, parábolas ou uma combinação de ambas, construindo caminhos onde o robô pode navegar sem colidir com os obstáculos. Ao armazenar as informações dessas rotas em um grafo, esses métodos utilizam algum algoritmo de busca em grafo, como A* e Dijkstra, para encontrar um caminho que vai de um ponto inicial até o alvo [Lynch e Park 2017]. Por exemplo, tem-se as técnicas baseadas no *Generalized Voronoi Diagram* (GVD) [Takahashi e Schilling 1989], como a encontrada mais recentemente em [Wang *et al.* 2015] que é aplicada para robôs não-holonômicos em corredores.

Em contrapartida, os métodos baseados em campos potenciais tem em sua essência a criação de um campo potencial artificial que direciona o robô ao alvo e o repele dos obstáculos. Dessa forma, é como se houvesse uma força atrativa e uma repulsiva, cuja resultante é aplicada ao robô. Esses métodos desempenham um papel importante na navegação de robôs móveis pela sua eficiência no desvio de obstáculos em tempo real, não necessitando de um mapa global e tendo baixo custo computacional [Adorno e Borges 2014]. Assim, desde o método APF (*Artificial Potential Field*) [Khatib 1986], outras abordagens ou extensões foram desenvolvidas em [Feder e Slotine 1997] e [Montano e Asensio 1997].

Outros métodos usados nos robôs de serviço são baseados em desvio de obstáculo, como o DWA (*Dynamic Window Approach*) [Fox, Burgard e Thrun 1997], o *Trajectory Rollout* [Gerkey e Konolige 2008], o DVZ (*Deformable Virtual Zone*) [Chavez *et al.* 2016] para o espaço 3D e os métodos baseados no VFH (*Vector Field Histogram*) [Borenstein e Koren 1990], [Ulrich e Borenstein 2000], [Qu *et al.* 2015] e [Li *et al.* 2016]. Esses métodos, normalmente, são combinados com um planejador global e geram informações seguras para o controle de navegação do robô, corrigindo o caminho fornecido por um planejador global a partir dos dados dos sensores.

Por fim, os métodos probabilísticos buscam encontrar, a partir da seleção de pontos amostrados no espaço livre do robô, um caminho que ligue o ponto de partida ao alvo. Em [Elbanhawi e Simic 2014] e [Karaman *et al.* 2011] encontra-se uma revisão dos vários planejadores probabilísticos, tais como o PRM (*Probabilistic Roadmap*) e os vários algoritmos baseados no RRT (*Rapidly-exploring Random Trees*), como o RRT* (*Optimal Rapidly Exploring Random Trees*) que é assintoticamente ótimo e probabilisticamente completo. Recentemente, encontra-se tentativas híbridas de acrescentar informações de um campo vetorial aos planejadores probabilísticos [Ko, Kim e Park 2014] e [Pereira, Choudhury e Scherer 2016]. Como será apresentado no Capítulo 3, no presente trabalho utilizou-se um planejador RRT* *anytime* com um funcional de custo baseado em um campo vetorial para gerar caminhos localmente.

2.5 Navegação

A combinação dos métodos apresentados anteriormente para a localização, mapeamento e planejamento de movimento define o que se chama de navegação de um robô. Há diversas maneiras de combinar esses métodos, definindo-se, portanto, uma determinada estratégia de navegação que execute a missão designada ao robô e que atenda às restrições impostas tanto pelo ambiente quanto pelo robô [Alves *et al.* 2011].

Dessa forma, as estratégias de navegação que buscam fazer com que um robô móvel de serviço possa locomover-se em um ambiente, seja ele conhecido, parcialmente conhecido ou desconhecido, foram agrupadas em três grupos: (i) Navegação local reativa: utilizam apenas métodos locais baseados em desvio de obstáculo que possibilitam o robô reagir às situações do ambiente, a partir das informações dos dados dos sensores, ou seja, não há um planejamento de movimento nem há a necessidade de um mapa global; (ii) Navegação global com o conhecimento *a priori* do mapa do ambiente: com base no mapa fornecido, utilizam, geralmente, planejadores globais, que determinam um caminho a ser seguido para o cumprimento da missão do robô, combinados com métodos locais de desvio de obstáculos; e (iii) Navegação local com mapeamento: a partir de um mapa gerado em tempo de execução, um planejador fornece caminhos a serem seguidos localmente que são válidos no campo de visão dos sensores. O mapeamento pode ser sem memória, se utilizar diretamente os dados dos sensores, ou com memória se durante a navegação o mapa é construído, usando, por exemplo, técnicas de SLAM.

O primeiro tipo de estratégia de navegação, ilustrado na Figura 2.4(a), é encontrado, principalmente, em robôs de serviço semiautônomos que são comandados por seres humanos. Nesse caso, é o usuário que define o que ele deseja fazer e a estratégia de navegação do robô foca-se no problema de desviar dos obstáculos localmente. Como exemplo desse tipo de navegação, pode-se citar o trabalho de Levine *et al.* [1999] que apresenta o desenvolvimento de uma cadeira de rodas inteligente capaz de locomover-se em ambientes internos. O cadeirante define as direções, utilizando um *joystick*, e, por meio de um método que combina campos potenciais com o VFH (*Vector Field Histogram*) e baseando-se nos dados de doze sonares, é fornecido os comandos necessários para o sistema de controle de velocidade da cadeira de rodas. Mais recente, o trabalho de Baklouti, Amor e Jallouli [2017], também para o desenvolvimento de uma cadeira de rodas inteligente, propõe um controle de desvio de obstáculos composto pela união de um controlador baseado em lógica fuzzy, que determina velocidades e acelerações necessárias para alcançar o alvo, e um método baseado no DVZ (*Deformable Virtual Zone*), que busca desviar dos possíveis obstáculos.

Já a navegação global com o conhecimento *a priori* de um mapa, ilustrada na Figura 2.4(b), é a mais utilizada, pois a maioria dos trabalhos assumem que o ambiente em que o robô será inserido é pré-definido, sendo possível obter um mapa prévio desse ambiente. Exemplo desse tipo de estratégia pode ser encontrado em robôs que navegam em museus [Burgard *et al.* 1999] [Thrun *et al.* 1999], em escritórios [Marder-Eppstein *et al.* 2010] e em prédios [Araujo,

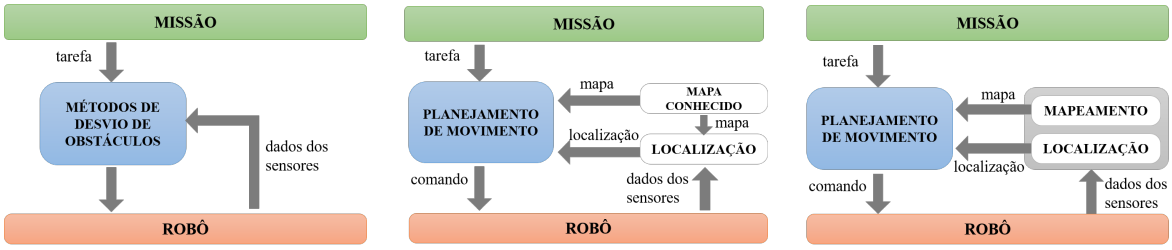


Figura 2.4 – Três estratégias de navegação: (a) Navegação local reativa. (b) Navegação global com conhecimento do mapa *a priori*. (c) Navegação local com mapeamento.

Caminhas e Pereira 2015]. Para a obtenção do mapa desses locais, alguns trabalhos utilizam, em uma etapa anterior à navegação, as técnicas de SLAM [Dayoub, Morris e Corke 2015] [Schwesinger *et al.* 2016]. Em [Araujo, Caminhas e Pereira 2015], o robô pode ser usado como entregador de encomendas, para tal, o ambiente foi representado por um mapa topológico e com base na missão especificada pode, por exemplo, ir de uma sala a outra. Um planejador global faz uma busca no grafo obtido pelo mapa topológico e determina uma sequência de passos que o robô deve seguir. Localmente, um controlador, baseado em um campo vetorial e nos dados dos sensores, determina os devidos comandos de velocidade para o robô, possibilitando-o chegar ao alvo sem se colidir.

Por último, a navegação local com mapeamento é encontrada, principalmente, em robôs exploradores. Stückler *et al.* [2015] propõem um planejador global para definir as direções a serem seguidas pelo robô explorador espacial NimbRo e um planejador Dijkstra que fornece caminhos a serem seguidos localmente pelo robô. Esse planejador baseia-se em uma grade de ocupação local e a cada intervalo de tempo replaneja esse caminho. Para fornecer os devidos comandos de velocidade ao robô é utilizado um método baseado em desvio de obstáculo, *Trajectory Rollout*, que leva em consideração uma janela de observação e a dinâmica do robô.

Uma comparação qualitativa e generalizada entre esses tipos de estratégias de navegação é apresentada na Tabela 2.3. Observe que nem todas estratégias garantem todos os requisitos desejados neste trabalho: desvio de obstáculos em ambientes dinâmicos, suavidade do movimento do robô e utilização de pouco recursos computacionais. Uma comparação mais detalhada com relação à navegação local com mapeamento com e sem memória pode ser encontrada no Capítulo 4.

No próximo capítulo será apresentada a metodologia adotada neste trabalho que baseia-se na navegação local com mapeamento, tendo como principais características: (i) necessitar apenas de um planejador que gera caminhos localmente para, ao mesmo tempo, desviar dos obstáculos e cumprir a missão; (ii) esse planejador busca fornecer um caminho suave, mesmo na presença de obstáculos dinâmicos, não requer recursos computacionais elevados e não exige o conhecimento de um mapa global; (iii) a missão fornecida por um usuário pode ser bem codificada pela utilização de campos vetoriais, sendo essa informação incorporada ao planejador; e (iv) apenas a

localização relativa, válida durante o intervalo de planejamento do robô, é necessária para sua navegação.

Tabela 2.3 – Comparação qualitativa e generalizada entre as estratégias de navegação quanto a três requisitos: dinâmica do ambiente, suavidade do movimento e custo computacional. O símbolo 👍👍 indica que a estratégia de navegação é muito boa com relação a um determinado critério, 👍 indica que é boa, 👎 indica que é ruim e por sua vez 👎👎 indica que é muito ruim.

Navegação	**Dinâmica do ambiente**	**Suavidade do movimento**	**Recursos computacionais**
local reativa	👍👍	👎👎	👍👍
global (mapa *a priori*)	👎👎	👍👍	👎👎
local com mapeamento com memória	👎	👍	👎
local com mapeamento sem memória	👍	👍	👍

CAPÍTULO

3

METODOLOGIA

"O importante na vida não é saber onde estamos, mas a direção para a qual nos movemos."

Oliver W. Holmes, 1809-1894

No Capítulo 2, foram apresentadas várias soluções e conceitos fundamentais relacionados a navegação de um robô de serviço que são base para a estratégia proposta neste trabalho. Essa estratégia de navegação procura solucionar o problema específico definido na Seção 1.1, sendo estruturada conforme a arquitetura híbrida apresentada na Figura 3.1. Embora os módulos de mapeamento e localização dessa arquitetura possam sugerir uma arquitetura puramente deliberativa tradicional, esses módulos dependem apenas de informações locais, a partir das quais, um único planejador RRT* *anytime* determina um caminho a ser seguido localmente, permitindo desviar de obstáculos e cumprir a missão fornecida pelo usuário ao mesmo tempo. Repare que essa missão é transmitida ao robô por meio de comandos simples do usuário e codificada por um campo vetorial que é utilizado pelo funcional de custo para o RRT*.

O presente capítulo pretende esclarecer essa estratégia que será detalhada na Seção 3.1. Os principais módulos da arquitetura de navegação serão apresentados nas seções: Planejador de Caminhos (Seção 3.2), Controlador de caminhos (Seção 3.3), Mapeamento local e Localização de curta duração (Seção 3.4), Gerenciador (Seção 3.5).

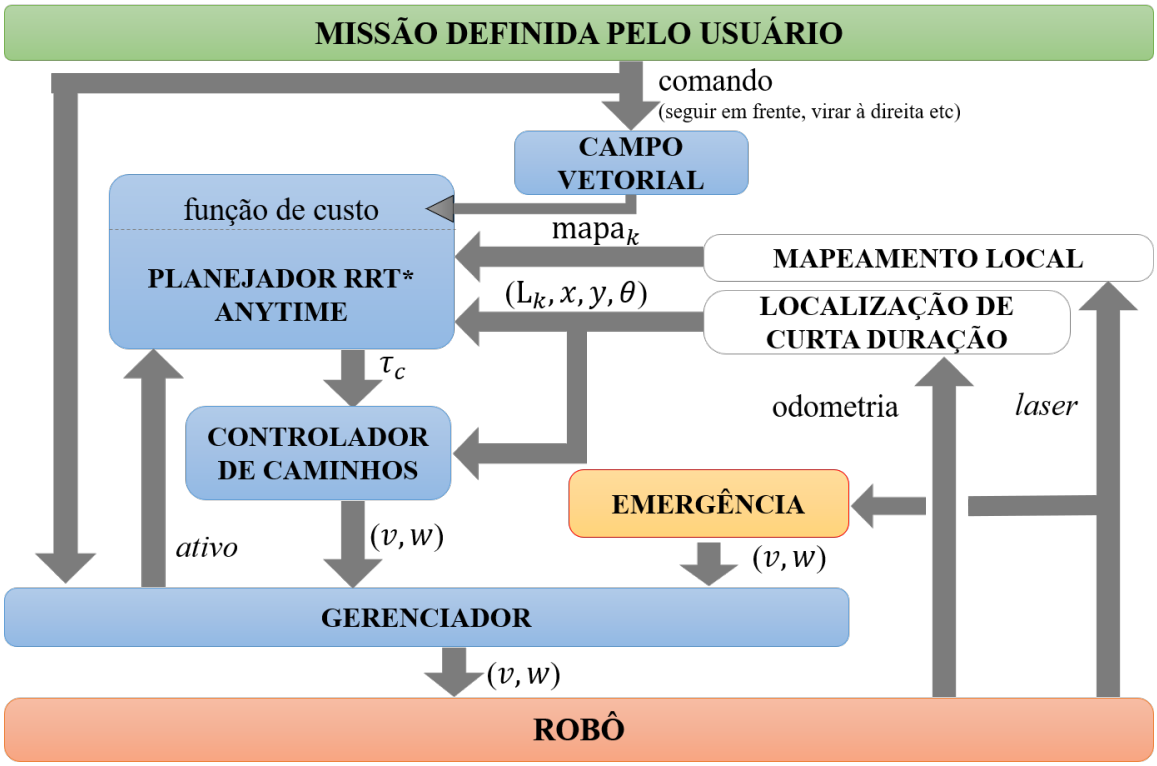


Figura 3.1 – Arquitetura híbrida de navegação local com mapeamento proposta neste trabalho.

3.1 Estratégia de Navegação

A ideia principal da estratégia de navegação deste trabalho pode ser expressa parafraseando o escritor americano Oliver W. Holmes citado na epígrafe: *"o importante para um robô semiautônomo não é saber onde ele está de forma global em um mapa, mas saber a direção para onde querem que ele se mova localmente"*. Utilizar esse tipo de ideia para a navegação do robô é buscar uma solução que se aproxima da maneira com que nos locomovemos em um ambiente, ou seja, normalmente, não precisamos ter a nossa localização global nem o conhecimento do mapa do ambiente para alcançarmos os nossos objetivos.

Pode-se elencar três características fundamentais dessa estratégia que pode facilitar o seu entendimento:

1. Mapeamento local: a partir dos dados de um sensor é obtido um mapa da região na qual o robô está se locomovendo;

2. Replanejamento a cada intervalo de tempo de planejamento (t_s): com base no mapa local fornecido e a cada intervalo t_s o planejador irá fornecer um caminho a ser seguido localmente pelo robô ;

Algoritmo 2: Estratégia de Navegação.

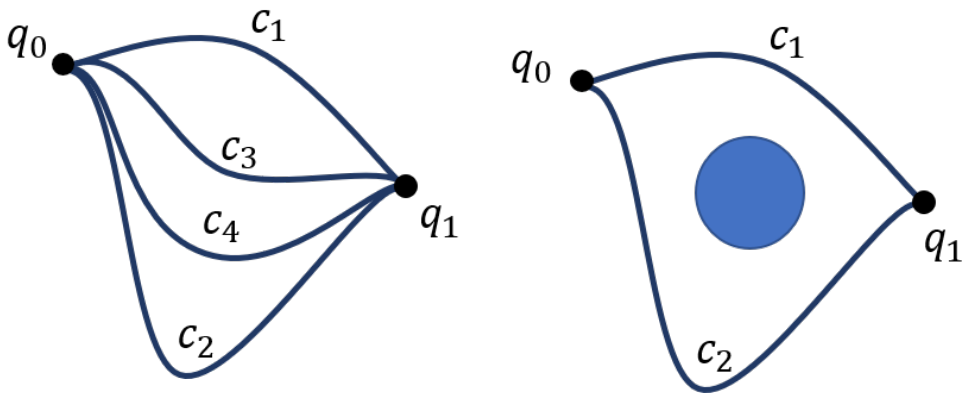
```
1  k = 0 ; v = 0.0 ; ω = 0.0 ; v_r = (v, ω) ; τ_0 ← ∅ ; t_s = tempo de planejamento desejado ;
2  L_0 = GeraReferencial();
3  mapa_0 = ObtemMapa(L_0);
4  k ← k + 1;
5  Enquanto ativo faça
6      L_k = GeraReferencial();
7      τ_c = TransformaCaminho(τ_{k-1}, L_k);
8      mapa_k = TransformaMapa(mapa_{k-1}, L_k);
9      o_c = Integra(τ_c, p_r , t_s, v_r );
10     Faça em paralelo enquanto t ≤ t_s:
11         τ_s = Planeja(mapa_k, o_c, t_s);
12         v_r = SegueCaminho(τ_c);
13     fim-paralelo
14     τ_k = ConcatenaCaminho(τ_c, τ_s, p_r, o_c);
15     mapa_k = ObtemMapa(L_k);
16     k ← k + 1;
17 fim-enquanto
```

3. Criação de referenciais (L_k): para o funcionamento local dessa estratégia é necessário a cada intervalo t_s criar um novo referencial, de forma que apenas utilizando a odometria relativa ao novo referencial seja suficiente para que o robô siga o caminho fornecido pelo planejador.

Nesse sentido, o Algoritmo 2 condensa a solução proposta neste trabalho. Considere que o tempo pode ser discretizado em intervalos t_k, $k = 1, 2, \ldots$, onde o período $\Delta t = t_k - t_{k-1}$ é correspondente ao tempo de cálculo do planejador escolhido (tempo de planejamento t_s). Veja que uma sequência de referenciais locais $\{L_k\}$ será gerada na execução desse algoritmo, sendo que a cada período de planejamento um referencial local é usado como referencial fixo (linha 6 do Algoritmo 2).

O laço de execução é realizado enquanto um sinal de *ativo* enviado pelo módulo gerenciador for verdadeiro. Os primeiros procedimentos do laço são gerar um novo referencial $\{L_k\}$ e em sequência transformar o caminho fornecido pelo planejador (τ_{k-1}) e o mapa local gerado a partir dos dados dos sensores ($\texttt{mapa}_{k-1}$), ambos obtidos em relação ao referencial $\{L_{k-1}\}$, para o novo referencial $\{L_k\}$ (linhas 7 e 8).

Após essa transformação, cujo tempo de cálculo é muito pequeno em relação a t_s e pode ser desprezado, uma nova instância do planejador é lançada (linha 11). Como parâmetros, esse planejador utiliza o mapa local que acaba de ser transformado para $\{L_k\}$ (linha 8) e o ponto de origem do novo caminho, $\mathbf{o}_c$. Pode-se estranhar a ausência de um ponto alvo como entrada do planejador. De fato, como um robô semiautônomo nem sempre conhece seu destino, caso esse ponto seja necessário para o planejador em questão, ele pode ser escolhido arbitrariamente



(a) (b)

Figura 3.2 – Ilustração de classes homotópicas. (a) Dois caminhos conectando os mesmos nós (inicial e final) são da mesma classe homotópica se eles podem ser deformados de forma suave um no outro, sem passar dentro de algum obstáculo. (b) Caso contrário eles pertencem a classes homotópicas diferentes.

e sem perda de generalidade como um ponto à frente do robô e no limite do campo de visão de seus sensores. O ponto de origem, $\mathbf{o}_c$, é obtido integrando-se (ou simulando-se) o caminho anterior (calculado em relação a $\{L_{k-1}\}$ e transformado para $\{L_k\}$), durante um tempo t_s a partir da posição real do robô, $\mathbf{p_r}$, representada no referencial atual (linha 9). Para essa integração é considerada a velocidade média do robô, $\mathbf{v_r}$. A ideia de se escolher um ponto do caminho anterior como sendo o início do próximo caminho é manter a continuidade do caminho do robô. Com essa escolha, também evita-se situações de caminhos pertencentes a classes homotópicas diferentes, como ilustrado na Figura 3.2. Observe que essa estratégia define um trecho do caminho anterior (entre a posição atual do robô $\mathbf{p_r}$ e $\mathbf{o_c}$) a ser seguido enquanto o novo caminho é calculado. Esse trecho do caminho é chamado na literatura de caminho comprometido (*commited path*) [Karaman *et al.* 2011].

Enquanto é solucionado o novo problema de planejamento (linha 11), durante o intervalo t_s, um Controlador de caminhos fornece os devidos comandos de velocidade para manter o robô seguindo o caminho comprometido (linha 12). Repare que esse caminho já está no referencial $\{L_k\}$, não havendo necessidade do Controlador de caminhos transformá-lo, utilizando apenas como realimentação a posição e orientação do robô em relação ao referencial atual ($\{L_k\}$).

Passado o tempo previsto de planejamento t_s, quando o planejador retorna o novo caminho, o caminho comprometido ainda não seguido pelo robô é concatenado com o caminho recém calculado (linha 14), um novo mapa com as últimas informações dos sensores é obtido (linha 15) e o processo se repete fazendo a transformação do novo caminho e do novo mapa para o recém criado referencial $\{L_{k+1}\}$.

É interessante ressaltar que, para que todo esse processo funcione adequadamente, sempre é necessário conhecer a transformação entre o referencial anterior e o referencial atual. Como t_s é, normalmente, um tempo muito pequeno (na ordem de 1 s a 2 s), uma boa odometria é geralmente

suficiente para calcular as transformações de coordenadas necessárias com baixa incerteza, como poderá ser visto no Capítulo 4.

Para ilustrar essa estratégia, considere um robô com velocidade, v_r, em um ambiente qualquer, contendo obstáculos estáticos e dinâmicos, como representado na Figura 3.3(a). Em $t = t_0$, Figura 3.3(b), um referencial $\{L_1\}$ é gerado e o caminho fornecido pelo planejador é transformado para esse referencial, válido até $t = t_0 + t_s$. Enquanto o robô segue esse caminho, utilizando a odometria relativa ao referencial L_1, o planejador inicia o replanejamento de um novo caminho, mas que se inicia em um ponto à frente do robô, $\mathbf{o}_{c1}$, que representa onde espera-se que o robô chegue, considerando sua velocidade máxima.

Pode-se definir três zonas, mostradas na Figura 3.3(c), para facilitar a análise da estratégia: (i) Zona Estática (Z_{est}), (ii) Zona de Emergência (Z_e) e (iii) Zona Dinâmica (Z_d). A Zona Estática, delimitada pelo raio R_s, é uma zona crítica, na qual o robô está percorrendo um trecho do caminho planejado durante o intervalo t_s, logo os obstáculos não devem se mover nessa zona. Caso hajam apenas obstáculos estáticos, a estratégia proposta garante a ausência de colisão, já que o robô seguirá o caminho planejado. Para garantir também que não haverá colisão mesmo que algum imprevisto ocorra nessa zona, foi definida uma zona de emergência em torno do robô, de maneira que, quando um obstáculo a invade, um sistema de emergência é acionado, evitando uma possível colisão de forma reativa. A Zona Dinâmica é delimitada externamente pelo raio de planejamento R_p e internamente pelo raio da Zona Estática R_s. O valor de R_p é, de fato, um parâmetro do planejador, que pode ser escolhido, por exemplo, em função da capacidade computacional (regiões maiores demandam mais cálculos), velocidade e número de obstáculos (um horizonte de planejamento maior pode resultar em trajetórias melhores) e campo de visão dos sensores (é necessário um conhecimento dos obstáculos dentro da região de planejamento). Independentemente da escolha de R_p, a estratégia proposta neste trabalho garantirá que o robô não entrará na Zona Dinâmica no intervalo de planejamento de trajetórias, t_s, e por isso, a ausência de colisões com obstáculos estáticos e dinâmicos nessa região será assegurada. Apesar do robô não se mover para a Zona Dinâmica, é interessante notar que o caminho planejado já prevê a existência de obstáculos nessa região, se esses forem detectados pelos sensores no instante inicial do planejamento.

Após t_s segundos, Figura 3.3(d), o robô percorreu o trecho de caminho marcado em amarelo, mas não alcançou por algum motivo o ponto no qual era esperado que ele chegasse. Nesse instante, também, um novo caminho a partir de $\mathbf{o}_{c1}$ é fornecido pelo planejador. Dessa forma, o novo caminho que o robô deve seguir é a junção do trecho do caminho comprometido que não foi cumprido com o novo caminho, formando um caminho completo a partir da posição atual do robô. Na Figura 3.3(e) é mostrado o novo referencial criado, $\{L_2\}$, para o qual o caminho completo é transformado. Assim, o ciclo recomeça: enquanto o robô segue esse caminho, o planejador, com base no mapa obtido nesse instante, recomeça o seu planejamento, considerando um ponto a frente do robô, $\mathbf{o}_{c2}$, enquanto o robô segue o caminho fornecido.

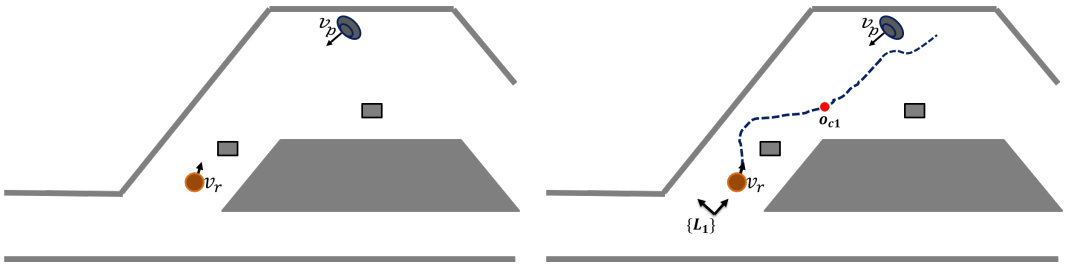
Na Figura 3.3(e), as zonas são novamente representadas para o instante $t = t_0 + t_s$. Repare que o obstáculo que se movia agora está dentro da Zona Estática. Dessa forma, para garantir o que o desvio seja de forma planejada, ele não poderá se mover nessa zona. Note que para um pior caso, quando um robô segue uma linha reta com velocidade máxima v_{max}, o raio R_s pode ser definido como:

$$R_s = v_{max} \cdot t_s \,, \tag{3.1}$$

ou seja, se a velocidade do robô e/ou o tempo de planejamento aumentarem, a Zona Estática torna-se maior. Como nessa zona o robô segue um plano calculado t_s segundos atrás (caminho comprometido), esse plano somente considera os obstáculos que não se moveram nesse período. Assim, para evitar colisão com obstáculos dinâmicos nessa zona, a navegação deverá também apresentar um caráter reativo, ou seja, se um obstáculo for detectado muito próximo ao robô, dentro da zona de emergência, este deverá parar imediatamente antes que ocorra uma colisão. Caso o tempo de planejamento diminua muito $(t_s \to 0)$, a área da zona estática também reduziria muito $(R_s \to 0)$ e, assim, a navegação tenderia a ser puramente deliberativa, já que sempre seria gerado, em um curto intervalo de tempo, um caminho completo a ser seguido pelo robô. Dessa forma, como ilustrado nas Figuras 3.3(g) e (h), sendo a zona estática menor, implica que a atuação de forma reativa da estratégia é reduzida, na existência de objetos móveis no ambiente.

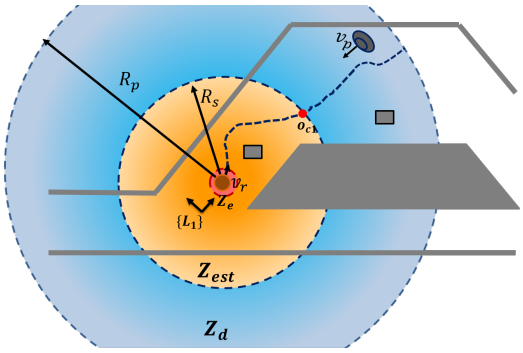
Pode-se, então, concluir que o desempenho da estratégia de navegação garante sempre desviar de obstáculos estáticos, desde que esses sejam percebidos pelos sensores do robô no tempo de planejamento. No entanto, para o desvio de obstáculos dinâmicos a estratégia de navegação é dependente do tempo de planejamento pré-estabelecido e das velocidades do robô e dos obstáculos. O desejável é, então, diminuir ao máximo o tempo de planejamento (por meio de algoritmos mais eficientes ou aumento do poder computacional) de forma a permitir que o robô se desloque com uma velocidade mais alta e mesmo assim seja capaz de desviar de obstáculos dinâmicos sem parar o seu movimento, de forma suave e planejada previamente.

As principais funções do Algoritmo 2: Planeja($\texttt{mapa}_k$, $\mathbf{o}_c$, t_s), SegueCaminho(τ_c), GeraReferencial() e ObtemMapa(L_k) são responsabilidades dos módulos tratados nas próximas seções.

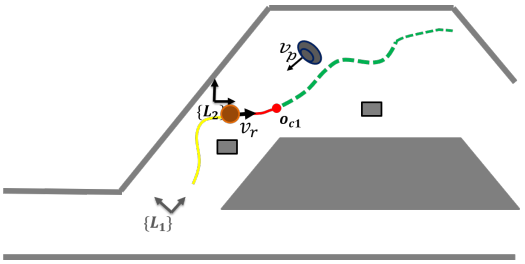


(a) $t = t_0$

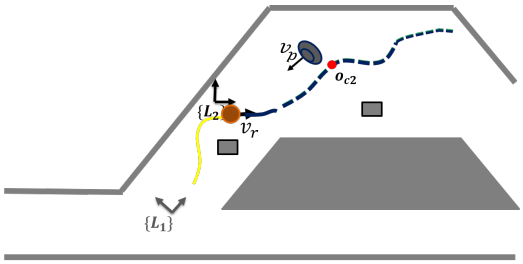
(b) $t = t_0$



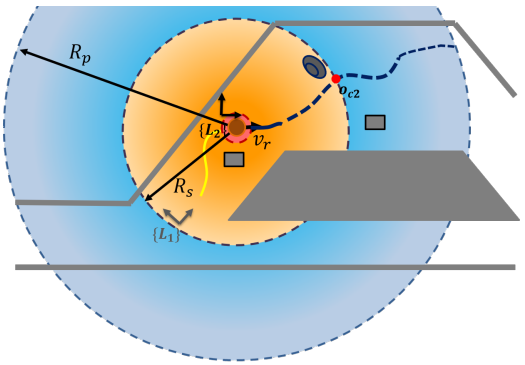
(c) $t = t_0$



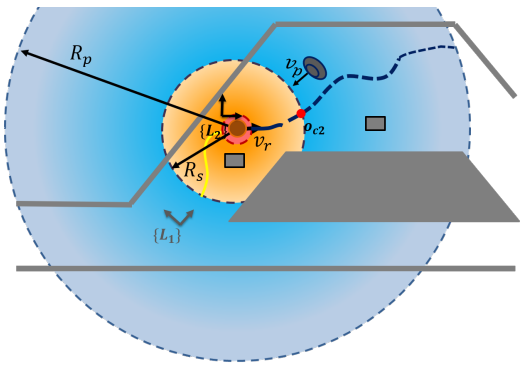
(d) $t = t_0 + t_s$



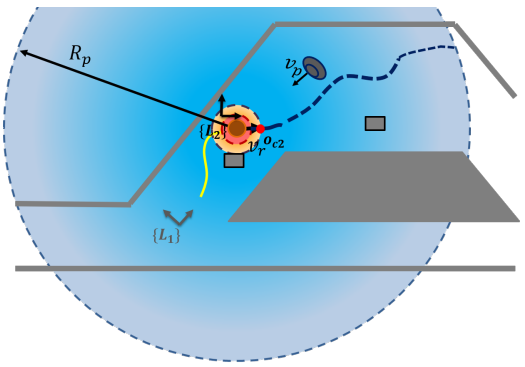
(e) $t = t_0 + t_s$



(f) $t = t_0 + t_s$



(g) $t = t_0 + t_{s1}$, para $t_{s1} < t_s$



(h) $t = t_0 + t_{s2}$, para $t_{s2} < t_{s1} < t_s$

Figura 3.3 – Ilustração do funcionamento da estratégia de navegação. O robô é representado como uma circunferência e possui uma velocidade $\mathbf{v}_r$. Os obstáculos são representados pela cor cinza e o caminho completo a ser seguido pelo robô por uma linha tracejada em azul escuro. O raio R_s define a Zona Estática (Z_{est}) e o raio R_p representa o raio de planejamento. A Zona Dinâmica (Z_d) é delimitada externamente por R_p e internamente por R_s e a Zona de Emergência é representada por um círculo em torno do robô.

