André Chaves Magalhães

**Planejamento Cinemático-Dinâmico de
Movimento com Desvio Local de Obstáculos
Utilizando Malhas de Estados**

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Mestre em Ciências, Programa de Engenharia Elétrica.

Área de Concentração: Sistemas Dinâmicos
Orientador: Prof. Dr. Valdir Grassi Jr.

São Carlos
2013

Trata-se da versão corrigida da dissertação. A versão original se encontra disponível na EESC/USP que aloja o Programa de Pós-Graduação de Engenharia Elétrica.

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

M188p Magalhães, André Chaves
Planejamento Cinemático-Dinâmico de Movimento com
Desvio Local de Obstáculo Utilizando Malhas de Estados
/ André Chaves Magalhães; orientador Valdir Grassi
Junior. São Carlos, 2013.

Dissertação (Mestrado) - Programa de Pós-Graduação
em Engenharia Elétrica e Área de Concentração em
Sistemas Dinâmicos -- Escola de Engenharia de São
Carlos da Universidade de São Paulo, 2013.

1. Robôs Móveis Autônomos. 2. Planejamento de
Movimento Cinemático-Dinâmico. 3. Malha de Estados. I.
Título.

Dedicatória

Dedico este trabalho aos meus pais.

Agradecimentos

Ao meu orientador, Prof. Dr. Valdir Grassi Junior, meus sinceros agradecimentos pelo empenho e dedicação demonstrados, pela competência, pelo incentivo, simpatia e presteza no auxílio às atividades e pelas valiosas discussões durante o andamento da pesquisa, pelo auxílio no desenvolvimento desta Dissertação e, principalmente, pela confiança depositada em mim.

A instituição Universidade de São Paulo em São Carlos por toda a infraestrutura oferecida.

A todos os professores pela dedicação e entusiasmo demonstrado ao longo das disciplinas que cursei e aos demais funcionários do Departamento de Engenharia Elétrica da USP São Carlos que de alguma forma, direta ou indiretamente, contribuíram com o trabalho.

Aos colegas do Laboratório de Sistemas Inteligentes - LASI pela recepção, pelos bons momentos que passamos juntos e pela espontaneidade na troca de informações e materiais numa grandiosa demonstração de amizade.

Aos colegas do Laboratório de Robótica Móvel - LRM/ICMC por todo apoio e suporte fornecidos.

A CAPES e a FAPESP pelo apoio financeiro concedido para o desenvolvimento deste trabalho.

A todos aqueles que através de uma simples sugestão, uma crítica ou um pensamento positivo, colaboraram de alguma forma com o sucesso deste trabalho.

Aos meus pais pela presença constante nas horas de alegria e pelo apoio incondicional nos momentos mais difíceis nesta e nas outras jornadas.

E, finalmente, a Deus pela benção, pela saúde, pela oportunidade e pelo privilégio que me foi concedido ao frequentar este curso e poder estabelecer mais esta etapa em minha vida.

*“Não sabemos quanto tempo nos resta,
não podemos desperdiçá-lo lamentando
coisas que não podemos mudar.”*

Stephen King

RESUMO

MAGALHAES, A. C. *Planejamento Cinemático-Dinâmico de Movimento com Desvio Local de Obstáculos Utilizando Malhas de Estados*. São Carlos, 2013, Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo.