3.2 Planejador de Caminhos

Para o módulo de planejador de caminhos foi escolhido o planejador probabilisticamente completo e assintoticamente ótimo RRT*, que possui a propriedade de ser também *anytime* [Karaman *et al.* 2011]. O Algoritmo 3 apresenta o seu funcionamento, que pode ser sintetizado como a construção de uma árvore G a partir de configurações aleatórias no espaço livre de posições do robô ($\mathscr{P}_{free}$) e conectados de forma a minimizar um funcional de custo (linhas 10 a 16). A Figura 3.4 ilustra a criação da árvore G. O caminho retornado por esse algoritmo (linha 26) é uma sequência de nós na árvore criada, partindo do alvo até chegar à raiz da árvore, que é a posição inicial. Note que ao utilizar um planejador *anytime* é definido um tempo máximo de solução, t_s, no qual é retornado o melhor caminho até esse instante. Quanto mais tempo houver para se obter a solução mais próximo do ótimo será o caminho. As funções utilizadas nesse algoritmo foram:

- $AmostraLivre()$: gera uma amostra aleatoriamente no espaço livre de configurações de posições do robô ($\mathscr{P}_{free}$).
- $ObtemMaisProximo(G = (V,E), q_i)$: encontra o nó da árvore G que é mais próximo a q_i em termos da distância Euclideana.
- $ObtemProximos(G = (V,E), q_i, r_{busca})$: retorna o conjunto de configurações que estão dentro de um círculo de busca de raio r_{busca} e centrado em q_i. Como mostrado em [Karaman *et al.* 2011], se esse raio for obtido por $min(\gamma(log(card(V))/card(V))^{1/d}, \eta))$, onde $\gamma > (2(2+1/d)^{1/d}(\mu(\mathscr{P}_{free})/\zeta_d)^{1/d}$, d é dimensão do espaço de configurações de posições do robô ($\mathscr{P}$), $\mu(\mathscr{P}_{free})$ é volume do espaço livre $\mathscr{P}_{free}$ e ζ_d é o volume do círculo unitário no espaço d-dimencional, é garantido que o planejador RRT* é assintoticamente ótimo.
- $Steer(q_i, q_j)$: retorna uma nova configuração obtida orientando q_i para q_j em linha reta, tal que $\|q_i, q_j\|$ é minimizado e $\|q_i, q_j\| \leq \eta$, onde η é o parâmetro chamado de *steering*.
- LivreDeColisão(q_i, q_j): retorna VERDADEIRO se o caminho em linha reta entre q_i e q_j está no espaço livre de posições do robô, caso contrário retorna FALSO.
- CustoAtual(q_i): retorna o custo do caminho que parte do nó raiz e vai até o nó q_i. Na implementação, esse custo pode ser armazenado na estrutura utilizada para a árvore, de forma a ser obtido mais rapidamente.
- Custo(q_i, q_j): retorna o custo do trecho de caminho entre q_i e q_j, utilizando, por exemplo, o funcional de custo detalhado na Subseção 3.2.1.
- Pai(q_i): retorna o nó pai de q_i.

- *ObtemAlvo*$(G = (V,E))$: retorna o nó na árvore G que possui o menor custo com relação ao funcional especificado e que está na borda de um círculo pré-especificado que representa a região de planejamento.

- *Caminho*(G, q_i, q_j): retorna uma sequência de nós, partindo do nó q_j e chegando ao nó q_i.

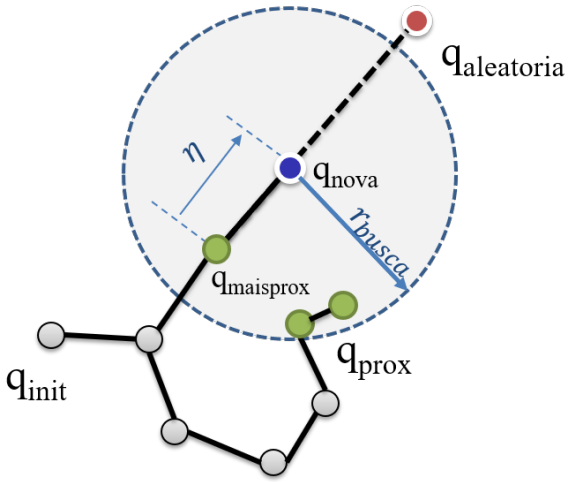


Figura 3.4 – Criação da árvore do RRT*. Uma amostra é sorteada, $q_{aleatoria}$, e com base no passo dado pelo parâmetro *steering*, η, uma nova amostra, q_{nova}, é gerada e conectada à árvore, inicialmente com a amostra mais próxima. Por meio da função *ObtemProximos*, dado um raio de busca, é encontrado um conjunto de amostras próximas a q_{nova}. Dessa forma, é verificado nesse conjunto o melhor pai para essa nova amostra, de maneira que o custo para chegar até ela seja o menor possível. Por fim, há uma reorganização das arestas da árvore, por meio do processo conhecido como *Rewire*.

Algoritmo 3: RRT* *anytime*. $\tau = RRT^*(t_s, q_{init}, [q_{alvo}])$

Entrada: t_s = tempo de planejamento, q_{init} = posição inicial, $[q_{alvo}]$ = alvo opcional.

Resultado: τ = caminho retornado.

$\mathtt{V} \leftarrow \{q_{init}\}$; $\mathtt{E} \leftarrow \emptyset$; $t = 0$; $G = (\mathtt{V}, \mathtt{E})$; // inicializações

Enquanto $t \leq t_s$ **faça**

  $q_{aleatorio} \leftarrow AmostraLivre()$; // amostra no espaço livre

  $q_{maisprox} \leftarrow ObtemMaisProximo(G = (V,E), q_{aleatorio})$;

  $q_{novo} \leftarrow Steer(q_{maisprox}, q_{aleat})$; /* retorna uma nova configuração guiada pela amostra mais próxima em uma linha reta. */

  Se $LivreDeColisão(q_{maisprox}, q_{novo})$ **então** // se entre $q_{maisprox}$ e q_{novo} está livre.

    $\mathtt{V} \leftarrow \mathtt{V} \cup \{q_{novo}\}$; // adiciona amostra nova no conjunto de vértices.

    // seleciona o pai da amostra q_{novo}, aquele que tem o menor custo:

    $r_{busca} = min(\gamma(log(card(V))/card(V))^{1/d}, \eta))$; /* raio de busca que garante otimalidade, sendo η chamado de *steering*. */

    $Q_{prox} \leftarrow ObtemProximos(G = (V,E), q_{novo}, r_{busca})$;

    $q_{pai} = q_{maisprox}$; $c_{min} \leftarrow \text{CustoAtual}(q_{maisprox}) + \text{Custo}(q_{maisprox}, q_{novo})$;

    para cada $q_{prox} \in Q_{prox}$ **faça**

      Se $LivreDeColisão(q_{prox}, q_{novo})$ **então**

        Se $CustoAtual(q_{prox}) + Custo(q_{prox}, q_{novo}) < c_{min}$ **então**

          $q_{pai} = q_{prox}$;

          $c_{min} \leftarrow \text{CustoAtual}(q_{prox}) + \text{Custo}(q_{prox}, q_{novo})$;

    $\mathtt{E} \leftarrow \mathtt{E} \cup \{(q_{pai}, q_{novo})\}$; // adiciona vértice entre o pai e a amostra nova.

    /* reorganiza as arestas da árvore, caso o caminho utilizando a amostra nova seja menor do que o custo atual da amostra de teste, etapa conhecida como *Rewire*: */

    para cada $q_{teste} \in Q_{prox}$ **faça**

      Se $LivreDeColisão(q_{teste}, q_{novo})$ **então**

        Se $CustoAtual(q_{novo}) + Custo(q_{novo}, q_{teste}) < Custo(q_{teste})$ **então**

          $q_{pai} = \text{Pai}(q_{teste})$; // retorna o pai da amostra de teste na árvore.

          $\mathtt{E} \leftarrow (\mathtt{E} \setminus \{(q_{pai}, q_{teste})\} \cup \{(q_{pai}, q_{novo})\}$; // atualiza arestas.

  $t \leftarrow t + \Delta t$;

Se q_{alvo} *não definido* **então**

  $q_{alvo} = ObtemAlvo(G = (V,E))$

$\mathtt{V} \leftarrow \mathtt{V} \cup \{q_{alvo}\}$; $\mathtt{E} \leftarrow \mathtt{E} \cup \{(q_{alvo}, ObtemMaisProximo(G = (V,E), q_{alvo}))\}$; $G \leftarrow (\mathtt{V}, \mathtt{E})$;

retorna $\tau = Caminho(G, q_{init}, q_{alvo})$

3.2.1 Funcional de custo do planejador

Seja um campo vetorial $\mathbf{f}$ e um caminho definido por uma função contínua $\tau(s) : [0,1] \to \mathscr{P}$, onde $\mathscr{P}$ é o espaço de posições do robô[8] no referencial $\{L_k\}$, deseja-se que esse caminho ao mesmo tempo que desvie de obstáculos siga o campo vetorial. Para tal, o caminho deve minimizar o seguinte funcional de custo:

$$F[\tau,\mathbf{f}] = \int_0^1 \left(1 - \frac{\tau'(s)}{\|\tau'(s)\|} \cdot \frac{\mathbf{f}(\tau(s))}{\|\mathbf{f}(\tau(s))\|}\right) \|\tau'(s)\|\, ds\,, \tag{3.2}$$

onde $\tau'(s)$ é um vetor paralelo ao caminho[9], referente a primeira derivada desse com respeito à variável de parametrização espacial s. Esse funcional de custo é o mesmo proposto em [Pereira, Choudhury e Scherer 2016] quando as variáveis a e b em [Pereira, Choudhury e Scherer 2016] são iguais a 1 (um), e similar ao *upstream criterion* proposto em [Ko, Kim e Park 2014], o qual é especificado para curvas em que $\|\tau'(s)\| = 1$.

Repare que o Funcional (3.2) é função tanto do comprimento do caminho quanto de uma medida de proximidade do caminho ao campo vetorial. Dessa forma, o menor custo possível, $F[\tau,\mathbf{f}] = 0$, acontece quando o produto escalar entre $\tau'(s)$ e $\mathbf{f}(\tau(s))$ é máximo para todos os pontos do caminho, ou seja, quando esses vetores são paralelos o caminho coincide com a integral do campo vetorial.

Com a utilização desse funcional, o planejador escolhido deve resolver o seguinte problema de otimização:

$$\begin{aligned} \underset{\tau}{\text{minimize}} \quad & F[\tau,\mathbf{f}] = \int_0^1 \left(1 - \frac{\tau'(s)}{\|\tau'(s)\|} \cdot \frac{\mathbf{f}(\tau(s))}{\|\mathbf{f}(\tau(s))\|}\right) \|\tau'(s)\|\, ds \\ \text{sujeito a:} \quad & \tau(0) = \mathbf{p}_0\,, \\ & \|\tau(1) - \tau(0)\| = r_{alvo}\,, \\ & \tau(s) \in \mathscr{P}_{\text{free}}, \forall s \in [0,1]\,. \end{aligned} \tag{3.3}$$

onde $\mathbf{p}_0$ é a posição inicial do caminho, r_{alvo} escolhido como sendo o raio de planejamento e $\mathscr{P}_{\text{free}}$ é o espaço de posições do robô livre de colisões[10].

O problema (3.3) apresenta uma grande diferença em relação ao problema padrão de planejamento de movimento, que é a substituição da restrição tradicional de se chegar à posição alvo por uma restrição que impõe a distância entre as posições inicial e final do caminho dentro de um círculo de raio r_{alvo}. Portanto, uma vez que não é necessário fornecer a posição do alvo, utiliza-se o Funcional (3.2) para determinar um alvo como sendo a configuração de menor custo na borda desse círculo. Isso pode ser visto como um alvo no infinito, tal que o robô nunca o alcançará, mas permite que o robô siga em sua direção. A Figura 3.5 ilustra um exemplo de um caminho fornecido pelo planejador, utilizando esse funcional para um dado campo vetorial.

[8] Esse espaço não é o espaço de configurações do robô, pois apenas a posição (x,y) do robô é considerada.

[9] O vetor $\tau'(s)$ também pode ser visto como um vetor ortogonal ao gradiente espacial do caminho.

[10] O espaço de posições do robô é uma aproximação do espaço de configurações, que neste caso pode ser obtido aproximando o robô por um disco.

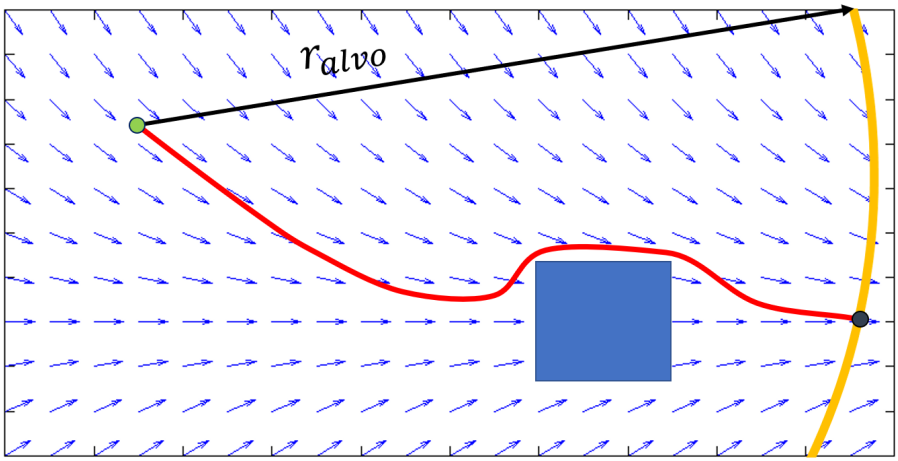


Figura 3.5 – Ilustração de um caminho (em vermelho) fornecido pelo planejador RRT*, utilizando o Funcional (3.2) para um dado campo vetorial representado pelas setas, em que r_{alvo} foi escolhido como sendo o raio de planejamento.

Diferentemente do proposto em [Pereira, Choudhury e Scherer 2016] e em [Ko, Kim e Park 2014] que calculam o funcional de custo avaliando o campo vetorial ao longo do trecho de caminho entre duas configurações, neste trabalho adotou-se a seguinte heurística: avaliar o funcional apenas na configuração inicial do trecho de caminho fornecido. Essa heurística é válida se considerarmos que o comprimento desse caminho é pequeno e que o campo vetorial varia pouco localmente. Essa heurística melhora o desempenho do algoritmo, já que é muito mais eficiente computacionalmente do que o proposto em [Pereira, Choudhury e Scherer 2016] e em [Ko, Kim e Park 2014], como será comprovado no Capítulo 4. Assim, o funcional de custo utilizado é definido pelo Algoritmo 4.

Algoritmo 4: Funcional de custo do caminho entre duas configurações, considerando que estão conectados por uma reta.

Entrada: q_0 = configuração inicial, q_1 = configuração final.
Resultado: c = custo do caminho.

Função *Custo*(q_0, q_1)
    `// heurística: avalia o funcional apenas para a configuração inicial.`
    $\tau = q_1 - q_0$;
    $f = \text{ObtemValorCampo}(q_0)$; `// obtém o valor do campo para a configuração inicial`
    $c = \left(1 - \left(\frac{\tau}{\|\tau\|} \cdot \frac{f}{\|f\|}\right)\right) \|\tau\|$; `// o operador (·) representa o produto escalar.`
    retorna c

3.3 Controlador de caminhos

Na Figura 3.6 é apresentado o diagrama de blocos do Controlador de caminhos adotado neste trabalho. Pode-se ver que suas entradas são: o referencial fixo no instante discretizado k, $\{L_k\}$, uma sequência de pontos nesse referencial, ou seja, o caminho a ser seguido, e a posição do robô no referencial $\{L_k\}$, $\vec{r}_k$. Com base nessas entradas, é determinada a referência, $\vec{z}_{ref}$, para um controlador proporcional que serve de entrada para a malha de linearização. Quando a norma do erro entre a posição do robô e a posição de referência é menor que um determinado limiar, o próximo ponto pertencente ao caminho é fornecido para o controlador. Note que, antes que o robô chegue em uma referência especificada, uma nova referência é fornecida para que ele a siga. Na Subseção 3.3.1 é apresentado o modelo cinemático, normalmente, utilizado para os robôs de serviço e na Subseção 3.3.2 é detalhado o controle linearizante.

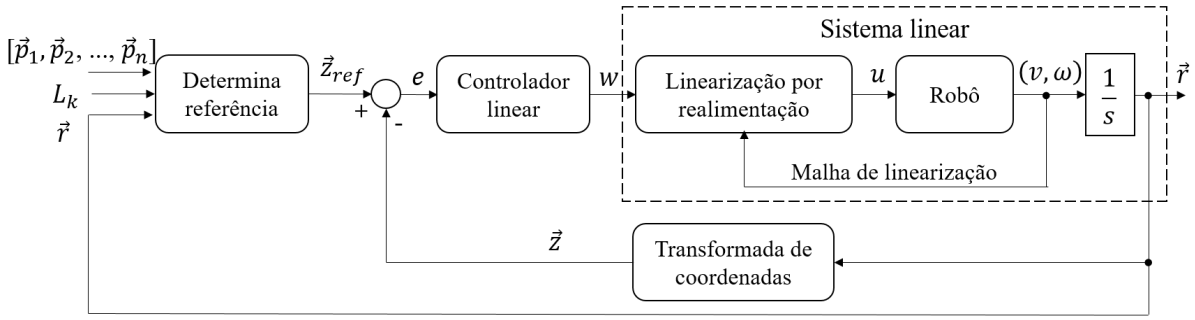


Figura 3.6 – Diagrama de blocos do Controlador de caminhos adotado neste trabalho.

3.3.1 Modelo cinemático

Muitos robôs de serviço terrestres podem ser modelados cinematicamente por um sistema não-linear, tendo como ação de controle, u, as velocidades linear, v, e angular, ω:

$$\dot{q} = \begin{bmatrix} \dot{x_c} \\ \dot{y_c} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.4}$$

sendo $q = (\vec{r}, \theta)$ a configuração do robô no referencial $\{L_k\}$, como ilustrado na Figura 3.7.

Esse modelo assume que as rodas do robô estão sempre em contato com o solo e que não acontecem derrapagens, ou seja, o modelo assume uma restrição de velocidade, que pode ser representada matematicamente pela seguinte restrição não-holonômica para cada roda:

$$\dot{x}\sin\theta - \dot{y}\cos\theta = 0\,, \tag{3.5}$$

ou seja, a roda não pode assumir velocidades na direção paralela ao seu eixo de rotação.

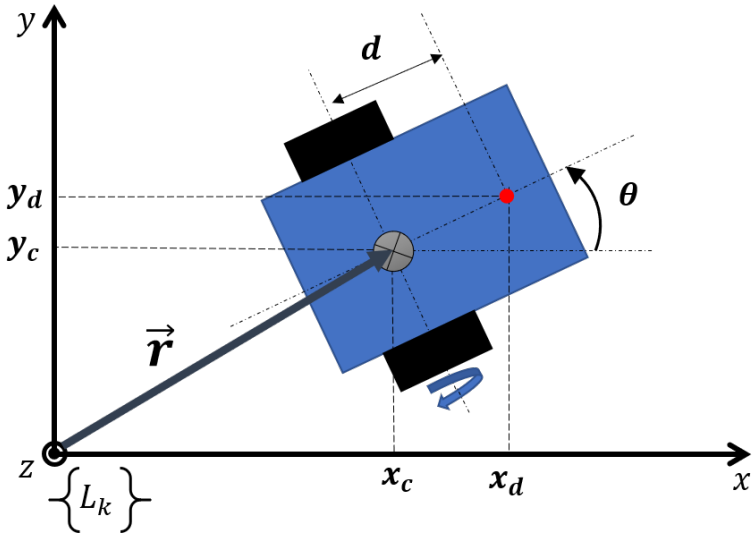


Figura 3.7 – Sistemas de coordenadas adotados. O centro do robô está fixado na interseção do eixo central do robô e o eixo das rodas, sendo determinado pelo vetor $\vec{r} = [x_c, y_c]^T$. Um ponto ao longo do eixo central do robô e a uma distância d do centro do robô é determinado pelas coordenadas (x_d, y_d) e indicado pelo círculo vermelho. A orientação do robô no sistema de coordenadas é fornecida pelo ângulo θ.

3.3.2 Controle linearizante

Para determinar uma lei de controle para o sistema não-linear da Equação (3.4), pode-se fazer a seguinte transformação de coordenadas, assumindo um ponto (x_d, y_d) ao longo do eixo central do robô e a uma distância d do centro do robô:

$$\vec{z} = \begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} x_c + d\cos\theta \\ y_c + d\sin\theta \end{bmatrix} \tag{3.6}$$

Derivando a Equação (3.6), pode-se obter o modelo cinemático do robô nas novas coordenadas:

$$\begin{bmatrix} \dot{x_d} \\ \dot{y_d} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -d\sin\theta \\ \sin\theta & d\cos\theta \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.7}$$

Como o sistema é sub-atuado, escolheu-se como variável livre a orientação do robô, ou seja, o controlador proposto irá garantir apenas que se alcance a posição desejada para o ponto (x_d, y_d), não garantindo uma orientação desejada para o robô. Assim, pode-se obter os comandos de velocidade linear e angular, a partir da velocidade desejada para o ponto (x_d, y_d) da seguinte forma:

$$u = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\frac{\sin\theta}{d} & \frac{\cos\theta}{d} \end{bmatrix} \cdot \begin{bmatrix} \dot{x_d} \\ \dot{y_d} \end{bmatrix} \tag{3.8}$$

Note que na Equação (3.8) a distância d do centro do robô aparece como um ganho na lei de controle, não podendo assumir um valor nulo. Percebe-se que quanto menor for o valor de d maior serão as variações sentidas pelo robô quando acontecerem pequenas variações no ponto que se está seguindo. Porém, quanto maior for o valor de d mais distante do caminho real

o centro do robô estará. Esse tipo de controle é conhecido como controle via linearização por realimentação de entrada-saída (*Feedback Linearization*) e mais detalhes podem ser encontrados em [Chernousko 2008].

Para completar o controle presente no Controlador de caminhos, a determinação das velocidades $(\dot{x_d}, \dot{y_d})$ pode ser especificada a partir de um controlador linear, por exemplo, um controlador proporcional ao erro:

$$w = \begin{bmatrix} \dot{x_d} \\ \dot{y_d} \end{bmatrix} = \frac{v_{max}}{\left\| \vec{z}_{ref} - \vec{z} \right\|} \cdot (\vec{z}_{ref} - \vec{z}) \tag{3.9}$$

em que $(\vec{z}_{ref} - \vec{z})$ é o erro entre os vetores referentes ao valor desejado e o valor medido para o ponto (x_d, y_d), v_{max} é a velocidade linear máxima desejada para o robô e o operador $\| \cdot \|$ representa a norma euclideana.

3.4 Mapeamento local e Localização de curta duração

O mapeamento local é responsável por gerar um mapa válido durante o tempo de planejamento, t_s, podendo ser obtido ao menos de duas formas: (i) sem memória: utilizando, diretamente, os dados de um *laser* ou (ii) com memória: utilizando algum método de SLAM. No primeiro caso, a obtenção do mapa local é mais simples, pois pode ser utilizada apenas uma matriz que contém as posições (x, y) dos obstáculos detectados pelo sensor no referencial $\{L_k\}$, válido naquele instante. No segundo caso, a obtenção é mais precisa, já que os métodos de SLAM fazem associação dos dados dos sensores e permitem a atualização da localização do robô no mapa que está sendo construído. No entanto, os métodos baseados em SLAM são mais lentos para atualização do mapa quando o ambiente modifica-se, dificultando, por exemplo, o desvio de obstáculos dinâmicos no ambiente. Repare que a cada intervalo t_s pode-se reiniciar o método de SLAM, já que um mapa global não é necessário para a metodologia. Com isso, é reduzido o custo de memória e o custo computacional.

Com respeito ao módulo de localização de curta duração, é importante ressaltar que o seu objetivo é: (i) gerar a cada intervalo de planejamento, t_s, um novo referencial, que pode ser baseado na sua posição atual ou em uma característica conhecida do ambiente e (ii) fornecer a localização do robô durante o curto intervalo t_s, tendo como referencial fixo o referencial $\{L_k\}$, ou seja, a odometria utilizada é relativa ao movimento após a fixação desse referencial. Repare que não é objetivo da metodologia corrigir os erros acumulados da odometria ao longo da navegação, mas garantir que o caminho fornecido seja cumprido localmente. Isso é possível, já que para efeitos do planejamento e do controlador de caminhos a localização utilizada é relativa aos referenciais que estão sendo criados a cada intervalo t_s, como visto no Algoritmo 2. No próximo Capítulo, será apresentada uma maneira de determinar os referenciais.

3.5 Gerenciador

A função do módulo gerenciador na arquitetura proposta é determinar os comandos de velocidades que serão enviados para o robô. Para tal, ele seleciona entre o comando fornecido pelo Controlador de caminhos, o comando enviado pelo bloco de emergência e o comando desejado do usuário.

O comando enviado pelo Controlador de caminhos é baseado em um planejamento, porém, alguns comandos desejados pelo usuário, como, parar, não precisam ser obtidos utilizando o planejador, o próprio gerenciador pode fornecer esse tipo de comando. Nesses casos, o sinal *ativo*, mostrado na Figura 3.1, é falso, parando a execução do planejador. O mesmo acontece quando uma situação inesperada, por exemplo, um obstáculo surge na frente do robô, o gerenciador seleciona o comando vindo do bloco de emergência que tem maior prioridade e desativa o planejador.

CAPÍTULO

4

RESULTADOS

Sejam quais forem os resultados, com êxito ou não, o importante é que no final cada um possa dizer: fiz o que pude.

Louis Pasteur, 1822-1895

Este capítulo tem como objetivo apresentar o ambiente escolhido para os experimentos (Seção 4.1), a implementação da metodologia apresentada no capítulo anterior (Seção 4.2), e os resultados obtidos para sua validação, tanto em simulação (Seção 4.3) quanto utilizando robôs de serviços semiautônomos reais (Seção 4.4).

4.1 Ambiente

O ambiente escolhido para os experimentos foi um cenário típico de prédios comerciais, escolas e hospitais: um conjunto de longos corredores e interseções, contendo alguns móveis e/ou objetos de decoração, mas que são essencialmente vazios, exceto pela presença de pessoas. A Figura 4.1 apresenta uma planta baixa de um hospital, na qual pode-se ver os vários corredores que o compõem.

Do ponto de vista da navegação de um robô, esse tipo de ambiente, por um lado, pode ser facilmente mapeado, devido à estrutura desses edifícios, com corredores longos e poucos obstáculos, mas por outro, o processo de localização é dificultado já que o ambiente possui simetria e regiões muito semelhantes. Quanto ao planejamento de movimento, se for considerado apenas um ambiente estático e conhecido, poderiam ser utilizadas abordagens clássicas, como busca em grafos e métodos baseados em campos potenciais. Mas, tendo em conta a presença de pessoas no ambiente, consideradas como obstáculos dinâmicos com movimento quase imprevisível, aumenta-se a dificuldade do problema, exigindo assim planejadores de

Figura 4.1 – Exemplo de planta baixa de um hospital. Fonte: <https://yukonhospitals.ca/yukon-hospital-corporation/design>.

movimento mais complexos. Nessas condições de dificuldade de localização, sem um mapa prévio e podendo haver obstáculos dinâmicos, esse ambiente foi considerado interessante e ideal para a validação da metodologia proposta neste trabalho.

4.2 Implementação

A estratégia proposta foi implementada em C++ e Python, utilizando o ROS (*Robot Operating System*) juntamente com a biblioteca OMPL[11] (*Open Motion Planning Library*). O processamento foi realizado por um notebook com processador i7@2.4GHz, com 8GB de RAM, rodando Linux Ubuntu 14.04.

Na Figura 4.2, é apresentada a arquitetura de navegação implementada no ROS, na qual pode-se ver os principais nós, tópicos e serviços desenvolvidos e o fluxo de dados entre eles. Note que da mesma forma que foi apresentado no Capítulo 3, o planejador, nó *planner*, tem apenas como entradas: um mapa local fornecido por um serviço do nó *laser_obstacles*, as transformadas entre os referenciais locais provida pelo ROS (tópico */tf)* e um sinal *booleano* de ativo que é recebido no tópico */state*.

4.2.1 Comandos do usuário

A missão do robô de serviço semiautônomo é definida pelo usuário por meio de uma sequência de comandos descrita em um arquivo texto lido pelo nó *task*. A missão pode ser

[11] Disponível em <http://ompl.kavrakilab.org/>.

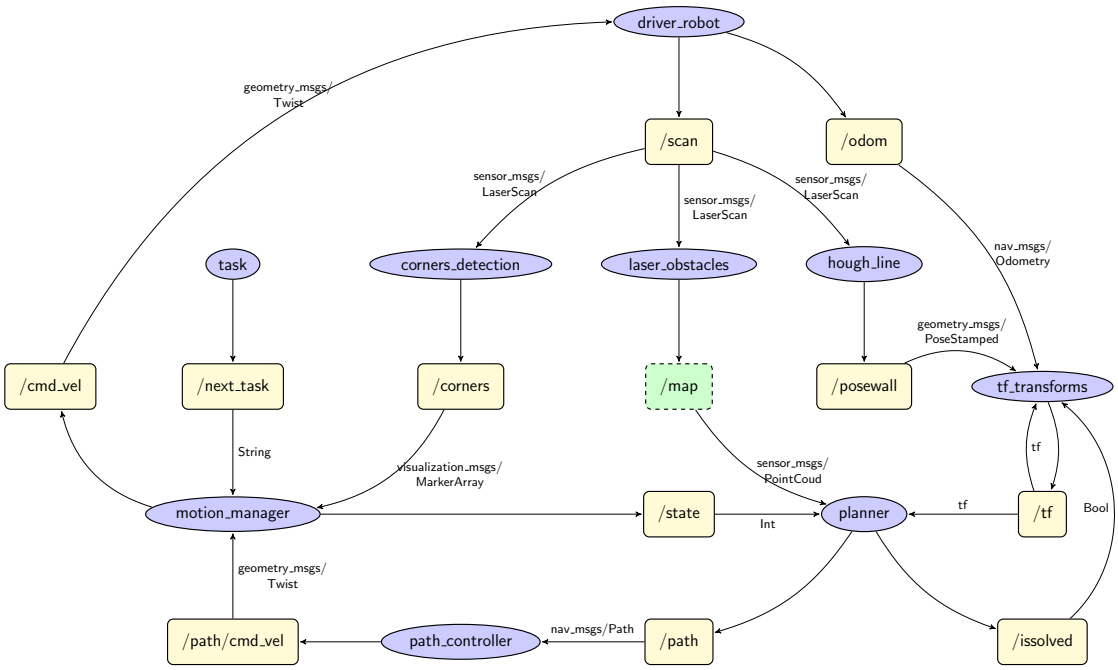


Figura 4.2 – Arquitetura de navegação implementada no ROS. As elipses representam os principais nós, os retângulos os tópicos, o retângulo pontilhado representa o único serviço implementado e as setas indicam o fluxo de dados.

alterada em tempo de execução e o robô, ao terminar de executar toda a sequência de comandos especificada no arquivo texto, fica aguardando por uma nova missão determinada pelo usuário. Os comandos possíveis para isso são:

- `siga #`: o robô irá seguir pelo corredor autonomamente. Se for fornecido um número, representado por #, o robô seguirá o corredor por # metros, caso contrário até que uma interseção seja detectada pelo robô.
- `vire à direita #`: o robô irá virar no primeiro corredor à direita e irá segui-lo autonomamente, como no comando `siga`.
- `vire à esquerda #`: o robô irá virar no primeiro corredor à esquerda e irá segui-lo autonomamente, como no comando `siga`.
- `retorne #`: o robô irá girar 180° e irá seguir o corredor autonomamente, como no comando `siga`.
- `pare`: o robô irá parar seu movimento.

Os comandos virar e retornar foram implementados em duas etapas: (i) o robô executa uma curva pré-estabelecida para virar ou para retornar; (ii) o robô executa o comando `siga`. Na primeira etapa não é utilizado um planejador pela sua simplicidade e a curta distância do

movimento, sendo, então, essa etapa determinada pelo módulo gerenciador, nó *motion_manager*, que também garante que não haverão colisões. Na segunda etapa, o planejador descrito pelo Algoritmo 3 combinado com o controlador de caminhos é selecionado pelo gerenciador para determinar as velocidades necessárias para que o robô siga o corredor.

Para o problema específico de navegar em longos corredores, o comando `siga` pode ser codificado facilmente por um campo vetorial bi-dimensional $\mathbf{f}(\vec{r}) = [f_x, f_y]^T$, onde $\vec{r}$ é a posição do centro do robô no referencial $\{L_k\}$, definido como:

$$\mathbf{f}(\vec{r}) = \begin{bmatrix} 1 \\ k(D_0 - D) \end{bmatrix}, \tag{4.1}$$

onde k é uma constante que determina o quão rápido o robô, estando a uma distância D da parede à sua direita, irá em direção à distância desejada D_0 dessa parede, como ilustrado na Figura 4.3. O valor de k escolhido neste trabalho foi de $0,35$, de forma que um robô seguindo o campo vetorial não fosse levado bruscamente para a distância D_0. Essa distância foi escolhida entre $0,70\,\text{m}$ e $1,0\text{m}$ de forma que o robô ficasse próximo à parede, considerando sua dimensão, e não atrapalhasse a circulação por esse corredor.

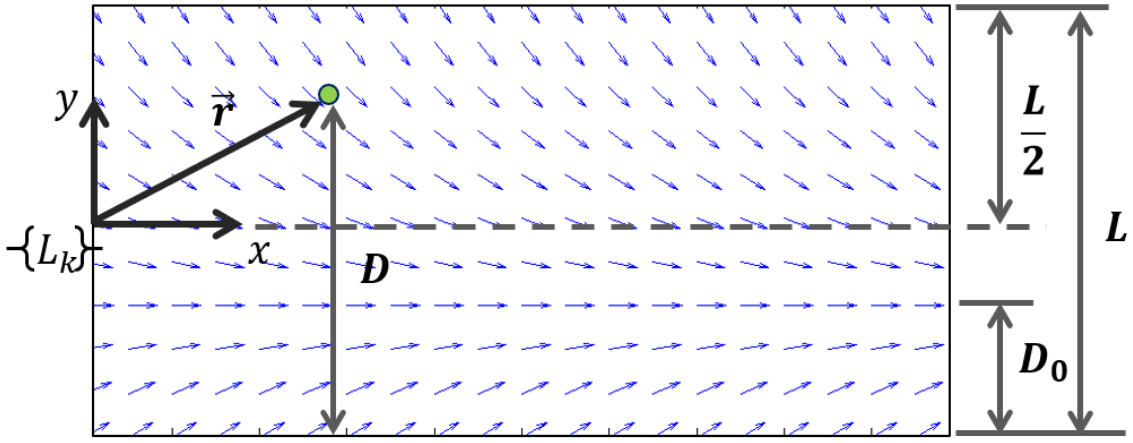
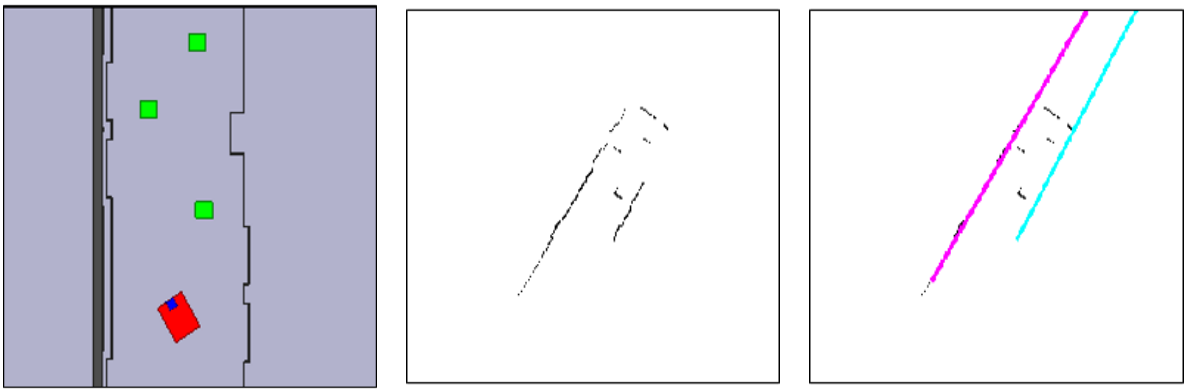


Figura 4.3 – Ilustração do campo vetorial definido pela Equação (4.1). Esse campo vetorial codifica o comando `siga` de forma a conduzir o robô para uma distância D_0 da parede à sua direita. O vetor $\vec{r}$ representa a posição do robô no referencial $\{L_k\}$ e a distância D representa a sua distância em relação à parede da direita do corredor, cuja largura é L.

4.2.2 Localização e determinação dos referenciais locais

A localização do robô é realizada por meio da odometria, calculada pelo sistema interno do robô a partir dos dados dos odômetros conectados às suas rodas. A cada intervalo de tempo, essa odometria é transformada para o referencial corrente, atualizando esse referencial a cada t_s segundos. Nesse problema específico, a orientação do referencial foi escolhida como sendo a orientação da parede para facilitar o cálculo do campo vetorial, que tem uma das componentes paralela ao corredor.

A detecção das paredes do corredor foi implementada utilizando um método baseado na Transformada de Hough Probabilística [Galamhos, Matas e Kittler 1999] pelo nó *hough_line*. Primeiramente, os dados do *laser* são convertidos em uma imagem binária, Figura 4.4(b). Dessa forma, por meio da Transformada de Hough, as retas mais significativas nessa imagem, que representam as paredes do corredor, são detectadas, como mostrado na Figura 4.4(c). Obtendo-se os parâmetros das retas mais significativas, é possível determinar a distância do robô à parede e a sua orientação com respeito às mesmas. Obtidas essas informações, define-se a orientação do referencial local atual, $\{L_k\}$, sempre paralela à parede e no centro do corredor. É importante mencionar que foi considerado nessa implementação que a largura do corredor é conhecida ou medida durante a navegação do robô, de maneira que fosse possível determinar esse referencial mesmo que apenas uma das paredes fosse detectada.



(a) (b) (c)

Figura 4.4 – Processo da utilização da Transformada de Hough para detecção das paredes do corredor. (a) Exemplo de ambiente simulado no instante em que os dados de (b) foram obtidos. (b) Imagem binária obtida utilizando os dados do laser do ponto de vista do robô. (c) Retas mais significativas detectadas por meio da Transformada de Hough aplicada em (b).

4.2.3 Detecção de fim de corredor

A detecção de fim de corredor é utilizada para a definição de término de alguns comandos fornecidos pelo usuário, como visto na Seção 4.2.1. Essa detecção é realizada pelo nó *corners_detection*, utilizando apenas os dados do *laser*.

O algoritmo executado por esse nó, primeiramente, encontra pontos de descontinuidades e de quinas nos dados do *laser*, formando, dois a dois, segmentos de retas que representam tanto a parede quanto os obstáculos no ambiente. Em seguida, são selecionados apenas os segmentos de retas que possuem um comprimento mínimo, gerando um conjunto de segmentos de retas que contém apenas aqueles com maior probabilidade de representar uma parede. Então, toma-se o segmento de reta mais significativo como referência e forma-se dois novos conjuntos: (A) um conjunto de segmentos paralelos ao segmento de referência e (B) um conjunto de segmentos perpendiculares ao segmento de referência. Por fim, quando houver ao menos três segmentos de retas com probabilidade de representar uma parede, sendo que dois deles fazem parte do conjunto

A e o terceiro do conjunto B, o nó informa que aconteceu uma detecção de fim de corredor, como apresentado nas três situações da Figura 4.5, em que as linhas em azul representam os segmentos de retas selecionados para verificação do fim do corredor.

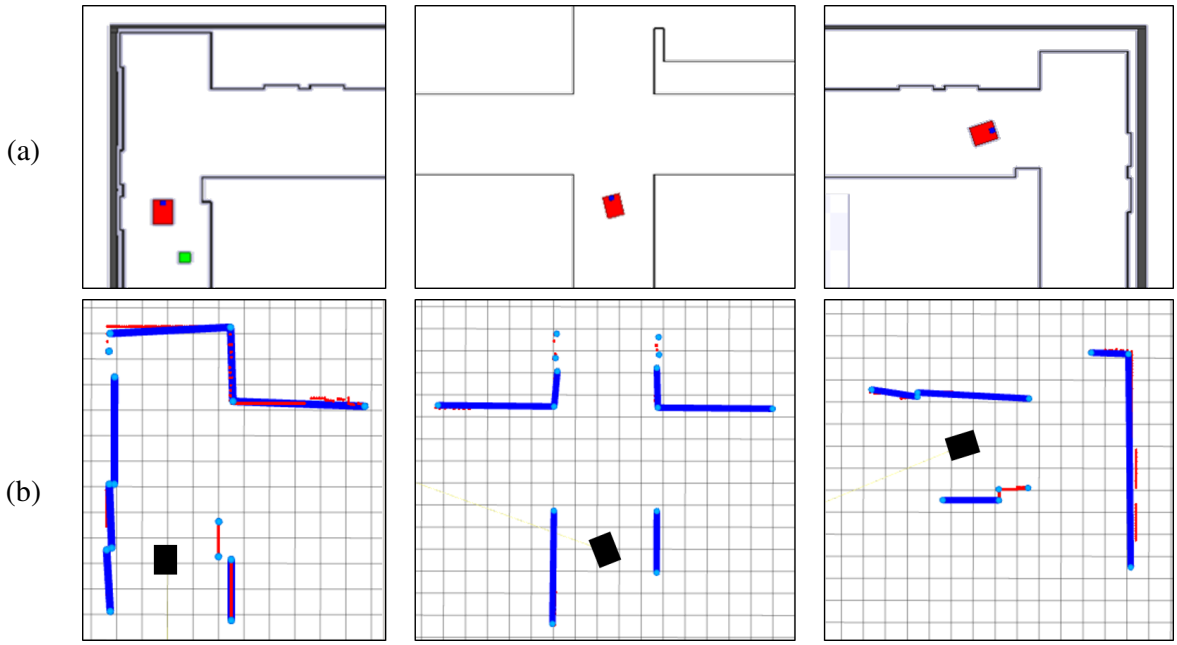


Figura 4.5 – Detecção de fim de corredor realizada pelo nó *corners_detection* para as três situações apresentadas em (a). O processo de detecção é ilustrado em (b) para cada uma dessas situações, em que os pontos em vermelho representam os dados do laser, o retângulo preto o robô, os pequenos círculos em azul claro representam a detecção de quinas e os pontos de descontinuidade e as linhas em azul são segmentos de retas selecionados para identificação da quantidade de retas ortogonais e paralelas.

4.3 Simulações

Por meio das simulações avaliou-se o funcionamento geral da estratégia a partir de uma sequência de comandos do usuário na Subseção 4.3.1. Também procurou-se verificar a vantagem de se utilizar o planejador proposto neste trabalho, apresentado na Seção 3.2 , em relação a outros possíveis planejadores baseados em métodos probabilísticos (Subseção 4.3.2) e comparar a estratégia de navegação proposta neste trabalho com outras estratégias: (i) estratégia local reativa utilizando campos potenciais (Subseção 4.3.3) e (ii) estratégia de navegação com mapeamento utilizando SLAM para o caso de desvio de obstáculos dinâmicos (Subseção 4.3.4).

4.3.1 Simulação do funcionamento geral da estratégia de navegação

Com o objetivo de avaliar o funcionamento geral da estratégia de navegação, um usuário definiu a seguinte sequência de comandos para alcançar dois alvos: (1) `siga`; (2) `vire à direita`; (3) `siga`; (4) `vire à direita`; (5) `siga`; (6) `pare`; (7) `retorne`; (8) `vire à esquerda`; (9) `vire à esquerda 5` (metros).

O caminho realizado pelo robô e os alvos são representados na Figura 4.6. Observe que na primeira parte do trajeto há obstáculos estáticos e o robô desvia deles com sucesso. A cada cruzamento de corredores é iniciado o seguimento de um novo comando da sequência definida pelo usuário. O vídeo dessa simulação pode ser encontrado em <https://youtu.be/kl01d0wOE3s>.

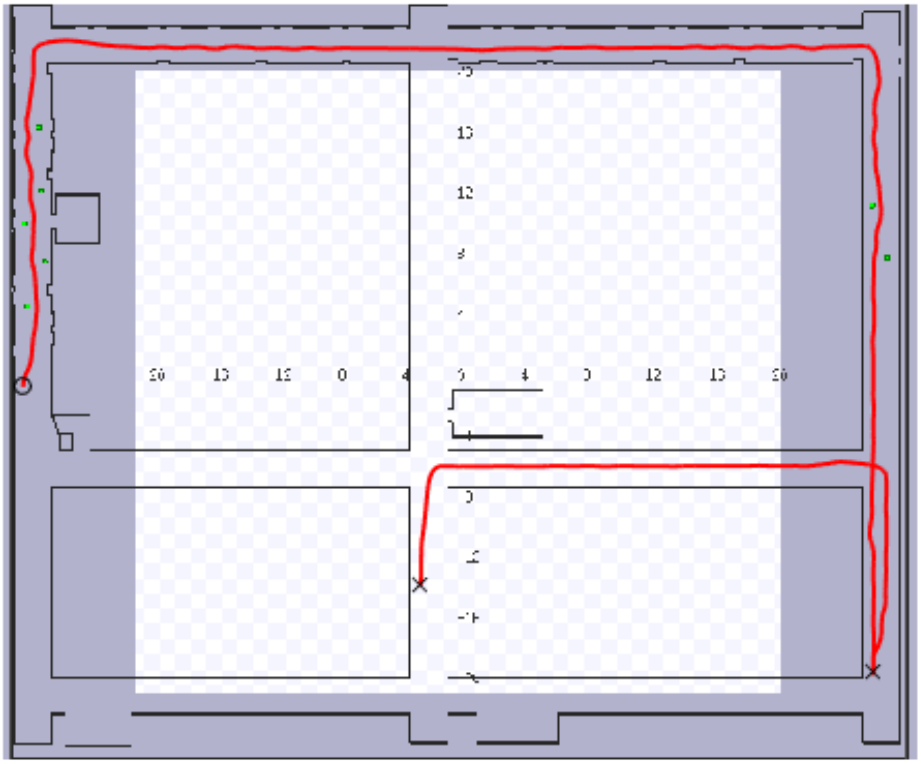


Figura 4.6 – Simulação do funcionamento geral da estratégia de navegação. O caminho realizado pelo robô (em vermelho), a partir do ponto inicial (pequeno círculo), demonstra que o robô seguiu corretamente a sequência definida pelo usuário para alcançar os dois alvos indicados por um x.

4.3.2 Comparação entre diferentes planejadores probabilísticos

Na Seção 3.2, vimos que o planejador adotado na metodologia é baseado no RRT*, usando a função de custo especificada no Algoritmo 4, que avalia o Funcional 3.2 apenas para a configuração inicial (heurística). Esse planejador, identificado como RRT* *ball_sampler*, foi configurado da seguinte maneira:

- O funcional de custo foi definido conforme o Algoritmo 4, utilizando o campo vetorial apresentado na Equação (4.1).
- O parâmetro *steering* (η) foi definido como 1,0 m. Esse parâmetro é utilizado na função $Steer(q_i, q_j)$ (linha 5 do Algoritmo 3) que gera uma nova amostra no grafo distante, no

máximo, η metros do nó mais próximo no grafo já existente. Com isso, um valor muito pequeno pode gerar um caminho pouco suave e para um valor grande a heurística utilizada pelo planejador pode não ser mais válida. Dessa forma, o valor de $\eta = 1{,}0\,\text{m}$, especificado empiricamente, é um valor que satisfaz tanto a suavidade quanto a validade da heurística. Esse valor também é adotado pela OMPL como o valor padrão para os planejadores baseados no RRT*.

- O raio de planejamento foi definido igual a 5 m. Assim, a função *AmostraLivre*() (linha 3 do Algoritmo 3) sorteia uma amostra no espaço livre dentro de um disco com esse raio. Observe que o planejador não utiliza o amostrador uniforme configurado como padrão da OMPL.

- O alvo é definido pelo próprio planejador, por meio da função $ObtemAlvo(G)$ (linha 24 do Algoritmo 3), que retorna o nó do grafo G que possui o menor custo com relação ao funcional especificado e que está na borda da região de planejamento.

A Figura 4.7 ilustra o caminho (em verde) e o grafo (em vermelho) gerado pelo planejador RRT* *ball_sampler* com essa configuração. Repare que no grafo gerado não há nós dentro de um raio em torno dos obstáculos, pois o robô foi aproximado por um disco de diâmetro igual ao seu comprimento. Essa aproximação garante que não haverá colisões se o robô seguir o caminho fornecido.

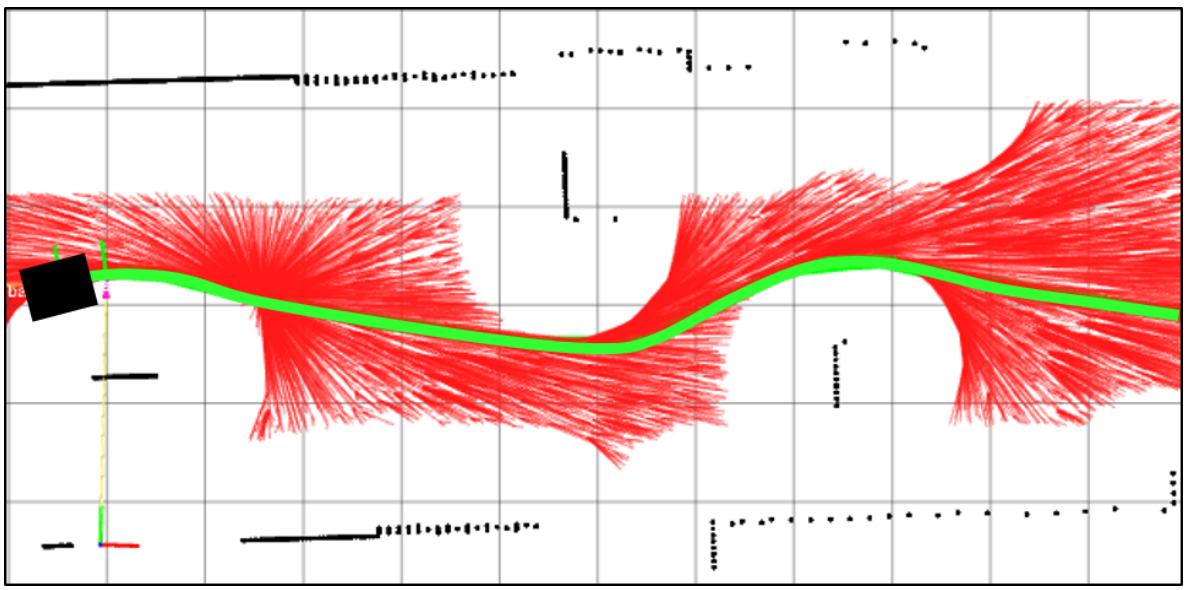


Figura 4.7 – Ilustração do caminho (em verde) e do grafo (em vermelho) gerado pelo planejador RRT* *ball_sampler* com as configurações escolhidas para este trabalho. Os pontos em preto representam os dados do *laser*.

Da maneira como foi proposta a estratégia de navegação, outros planejadores poderiam ter sido utilizados. Dessa forma, quatro planejadores com características diferentes foram escolhidos para serem comparados com o planejador adotado neste trabalho:

- VF-RRT (*Vector Field Rapidly Exploring Random Tree*) [Ko, Kim e Park 2014]: é um planejador baseado no RRT (*Rapidly Exploring Random Tree*) clássico [Kuffner e LaValle 2000], porém busca minimizar o Funcional (3.2), ou seja, levando em consideração um campo vetorial fornecido pelo usuário. Esse planejador não é ótimo e fornece uma solução apenas quando é alcançada uma região em torno do alvo, não sendo um planejador *anytime*. Um parâmetro importante desse planejador é o parâmetro de exploração (E_s), valor entre 0 e 1, que determina a transição entre o algoritmo clássico do RRT ($E_s = 0$) e um planejador de caminho que tenta seguir rigorosamente o campo vetorial ($E_s = 1$). Dessa forma, esse planejador foi comparado utilizando os dois valores sugeridos pelos autores: $E_s = 0,35$ e $E_s = 0,85$, sendo que esse último foi o valor que obteve na maioria dos casos o melhor resultado no trabalho de [Ko, Kim e Park 2014].

- RRT* (*Optimal Rapidly Exploring Random Tree*) clássico: é um planejador que cria uma árvore, utilizando o Algoritmo 3, de forma a minimizar otimamente a distância euclideana, sendo um planejador assintoticamente ótimo e *anytime*. Os parâmetros utilizados foram aqueles definidos como padrão pela OMPL, sendo o principal $\eta = 1,0\,\text{m}$.

- RRT* *upstream*: é um planejador que também utiliza o Algoritmo 3, mas busca minimizar otimamente o Funcional (3.2). Esse planejador foi configurado usando dois valores diferentes para o parâmetro *steering*: $\eta = 1,0\,\text{m}$ e $2,5\,\text{m}$. O último valor foi escolhido para verificar se o aumento desse parâmetro interfere no cálculo do funcional a ser otimizado.

- RRT* *heuristic*: é um planejador semelhante aos dois últimos, mas que utiliza a heurística proposta neste trabalho, conforme o Funcional de custo (4). Diferencia da metodologia apenas por utilizar o amostrador uniforme padrão da OMPL. Também foi configurado para os mesmos valores anteriores de η.

Outros possíveis planejadores, como o BIT* [Gammell, Srinivasa e Barfoot 2015] que cria uma árvore informada e é, normalmente, mais rápido que o RRT*, não foram testados, por não ficar claro como uma heurística que leva em consideração um campo vetorial poderia ser implementada.

Assim, para realizar comparações entre os planejadores, definiu-se as mesmas situações de simulação: ambiente, campo vetorial, ponto de partida, ponto de chegada, tempo de planejamento e a mesma CPU. O ambiente de simulação escolhido para os testes foi um corredor de largura de três metros e comprimento de seis metros[12]. O campo vetorial para todos os testes foi definido conforme a Equação (4.1), considerando $k = 0,35$ e uma distância desejada da parede $D_0 = 1,0\,\text{m}$. Definiu-se, por sua vez, um referencial fixo no centro do corredor, o ponto de partida sendo $(x = 0,0\,;\, y = 1,5)\,\text{m}$ e o alvo sendo $(x = 5,0\,;\, y = -0,5)\,\text{m}$ com uma tolerância de $0,1\,\text{m}$. Também, é importante mencionar que todos os planejadores foram testados utilizando

[12] Foi escolhido um comprimento de seis metros, levando em consideração o raio de planejamento escolhido na implementação da estratégia deste trabalho.

a implementação sem modificações da OMPL e que o robô foi considerado, nesse caso, como um ponto no espaço 2D.

Foram realizados 100 (cem) testes com cada planejador para as mesmas três configurações de ambiente: (i) sem obstáculos, (ii) com vinte e cinco obstáculos e (iii) com cinquenta obstáculos. Para cada teste, os obstáculos foram gerados aleatoriamente, tanto a sua posição no ambiente, sorteada dentro da faixa de $x \in [0{,}5\,;\,4{,}5]\,\text{m}$ e $y \in [-1{,}0\,;\,1{,}0]\,\text{m}$, quanto o valor de seu raio, sorteado entre $[0{,}05\,;\,0{,}2]\,\text{m}$. Os planejadores foram comparados em relação a cinco critérios:

- Porcentagem das vezes em que o planejador conseguiu encontrar uma solução até o alvo, no intervalo de tempo definido e com uma tolerância da norma do erro de 0,1 m.
- Número de nós no grafo gerado pelo algoritmo do planejador;
- Custo euclideano do caminho em metros (somatório das distâncias euclideanas de cada segmento do caminho encontrado pelo algoritmo);
- Custo *upstream* definido pelo Funcional (3.2), que mede o custo do caminho em relação ao seguimento do campo vetorial;
- Suavidade do caminho definida pela OMPL[13] como:

$$\texttt{suavidade} = \sum_{i=2}^{n-1} \left(\frac{2(\pi - \arccos(a_i^2 + b_i^2 + c_i^2))}{a_i + b_i} \right)^2 \tag{4.2}$$

onde $a_i = \texttt{dist}(s_{i-2}, s_{i-1})$, $b_i = \texttt{dist}(s_{i-1}, s_i)$, $c_i = \texttt{dist}(s_{i-2}, s_i)$, s_i o elemento i-ésimo do caminho e $\texttt{dist}(s_i, s_j)$ é a distância entre dois elementos do caminho.

As Tabelas 4.1, 4.2 e 4.3 apresentam uma síntese dos resultados obtidos para cada planejador no ambiente, respectivamente, sem obstáculos, com vinte e cinco obstáculos e com cinquenta obstáculos, testados para os seguintes tempos de planejamento[14] (t_s) iguais à 0,45 s, 1,0 s, 2,0 s e 5,0 s. Esses resultados podem ser visualizados graficamente no Apêndice A.

A distância euclideana calculada entre o ponto de partida até o ponto de chegada para o ambiente sem obstáculos é de $\sqrt{29} \approx 5{,}38\,\text{m}$. Percebe-se pela Tabela 4.1 que para o menor tempo de planejamento testado, $t_s = 0{,}45\,\text{s}$, o caminho fornecido pelo planejador RRT* já tem um custo euclideano muito próximo do ótimo. Claramente, esse planejador também é o que fornece o caminho mais suave, já que o caminho ótimo para ele é uma reta que parte do ponto de partida até o ponto de chegada, como visto na Figura 4.8. Os demais planejadores tendem

[13] A ideia da Equação 4.2 é calcular o ângulo externo do triângulo formado por segmentos de caminho consecutivos, usando o teorema de Pitágoras. Dessa forma, o ângulo externo é normalizado pela distância do segmento e contribui para a suavidade. O somatório de todas essas contribuições ao longo do caminho é definido como suavidade. Quanto mais próximo o valor for de zero, mais suave é considerado o caminho.

[14] O tempo de planejamento especificado é garantido pela OMPL.

Tabela 4.1 – Comparação entre os planejadores para um ambiente sem obstáculos.

tempo de		VF-RRT		RRT*	RRT* *upstream*		RRT* *heuristic*		RRT* *ball_sampler*	
planejamento		$E_s = 0,35$	$E_s = 0,85$		$\eta = 1,0$	$\eta = 2,5$	$\eta = 1,0$	$\eta = 2,5$	$\eta = 1,0$	$\eta = 2,5$
	solução (%)	82	6	**100**	79	78	**100**	**100**	**100**	**100**
	número de nós	12454	16082	1116	147	139	1087	912	1074	942
$t_s = 0,45s$	custo euclidiano	5,74	5,85	**5,39**	5,5	5,43	5,45	5,47	5,45	5,48
	custo *upstream*(x10^{-3})	378,63	409,8	96,02	126,31	48,56	**45,57**	63,73	45,91	65,11
	suavidade	118017,99	54595,4	**3,00**	125,23	17,58	18,46	39,59	19,03	40,21
	solução (%)	98	41	**100**	88	92	**100**	**100**	**100**	**100**
	número de nós	19464	30813	2520	234	210	2508	3290	2642	3348
$t_s = 1s$	custo euclidiano	5,74	5,77	**5,39**	5,46	5,43	5,45	5,48	5,45	5,48
	custo *upstream*(x10^{-3})	377,56	326,16	93,6	77,2	45,66	44,89	66,46	**44,54**	64,71
	suavidade	113328,79	50163,01	**1,00**	61,92	**14,71**	16,67	42,5	17,6	40,74
	solução (%)	**100**	95	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	número de nós	33427	47825	5905	422	394	6887	8291	6983	8485
$t_s = 2s$	custo euclidiano	5,74	5,74	**5,38**	5,44	5,43	5,45	5,48	5,45	5,48
	custo *upstream*(x10^{-3})	380,25	300,91	92,75	53,98	**43,76**	44,65	64,36	44,75	67,46
	suavidade	113441,23	44698,17	**0,58**	33,58	12,91	16,26	42,11	18,04	42,77
	solução (%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	número de nós	41880	50373	20127	1260	1245	22940	21536	22929	21041
$t_s = 5s$	custo euclidiano	5,75	5,75	**5,38**	5,43	5,43	5,45	5,48	5,45	5,48
	custo *upstream*(x10^{-3})	399,21	303,2	92,34	42,2	**41,57**	44,47	64,32	44,35	68,86
	suavidade	140374,84	44155,52	**0,41**	10,18	9,91	17,42	41,69	17,7	45,91

Tabela 4.2 – Comparação entre os planejadores para um ambiente com 25 obstáculos.

tempo de		VF-RRT		RRT*	RRT* *upstream*		RRT* *heuristic*		RRT* *ball_sampler*	
planejamento		$E_s = 0,35$	$E_s = 0,85$		$\eta = 1,0$	$\eta = 2,5$	$\eta = 1,0$	$\eta = 2,5$	$\eta = 1,0$	$\eta = 2,5$
	solução (%)	96	29	**100**	76	81	**100**	**100**	**100**	**100**
	número de nós	8762	11321	1010	147	136	984	561	976	567
$t_s = 0,45s$	custo euclidiano	6,23	5,81	**5,43**	5,76	5,55	5,49	5,53	5,49	5,53
	custo *upstream*(x10^{-3})	845,34	378,85	156,21	398,81	197,02	**99,2**	135,59	102,96	128,87
	suavidade	210134,66	59209,72	**50,78**	412,52	142,96	72,6	100,12	80,23	105,29
	solução (%)	98	51	**100**	90	96	**100**	**100**	**100**	**100**
	número de nós	14891	20112	1947	225	210	1963	1726	1968	1730
$t_s = 1s$	custo euclidiano	6,28	5,83	**5,43**	5,57	5,52	5,48	5,51	5,49	5,52
	custo *upstream*(x10^{-3})	908,72	396,42	150,08	212,68	160,28	**91,12**	112,05	91,79	115,15
	suavidade	213306,59	61618,49	**39,81**	220,53	159,85	66,06	80,4	66,64	78,81
	solução (%)	**100**	63	**100**	99	**100**	**100**	**100**	**100**	**100**
	número de nós	24673	33000	4597	421	381	5027	4678	5029	4705
$t_s = 2s$	custo euclidiano	6,23	5,81	**5,42**	5,53	5,49	5,48	5,51	5,48	5,51
	custo *upstream*(x10^{-3})	860,79	385,57	134,16	156,26	113,34	**83,49**	108,77	84,51	111,18
	suavidade	200148,46	61826,26	**33,13**	137,36	82,19	67,95	82,18	129,31	80,11
	solução (%)	**100**	91	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	número de nós	24282	56497	14601	1244	1115	17836	15953	17999	15309
$t_s = 5s$	custo euclidiano	6,24	5,79	**5,42**	5,47	5,47	5,47	5,5	5,47	5,5
	custo *upstream*(x10^{-3})	875,82	365,86	136,73	92,97	92,62	80,56	103,65	**80,55**	105,28
	suavidade	233035,38	63344,74	27,67	**72,79**	61,16	54,7	69,75	51,39	77,9

a seguir o campo vetorial, dessa forma, mesmo considerando $t_s = 5\,\text{s}$, esses planejadores não alcançam o mesmo custo euclideano do RRT*, como era esperado. Dado que o campo vetorial escolhido é um campo que varia pouco e considerando o ambiente sem obstáculos, repare que o custo *upstream* também é pequeno para o RRT*. Dessa forma, nessas condições, o RRT* poderia ser utilizado, aproveitando a simplicidade da sua função de custo e o caminho suave que ele fornece para todos os tempos de planejamento testados na Tabela 4.1.