Planejamento de movimento tem o propósito de determinar quais movimentos o robô deve realizar para que alcance posições ou configurações desejadas no ambiente sem que ocorram colisões com obstáculos. É comum na robótica móvel simplificar o planejamento de movimento representando o robô pelas coordenadas do seu centro e desconsiderando qualquer restrição cinemática e dinâmica de movimento. Entretanto, a maioria dos robôs móveis possuem restrições cinemáticas não-holonômicas, e para algumas tarefas e robôs, é importante considerar tais restrições juntamente com o modelo dinâmico do robô na tarefa de planejamento. Assim é possível determinar um caminho que possa ser de fato seguido pelo robô. Nesse trabalho é proposto um método de planejamento cinemático-dinâmico que permite planejar trajetórias para robôs móveis usando malhas de estados. Essa abordagem considera a cinemática e a dinâmica do robô para gerar trajetórias possíveis de serem executadas e livre de colisões com obstáculos. Quando obstáculos não representados no mapa são detectados pelos sensores do robô, uma nova trajetória é gerada para desviar desses obstáculos. O planejamento de movimento utilizando malhas de estados foi associado a um algoritmo de desvio de obstáculos baseado no método da janela dinâmica (DWA). Esse método é responsável pelo controle de seguimento de trajetória, garantindo a segurança na realização da tarefa durante a navegação. As trajetórias planejadas foram executadas em duas plataformas distintas. Essas plataformas foram utilizadas em tarefas de navegação em ambientes simulados interno e externo e em ambientes reais. Para navegação em ambientes internos utilizou-se o robô móvel Pioneer 3AT e para navegação em ambientes externos utilizou-se o veículo autônomo elétrico CaRINA I que está sendo desenvolvido no ICMC-USP com apoio do Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos (INCT-SEC).

Palavras-Chave: *Robôs Móveis Autônomos, Planejamento de Movimento Cinemático-Dinâmico, Malha de Estados.*

ABSTRACT

MAGALHAES, A. C. *Kinematic-Dynamic Motion Planning with Local Deviation of Obstacles Using Lattice States*. São Carlos, 2013, Dissertation (Master) - School of Engineering of São Carlos, University of São Paulo.

Motion planning aims to determine which movements the robot must accomplish to reach a desired position or configuration in the environment without the occurrence of collisions with obstacles. It is common in mobile robotics to simplify the motion planning representing the robot by the coordinates of its center of gravity and ignoring any kinematic and dynamic constraint motion. However, most mobile robots have non-holonomic kinematic constraints, and for some tasks and robots, it is important to consider these constraints together with the dynamic model of the robot in task planning. Thus it is possible to determine a path that can actually be followed by the robot. Here we propose a method for kinematic-dynamic path planning using lattice states. This approach considers the kinematic and dynamic of the robot to generate generate feasible trajectories free of collisions with obstacles. When obstacles not represented on the map are detected by the sensors of the robot, a new trajectory is generated to avoid these obstacles. The motion planning using lattice state was associated with an obstacle avoidance algorithm based on the dynamic window approach (DWA). This method is responsible for trajectory tracking to ensure safety in navigation tasks. This method was applied in two distinct platforms. These platforms were used for navigation tasks in both indoor and outdoor simulated environments, as well as, in real environments. For navigation in indoor environments we used a Pioneer 3AT robot and for outdoor navigation we used the autonomous electric vehicle CaRINA1 being developed at ICMC-USP with support National Institute of Science and Technology in Critical Embedded Systems (INCT-SEC).

Keywords: *Autonomous Mobile Robots, Kinematic-Dynamic Motion Planning, Lattice States*

LISTA DE FIGURAS

2.1	Grade de Ocupação (Wolf et al., 2009).	10
2.2	Diferença entre espaço de trabalho e espaço de configurações.	13
2.3	Planejamento local de caminhos alternativos (Thrun et al., 2006).	15
2.4	Movimentos Primitivos. Cada estado S é um estado possível de ser alcançado pelo robô.	18
2.5	Exemplo de uma malha	19
2.6	Malha de Múltipla Resolução	20
2.7	Algoritmo Anytime Dynamic A* (Likhachev et al., 2005)	22
2.8	Exemplo de Navegação com AD* (Likhachev et al., 2005).	24
2.9	Heurística de restrições mecânicas (linha contínua) e de restrições no ambiente (linha tracejada). As posições inicial e desejada do veículo são mostradas como retângulos azuis e vermelhos, respectivamente. (a) A heurística de restrições mecânicas perfeitamente informa quando não há obstáculos presentes no ambiente. (b) A heurística com restrição do ambiente pode proporcionar benefícios significativos quando existem obstáculos (Likhachev e Ferguson, 2009).	25
2.10	Instante de movimento do robô Synchro-Drive (Lima, 2010).	26