Com relação a suavidade do caminho fornecido pelos planejadores, percebe-se que o caminho fica mais suave ao aumentar o tempo de planejamento, já que nessa condição o grafo

Tabela 4.3 – Comparação entre os planejadores para um ambiente com 50 obstáculos.

tempo de planejamento		VF-RRT $E_s = 0,35$	VF-RRT $E_s = 0,85$	RRT*	RRT* *upstream* $\eta = 1,0$	RRT* *upstream* $\eta = 2,5$	RRT* *heuristic* $\eta = 1,0$	RRT* *heuristic* $\eta = 2,5$	RRT* *ball_sampler* $\eta = 1,0$	RRT* *ball_sampler* $\eta = 2,5$
$t_s = 0,45s$	solução (%)	99	35	**100**	76	84	**100**	**100**	**100**	**100**
	número de nós	3331	10380	967	146	131	939	457	941	454
	custo euclidiano	6,85	5,99	**5,54**	6,08	5,83	5,65	5,7	5,62	5,71
	custo *upstream*(x10^{-3})	1497,58	553,09	319,23	838,78	566,9	279,27	347,1	**252,24**	345,87
	suavidade	320356,45	73111,75	**163,92**	548,88	415,75	314,87	325,23	299,68	310,83
$t_s = 1s$	solução (%)	**100**	41	**100**	96	95	**100**	**100**	**100**	**100**
	número de nós	4453	16926	1869	224	203	1755	1127	1740	1128
	custo euclidiano	6,86	6	**5,52**	5,86	5,73	5,6	5,64	5,59	5,64
	custo *upstream*(x10^{-3})	1506,85	565,1	269,95	611,43	458,28	227,46	269,51	**213,5**	268,43
	suavidade	295188,52	80006,46	**150,6**	524,53	372,81	240,64	271,57	238,33	255,9
$t_s = 2s$	solução (%)	**100**	62	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	número de nós	5001	30668	3721	440	350	3907	3186	3922	3216
	custo euclidiano	6,83	5,93	**5,5**	5,66	5,7	5,57	5,61	5,57	5,61
	custo *upstream*(x10^{-3})	1478,72	509,47	240,83	358,64	374,76	**189,45**	220,72	190,31	224,52
	suavidade	303344,8	74992,56	**130,66**	374,88	373,63	203,44	240,92	174,84	235,08
$t_s = 5s$	solução (%)	**100**	80	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	número de nós	4597	47446	11422	1087	931	13803	11245	13449	11717
	custo euclidiano	6,85	5,94	**5,49**	5,6	5,6	5,56	5,58	5,56	5,58
	custo *upstream*(x10^{-3})	1501,86	522,55	214,64	255,59	249,1	**171,74**	194,52	172,35	196,64
	suavidade	308461,22	69493,97	**128,77**	297,52	277,3	178,64	198,51	173,24	194,44

gerado tem mais nós, aumentando a probabilidade de conexões nesse grafo, de forma a deixar o caminho mais suave.

Era esperado que o planejador RRT* *upstream* apresentasse o menor custo *upstream*, já que ele utiliza diretamente o Funcional (3.2). Porém o que se nota é que apenas após o tempo de planejamento ser igual ou superior a dois segundos é que ele obtém os melhores resultados, no caso de um ambiente sem obstáculos, sendo o menor custo *upstream* igual a $41,57 \times 10^{-3}$, obtido para um tempo de planejamento de cinco segundos. Isso é justificado pelo cálculo desse funcional requerer mais processamento computacional, como pode ser percebido pelo fato desse planejador apresentar o menor número de nós no grafo em relação aos demais planejadores. Dessa forma, verifica-se que para um tempo de planejamento inferior a dois segundos esse planejador não é adequado.

Ao aumentar o número de obstáculos no ambiente, percebe-se a diminuição do número de nós presentes nos grafos de todos planejadores, relacionado, principalmente, com aumento do custo computacional envolvido na verificação de um caminho livre de colisão. Também, percebe-se que fica mais difícil seguir o campo vetorial, como visto nas Figuras 4.9 e 4.10, o que justifica o aumento do custo *upstream* nas Tabelas 4.2 e 4.3. Nesses casos, os menores custos *upstream* são obtidos pelos planejadores RRT* *heuristic* e RRT* *ball_sampler*, ambos com $\eta = 1,0$. Esse fato evidencia a importância da heurística adotada neste trabalho com relação à melhora do desempenho desses planejadores comparado com o planejador RRT* *upstream*, que utiliza o Funcional (3.2). Observe, também, que a heurística utilizada nos planejadores RRT* *heuristic* e RRT* *ball_sampler*, como comentada no Capítulo 3, é válida quando a distância entre os nós é pequena e o campo vetorial varia pouco localmente. Dessa forma, constata-se que quanto menor for o parâmetro *steering*, η, melhor será a aproximação do Funcional (3.2) feita

pela heurística, já que esse parâmetro é utilizado para gerar um novo nó no grafo distante, no máximo, η metros do nó mais próximo no grafo já existente.

Outro aspecto que se observa é o aumento da variância nos resultados obtidos para os critérios analisados à medida que há mais obstáculos no ambiente, representada nos diagramas das Figuras A.1 - A.9, pois, tendo gerado em cada teste os obstáculos aleatoriamente no ambiente, pode-se ter situações em que ficou bem mais simples encontrar um caminho e outras em que ficou mais complicado encontrá-lo.

Pode-se concluir pelos resultados que o planejador VF-RRT não apresenta uma solução satisfatória para o problema proposto neste trabalho, gerando um caminho pouco suave e um custo *upstream* superior aos demais planejadores. Além do fato que, mesmo gerando uma quantidade mais elevada de nós, quase seis vezes mais em alguns casos, nem sempre encontra uma solução a tempo. O parâmetro de exploração (E_s) mais adequado foi de 0,35, para o qual o planejador encontrou solução para a maioria das vezes, diferentemente de quando utilizado o valor $E_s = 0{,}85$. O fato desse planejador não ser *anytime* e obter uma solução em um tempo médio em torno de 2 s, também não possibilita utilizá-lo para tempos inferiores a esse, dada a menor chance de se encontrar um caminho, diferentemente dos planejadores *anytime* que melhoram o caminho à medida que se aumenta o tempo de planejamento, como mostrado nas Figuras 4.8 - 4.10.

Por fim, analisando os valores médios, o planejador proposto neste trabalho, RRT* *ball_sampler*, juntamente com o planejador RRT* *heuristic* são os mais adequados para a metodologia proposta no Capítulo 3, considerando, principalmente, o desempenho desses planejadores no seguimento de um campo vetorial para um ambiente com obstáculos e um tempo de planejamento pequeno. Esse aspecto é essencial para a metodologia, já que o campo vetorial a ser seguido codifica um comando do usuário. A diferença entre esses dois planejadores está no amostrador, que não gera uma diferença expressiva nos resultados obtidos. A vantagem mais significativa do RRT* *ball_sampler* em relação ao RRT* *heuristic* é o fato de não necessitar explicitamente de um alvo, tornando a sua utilização mais simples, principalmente, quando o objetivo é seguir comandos de alto nível (siga em frente, vira à direita etc) fornecidos por um usuário, como é o caso deste trabalho. Observou-se também que o valor de η que apresentou o melhor resultado para os dois planejadores foi com $\eta = 1{,}0$.

Para verificar estatisticamente se os valores médios apresentados nas Tabelas 4.1-4.3 são significativamente diferentes em relação aos valores obtidos pelo planejador usado neste trabalho, foi realizado o Teste-T [Montgomery e Runger 2003] dois a dois com esse planejador utilizando $\eta = 1{,}0$. O `valor-p` obtido por esse teste para cada uma das situações e para cada planejador comparado é apresentado nas Tabelas 4.4-4.6. Observe que quando `valor-p` $< \alpha$, em que α é o nível de significância, há evidência estatística suficiente para rejeitar a hipótese nula (valores médios entre os critérios comparados são iguais). Dessa forma, considerando $\alpha = 0{,}05$, pode-se confirmar estatisticamente a escolha do planejador conforme analisado anteriormente,

levando em consideração os critérios de suavidade e custo *upstream*, principalmente, em um ambiente com obstáculos.

Tabela 4.4 – `Valor-p` obtido utilizando o Teste-T dois a dois com o planejador de referência: RRT* *ball_sampler* ($\eta = 1{,}0$), para a condição do ambiente sem obstáculos. Em negrito estão evidenciados os valores em que não há evidência estatística suficiente para dizer que os valores médios são diferentes.

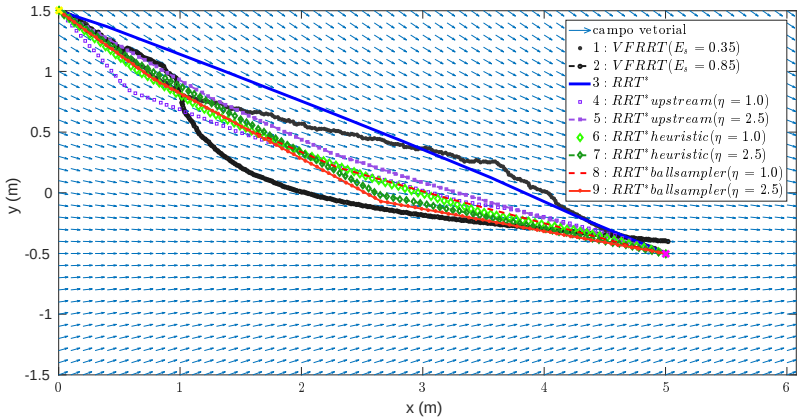
planejador	comparado:	VF-RRT		RRT*	RRT* *upstream*		RRT* *heuristic*		RRT* *ball_sampler*
		$E_s = 0,35$	$E_s = 0,85$		$\eta = 1{,}0$	$\eta = 2,5$	$\eta = 1{,}0$	$\eta = 2,5$	$\eta = 2,5$
$t_s = 0.45s$	custo *upstream*	5.9934e-91	3.2323e-68	3.4932e-62	1.4726e-24	0.0040215	**0.4094**	5.3319e-22	7.5476e-24
	suavidade	2.0432e-86	8.1563e-59	1.2468e-40	4.5897e-23	**0.48222**	**0.55632**	1.2829e-23	1.7136e-27
$t_s = 1.0s$	custo *upstream*	1.5992e-94	7.3606e-79	1.0755e-91	1.5819e-21	**0.23727**	**0.23175**	6.5225e-26	2.3725e-29
	suavidade	2.0128e-92	1.0502e-58	2.9356e-33	3.9465e-16	0.009106	**0.3577**	6.0221e-29	1.8133e-27
$t_s = 2.0s$	custo *upstream*	9.7498e-87	1.01e-72	2.0404e-103	6.2515e-16	0.0015982	**0.73229**	1.5159e-25	6.1923e-29
	suavidade	1.9533e-87	5.6084e-56	2.4365e-39	0.0097312	3.18e-08	0.032542	2.8824e-23	9.3489e-31
$t_s = 5.0s$	custo *upstream*	9.4247e-68	8.5775e-74	1.6453e-117	4.7589e-19	8.5739e-27	**0.62522**	1.7699e-29	8.3347e-33
	suavidade	1.5968e-31	2.2302e-61	4.3335e-32	6.1221e-11	4.9711e-12	**0.80928**	7.0431e-24	8.232e-22

Tabela 4.5 – `Valor-p` obtido utilizando o Teste-T dois a dois com o planejador de referência: RRT* *ball_sampler* ($\eta = 1{,}0$), para a condição do ambiente com 25 obstáculos. Em negrito estão evidenciados os valores em que não há evidência estatística suficiente para dizer que os valores médios são diferentes.

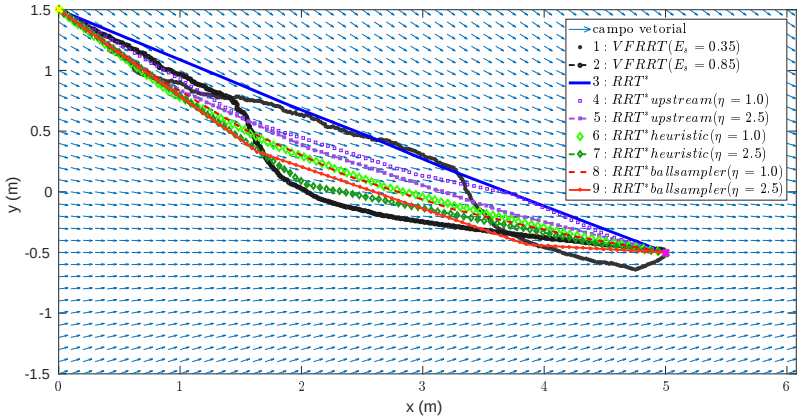
planejador	comparado:	VF-RRT		RRT*	RRT* *upstream*		RRT* *heuristic*		RRT* *ball_sampler*
		$E_s = 0,35$	$E_s = 0,85$		$\eta = 1{,}0$	$\eta = 2,5$	$\eta = 1{,}0$	$\eta = 2,5$	$\eta = 2,5$
$t_s = 0.45s$	custo *upstream*	2.5781e-34	2.3376e-58	1.3739e-11	5.3778e-26	5.6883e-10	**0.11127**	1.0503e-05	1.2323e-07
	suavidade	1.1975e-40	4.7479e-90	0.0007197	1.1576e-19	0.00041019	**0.25277**	0.025336	0.0151
$t_s = 1.0s$	custo *upstream*	4.0915e-33	1.2343e-48	1.1535e-12	1.0353e-13	1.1153e-07	**0.57989**	5.8588e-11	4.9366e-08
	suavidade	2.0205e-35	1.2845e-67	0.00069256	3.6386e-12	0.0048523	**0.93725**	0.010145	**0.067425**
$t_s = 2.0s$	custo *upstream*	1.4842e-38	4.389e-45	1.2629e-13	2.292e-09	1.4069e-12	**0.29216**	1.3642e-12	4.2403e-13
	suavidade	1.1791e-45	8.5586e-71	**0.17733**	**0.9119**	**0.51049**	**0.38902**	**0.50667**	**0.48699**
$t_s = 5.0s$	custo *upstream*	1.3188e-40	1.2611e-42	3.0675e-14	1.8592e-08	7.2285e-08	**0.98017**	2.9867e-12	8.3866e-11
	suavidade	3.1571e-36	3.7847e-62	2.4047e-06	0.013059	**0.052722**	**0.23366**	2.8e-06	0.0051002

Tabela 4.6 – `Valor-p` obtido utilizando o Teste-T dois a dois com o planejador de referência: RRT* *ball_sampler* ($\eta = 1{,}0$), para a condição do ambiente com 50 obstáculos. Em negrito estão evidenciados os valores em que não há evidência estatística suficiente para dizer que os valores médios são diferentes.

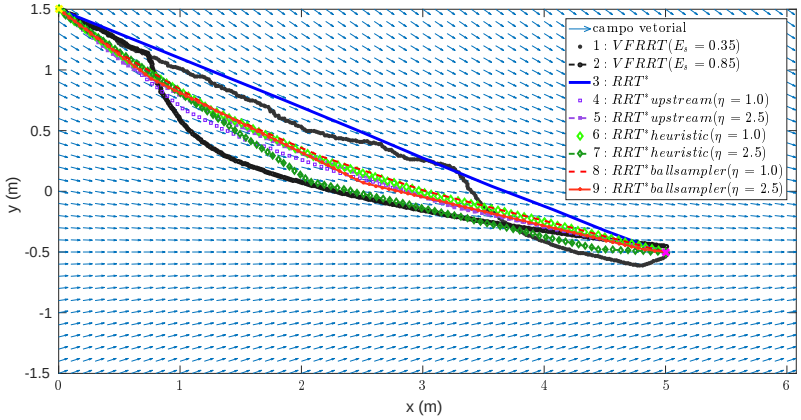
planejador	comparado:	VF-RRT		RRT*	RRT* *upstream*		RRT* *heuristic*		RRT* *ball_sampler*
		$E_s = 0,35$	$E_s = 0,85$		$\eta = 1{,}0$	$\eta = 2,5$	$\eta = 1{,}0$	$\eta = 2,5$	$\eta = 2,5$
$t_s = 0.45s$	custo *upstream*	6.2326e-38	5.4226e-23	1.3344e-06	3.4417e-26	3.4428e-17	0.022391	8.8097e-07	3.4658e-09
	suavidade	3.495e-45	1.7661e-71	5.012e-07	3.4817e-06	0.012499	**0.58056**	**0.61927**	**0.69567**
$t_s = 1.0s$	custo *upstream*	5.4568e-43	1.4788e-23	6.0968e-08	3.4231e-23	2.1518e-17	0.0077017	7.3079e-12	1.3444e-07
	suavidade	8.4108e-54	1.4613e-41	0.00028126	5.1848e-09	0.021639	**0.90708**	**0.15786**	**0.36589**
$t_s = 2.0s$	custo *upstream*	1.2553e-39	1.6759e-21	4.325e-08	3.0355e-18	3.47e-14	**0.75796**	3.5793e-10	1.2522e-09
	suavidade	5.1869e-52	2.3816e-53	0.0014171	1.6884e-05	3.6952e-06	0.036579	0.00079386	0.00025453
$t_s = 5.0s$	custo *upstream*	6.6969e-40	1.2132e-20	2.8996e-08	3.2131e-08	6.9589e-13	**0.59543**	2.2564e-10	8.886e-10
	suavidade	1.6167e-59	9.5337e-50	0.0043335	4.0854e-05	0.00017652	**0.65642**	**0.029744**	**0.14719**



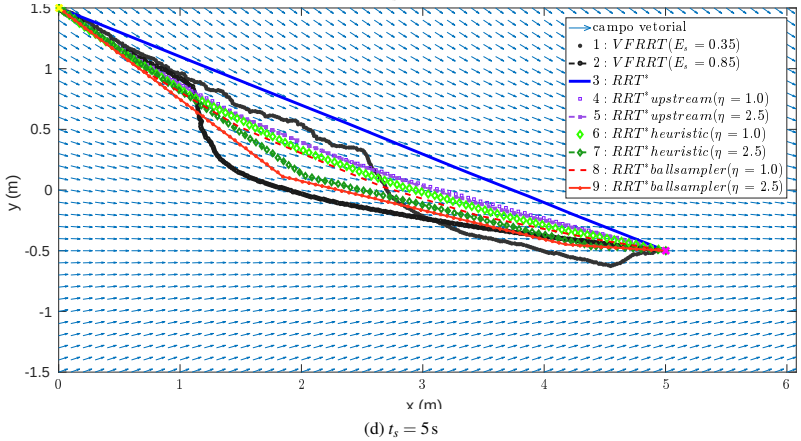
(a) $t_s = 0,45\,\text{s}$.



(b) $t_s = 1\,\text{s}$

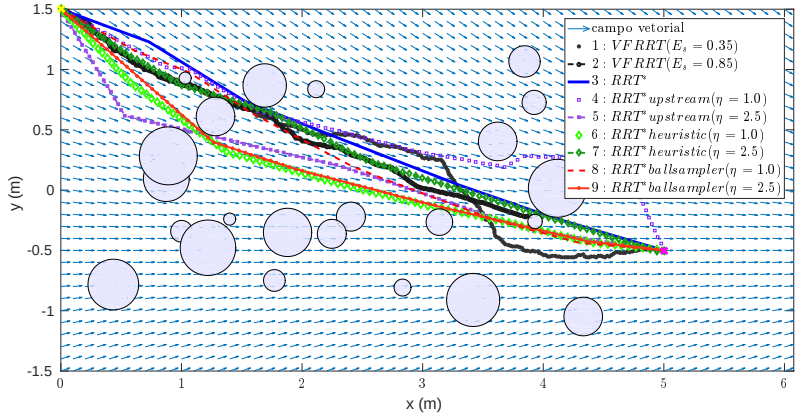


(c) $t_s = 2\,\text{s}$

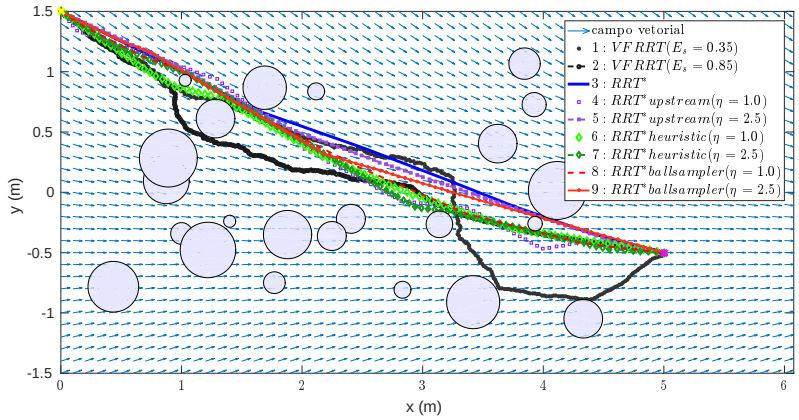


(d) $t_s = 5\,\text{s}$

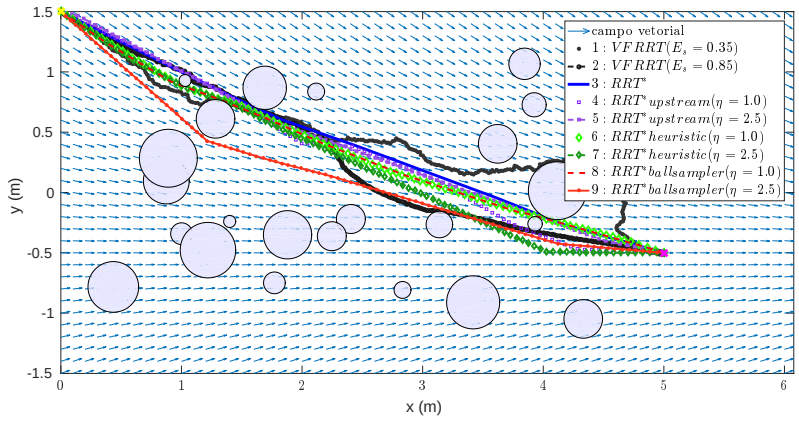
Figura 4.8 – Comparativo entre os caminhos obtidos por cada planejador em um ambiente sem obstáculos ao variar o tempo de planejamento. Os limites inferior e superior representam as paredes do corredor.



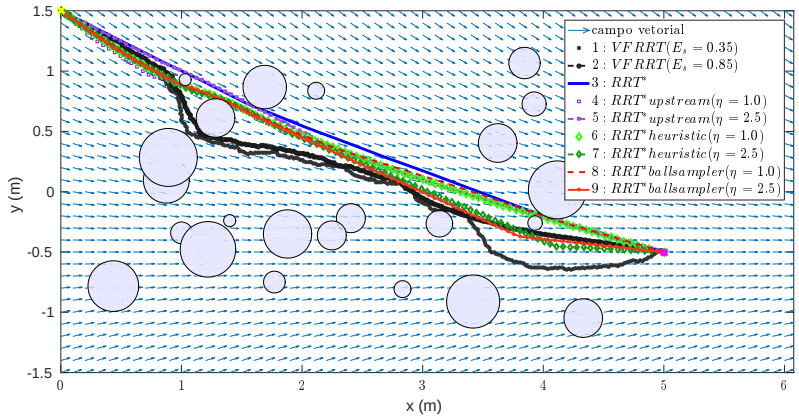
(a) $t_s = 0,45\,\text{s}$.



(b) $t_s = 1\,\text{s}$

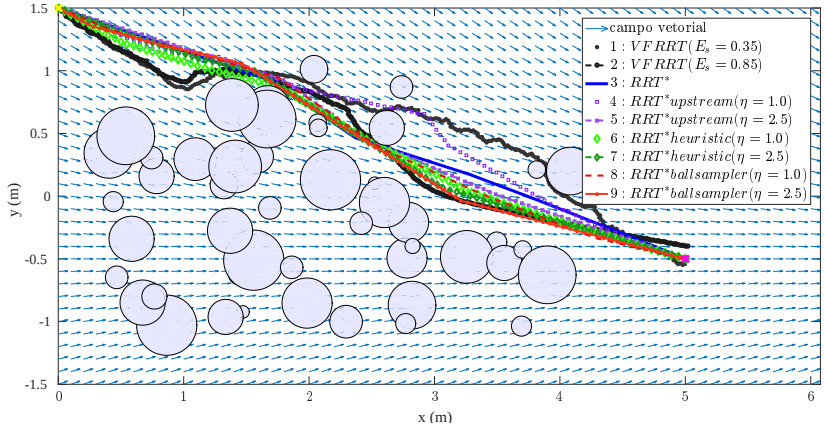


(c) $t_s = 2\,\text{s}$

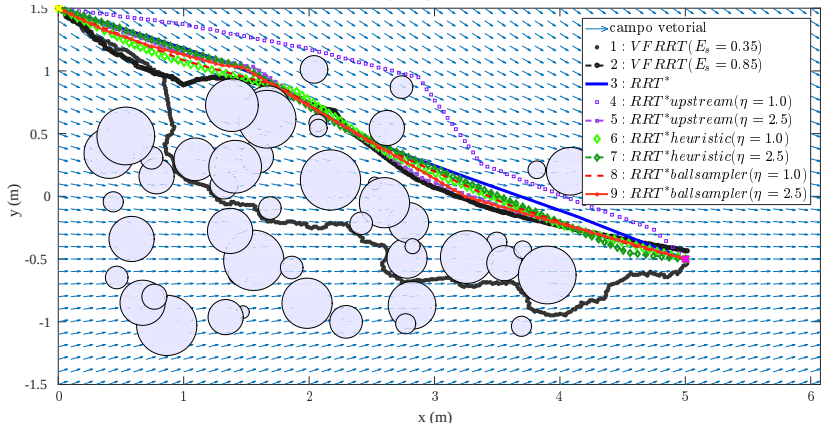


(d) $t_s = 5\,\text{s}$

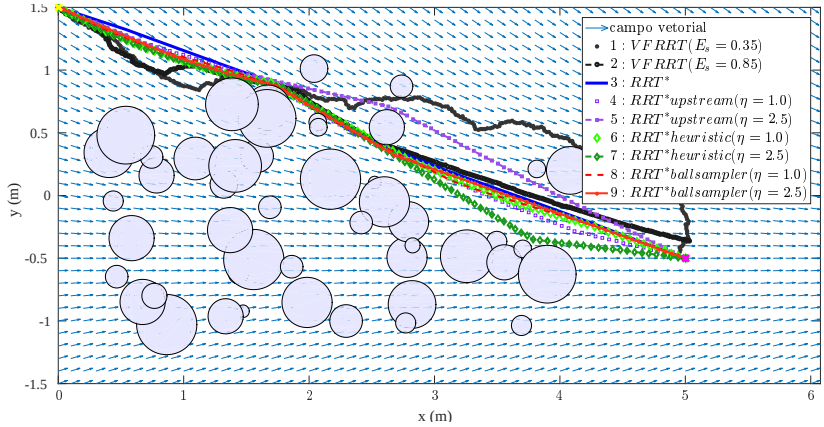
Figura 4.9 – Comparativo entre os caminhos obtidos por cada planejador em um ambiente com 25 obstáculos ao variar o tempo de planejamento. Os limites inferior e superior representam as paredes do corredor.



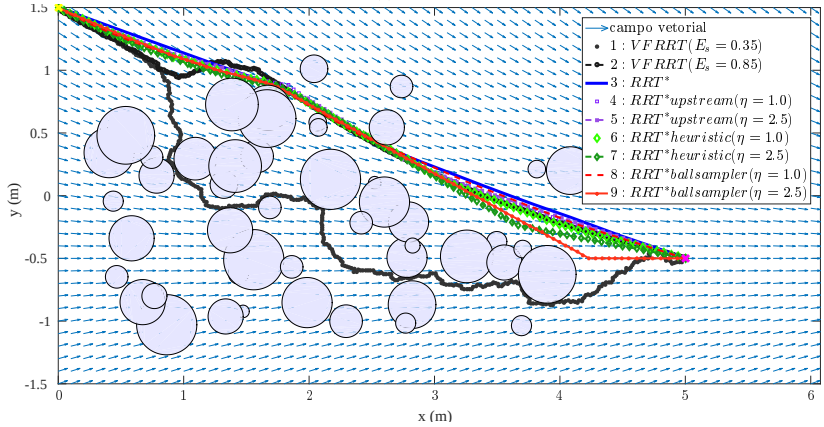
(a) $t_s = 0,45\,\mathrm{s}$.



(b) $t_s = 1\,\mathrm{s}$



(c) $t_s = 2\,\mathrm{s}$



(d) $t_s = 5\,\mathrm{s}$

Figura 4.10 – Comparativo entre os caminhos obtidos por cada planejador em um ambiente com 50 obstáculos ao variar o tempo de planejamento. Os limites inferior e superior representam as paredes do corredor.

4.3.3 Comparação entre a estratégia proposta e uma estratégia de navegação local reativa

Com o objetivo de comparar a estratégia proposta com uma estratégia de navegação local reativa, foi implementado um método de desvio de obstáculos a partir de campos potenciais, como descrito em [Choset *et al.* 2005]. Dessa forma, o somatório das contribuições do gradiente do campo potencial atrativo e do campo potencial repulsivo determinam a velocidade que o robô deve seguir em cada posição do espaço. Para tal, o campo vetorial normalizado descrito pela Equação (4.1) foi utilizado como o gradiente do campo potencial atrativo, de maneira que o robô seguirá esse campo vetorial na ausência de obstáculos. Já o gradiente do campo potencial repulsivo foi obtido por meio da soma da contribuição de cada obstáculo detectado pelo *laser*:

$$\nabla U_i(\vec{r}) = \begin{cases} k_r \left(\dfrac{1}{Q^*} - \dfrac{1}{d_i} \right), d_i \leq Q^* \\ 0, d_i > Q^* \end{cases}, \tag{4.3}$$

sendo que $\nabla U_i(\vec{r})$ é a contribuição para o gradiente do campo potencial repulsivo referente à um obstáculo i, quando o robô está na posição $\vec{r}$ a uma distância d_i desse obstáculo. A distância máxima a ser considerada para contribuir na repulsão é dada por Q^* e o ganho do gradiente por k_r.

Para as mesmas condições, foram obtidos os caminhos realizados pelo robô tanto utilizando a estratégia deste trabalho quanto utilizando campos potenciais, como apresentado na Figura 4.11. Por meio da função de suavidade da OMPL, definida pela Equação (4.2), e repetindo essa simulação por 10 (dez) vezes, foram obtidos os valores médios do custo euclideano, do custo *upstream* e da suavidade dos caminhos realizados, conforme apresentado na Tabela 4.7.

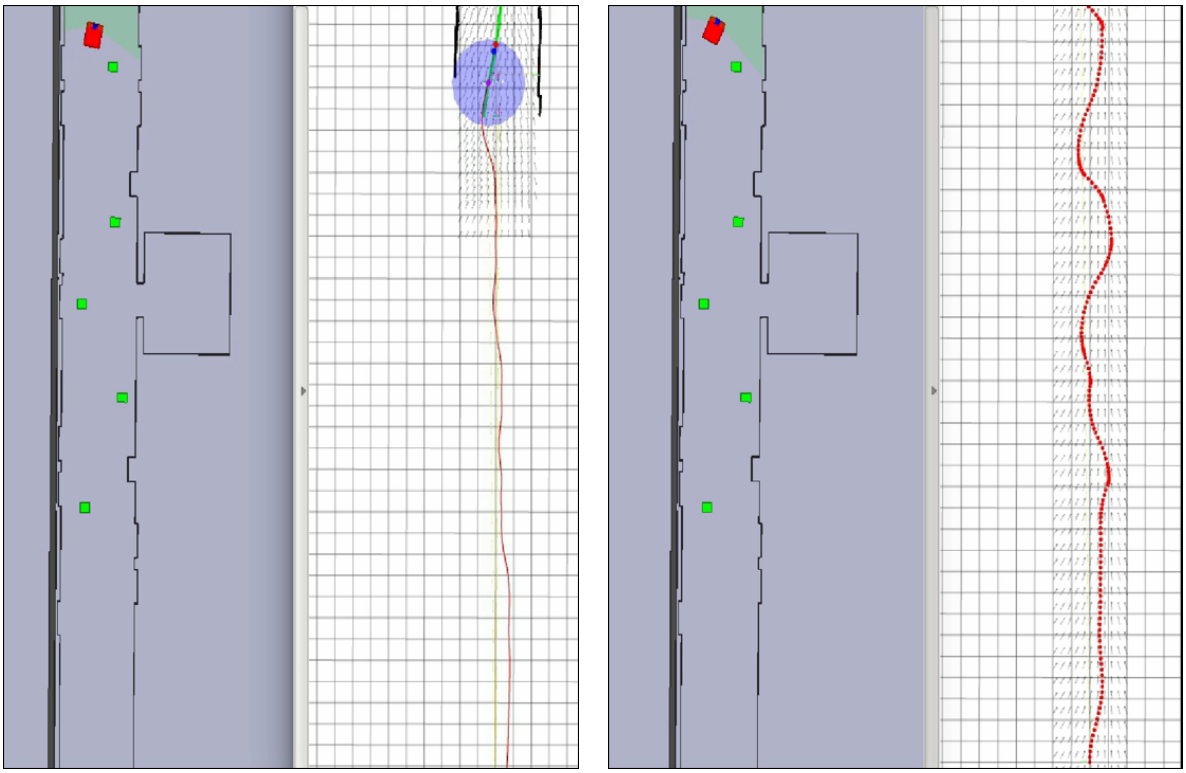
Observa-se que tanto o custo *upstream*, relativo ao seguimento do campo vetorial, quanto a suavidade dos caminhos realizados são bem maiores quando foi utilizado a estratégia local reativa. Realizando o Teste-T, verifica-se que esse dois critérios também são estatisticamente diferentes quando é considerada uma significância de 5%. Por outro lado, o custo euclideano das duas estratégias é o mesmo, considerando-se 5% de significância.

Tabela 4.7 – Comparação com relação ao custo euclideano, ao custo *upstream* e à suavidade dos caminhos realizados utilizando uma estratégia local reativa e a estratégia deste trabalho. O `valor-p` foi obtido pelo Teste-T aplicado a cada critério.

	estratégia local reativa (campos potenciais)	estratégia deste trabalho	`valor-p`
custo euclideano	20,97	21,05	0,62
custo *upstream*	1,97	0.31	1,82e-09
`suavidade`	362.868,13	17,71	6,92e-03

Foi percebido que a estratégia local reativa utilizando campos potenciais é de fácil implementação, porém pode gerar um caminho pouco suave quando comparado com a estratégia deste trabalho, como fica mais evidenciado no vídeo de comparação (<https://youtu.be/

JyNZmqzTzp0>). Além disso, dependendo da posição dos obstáculos, a estratégia reativa pode apresentar posições onde as somas dos vetores é nula (mínimos locais), o que pode parar o robô indefinidamente.



(a) (b)

Figura 4.11 – Comparação entre a estratégia de navegação proposta neste trabalho (a) e uma estratégia de navegação local reativa (b). Em vermelho é indicado o caminho realizado pelo robô. Mais detalhes podem ser vistos no vídeo de comparação: <https://youtu.be/JyNZmqzTzp0>.

4.3.4 Comparação entre as estratégias de navegação local com mapeamento para o desvio de obstáculos dinâmicos

Neste trabalho, optou-se pelo uso do mapeamento sem memória, como mostrado na Figura 4.2 em que o nó *laser_obstacles* é responsável por gerar um mapa local diretamente dos dados do *laser*. Para comparação, também foi implementada uma estratégia de navegação local com mapeamento com memória, utilizando o pacote *hector_slam*[15], que gera uma grade de ocupação enquanto o robô se locomove utilizando técnicas de SLAM.

Para comparar essas duas estratégias, realizou-se o seguinte experimento no ambiente de simulação Stage/ROS: um usuário determina o comando `siga` e, logo que o robô começa a se mover na simulação, dois obstáculos retangulares, que representam as pernas de uma pessoa, se movem próximos à parede da direita do robô no sentido contrário em que o robô está seguindo.

[15] Disponível no ROS em <http://wiki.ros.org/hector_slam> e configurado com os valores sugeridos pelos autores do pacote.

Uma sequência de capturas desse experimento, utilizando o mapeamento com memória, é apresentado na Figura 4.12 e o vídeo completo dessa simulação pode ser encontrado em <https://youtu.be/pWG7zFOviDI>. Repare que há um atraso na atualização do mapa em relação a posição em que o obstáculo está na cena. Isso acontece porque o algoritmo de mapeamento para definir a probabilidade de uma célula estar ocupada ou livre necessita de uma sequência de medições do *laser*, gastando mais tempo para atualizar o mapa. Assim, como o obstáculo se move, note na Figura 4.12(d), em que o robô está muito próximo do obstáculo, o mapa fornecido informa ao robô que há um obstáculo há quase 2 (dois) metros de onde verdadeiramente o obstáculo está na cena, assim, como mostrado na Figura 4.12(e), para essa condição de velocidade do obstáculo e do robô, acontece uma colisão, já que no momento do planejamento levou-se em consideração que não havia obstáculos naquele local.

Esse experimento foi repetido por 10 (dez) vezes, tanto utilizando a estratégia de navegação com mapeamento com memória via SLAM quanto a estratégia de navegação deste trabalho, variando a velocidade dos obstáculos entre $0,1\,\text{m/s}$ e $0,5\,\text{m/s}$ e mantendo a velocidade máxima alcançada com o robô simulado em $0{,}9\,\text{m/s}$. A porcentagem de sucessos do robô em desviar desses obstáculos dinâmicos é apresentada na Tabela 4.8.

Tabela 4.8 – Porcentagem de sucessos do robô em desviar dos obstáculos dinâmicos, utilizando a estratégia de navegação com memória via SLAM e a estratégia de navegação proposta neste trabalho, considerando um tempo de planejamento de $1,0\,\text{s}$, a velocidade do robô sendo $0{,}9\,\text{m/s}$ e os obstáculos com velocidade v_{ob}.

estratégia de navegação com mapeamento	$v_{ob} = 0,1\,\text{m/s}$	$v_{ob} = 0,25\,\text{m/s}$	$v_{ob} = 0,4\,\text{m/s}$	$v_{ob} = 0,5\,\text{m/s}$
com memória (SLAM)	100 %	60 %	20 %	0 %
sem memória (metodologia deste trabalho)	100 %	100 %	90 %	70 %

Nota-se que a estratégia de navegação proposta neste trabalho tem maior capacidade de proporcionar que o robô desvie de obstáculos dinâmicos, já que para obstáculos se movendo em velocidades maiores essa estratégia obtém mais sucesso do que a estratégia de navegação com mapeamento via SLAM. Pode-se, ainda, elencar outra vantagem do mapeamento sem memória que é o fato de consumir menos recursos computacionais, já que o mapa criado pelo SLAM cresce à medida que o robô se locomove. Por outro lado, percebe-se a qualidade de detalhes do mapa fornecido pelo SLAM, o que facilita, por exemplo, a detecção das paredes e a determinação do referencial de localização, já que o mapa guarda o histórico do corredor.

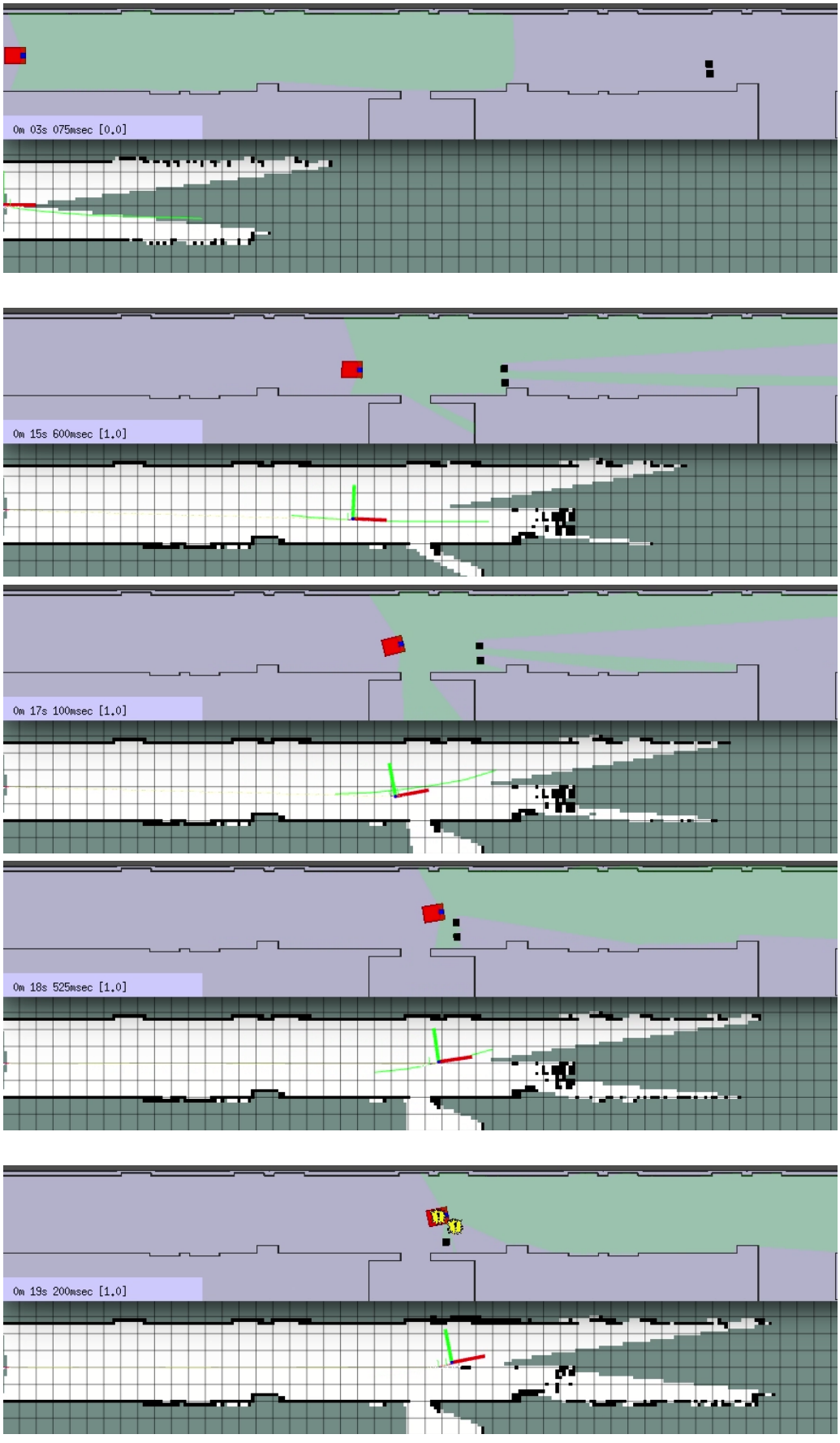
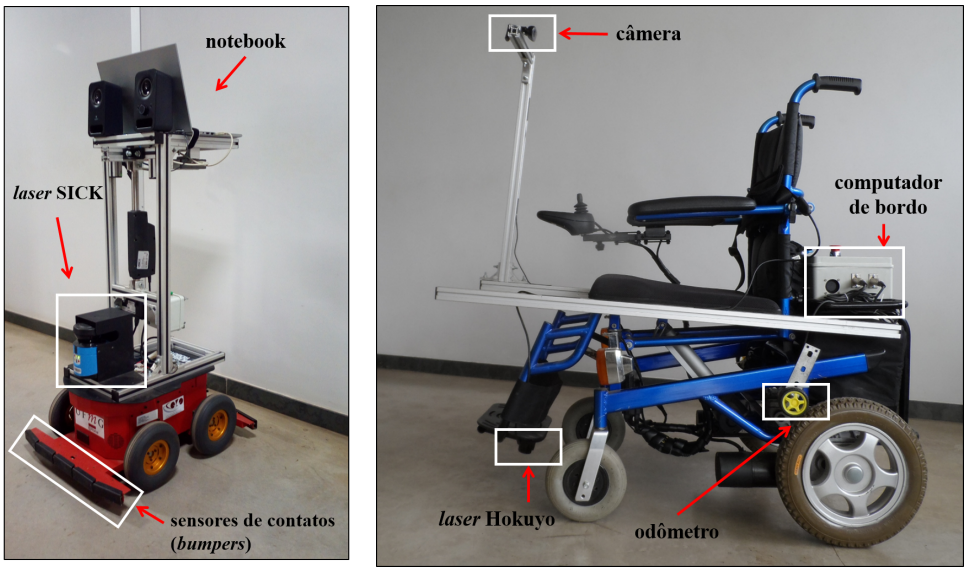


Figura 4.12 – Desvio de obstáculos dinâmicos usando a estratégia de navegação com mapeamento com memória via SLAM. Cada imagem é composta por uma cena obtida do simulador Stage/ROS e pela grade de ocupação criada, utilizando o pacote *hector_slam* no instante retratado pela respectiva cena e visualizada pelo Rviz/ROS. A velocidade do robô foi configurada em 0,9 m/s, a velocidade dos obstáculos em 0,5 m/s e um tempo de planejamento de 1,0 s. Mais detalhes em: <https://youtu.be/pWG7zFOviDI>.

4.4 Experimentos em robôs reais

Foram escolhidos dois robôs de serviço semiautônomos para testar a estratégia de navegação deste trabalho. O primeiro é o robô MARIA (*Manipulator Robot for Interaction and Assistance*), mostrado na Figura 4.13(a), que tem como base móvel um Pioneer3-AT, sendo equipado com sensores de contato (*bumpers*), um sensor de distância à *laser* SICK-LMS 100, com campo de visão de 270°, alcance máximo de 20 m e frequência de amostragem de 50 Hz e um sistema de odometria composto por dois odômetros e um girômetro que fornece dados a uma taxa de 10 Hz. A base desse robô tem 0,625 m de comprimento por 0,49 m de largura. O segundo robô é uma cadeira de rodas inteligente em desenvolvimento no (Laboratório de Sistemas de Computação e Robótica (CORO)), mostrada na Figura 4.13(b), que é equipada, por sua vez, com um sensor de distância à *laser* Hokuyo URG-04LX-UG01, com campo de visão de 240°, alcance máximo de 5,6 m e frequência de amostragem de 10 Hz e um sistema de odometria composto, atualmente, apenas por um par de odômetros. O controle de velocidade interno da cadeira é realizado por um computador de bordo (Raspberry Pi3[16] com um processador *quadcore* de 1.2 GHz) e a comunicação com esse sistema é feita por meio de um roteador *wireless*. Uma *webcam* também está disponível para possibilitar uma interface com usuário. A base da cadeira de rodas tem, aproximadamente, 1,0 m de comprimento por 0,70 m de largura.



(a) (b)

Figura 4.13 – Robôs de serviço semiautônomos utilizados nos experimentos reais. (a) Robô de serviço MARIA. (b) Cadeira de rodas inteligente em desenvolvimento no laboratório CORO.

Os experimentos com os robôs semiautônomos, MARIA e a cadeira de rodas inteligente, foram realizados em um dos andares da Escola de Engenharia da UFMG. Esses experimentos visam apresentar o funcionamento completo da estratégia de navegação no mundo real e validar essa estratégia mesmo com as incertezas do sistema de odometria (Subseção 4.4.1) e mesmo na presença de uma pessoa caminhando no ambiente (Subseção 4.4.2).

[16] Disponível em <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>.

4.4.1 Resultados com relação às incertezas do sistema de odometria

Foi realizado o seguinte experimento utilizando o robô MARIA: um usuário definiu a seguinte sequência de comandos: `siga`, `vire à direita` (x11), de tal maneira que o robô completasse três voltas no ambiente. Nesse experimento, a velocidade máxima do robô foi definida como sendo 0,9 m/s.

A odometria coletada do robô MARIA durante a execução desse experimento é apresentada na Figura 4.14, cujo vídeo completo pode ser encontrado em: <https://youtu.be/loS_TxmtQMI>. Como mostrado Figura 4.15, o robô foi capaz de desviar dos obstáculos existentes no ambiente e cumprir os comandos fornecidos pelo usuário. É importante mencionar que o robô finalizou a missão próximo ao ponto de partida. Dessa forma, repare que mesmo com o erro acumulativo da odometria, o robô MARIA locomoveu-se pelos corredores sem nenhum problema. Isso foi possível pelo fato da metodologia proposta nesta dissertação utilizar uma localização de curta duração, na qual as incertezas do sistema de odometria não são acumuladas durante toda a missão. Para o cumprimento geral da missão, a maneira como como os referenciais utilizados na localização de curta duração foram definidos foi importante, já que esses são alinhados com o corredor a cada intervalo de planejamento.

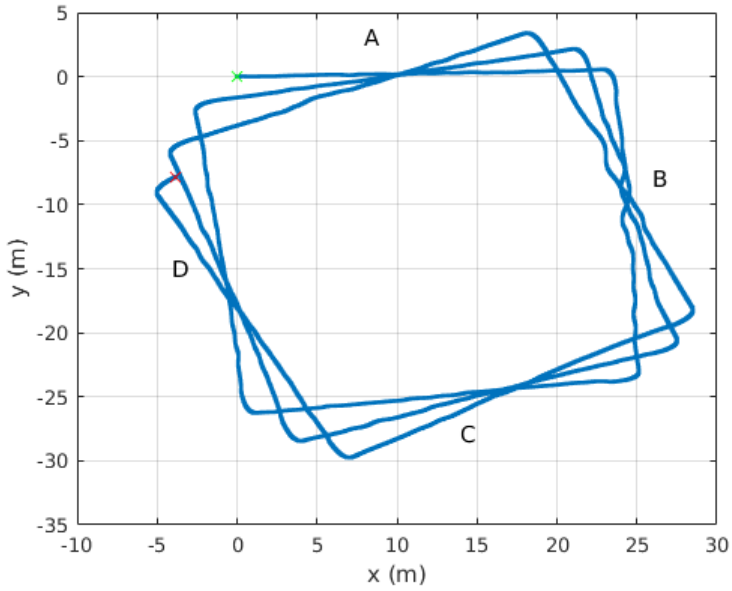


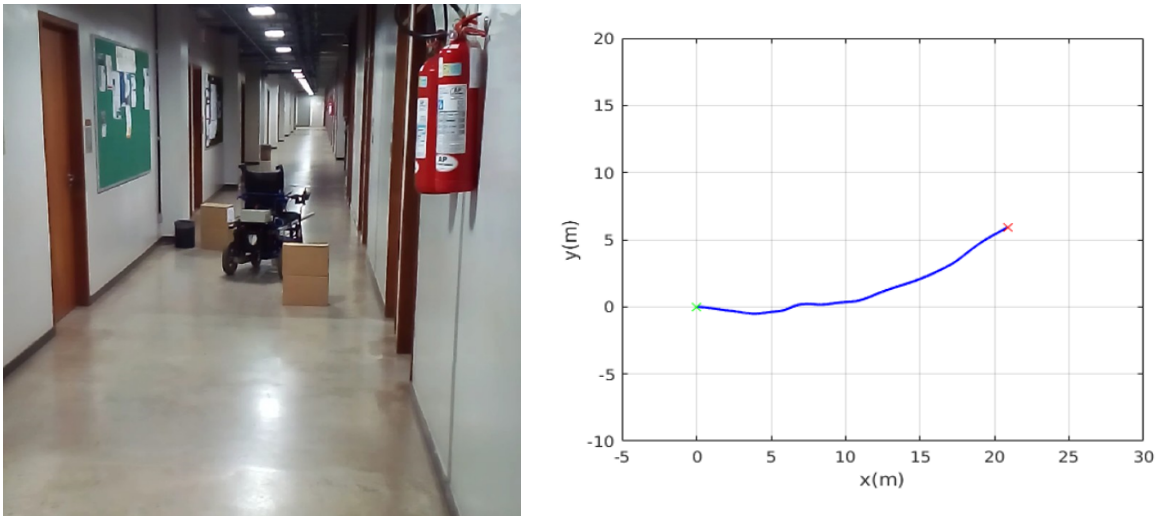
Figura 4.14 – Odometria do robô MARIA coletada durante a execução de três voltas em um ambiente composto por quatro corredores identificados por A,B,C,D. No corredor B foram colocados obstáculos fixos (cones) no caminho do robô. O robô inicia sua trajetória no ponto em verde (corredor A), finalizando no ponto em vermelho.

(a) (b)

(c) (d)

Figura 4.15 – Imagens do experimento no qual o robô MARIA navegava pelos corredores do ambiente. (a) Momento em que o robô navegava pelo corredor A. (b) Momento em que o robô navegava pelo corredor B e desviava dos obstáculos fixos em seu caminho. (c) Momento em que o robô navegava pelo corredor C. (d) Momento em que o robô navegava pelo corredor D.

Similarmente ao experimento anterior, o segundo experimento foi realizado com a cadeira de rodas inteligente, sendo que o comando fornecido pelo usuário foi apenas `siga`. Dessa forma, a cadeira percorreu o corredor inteiro, mantendo-se próxima a parede à sua direita, até encontrar a primeira interseção com outro corredor. Durante esse trajeto, ela desviou dos obstáculos estáticos presentes no ambiente, como ilustrado na Figura 4.16(a). Repare na Figura 4.16(b) que o sucesso da missão ocorreu mesmo com o erro acumulado da odometria, que chegou a quase 5 m no eixo y em relação à posição em que a cadeira deveria estar ao chegar ao fim do corredor. Percebe-se, também, que o sistema de odometria da cadeira de rodas é muito pior do que o do robô MARIA.



(a) (b)

Figura 4.16 – Experimento realizado em um corredor, utilizando a cadeira de rodas inteligente, como ilustrado em (a) em que a cadeira de rodas está desviando dos obstáculos presentes nesse corredor. Em (b) é apresentada a odometria da cadeira de rodas inteligente coletada durante o experimento.

Para estimar como a incerteza da localização fornecida pela odometria desses robôs de serviço variou durante os experimentos, tomou-se como valor médio a própria odometria do robô e propagou-se as incertezas associadas às velocidades linear e angular dos robôs, obtidas experimentalmente[17], utilizando apenas a etapa de predição do Filtro de Kalman EKF (*Extended Kalman Filter*). O cálculo da matriz de covariância na etapa de predição do filtro foi implementada utilizando a seguinte equação, conforme [Corke 2011]:

$$P[k+1] = F_x[k]\,P[k]\,F_x[k]^T + F_v[k]\,V\,F_v[k]^T\,, \tag{4.4}$$

onde $P[k]$ é matriz de covariância da odometria no instante k e V é a estimativa da covariância referente às incertezas nas entradas, v e ω, do seguinte modelo, obtido a partir da discretização

[17] Os robôs foram programados para movimentar com velocidades conhecidas durante um intervalo de tempo. Com os dados coletados da odometria, foram obtidos valores para a variância com relação ao valor médio medido das velocidades linear e angular para os vários testes. Por fim, definiu-se um valor médio para a variância de cada velocidade, obtendo a matriz V da equação (4.4).

do Modelo (3.4) e assumindo um ruído gaussiano $\mathtt{v} = (\mathtt{v}_v, \mathtt{v}_\omega) \sim \mathtt{N}(0, V)$:

$$\xi[k+1] = \begin{bmatrix} x[k] + \Delta t(v[k] + \mathtt{v}_v)(\cos(\theta[k] + \Delta t(\mathtt{v}_\omega + \omega)) \\ y[k] + \Delta t(v[k] + \mathtt{v}_v)(\sin(\theta[k] + \Delta t(\mathtt{v}_\omega + \omega)) \\ \theta[k] + \Delta t(\mathtt{v}_\omega + \omega) \end{bmatrix}, \tag{4.5}$$

a matriz F_x é o Jacobiano obtido derivando o Modelo (4.5) em relação aos estados e considerando o ruído nulo:

$$F_x[k] = \begin{bmatrix} 1 & 0 & -\Delta t\, v[k](\sin(\theta[k] + \Delta t\, \omega[k])) \\ 0 & 1 & \Delta t\, v[k](\cos(\theta[k] + \Delta t\, \omega[k])) \\ 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

e F_v é o Jacobiano obtido derivando o Modelo (4.5) em relação ao ruído:

$$F_v[k] = \begin{bmatrix} \cos(\theta[k] + \Delta t\, \omega[k]) & -\Delta t\, v[k](\sin(\theta[k] + \Delta t\, \omega[k])) \\ \sin(\theta[k] + \Delta t\, \omega[k]) & \Delta t\, v[k](\cos(\theta[k] + \Delta t\, \omega[k])) \\ 0 & 1 \end{bmatrix} \tag{4.7}$$

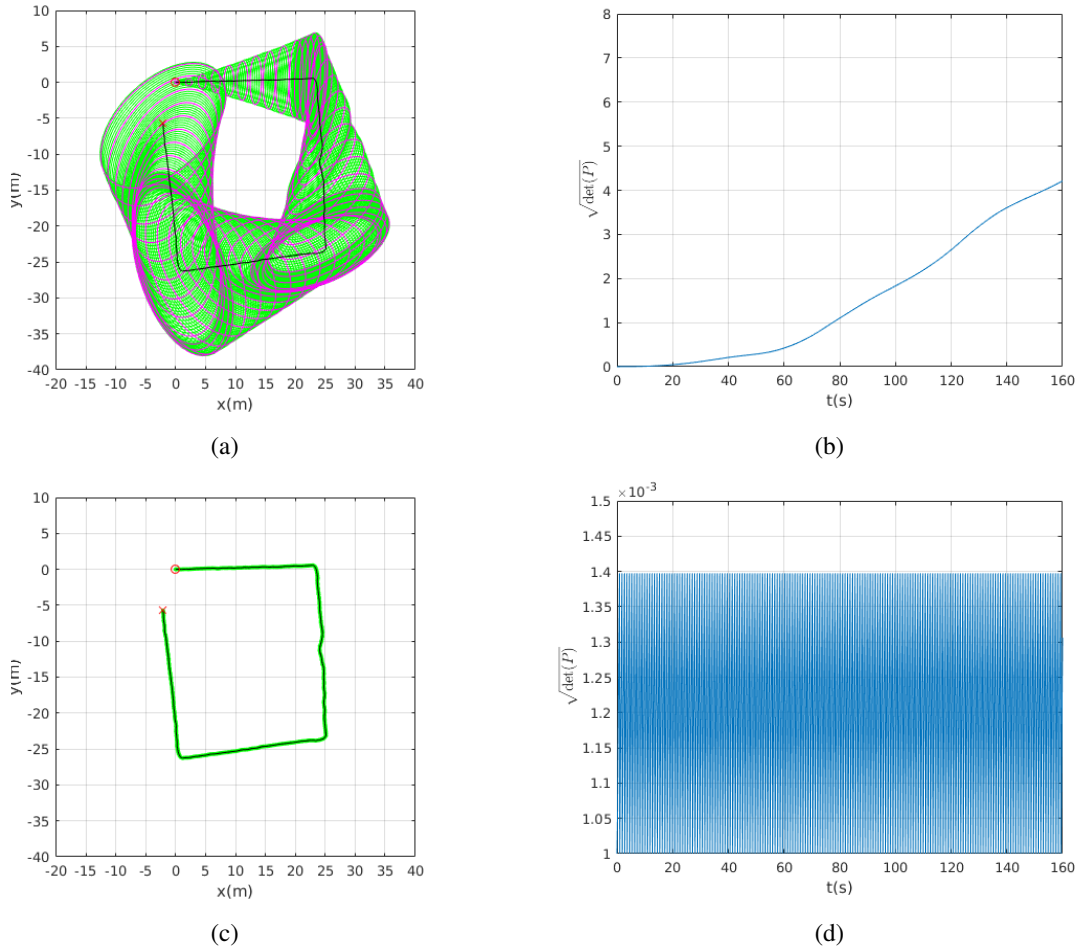
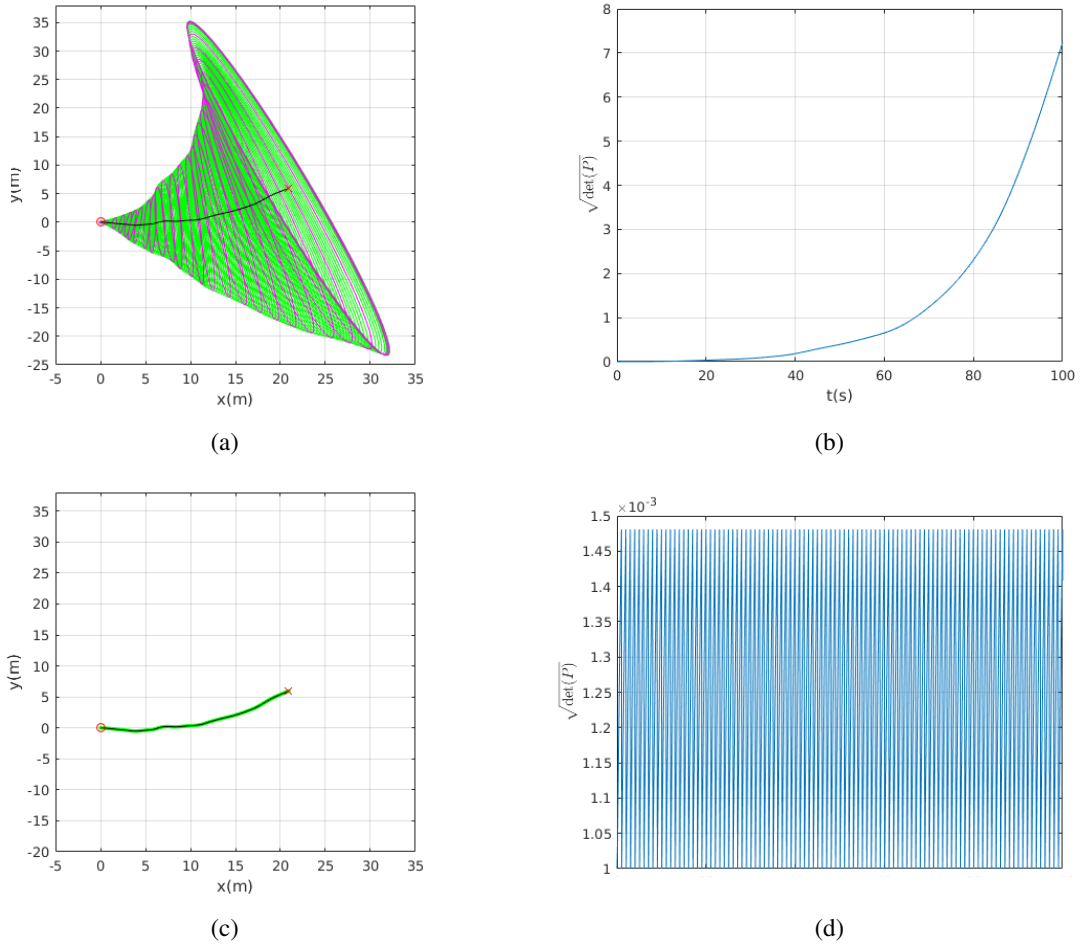


Figura 4.17 – Estimativa da covariância da odometria durante a primeira volta da missão, utilizando o robô MARIA. Em (a) é apresentada a estimativa da incerteza da posição do robô, representada pelas elipses, usando dois desvios padrão, ao longo dos dados da odometria em preto. Em (b) é apresentado a variação da estimativa total da incerteza de localização em relação ao tempo. Em (c) e (d) é ilustrado como a estratégia de navegação proposta neste trabalho percebe a incerteza de localização para um tempo de planejamento $t_s = 1,0\,\mathrm{s}$.



(a) (b)

(c) (d)

Figura 4.18 – Estimativa da covariância da odometria durante a navegação da cadeira de rodas inteligente por um corredor. Em (a) é apresentada a estimativa da incerteza da posição da cadeira de rodas, representada pelas elipses, usando dois desvios padrão, ao longo dos dados da odometria em preto.. Em (b) é apresentado a variação da estimativa total da incerteza de localização em relação ao tempo. Em (c) e (d) é ilustrado como a estratégia de navegação proposta neste trabalho percebe a incerteza de localização para um tempo de planejamento $t_s = 1,0\,\text{s}$.

A incerteza da localização em cada instante k com relação à posição (x,y) é descrita pela parte superior à esquerda da matriz de covariância $P[k]$ e pode ser interpretada como uma elipse que define um limite de confiança em relação ao valor médio. Nas Figuras 4.17(a) e 4.18(a) são mostradas essas elipses para dois desvios padrão, o que representa um limite de aproximadamente 95 %. Observe que as áreas das elipses crescem a partir da posição inicial do robô, significando que há um aumento gradativo da incerteza da localização do robô, causado pelo acúmulo de erro do sistema de odometria. Esse aumento ao passar do tempo pode ser visualizado nas Figuras 4.17(b) e Figura 4.18(b), cujo valor no eixo vertical do gráfico, $\sqrt{\det(P)}$, representa a incerteza total da localização, considerando a posição e a orientação do robô.

Do ponto de vista da estratégia de navegação, apenas o erro acumulado durante o tempo de planejamento t_s interfere na navegação. Dessa forma, é como se fosse sempre propagado o erro inicial apenas durante t_s segundos, ou seja, a incerteza de localização fica limitada, como visto nas Figuras 4.17(d) e 4.18(d). Conclui-se que a estimativa da incerteza durante esse intervalo

é muito pequena, como apresentado nas Figuras 4.17(c) e 4.18(c), em que praticamente não se nota o crescimento das áreas das elipses, validando a estratégia proposta neste trabalho com relação à odometria.

4.4.2 Desvio de obstáculos dinâmicos

Foi realizado o seguinte experimento para validar a estratégia de navegação com relação ao desvio de obstáculos dinâmicos: enquanto o robô navegava pelo corredor, uma pessoa caminhou no sentido oposto ao robô, junto à parede à direita na qual o robô deveria ficar próximo. Esse experimento foi realizado, primeiramente, com o robô MARIA, sendo que o robô foi configurado para locomover-se com sua velocidade máxima, $v_{max} = 0,9\,\text{m/s}$, e um tempo de planejamento de $t_s = 0,45\,\text{s}$, reduzido para garantir o desvio de uma pessoa andando normalmente, ou seja, com uma velocidade em torno de $1,0\,\text{m/s}$. Também esse experimento foi realizado utilizando a cadeira de rodas, porém dada a dificuldade de se controlar a velocidade dessa cadeira com um tempo de resposta pequeno, seja pela sua dinâmica seja pelo fato de não ser possível acionar diretamente os motores da mesma, foi configurada uma velocidade menor, $v_{max} = 0,3\,\text{m/s}$, e a pessoa caminhou com uma velocidade em torno de $0,25\,\text{m/s}$[18].

Utilizando o robô MARIA, esse experimento foi repetido 10 (dez) vezes em sequência, sendo que em todas elas o robô foi capaz de desviar da pessoa em movimento. Na Figura 4.19, é apresentada uma sequência de capturas relativas a uma dessas repetições, contendo cenas de alguns momentos durante a execução do experimento pelo robô MARIA e a visualização correspondente obtida pelo visualizador Rviz/ROS, que mostra o caminho planejado em verde, o centro do robô representado pelo ponto vermelho, os dados do laser representados pelo conjunto de pontos em preto e a Zona Estática indicada pelo disco em azul.

Observe na Figura 4.19(a) que a pessoa começa a caminhar próximo à parede, porém, o caminho planejado ainda visa apenas fazer com que o robô siga o corredor à $D_0 = 1,0\,\text{m}$ da parede da direita, já que o obstáculo está fora do raio de planejamento. Nas Figuras 4.19(b) e 4.19(c), é possível perceber que a pessoa está dentro da região de planejamento de movimento e que o caminho já prevê o desvio dessa pessoa que continua em movimento. Repare que, na Figura 4.19(d), o caminho planejado está prevendo o desvio de um obstáculo que estava a alguns centímetros atrás do obstáculo detectado pelo *laser* mostrado na imagem, pois esse caminho foi gerado com base nas informações obtidas em um instante, cerca de $0,45\,\text{s}$, antes da captura desse momento. Dessa forma, o robô consegue desviar da pessoa andando em sua direção, pois já havia começado a desviar dela momentos antes.

Claramente, se a pessoa mudasse a direção do seu movimento ou fosse com uma velocidade maior, após o caminho ter sido fornecido, o robô não conseguiria desviar, parando antes que acontecesse a colisão. Isso pode ser justificado pelo fato que a estratégia adotada neste trabalho

[18] A velocidade da pessoa foi estimada visualmente no Rviz/ROS, dividindo a distância percorrida durante um intervalo de tempo.

garante que não haverá colisões fora da Zona Estática para quaisquer tipo de obstáculos, porém, nesse caso, o obstáculo estaria dentro da Zona Estática e estaria se movimentando, ou seja, nessa situação não seria garantido o desvio desse obstáculo de forma planejada pela estratégia de navegação. A solução para isso seria diminuir o tempo de planejamento ou no momento de gerar o mapa local levar em consideração a previsão do movimento da pessoa.

Da mesma forma, pode-se analisar o resultado obtido com a cadeira de rodas inteligente, apresentado na Figura 4.20. Constata-se que também ela foi capaz de desviar da pessoa em movimento, mas vale ressaltar que, nesse caso, para que haja também o desvio de pessoas caminhando com velocidades maiores, seria necessário ainda melhorar o controle de velocidade interno da cadeira de rodas, o tempo de latência da comunicação entre o notebook e o sistema interno da cadeira e, seria, conveniente utilizar um *laser* com uma taxa de dados maior.

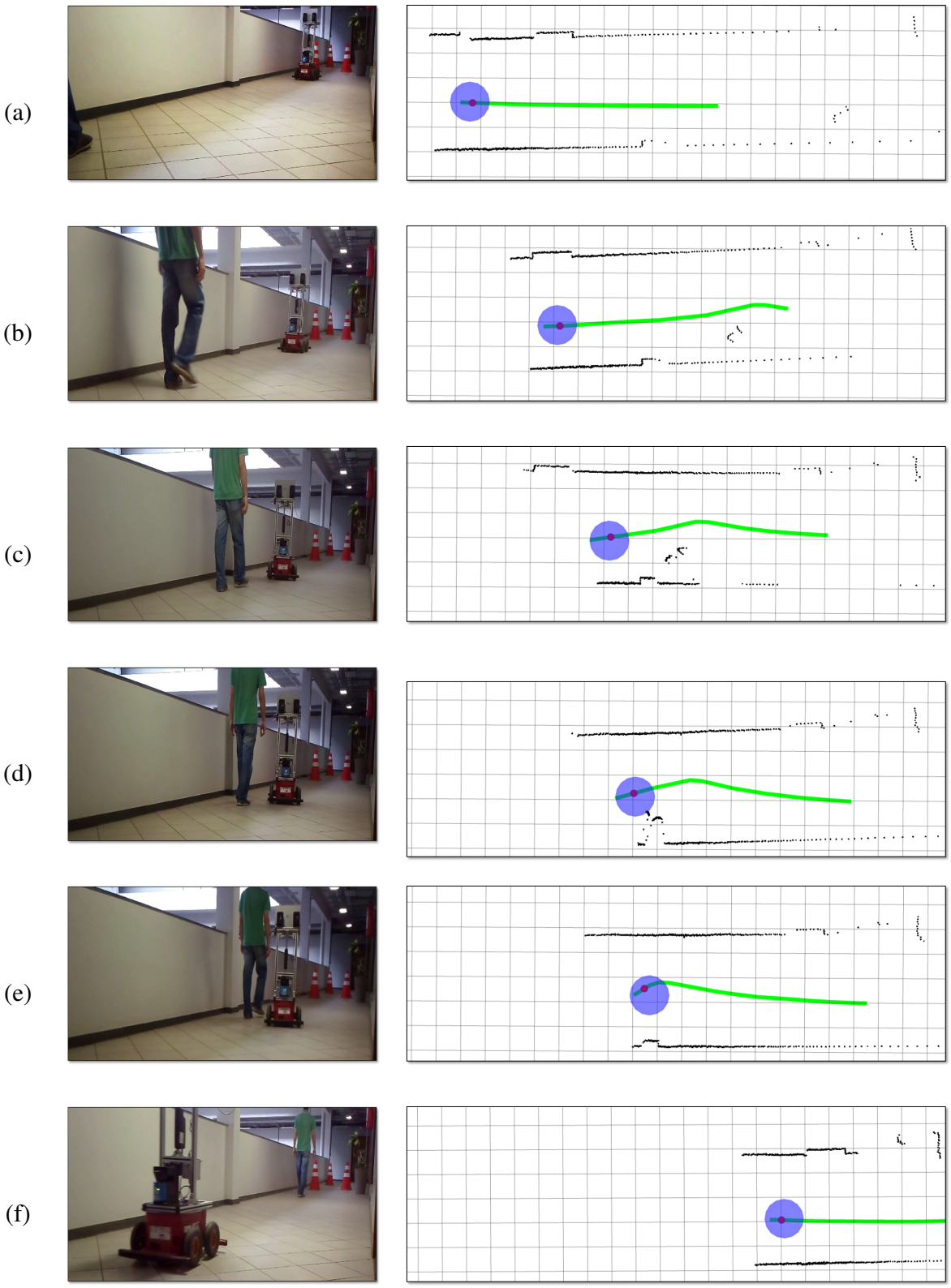


Figura 4.19 – Desvio de obstáculos dinâmicos utilizando o robô MARIA. As imagens à esquerda da página são capturas de momentos na sequência da navegação do robô em um corredor na presença de uma pessoa. As imagens à direita representam a visualização desse momento obtidas por meio do visualizador Rviz/ROS, sendo que em verde está representado o caminho planejado, o ponto vermelho indica o centro do robô, o conjunto de pontos pretos representam os dados do laser e o disco em azul representa a Zona Estática para uma velocidade $v_{max} = 0,9\,\text{m/s}$ e um tempo de planejamento $t_s = 0,45\,\text{s}$. O vídeo completo desse experimento pode ser encontrado em <https://youtu.be/DONxBJJ5OT8>.

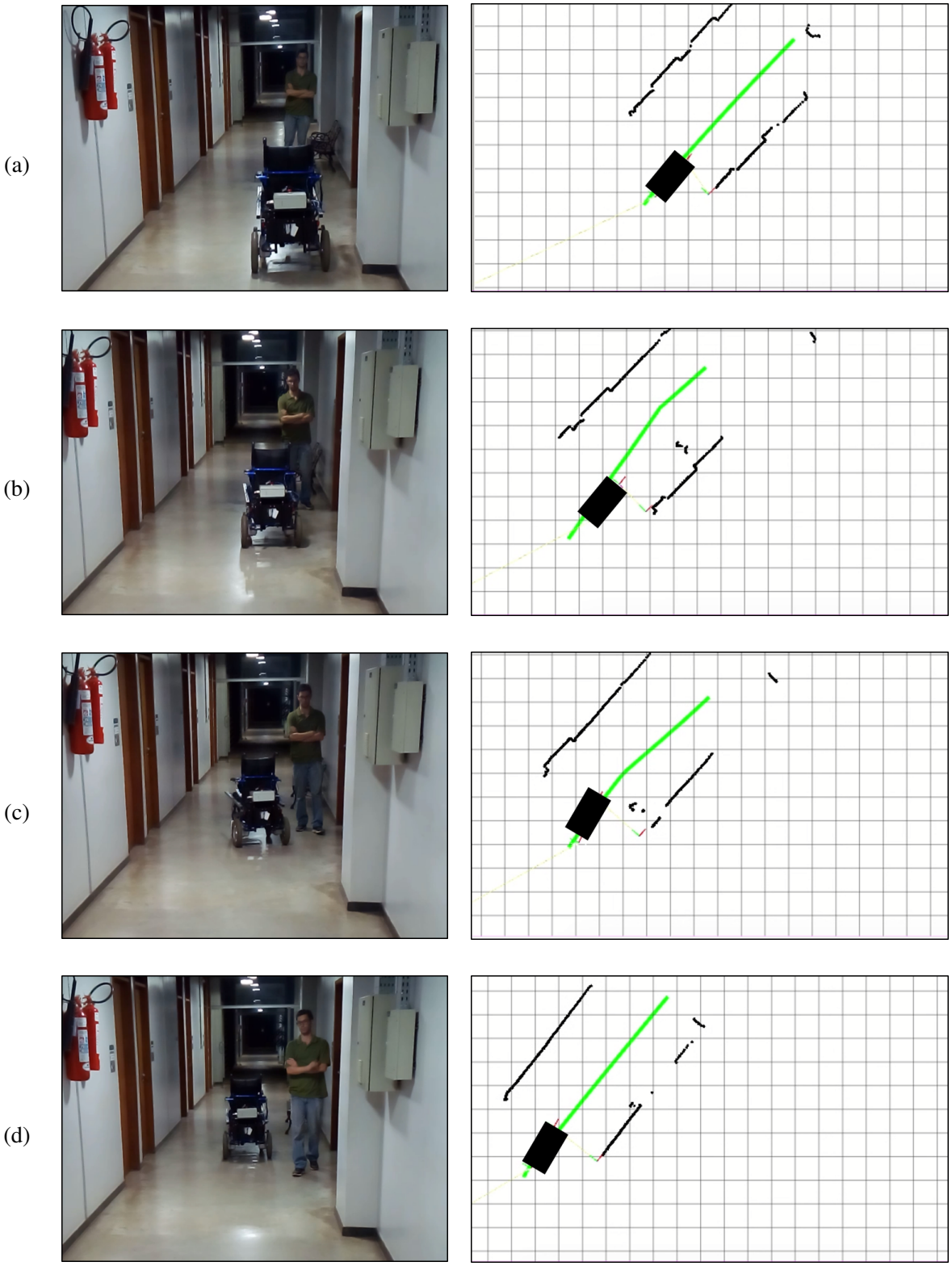


Figura 4.20 – Desvio de obstáculos dinâmicos utilizando a cadeira de rodas. As imagens à esquerda da página são capturas de momentos na sequência da navegação da cadeira de rodas em um corredor na presença de uma pessoa. As imagens à direita representam a visualização desse momento obtidas por meio do visualizador Rviz/ROS, sendo que em verde está representado o caminho planejado, o conjunto de pontos pretos representam os dados do laser e o retângulo preto representa a cadeira de rodas que foi configurada para uma velocidade $v_{max} = 0,25\,\text{m/s}$ e um tempo de planejamento $t_s = 1,0\,\text{s}$. O vídeo completo desse experimento pode ser encontrado em <https://youtu.be/DiIcPul18kc>.

CAPÍTULO

5

CONCLUSÕES E TRABALHOS FUTUROS

"Se cada um de seus dias for uma centelha, você terá iluminado o mundo."

Irmã Tecla Merlo, 1894-1964

Neste trabalho propôs-se uma estratégia de navegação para robôs móveis terrestres que utiliza informação local obtida dos sensores instalados no robô e um planejador probabilístico, que fornece caminhos a serem seguidos localmente pelo robô. Essa estratégia é destinada, principalmente, aos robôs de serviço semiautônomos, já que esses robôs nem sempre possuem o mapa do ambiente nem um sistema de localização global, uma vez que operam em ambientes desconhecidos e sua missão pode ser controlada por um operador humano. A estratégia de navegação proposta diferencia-se da maioria das soluções previamente existentes, que assumem o conhecimento do mapa do ambiente e combinam um planejador de caminho com algum método reativo, como o DWA (*Dynamic Window Approach*), para garantir o desvio de obstáculos que não foram previstos no mapa.

Para a validação da estratégia de navegação deste trabalho, foi escolhido um ambiente composto por corredores e interseções, que é um cenário típico de ambientes como hospitais, escolas e prédios comerciais, nos quais os robôs de serviço poderiam operar. A partir dessa escolha, foram definidos alguns comandos para o usuário determinar uma sequência de tarefas para o robô. Dentre eles, o comando `siga` foi codificado por meio de um campo vetorial. Esse campo vetorial é simples, mas suficiente para orientar o robô a seguir a mão convencional de circulação em um corredor, não atrapalhando a passagem de pessoas nesse ambiente.

Verificou-se que incorporar a informação de um campo vetorial a um planejador de caminhos baseado no RRT*, como proposto pela metodologia deste trabalho, permite que o robô, ao seguir o caminho gerado pelo planejador, alcance um alvo especificado e fique condicionado

a seguir o mais próximo possível um determinado comportamento de circulação especificado pelo campo vetorial. Esse tipo de comportamento não é obtido, por exemplo, ao se utilizar um planejador clássico RRT*, que otimiza a distância euclideana, levando em consideração apenas o alvo especificado. Nos resultados obtidos, o planejador proposto neste trabalho obtém melhores resultados que o planejador VF-RRT, que constrói uma árvore baseada no campo vetorial fornecido. Isso pode ser justificado pelo fato do planejador VF-RRT não ser um planejador ótimo, assim, não garante um seguimento tão próximo ao campo vetorial, tendo um custo *upstream* muito superior aos demais planejadores comparados, além de fornecer um caminho pouco suave.

Também, verificou-se na comparação entre os planejadores probabilísticos que a escolha feita neste trabalho de se utilizar uma heurística simples para o cálculo do Funcional (3.2), avaliando o campo vetorial apenas na amostra inicial, permite que a estratégia de navegação posposta nesta dissertação assuma tempos de planejamento menores do que os obtidos quando a funcional é calculado em todo o caminho. Nessa condição, foram obtidos melhores resultados quanto à suavidade do caminho e ao custo *upstream*.

No ambiente escolhido, a estratégia foi utilizada para controlar um robô simulado e dois robôs de serviço semiautônomos reais. Os resultados apresentados no Capítulo 4 demostram que os robôs foram capazes de executar a missão definida pelo usuário, desviando eficientemente e de forma planejada de obstáculos estáticos e, para algumas situações bem definidas, também foi possível desviar de obstáculos dinâmicos, incluindo pessoas se locomovendo no ambiente durante a missão. A estratégia de navegação deste trabalho, além de ser mais simples e mais eficiente computacionalmente, apresentou melhores resultados do que utilizando uma estratégia baseada em SLAM, quando o robô está na presença de obstáculos dinâmicos. Comparada com uma estratégia reativa, verificou-se que este trabalho proporciona de forma planejada um caminho mais suave e sem mínimos locais. A importância dessa suavidade fica evidenciada quando aplicada a uma cadeira de rodas inteligente, que é a aplicação motivacional deste trabalho, já que quanto mais suave for o caminho realizado pelo robô maior será o conforto do usuário.

Mesmo com todos os bons resultados obtidos pela estratégia de navegação proposta neste trabalho, pode-se elencar algumas limitações:

- Não foi tratado nenhum tipo de restrição temporal com relação ao movimento do robô, como, por exemplo, chegar até o fim do corredor dentro de um tempo especificado, já que o planejamento de movimento foi abordado pela metodologia como um planejamento de caminhos, que gera apenas uma sequência de pontos sem informação temporal.
- O desvio de obstáculos dinâmicos de forma planejada não é garantido para todas situações, principalmente, se for considerado o movimento imprevisível de uma pessoa.
- O mapa local utilizado pelo planejador de caminhos considera apenas a informação mais recente obtida pelo *laser*, adquirida instantes antes do planejador iniciar um novo

planejamento. Dessa forma, pode ocorrer uma situação na qual o dado mais recente do *laser* não contém a informação de um objeto detectado anteriormente durante o tempo de planejamento, por causa, por exemplo, de uma oclusão do objeto no instante de aquisição do mapa, podendo gerar um caminho indesejado.

- A implementação da metologia proposta neste trabalho limitou-se ao ambiente escolhido: longos corredores com interseções.

- A estratégia de navegação proposta guia o robô pelo corredor de acordo com a mão convencional de circulação, mas não garante outras regras sociais como, por exemplo, as elaboradas por Pandey e Alami [2010]: (i) o robô deve desviar de uma pessoa pelo lado esquerdo, (ii) o robô deve possuir um conjunto de velocidades permitidas, de tal forma, que dependendo da situação que ele se encontra uma determinada velocidade é mais adequada que outra, e (iii) o robô não deve mover-se muito próximo de um obstáculo.

- Os testes foram realizados considerando a velocidade máxima do robô de 0,9 m/s. Dessa forma, se a estratégia de navegação fosse aplicada, por exemplo, a um carro autônomo, que pode rodar com velocidade muito superior a essa, seria necessário diminuir muito o tempo de planejamento e adaptar a estratégia, de tal forma, que se houvesse a detecção de algum obstáculo à frente desse veículo, a sua velocidade máxima deveria diminuir, garantindo o funcionamento da estratégia de navegação.

Dado esses elementos, algumas propostas de trabalhos futuros são sugeridas afim de aprimorar ainda mais a estratégia de navegação apresentada neste trabalho:

- Tratar o problema de planejamento de movimento como um problema de planejamento de trajetórias. Por exemplo, baseando-se no trabalho de Richter, Bry e Roy [2016], um caminho gerado pelo planejador poderia ser interpolado por meio de funções polinomiais e, a partir de restrições temporais desejadas, uma trajetória que minimize, por exemplo, a arrancada[19] (do inglês, *jerk*) pode ser gerada. Assim, além de incorporar a informação temporal à tarefa a ser executada pelo robô, a trajetória gerada pode permitir que o robô evite saltos bruscos em sua aceleração. Isso é muito importante para garantir o conforto da navegação, principalmente, quando consideramos a situação específica de um usuário utilizando uma cadeira de rodas inteligente.

- Definir um conjunto de novos comandos para o usuário, como: siga corredor circular, atravesse espaço aberto, entre na primeira porta etc. Para isso, seria também necessário elaborar um conjunto de campos vetoriais que satisfaçam esses comandos e novos detectores para essas novas situações.

[19] Arrancada é definida como sendo a terceira derivada da posição com relação ao tempo.

- Acrescentar à estratégia um detector de pessoas, utilizando, por exemplo, um sensor RGB-D. Com as informações obtidas por esse detector, seria possível adicionar uma previsão do movimento da pessoa no mapa que é utilizado pelo planejador, permitindo que o planejamento fosse mais eficiente no desvio de pessoas.

- Verificar a possibilidade de gerar um mapa local, combinando as informações do *laser* adquiridas durante um intervalo de planejamento, sendo essa uma solução intermediária entre o mapeamento sem memória e o mapeamento com memória do mapa global.

- Propor novos funcionais de custo ou outros campos vetoriais que possibilitem considerar mais regras sociais para a navegação do robô.

- Avaliar a navegação do robô em ambientes externos, propondo outras formas de definir os referenciais de curta duração, que são válidos durante um intervalo de planejamento, como, por exemplo, por meio da detecção do meio-fio.

- Avaliar a possibilidade de se utilizar o espaço de *Dubins* no planejamento de movimento. Mesmo que esse espaço aumente o custo computacional do planejador, seria possível incorporar ao caminho a ser fornecido a restrição de movimento do robô, definindo um raio mínimo de curvatura.

- Desenvolver novas interfaces entre usuário-robô que sejam interativas e intuitivas para os usuários. Existem várias possibilidades para transmitir a sequência de comandos ao robô, como: (i) por meio de comandos de voz [Nishimori, Saitoh e Konishi 2007], (ii) por meio de gestos faciais [Kawarazaki e Diaz 2013] ou, ainda, (iii) por meio de sinais cerebrais [TANNUS *et al.* 2014].

REFERÊNCIAS

ADORNO, B. V.; BORGES, G. A. Robótica móvel. In: VARIOS (Ed.). Rio de Janeiro: LTC, 2014. cap. 6, p. 84–112. ISBN 978-8521623038. Citado na página 30.

ALVES, S. F. R.; ROSARIO, J. M.; FILHO, H. F.; RINCON, L. K. A.; YAMASAKI, R. A. T. Conceptual bases of robot navigation modeling, control and applications. In: BARRERA, A. (Ed.). **Advances in Robot Navigation**. Rijeka: InTech, 2011. cap. 1. Disponível em: <http://dx.doi.org/10.5772/20955>. Citado na página 31.

ARAUJO, A. R.; CAMINHAS, D. D.; PEREIRA, G. A. An architecture for navigation of service robots in human-populated office-like environments. **IFAC-PapersOnLine**, v. 48, n. 19, p. 189–194, 2015. Disponível em: <https://doi.org/10.1016/j.ifacol.2015.12.032>. Citado 6 vezes nas páginas: 24, 26, 27, 28, 31 e 32.

ARKIN, R. C. **Behavior-based robotics**. Cambridge: The MIT press, 1998. ISBN 978-0262011655. Citado na página 23.

BAKLOUTI, E.; AMOR, N. B.; JALLOULI, M. Reactive control architecture for mobile robot autonomous navigation. **Robotics and Autonomous Systems**, v. 89, p. 9–14, 2017. Disponível em: <https://doi.org/10.1016/j.robot.2016.09.001>. Citado 3 vezes nas páginas: 23, 26 e 31.

BELLARBI, A.; KAHLOUCHE, S.; ACHOUR, N.; OUADAH, N. A social planning and navigation for tour-guide robot in human environment. In: **Proceedings of the 8th International Conference on Modelling, Identification and Control (ICMIC)**. [s.n.], 2016. p. 622–627. Disponível em: <https://doi.org/10.1109/icmic.2016.7804186>. Citado na página 26.

BOAL, J.; SÁNCHEZ-MIRALLES, Á.; ARRANZ, Á. Topological simultaneous localization and mapping: a survey. **Robotica**, Cambridge University Press (CUP), v. 32, n. 05, p. 803–821, 2013. Disponível em: <https://doi.org/10.1017/s0263574713001070>. Citado na página 26.

BORENSTEIN, J.; KOREN, Y. Real-time obstacle avoidance for fast mobile robots in cluttered environments. In: **Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)**. IEEE, 1990. v. 1, p. 572–577. Disponível em: <https://doi.org/10.1109/robot.1990.126042>. Citado na página 30.

____. Teleautonomous guidance for mobile robots. **IEEE Transactions on Systems, Man, and Cybernetics (SMC)**, IEEE, v. 20, n. 6, p. 1437–1443, 1990. Disponível em: <https://doi.org/10.1109/21.61212>. Citado na página 17.

BROOKS, R. A robust layered control system for a mobile robot. **IEEE Journal on Robotics and Automation**, IEEE, v. 2, n. 1, p. 14–23, 1986. Disponível em: <https://doi.org/10.1109/jra.1986.1087032>. Citado na página 22.

BURGARD, W.; CREMERS, A. B.; FOX, D.; HäHNEL, D.; LAKEMEYER, G.; SCHULZ, D.; STEINER, W.; THRUN, S. Experiences with an interactive museum tour-guide robot. **Artificial Intelligence**, v. 114, n. 1-2, p. 3–55, 1999. Disponível em: <https://doi.org/10.1016/s0004-3702(99)00070-3>. Citado 3 vezes nas páginas: 26, 27 e 31.

CEN, G.; MATSUHIRA, N.; HIROKAWA, J.; OGAWA, H.; HAGIWARA, I. Service robot localization using improved particle filter. In: **2008 IEEE International Conference on Automation and Logistics**. IEEE, 2008. p. 2454–2459. Disponível em: <https://doi.org/10.1109/ical.2008.4636580>. Citado na página 27.

CHAVEZ, J. R.; VIDAL, F. D. B.; VIANA, D. M.; KOIKE, C. M. E. C. Deformable virtual zone and particle filters applied to obstacle avoidance in mobile robotics. In: **2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)**. [s.n.], 2016. p. 205–210. Disponível em: <https://doi.org/10.1109/lars-sbr.2016.41>. Citado na página 30.

CHERNOUSKO, F. **Control of Nonlinear Dynamical Systems**. 1. ed. Berlin: Springer-Verlag, 2008. ISBN 978-3540707844. Citado na página 48.

CHIK, S.; YEONG, C.; SU, E.; LIM, T.; SUBRAMANIAM, Y.; CHIN, P. A review of social-aware navigation frameworks for service robot in dynamic human environments. **Journal of Telecommunication, Electronic and Computer Engineering**, v. 8, n. 11, p. 41–50, 2016. ISSN 2180-1843. Disponível em: <http://journal.utem.edu.my/index.php/jtec/article/view/1408>. Citado na página 23.

CHOSET, H.; LYNCH, K. M.; HUTCHINSON, S.; KANTOR, G. A.; BURGARD, W.; KAVRAKI, L. E.; THRUN, S. **Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents series)**. Cambridge: The MIT Press, 2005. ISBN 978-0262033275. Citado 3 vezes nas páginas: 23, 29 e 67.

CLARK, J.; ROEMER, R. Voice controlled wheelchair. **Arch Phys Med Rehabil**, p. 169–175, 1977. Citado na página 20.

CORKE, P. **Robotics, Vision and Control**. Berlin: Springer Berlin Heidelberg, 2011. ISBN 978-3642201431. Disponível em: <https://doi.org/10.1007/978-3-642-20144-8>. Citado na página 74.

DAYOUB, F.; MORRIS, T.; CORKE, P. Rubbing shoulders with mobile service robots. **IEEE Access**, IEEE, v. 3, p. 333–342, 2015. Disponível em: <https://doi.org/10.1109/access.2015.2424273>. Citado 2 vezes nas páginas: 26 e 32.

DIB, A.; BEAUFORT, N.; CHARPILLET, F. A real time visual SLAM for RGB-d cameras based on chamfer distance and occupancy grid. In: **2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics**. IEEE, 2014. p. 652–657. Disponível em: <https://doi.org/10.1109/aim.2014.6878153>. Citado na página 27.

EL-LAITHY, R. A.; HUANG, J.; YEH, M. Study on the use of microsoft kinect for robotics applications. In: **Proceedings of the IEEE/ION Position, Location and Navigation Symposium**. IEEE, 2012. p. 1280–1288. Disponível em: <https://doi.org/10.1109/plans.2012.6236985>. Citado na página 25.

ELBANHAWI, M.; SIMIC, M. Sampling-based robot motion planning: A review. **IEEE Access**, IEEE, v. 2, p. 56–77, 2014. Disponível em: <https://doi.org/10.1109/access.2014.2302442>. Citado na página 30.

ELFES, A. Using occupancy grids for mobile robot perception and navigation. **Computer**, IEEE, v. 22, n. 6, p. 46–57, 1989. Disponível em: <https://doi.org/10.1109/2.30720>. Citado na página 27.

FEDER, H.; SLOTINE, J.-J. Real-time path planning using harmonic potentials in dynamic environments. In: **Proceedings of the IEEE International Conference on Robotics and Automation**. IEEE, 1997. p. 874–881. Disponível em: <https://doi.org/10.1109/robot.1997.620144>. Citado na página 30.

FERNANDEZ, E.; CRESPO, L. S.; MAHTANI, A.; MARTINEZ, A. **Learning ROS for Robotics Programming**. 2. ed. Birmingham: Packt Publishing, 2015. ISBN 978-1783987580. Citado na página 24.

FOTINEA, S.-E.; EFTHIMIOU, E.; GOULAS, T.; DIMOU, A.-L.; TZAFESTAS, C.; PITSIKALIS, V. The MOBOT human-robot interaction. In: ACM PRESS. **Proceedings of the 20th Pan-Hellenic Conference on Informatics - PCI '16**. 2016. p. 1–6. Disponível em: <https://doi.org/10.1145/3003733.3003812>. Citado na página 26.

FOX, D.; BURGARD, W.; THRUN, S. The dynamic window approach to collision avoidance. **IEEE Robotics & Automation Magazine**, IEEE, v. 4, n. 1, p. 23–33, 1997. Disponível em: <https://doi.org/10.1109/100.580977>. Citado na página 30.

FREITAS, E. J. R.; PASSOS, H. A.; PEREIRA, G. A. S. Desvio de obstáculos por robôs semiautônomos usando planejamento de caminhos. **XIII Simpósio Brasileiro de Automação Inteligente**, Porto Alegre, p. 1043–1048, 2017. Citado na página 21.

FUENTES-PACHECO, J.; RUIZ-ASCENCIO, J.; RENDÓN-MANCHA, J. M. Visual simultaneous localization and mapping: a survey. **Artificial Intelligence Review**, v. 43, n. 1, p. 55–81, 2012. Disponível em: <https://doi.org/10.1007/s10462-012-9365-8>. Citado na página 28.

G1. **Idosos já são 13% da população e país tem menos crianças, diz Pnad**. 2014. G1 Portal de Notícias. Disponível em: <http://g1.globo.com/economia/noticia/2014/09/idosos-ja-sao-13-da-populacao-e-pais-tem-menos-criancas-diz-pnad.html>. Acesso em: 2015-05-10. Citado na página 19.

GALAMHOS, C.; MATAS, J.; KITTLER, J. Progressive probabilistic hough transform for line detection. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. IEEE, 1999. v. 1, p. 1554–1560. Disponível em: <https://doi.org/10.1109/cvpr.1999.786993>. Citado na página 54.

GAMMELL, J. D.; SRINIVASA, S. S.; BARFOOT, T. D. Batch informed trees (BIT∗): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In: **Proceedings of the IEEE International Conference on Robotics and Automation**. IEEE, 2015. p. 3067–3074. Disponível em: <https://doi.org/10.1109/icra.2015.7139620>. Citado na página 58.

GERKEY, B. P.; KONOLIGE, K. Planning and control in unstructured terrain. In: **Workshop on Path Planning on Costmaps, Proceedings of the IEEE International Conference on Robotics and Automation**. IEEE, 2008. p. 1–6. Disponível em: <http://pub1.willowgarage.com/apubdb_html/files_upload/8.pdf>. Citado na página 30.

GOLDBERG, S.; MAIMONE, M.; MATTHIES, L. Stereo vision and rover navigation software for planetary exploration. In: IEEE. **Proceedings of the IEEE Aerospace Conference**. 2002. v. 5, p. 2025–2036. Disponível em: <https://doi.org/10.1109/aero.2002.1035370>. Citado na página 17.

GOLLER, M.; STEINHARDT, F.; KERSCHER, T.; ZOLLNER, J. M.; DILLMANN, R. Proactive avoidance of moving obstacles for a service robot utilizing a behavior-based control. In: **2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. IEEE, 2010. p. 5984–5989. Disponível em: <https://doi.org/10.1109/iros.2010.5651506>. Citado na página 24.

GRASSI, V.; OKAMOTO, J. Robótica móvel. In: VARIOS (Ed.). Rio de Janeiro: LTC, 2014. cap. 4, p. 47–60. ISBN 978-8521623038. Citado 2 vezes nas páginas: 22 e 23.

____. ____. In: VARIOS (Ed.). Rio de Janeiro: LTC, 2014. cap. 5, p. 61–80. ISBN 978-8521623038. Citado na página 23.

GRISETTI, G.; STACHNISS, C.; BURGARD, W. Improved techniques for grid mapping with rao-blackwellized particle filters. **IEEE Transactions on Robotics**, IEEE, v. 23, n. 1, p. 34–46, 2007. Disponível em: <https://doi.org/10.1109/tro.2006.889486>. Citado na página 27.

IFR. **Industrial robots post a new sales record in 2015**. 2016. Disponível em: <https://ifr.org/img/uploads/Executive_Summary_WR_Industrial_Robots_20161.pdf>. Acesso em: 01-08-2017. Citado na página 16.

ISO. **8373: 2012 – Robots and robotic devices – Vocabulary**. Geneva, Switzerland, 2012. 1–38 p. Disponível em: <https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en>. Citado na página 16.

KARAMAN, S.; WALTER, M. R.; PEREZ, A.; FRAZZOLI, E.; TELLER, S. Anytime motion planning using the RRT*. In: **Proceedings of the IEEE International Conference on Robotics and Automation**. IEEE, 2011. p. 1478–1483. Disponível em: <https://doi.org/10.1109/icra.2011.5980479>. Citado 3 vezes nas páginas: 30, 37 e 41.

KAWARAZAKI, N.; DIAZ, A. I. B. Gesture recognition system for wheelchair control using a depth sensor. In: **2013 IEEE Symposium on Computational Intelligence in Rehabilitation and Assistive Technologies (CIRAT)**. IEEE, 2013. Disponível em: <http://dx.doi.org/10.1109/CIRAT.2013.6613822>. Citado na página 84.

KHATIB, O. Real-time obstacle avoidance for manipulators and mobile robots. **The international journal of robotics research**, v. 5, n. 1, p. 90–98, 1986. Disponível em: <https://doi.org/10.1177/027836498600500106>. Citado 2 vezes nas páginas: 23 e 30.

KO, I.; KIM, B.; PARK, F. C. Randomized path planning on vector fields. **The International Journal of Robotics Research**, v. 33, n. 13, p. 1664–1682, 2014. Disponível em: <https://doi.org/10.1177/0278364914545812>. Citado 5 vezes nas páginas: 20, 30, 44, 45 e 58.

KOHLBRECHER, S.; STRYK, O. von; MEYER, J.; KLINGAUF, U. A flexible and scalable SLAM system with full 3D motion estimation. In: **2011 IEEE International Symposium on Safety, Security, and Rescue Robotics**. IEEE, 2011. p. 155–160. Disponível em: <https://doi.org/10.1109/ssrr.2011.6106777>. Citado na página 27.

KUFFNER, J.; LAVALLE, S. RRT-connect: An efficient approach to single-query path planning. In: **Proceedings of the IEEE International Conference on Robotics and Automation**. IEEE, 2000. p. 995–1001. Disponível em: <https://doi.org/10.1109/robot.2000.844730>. Citado na página 58.

LEVINE, S.; BELL, D.; JAROS, L.; SIMPSON, R.; KOREN, Y.; BORENSTEIN, J. The NavChair assistive wheelchair navigation system. **IEEE Transactions on Rehabilitation Engineering**, IEEE, v. 7, n. 4, p. 443–451, 1999. Disponível em: <https://doi.org/10.1109/86.808948>. Citado 2 vezes nas páginas: 26 e 31.

LI, X.; GONG, Y.; JIA, S.; QIU, H. VFHF: A combining obstacle avoidance method based on laser rangefinder. In: **2016 IEEE International Conference on Information and Automation (ICIA)**. IEEE, 2016. p. 825–830. Disponível em: <https://doi.org/10.1109/icinfa.2016.7831933>. Citado na página 30.

LOPES, A.; RODRIGUES, J.; PERDIGAO, J.; PIRES, G.; NUNES, U. A new hybrid motion planner: Applied in a brain-actuated robotic wheelchair. **IEEE Robotics & Automation Magazine**, IEEE, v. 23, n. 4, p. 82–93, 2016. Disponível em: <https://doi.org/10.1109/mra.2016.2605403>. Citado 2 vezes nas páginas: 24 e 26.

LYNCH, K. M.; PARK, F. C. **Modern Robotics: Mechanics, Planning, and Control**. Cambridge: Cambridge University Press, 2017. ISBN 978-1107156302. Citado 2 vezes nas páginas: 29 e 30.

MARDELL, M. **2017 marcará o início da era dos robôs?** 2017. BBC News. Disponível em: <http://www.bbc.com/portuguese/geral-38461827>. Acesso em: 2017-06-28. Citado na página 16.

MARDER-EPPSTEIN, E.; BERGER, E.; FOOTE, T.; GERKEY, B.; KONOLIGE, K. The office marathon: Robust navigation in an indoor office environment. In: **Proceedings of the IEEE International Conference on Robotics and Automation**. IEEE, 2010. p. 300–307. Disponível em: <https://doi.org/10.1109/robot.2010.5509725>. Citado 3 vezes nas páginas: 26, 27 e 31.

MONTANO, L.; ASENSIO, J. Real-time robot navigation in unstructured environments using a 3D laser rangefinder. In: **Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. IEEE, 1997. p. 526–532. Disponível em: <https://doi.org/10.1109/iros.1997.655062>. Citado na página 30.

MONTGOMERY, D. C.; RUNGER, G. C. **Applied Statistics and Probability for Engineers**. 6. ed. [S.l.]: John Wiley and Sons, 2003. Citado na página 62.

MÜLLER, J.; STACHNISS, C.; ARRAS, K. O.; BURGARD, W. Socially inspired motion planning for mobile robots in populated environments. In: **Proceedings of the International Conference on Cognitive Systems**. Springer, 2008. p. 85–90. Disponível em: <http://www2.informatik.uni-freiburg.de/~arras/papers/muellerCOGSYS08.pdf>. Citado na página 17.

NAKHAEINIA, D.; PAYEUR, P.; HONG, T. S.; KARASFI, B. A hybrid control architecture for autonomous mobile robot navigation in unknown dynamic environment. In: **2015 IEEE International Conference on Automation Science and Engineering (CASE)**. IEEE, 2015. v. 5, p. 1274–1281. Disponível em: <https://doi.org/10.1109/coase.2015.7294274>. Citado 2 vezes nas páginas: 24 e 27.

NISHIMORI, M.; SAITOH, T.; KONISHI, R. Voice controlled intelligent wheelchair. In: **SICE Annual Conference 2007**. [s.n.], 2007. p. 336 – 340. Disponível em: <http://dx.doi.org/10.1109/SICE.2007.4421003>. Citado na página 84.

PANDEY, A. K.; ALAMI, R. A framework towards a socially aware mobile robot motion in human-centered dynamic environment. In: **2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. IEEE, 2010. Disponível em: <https://doi.org/10.1109/iros.2010.5649688>. Citado na página 83.

PEREIRA, G. A.; CHOUDHURY, S.; SCHERER, S. A framework for optimal repairing of vector field-based motion plans. In: **2016 International Conference on Unmanned Aircraft Systems (ICUAS)**. [s.n.], 2016. p. 261–266. Disponível em: <https://doi.org/10.1109/ICUAS.2016.7502525>. Citado 4 vezes nas páginas: 20, 30, 44 e 45.

QU, P.; XUE, J.; MA, L.; MA, C. A constrained VFH algorithm for motion planning of autonomous vehicles. In: **2015 IEEE Intelligent Vehicles Symposium (IV)**. IEEE, 2015. p. 700–705. Disponível em: <https://doi.org/10.1109/ivs.2015.7225766>. Citado na página 30.

QUIGLEY, M.; CONLEY, K.; GERKEY, B. P.; FAUST, J.; FOOTE, T.; LEIBS, J.; WHEELER, R.; NG, A. Y. ROS: an open-source robot operating system. In: **ICRA Workshop on Open Source Software**. [s.n.], 2009. p. 1–6. Disponível em: <http://www.willowgarage.com/sites/default/files/icraoss09-ROS.pdf>. Citado na página 24.

RENAWI, A.; JARADAT, M. A.; ABDEL-HAFEZ, M. ROS validation for non-holonomic differential robot modeling and control: Case study: Kobuki robot trajectory tracking controller. In: **7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)**. [s.n.], 2017. p. 1–5. Disponível em: <https://doi.org/10.1109/icmsao.2017.7934880>. Citado na página 24.

RICHTER, C.; BRY, A.; ROY, N. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In: ____. **Robotics Research: The 16th International Symposium ISRR**. Springer International Publishing, 2016. p. 649–666. ISBN 978-3-319-28872-7. Disponível em: <https://doi.org/10.1007/978-3-319-28872-7_37>. Citado na página 83.

ROS. **GMapping package**. 2017. Disponível em: <http://wiki.ros.org/gmapping>. Acesso em: 2017-08-01. Citado na página 27.

____. **Hector mapping package**. 2017. Disponível em: <http://wiki.ros.org/hector_slam>. Acesso em: 2017-08-01. Citado na página 27.

SCHWESINGER, D.; SHARIATI, A.; MONTELLA, C.; SPLETZER, J. A smart wheelchair ecosystem for autonomous navigation in urban environments. **Autonomous Robots**, Springer Nature, v. 41, n. 3, p. 519–538, 2016. Disponível em: <https://doi.org/10.1007/s10514-016-9549-1>. Citado 2 vezes nas páginas: 26 e 32.

SCUDELLARI, M. **Self-Driving Wheelchairs Debut in Hospitals and Airports**. 2017. Disponível em: <https://spectrum.ieee.org/the-human-os/biomedical/devices/selfdriving-wheelchairs-debut-in-hospitals-and-airports>. Acesso em: 2017-08-17. Citado na página 20.

SHOVAL, S.; ULRICH, I.; BORENSTEIN, J. Navbelt and the guide-cane [obstacle-avoidance systems for the blind and visually impaired]. **IEEE Robotics & Automation Magazine**, IEEE, v. 10, n. 1, p. 9–20, 2003. Disponível em: <https://doi.org/10.1109/mra.2003.1191706>. Citado na página 26.

SIEGWART, R.; ARRAS, K. O.; BOUABDALLAH, S.; BURNIER, D.; FROIDEVAUX, G.; GREPPIN, X.; JENSEN, B.; LOROTTE, A.; MAYOR, L.; MEISSER, M.; PHILIPPSEN, R.; PIGUET, R.; RAMEL, G.; TERRIEN, G.; TOMATIS, N. Robox at expo.02: A large-scale installation of personal robots. **Robotics and Autonomous Systems**, v. 42, n. 3-4, p. 203–222, 2003. Disponível em: <https://doi.org/10.1016/s0921-8890(02)00376-7>. Citado 2 vezes nas páginas: 26 e 27.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. **Introduction to Autonomous Mobile Robots (Intelligent Robotics and Autonomous Agents series)**. 2. ed. Cambridge: The MIT Press, 2011. ISBN 978-0262015356. Citado na página 24.

SIMPSON, R. C.; LOPRESTI, E. F.; COOPER, R. A. How many people would benefit from a smart wheelchair? **Journal of Rehabilitation Research and Development**, Journal of Rehabilitation Research & Development, v. 45, n. 1, p. 53–72, 2008. Disponível em: <https://doi.org/10.1682/jrrd.2007.01.0015>. Citado na página 17.

SOUZA, A. A. d. S.; SANTANA, A. M.; GONCALVES, L. M. G.; MEDEIROS, A. A. D. Robótica móvel. In: VARIOS (Ed.). Rio de Janeiro: LTC, 2014. cap. 9, p. 159–177. ISBN 978-8521623038. Citado na página 27.

STüCKLER, J.; SCHWARZ, M.; SCHADLER, M.; TOPALIDOU-KYNIAZOPOULOU, A.; BEHNKE, S. NimbRo explorer: Semiautonomous exploration and mobile manipulation in rough terrain. **Journal of Field Robotics**, v. 33, n. 4, p. 411–430, 2015. Disponível em: <https://doi.org/10.1002/rob.21592>. Citado 2 vezes nas páginas: 26 e 32.

TAKAHASHI, O.; SCHILLING, R. Motion planning in a plane using generalized voronoi diagrams. **IEEE Transactions on Robotics and Automation**, IEEE, v. 5, n. 2, p. 143–150, 1989. Disponível em: <https://doi.org/10.1109/70.88035>. Citado na página 30.

TANNUS, A. M.; TIERRA-CRIOLLO, C. J.; PEREIRA, G. A. S.; MELGES, D. B. Controle de cadeira de rodas usando potencial evocado visual em regime permanente. In: **XXIV Congresso Brasileiro de Engenharia Biomédica**. Uberlândia: [s.n.], 2014. p. 1003–1006. Citado na página 84.

THRUN, S. Particle filters in robotics. In: **Proceedings of the Eighteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)**. Morgan Kaufmann, 2002. v. 1, p. 511–518. ISBN 1-558608974. Disponível em: <https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=3&smnu=2&article_id=898>. Citado na página 27.

THRUN, S.; BENNEWITZ, M.; BURGARD, W.; CREMERS, A.; DELLAERT, F.; FOX, D.; HAHNEL, D.; ROSENBERG, C.; ROY, N.; SCHULTE, J.; SCHULZ, D. MINERVA: a second-generation museum tour-guide robot. In: **Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)**. IEEE, 1999. v. 3, p. 1999–2005. Disponível em: <https://doi.org/10.1109/robot.1999.770401>. Citado 3 vezes nas páginas: 26, 27 e 31.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)**. Cambridge: The MIT Press, 2005. ISBN 978-0262201629. Citado na página 26.

ULRICH, I.; BORENSTEIN, J. VFH*: local obstacle avoidance with look-ahead verification. In: **Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)**. IEEE, 2000. v. 3, p. 2505–2511. Disponível em: <https://doi.org/10.1109/robot.2000.846405>. Citado na página 30.

WANG, Q.; WIEGHARDT, C. S.; JIANG, Y.; GONG, J.; WAGNER, B. Generalized path corridor-based local path planning for nonholonomic mobile robot. In: **2015 IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)**. IEEE, 2015. p. 1922–1927. Disponível em: <https://doi.org/10.1109/itsc.2015.311>. Citado na página 30.

XIE, X.; YU, Y.; LIN, X.; SUN, C. An EKF SLAM algorithm for mobile robot with sensor bias estimation. In: **2017 32nd Youth Academic Annual Conference of Chinese Association of Automation (YAC)**. [s.n.], 2017. p. 281–285. Disponível em: <https://doi.org/10.1109/yac.2017.7967420>. Citado na página 27.

YANG, H.; LIM, J.; YOON, S. eui. Anytime RRBT for handling uncertainty and dynamic objects. In: **2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. IEEE, 2016. p. 4786–4793. Disponível em: <https://doi.org/10.1109/IROS.2016.7759703>. Citado na página 23.

ZHIWEI, S.; YIYAN, W.; CHANGJIU, Z.; YI, Z. A new sensor fusion framework to deal with false detections for low-cost service robot localization. In: **2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)**. IEEE, 2013. p. 197–202. Disponível em: <https://doi.org/10.1109/robio.2013.6739458>. Citado na página 25.

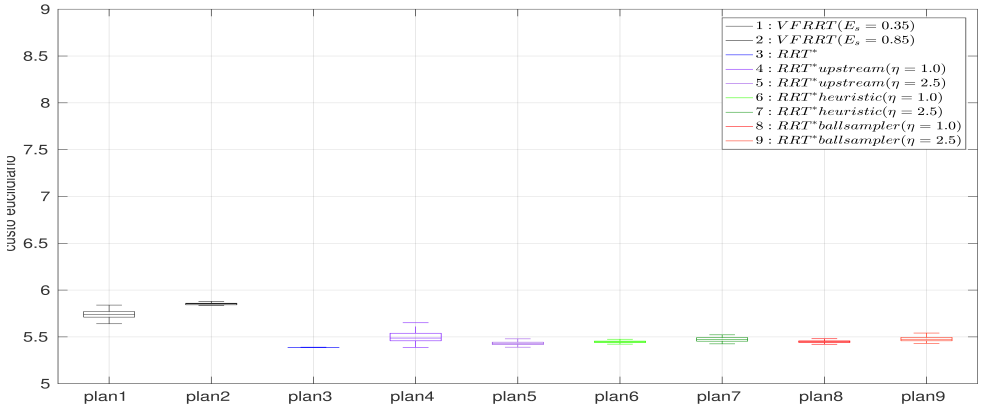
APÊNDICE

A

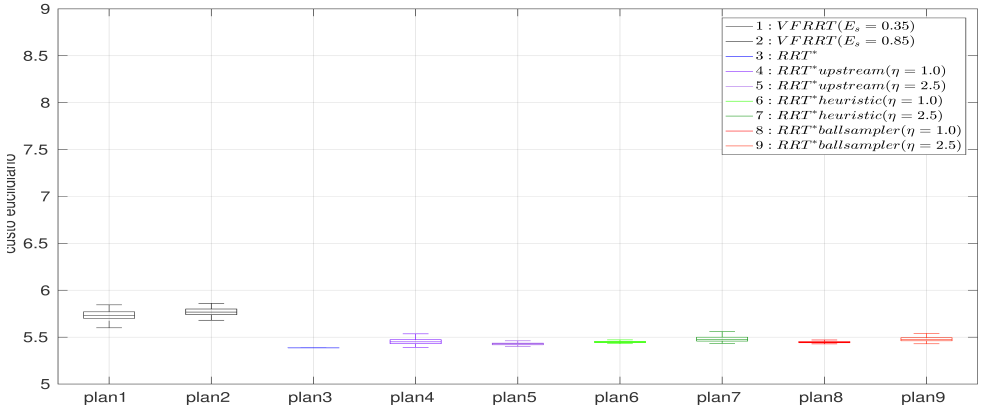
DIAGRAMA DE CAIXA PARA OS RESULTADOS OBTIDOS EM SIMULAÇÃO

O diagrama de caixa, do inglês *box plot*, popularizado pelo matemático John W. Tukey, é uma ferramenta que permite a representação visual de informações de probabilidade dada uma determinada amostra, tais como a mediana, o grau de dispersão e o limite inferior e o superior. Por meio desse tipo de diagrama, este apêndice tem como objetivo apresentar os resultados obtidos no experimento da Seção 4.3.2.

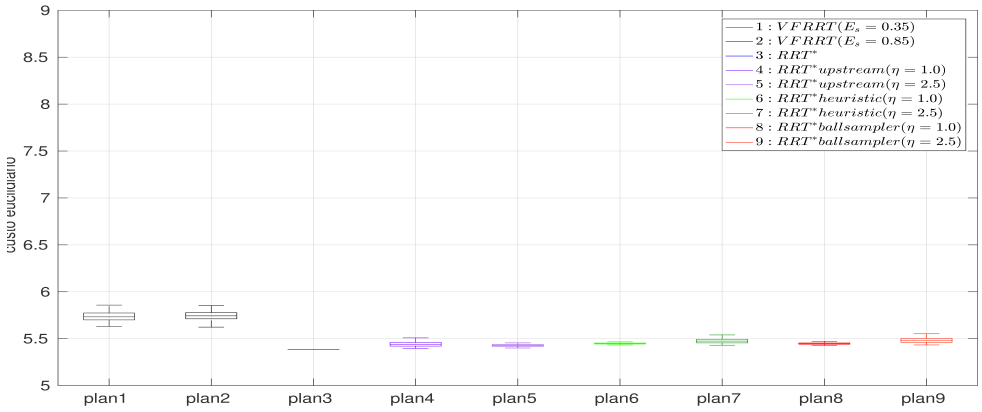
As Figuras de A.1 a A.3 apresentam os resultados obtidos pelos planejadores utilizados nesse experimento para a situação em que o ambiente não possui obstáculos, para os critérios: (i) custo euclideano, (ii) custo *upstream* e (iii) suavidade, variando o tempo de planejamento entre 0,45 s e 5,0 s. Similarmente, as Figuras de A.4 a A.6 apresentam os resultados obtidos em um ambiente com 25 obstáculos e as Figuras de A.7 a A.9 apresentam os resultados obtidos em ambiente com 50 obstáculos. É válido ressaltar que a largura da caixa desses diagramas representa a faixa referente ao resultado de 50% dos dados da amostra e que a linha dentro dessa caixa representa a mediana da amostra.



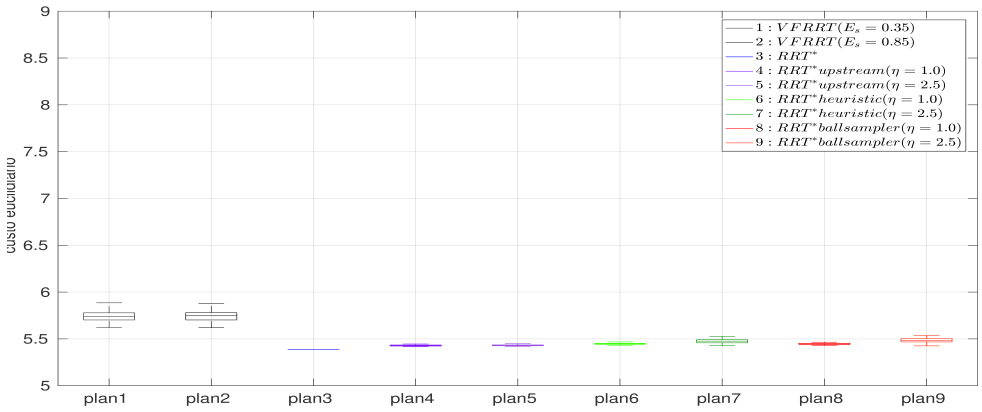
(a) $t_s = 0,45\,\text{s}$.



(b) $t_s = 1\,\text{s}$

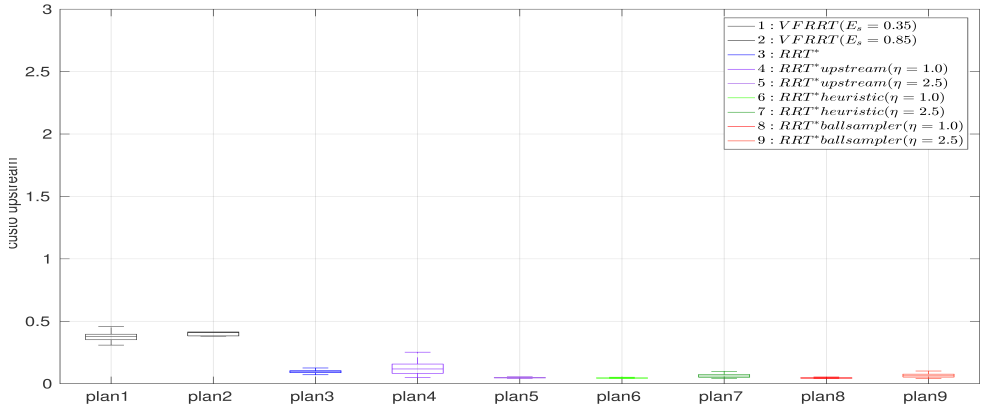


(c) $t_s = 2\,\text{s}$

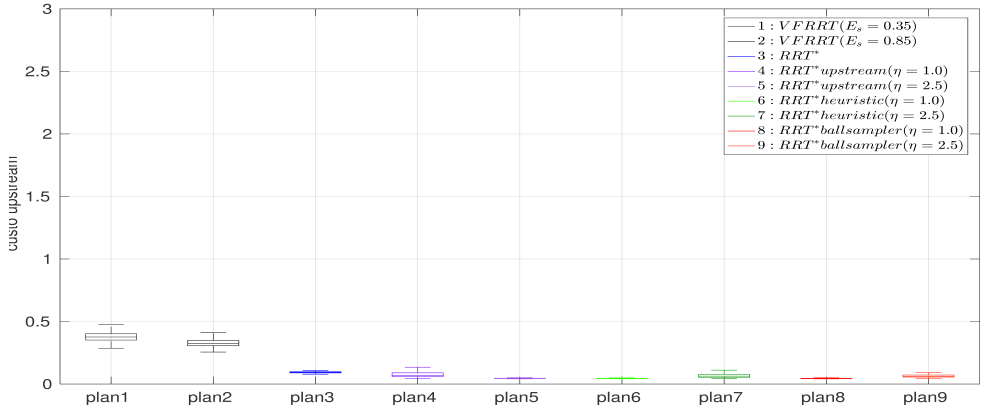


(d) $t_s = 5\,\text{s}$

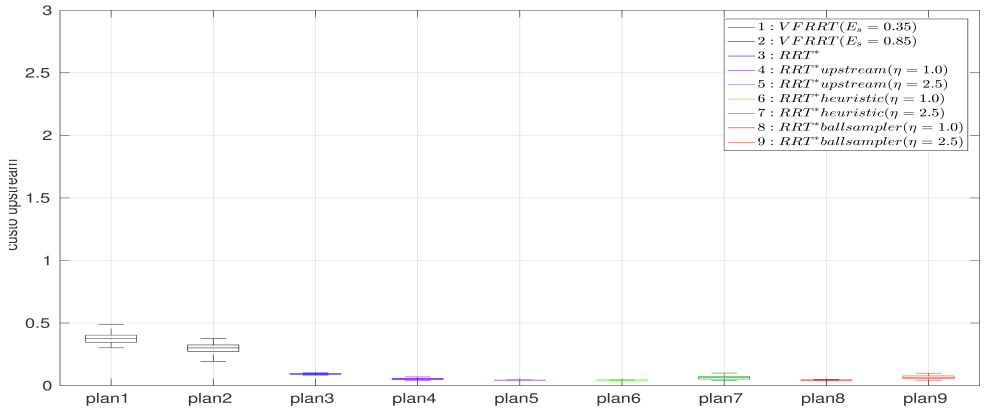
Figura A.1 – Diagramas de caixa para comparação entre os planejadores quanto ao custo euclideano, considerando um ambiente sem obstáculos ao variar o tempo de planejamento.



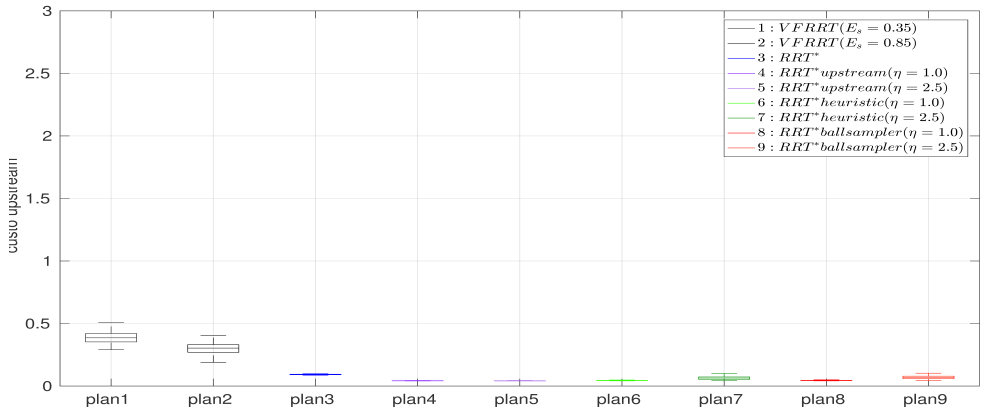
(a) $t_s = 0,45\,\text{s}$.



(b) $t_s = 1\,\text{s}$

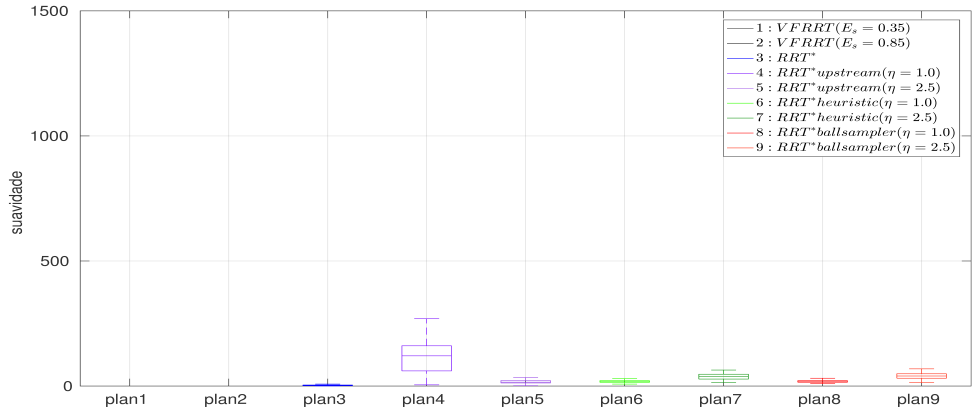


(c) $t_s = 2\,\text{s}$

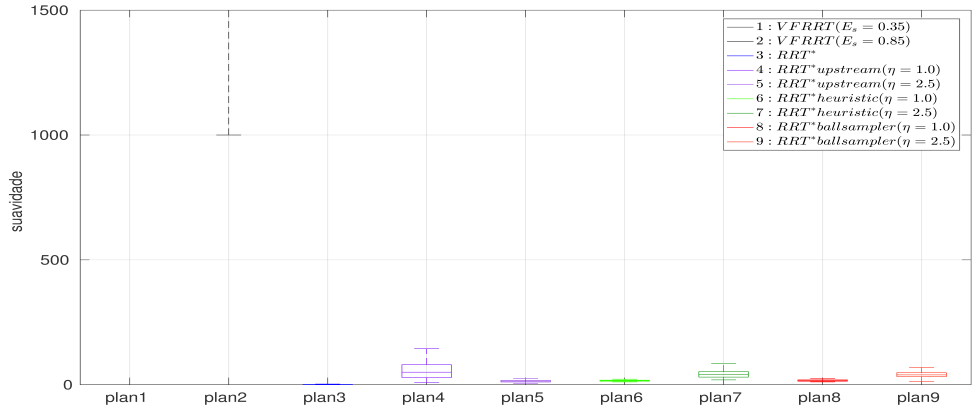


(d) $t_s = 5\,\text{s}$

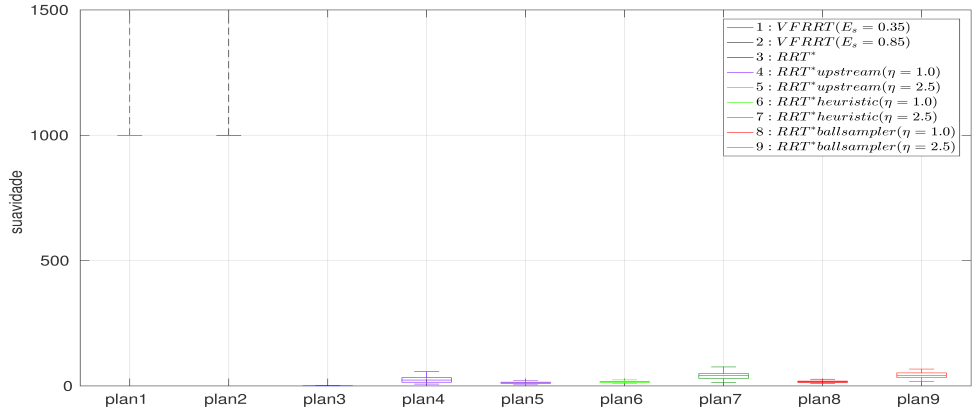
Figura A.2 – Diagramas de caixa para comparação entre os planejadores quanto ao custo *upstream*, considerando um ambiente sem obstáculos ao variar o tempo de planejamento.



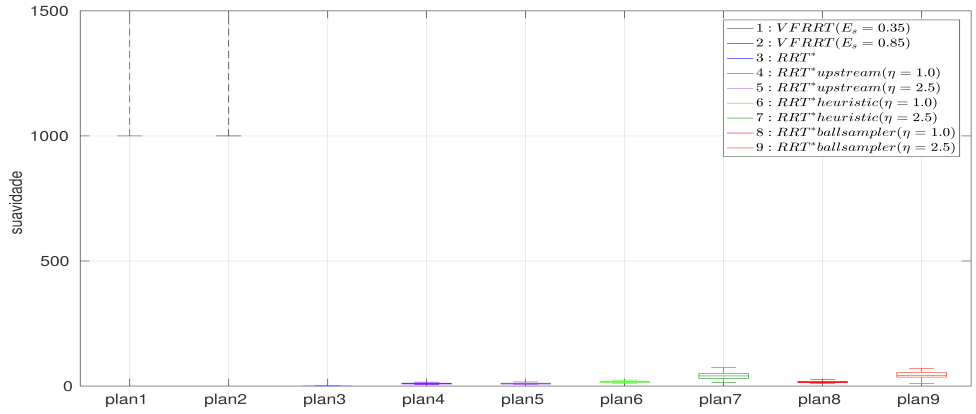
(a) $t_s = 0,45\,\text{s}$.



(b) $t_s = 1\,\text{s}$

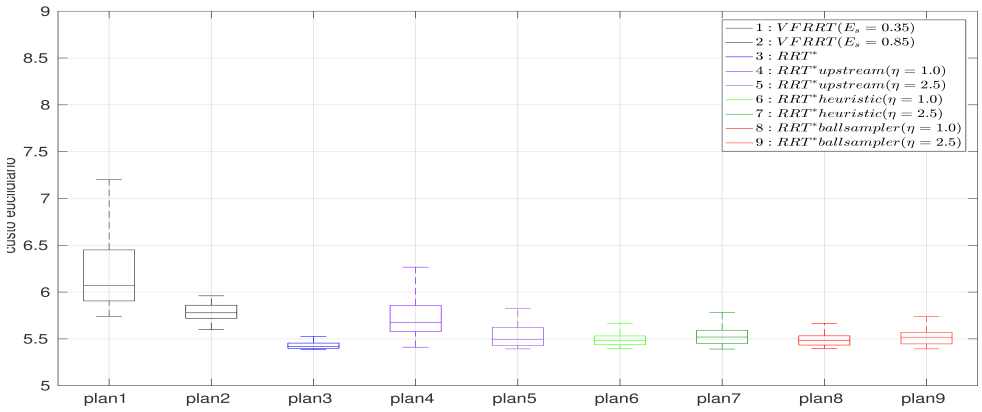


(c) $t_s = 2\,\text{s}$

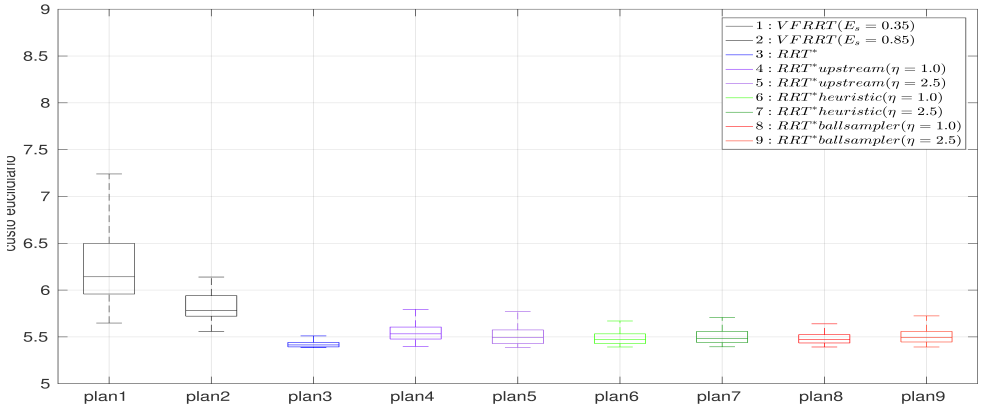


(d) $t_s = 5\,\text{s}$

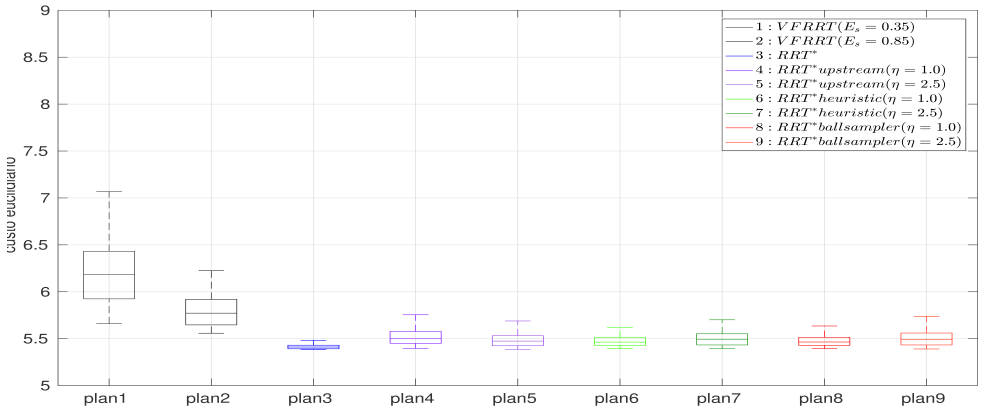
Figura A.3 – Diagrama de caixa para comparação entre os planejadores quanto à suavidade do caminho, considerando um ambiente sem obstáculos ao variar o tempo de planejamento.



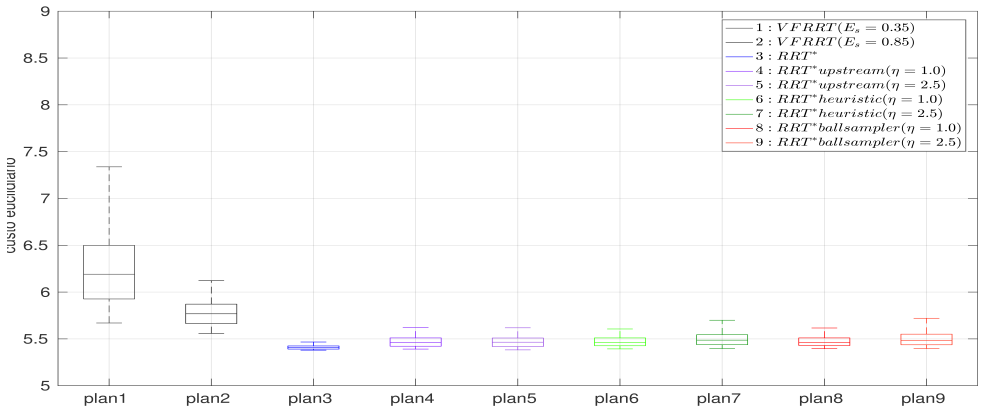
(a) $t_s = 0,45\,\text{s}$.



(b) $t_s = 1\,\text{s}$

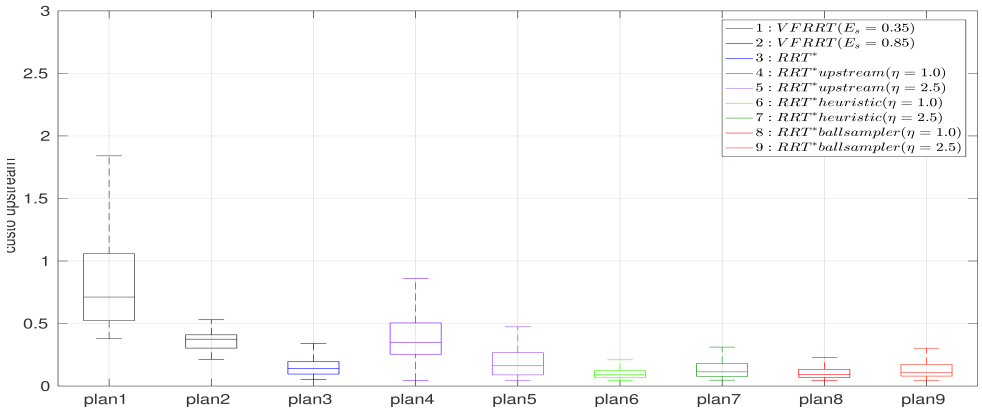


(c) $t_s = 2\,\text{s}$

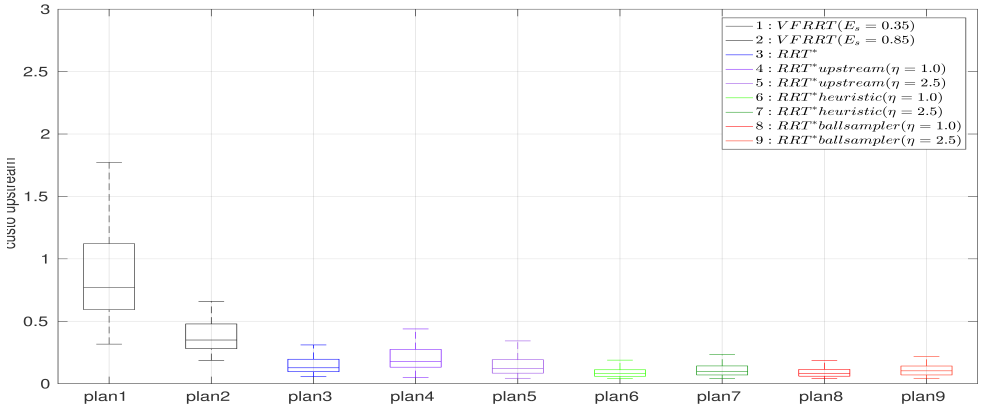


(d) $t_s = 5\,\text{s}$

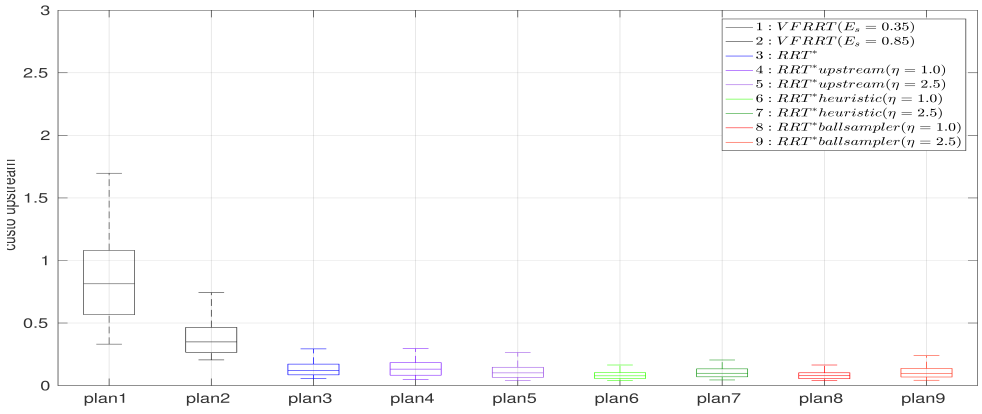
Figura A.4 – Diagramas de caixa para comparação entre os planejadores quanto ao custo euclideano, considerando um ambiente com 25 obstáculos ao variar o tempo de planejamento.



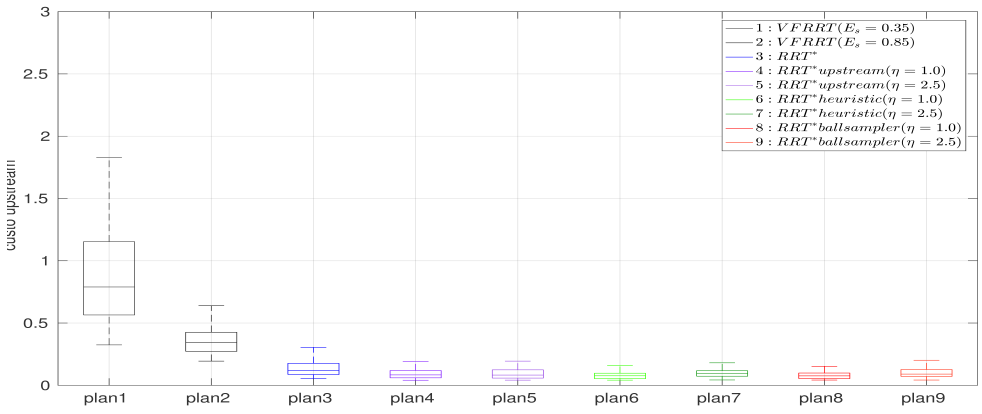
(a) $t_s = 0,45\,\text{s}$.



(b) $t_s = 1\,\text{s}$

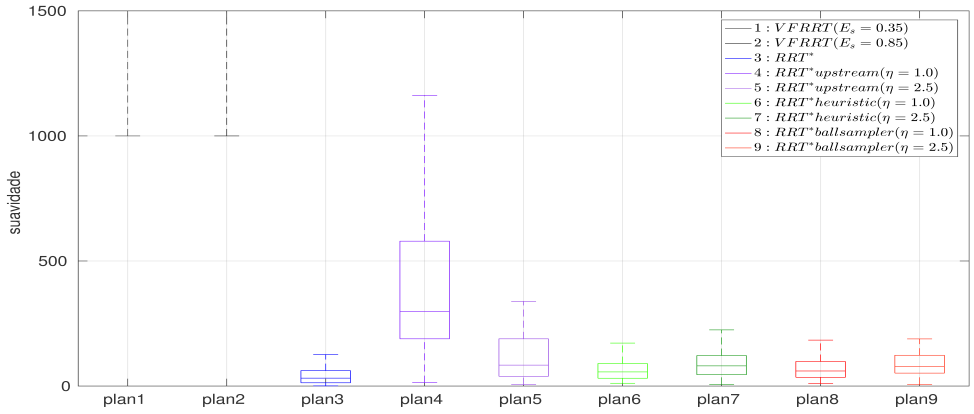


(c) $t_s = 2\,\text{s}$

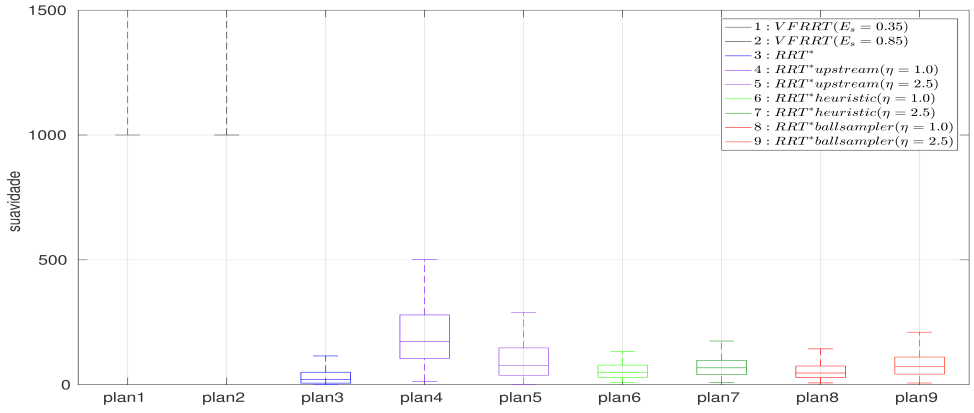


(d) $t_s = 5\,\text{s}$

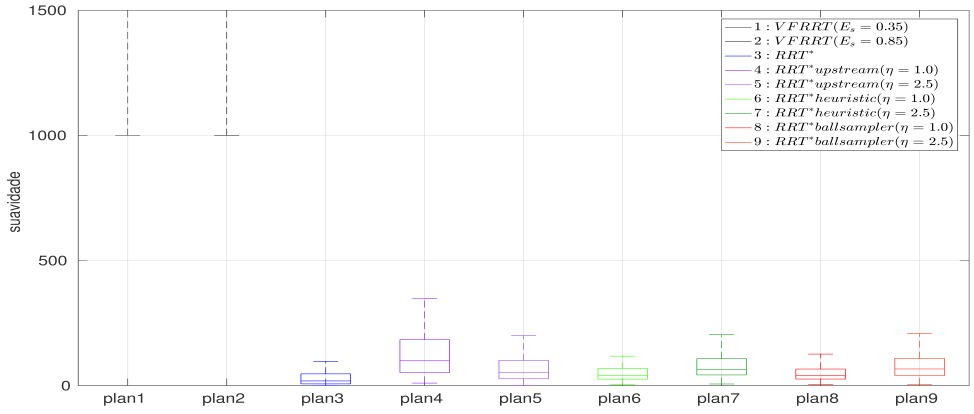
Figura A.5 – Diagramas de caixa para comparação entre os planejadores quanto ao custo *upstream*, considerando um ambiente com 25 obstáculos ao variar o tempo de planejamento.



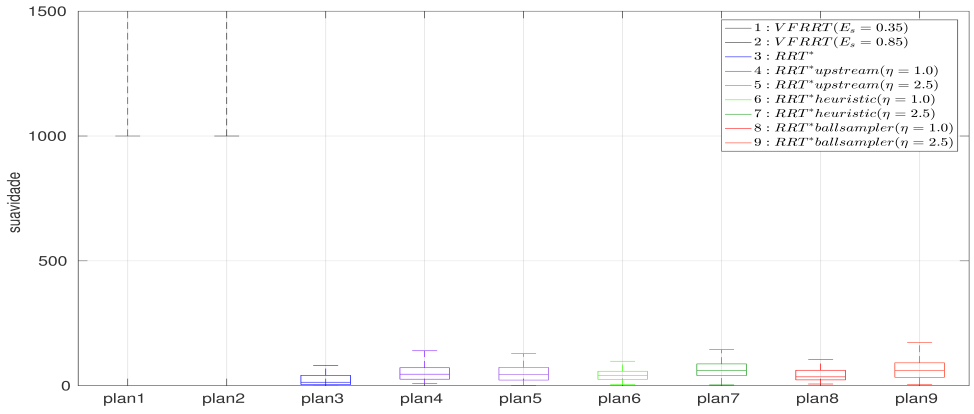
(a) $t_s = 0,45\,\text{s}$.



(b) $t_s = 1\,\text{s}$

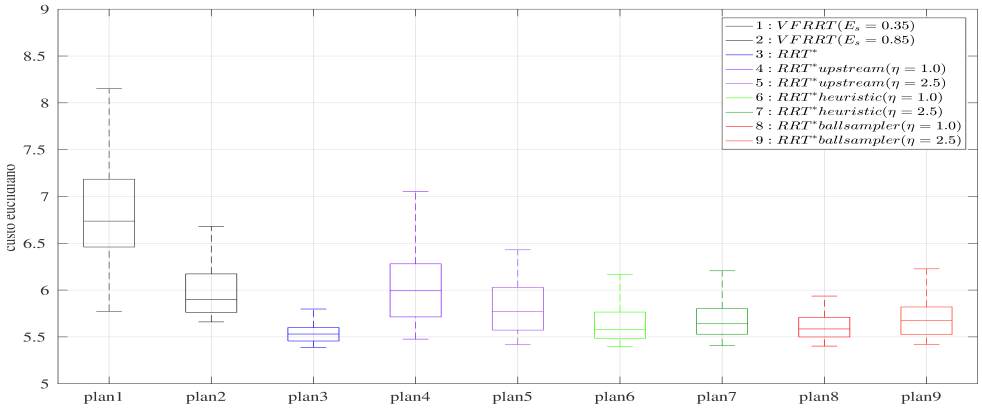


(c) $t_s = 2\,\text{s}$

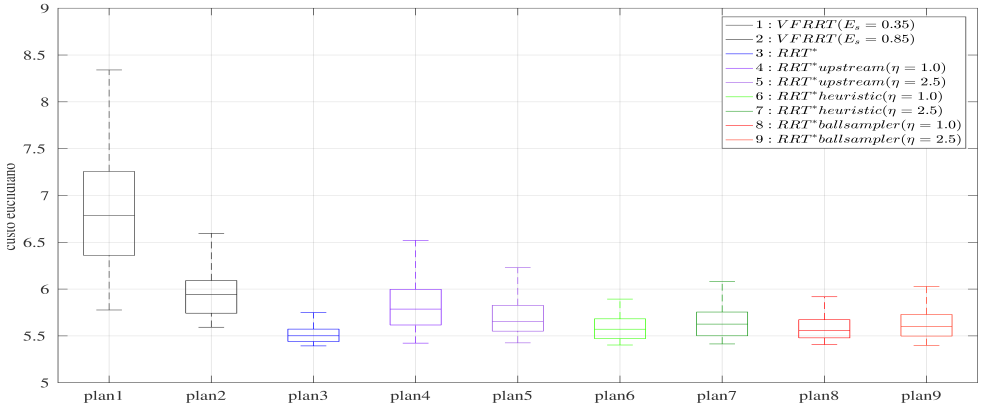


(d) $t_s = 5\,\text{s}$

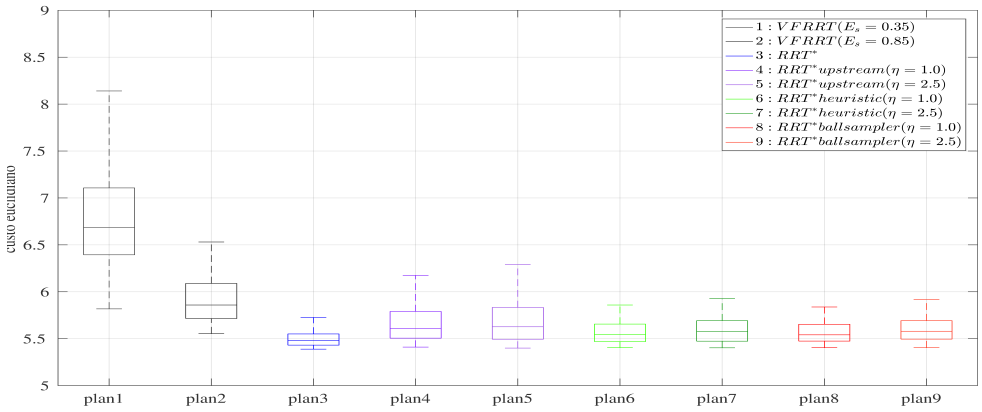
Figura A.6 – Diagrama de caixa para comparação entre os planejadores quanto à suavidade do caminho, considerando um ambiente com 25 obstáculos ao variar o tempo de planejamento.



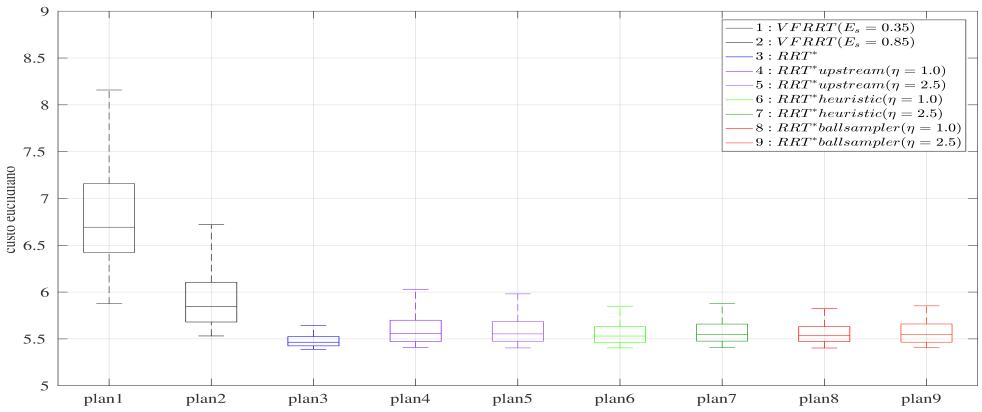
(a) $t_s = 0,45\,\text{s}$.



(b) $t_s = 1\,\text{s}$

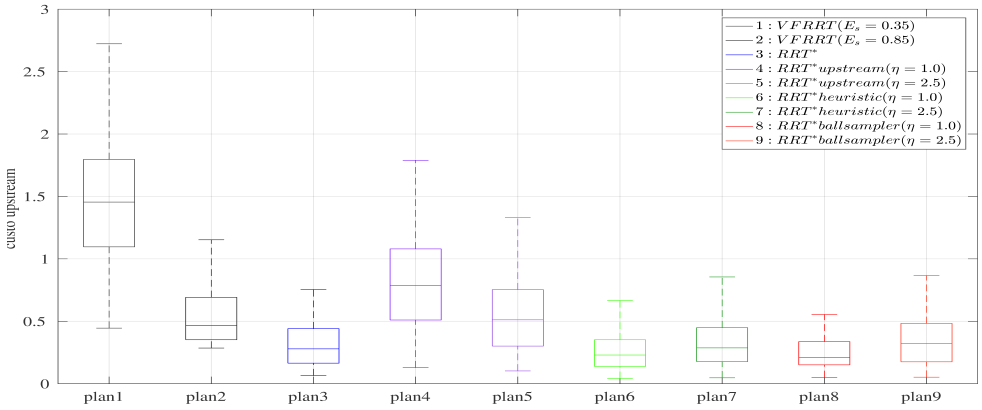


(c) $t_s = 2\,\text{s}$

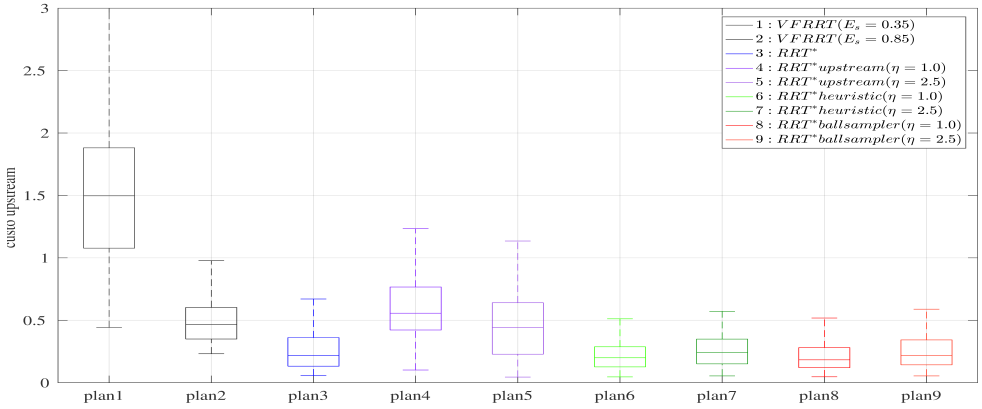


(d) $t_s = 5\,\text{s}$

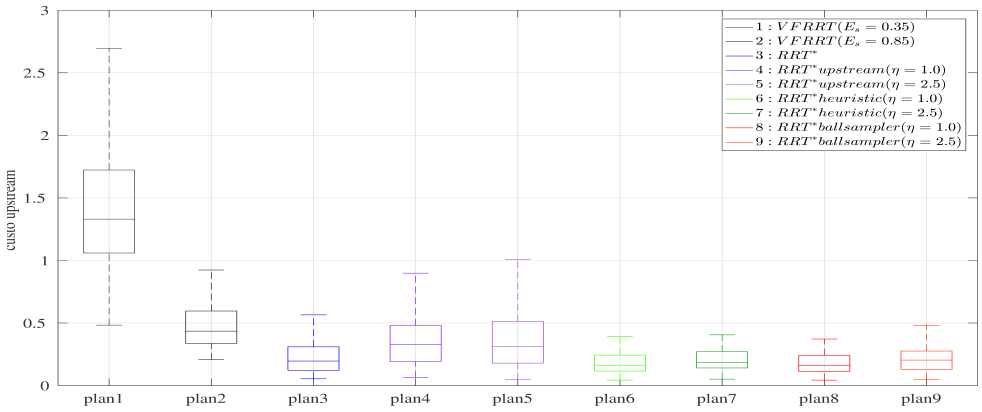
Figura A.7 – Diagramas de caixa para comparação entre os planejadores quanto ao custo euclideano, considerando um ambiente com 50 obstáculos ao variar o tempo de planejamento.



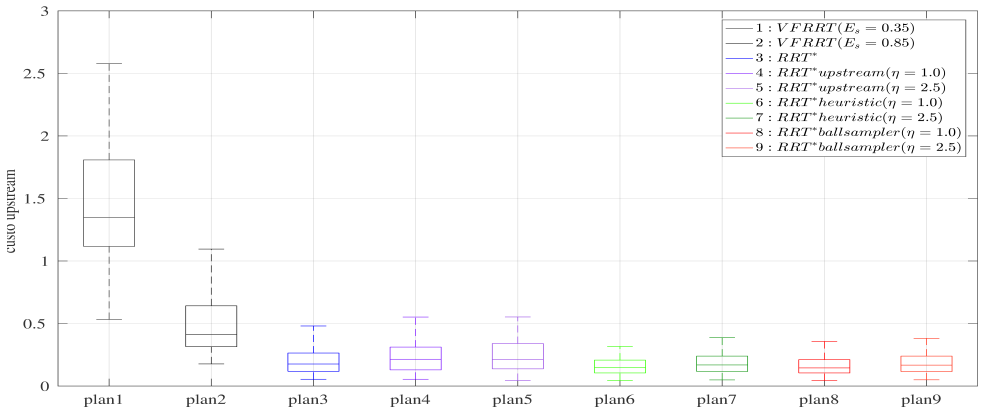
(a) $t_s = 0,45\,\text{s}$.



(b) $t_s = 1\,\text{s}$

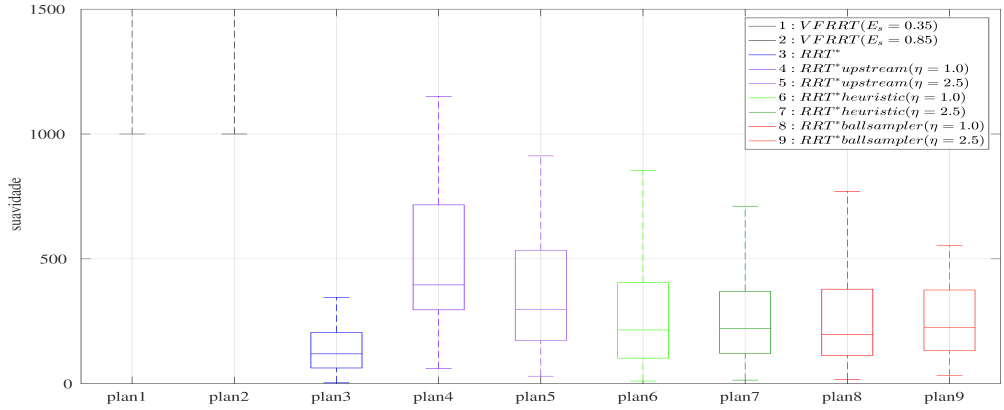


(c) $t_s = 2\,\text{s}$

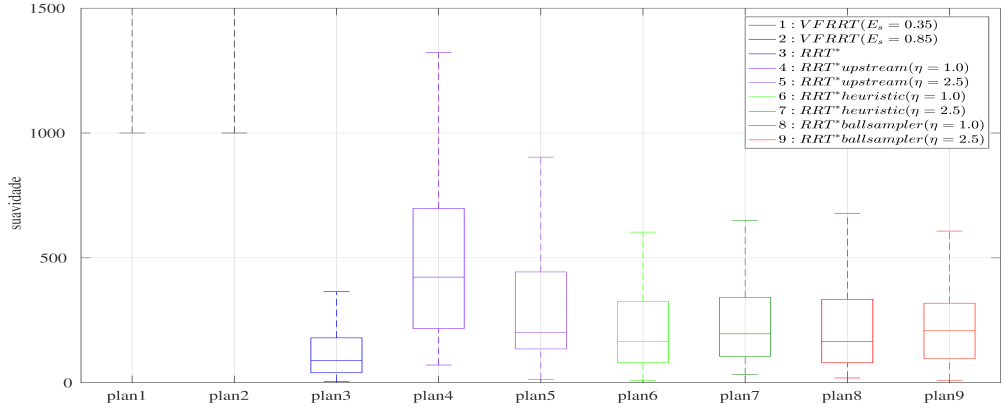


(d) $t_s = 5\,\text{s}$

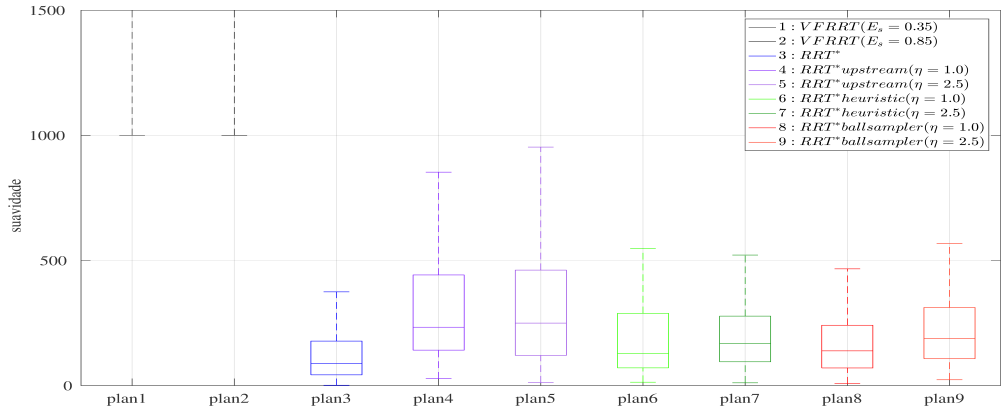
Figura A.8 – Diagramas de caixa para comparação entre os planejadores quanto ao custo *upstream*, considerando um ambiente com 50 obstáculos ao variar o tempo de planejamento.



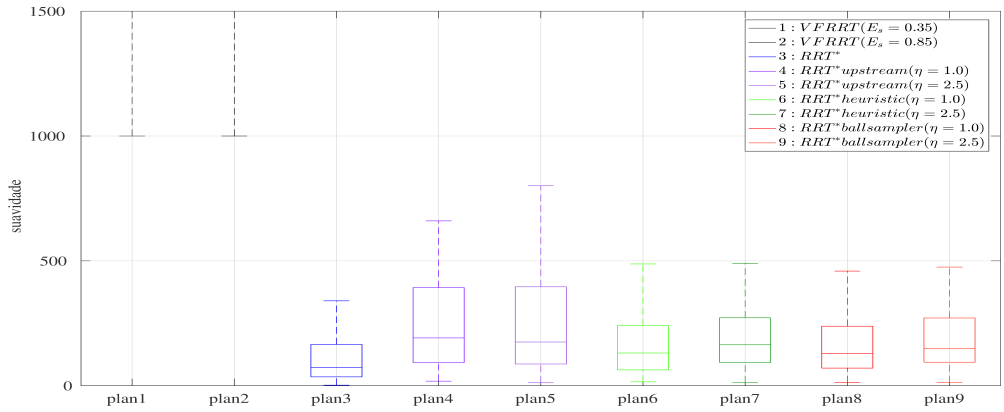
(a) $t_s = 0,45\,\text{s}$.



(b) $t_s = 1\,\text{s}$



(c) $t_s = 2\,\text{s}$



(d) $t_s = 5\,\text{s}$

Figura A.9 – Diagrama de caixa para comparação entre os planejadores quanto à suavidade do caminho, considerando um ambiente com 50 obstáculos ao variar o tempo de planejamento.