2.11	Exemplo dos conjuntos V_s , V_a , V_d e V_r , necessários para a aplicação do Método da Janela Dinâmica em um robô do tipo Synchro-Drive. Estes conjuntos foram obtidos por meio da leitura sensorial no ambiente da Figura 2.10, com os valores atuais das velocidades representados por (v_a, ω_a) . Em cinza escuro estão todos os pares de (v, ω) que implicam em um movimento inválido no robô, com possíveis colisões. A região destacada em verde são as velocidades válidas para a janela dinâmica.(Lima, 2010).	27
3.1	Laser SICK embarcado ao Pioneer.	30
3.2	Robô Pioneer 3AT	31
3.3	Geometria do robô móvel (Vaz, 2011).	32
3.4	Forças de tração e resistivas do robô móvel (Vaz, 2011).	34
3.5	Sensores embarcados no CaRINA 1.	37
3.6	Veículo Autônomo CARina 1	38
3.7	Sistema de percepção do CaRINA 1 (Prado, 2013). (a) Posicionamento e alcance de cada sensor laser, (b) Fusão dos sensores laser	39
3.8	Visão de 360° dos sensores laser. Pontos vermelhos representam a detecção de obstáculos pela fusão dos sensores laser.	39
3.9	Modelo Cinemático	40
4.1	Diagrama de troca de informações entre nós para uma determinada aplicação robótica.	45
4.2	Ambiente de simulação do ROS . (a) Simulador STAGE. (b) Simulador GAZEBO (c) Visualizador de ambientes 3D do ROS.	45
4.3	Representação em grafo para planejamento 2D	46
4.4	Representação em grafo baseado em malha de estados para planejamento de movimento 3D (Likhachev, 2009).	47
4.5	Estimação da posição pelo AMCL	48

5.1	Representação tridimensional da plataforma Pioneer 3AT. (a) Modelo 3D. (b) Modelo em estrutura de árvore. (c) Pioneer 3AT.	50
5.2	Representação tridimensional da plataforma CaRINA 1 (Prado, 2013). (a) Modelo 3D. (b) Modelo em estrutura de árvore. (c) CaRINA 1.	50
5.3	Navegação com o Pioneer	51
5.4	Mapa de ambiente em grade de ocupação.	51
5.5	Localização de Monte Carlo no Pioneer	52
5.6	Primitivas de Movimento para o Pioneer	54
5.7	Janela de navegação para o Pioneer.	56
5.8	Navegação com o CaRINA	56
5.9	Ambiente simulado no Gazebo	57
5.10	Estimativa de posição obtida com o filtro de Kalman	58
5.11	Primitivas de movimento CaRINA 1.	59
5.12	Janela de navegação para o CaRINA 1.	61
6.1	Navegação em ambiente interno simulado.(a) Mundo simulado STAGE . (b) Planejamento de trajetória. (c) Detecção de obstáculo (I). (d) Replanejamento (I). (e) Detecção de Obstáculo (II). (f) Replanejamento (II). (g)Passagem estreita. (h) Meta atingida.	65
6.2	Navegação em ambiente interno real. (a) Planejamento de trajetória. (b) Passagem estreita. (c) Detecção de obstáculo. (d) Replanejamento. (e) Desvio de obstáculo. (f) Meta atingida.	67
6.3	Tarefa de estacionamento perpendicular. (a) Representação inicial da posição inicial do veículo no GAZEBO. (b) Planejamento de movimento. (c) Detecção de obstáculo. (d) Replanejamento. (e) Seguimento de trajetória (f) Estacionamento	70

6.4	Tarefa de estacionamento diagonal. a) Representação inicial da posição inicial do veículo no GAZEBO. (b) Planejamento de movimento. (c) Detecção de obstáculo (I). (d) Replanejamento (I). (e) Detecção de Obstáculo (II). (f) Replanejamento (II). (g) Estacionamento na vaga.	71
6.5	Manobra com marcha Ré. (a) Representação inicial da posição inicial do veículo no GAZEBO. (b) Planejamento de movimento. (c) Detecção de obstáculo. (d) Replanejamento. (e) Meta atingida.	73
6.6	Navegação com o veículo CaRINA 1 em ambiente real. (a) Planejamento de movimento. (b) Detecção de obstáculos. (c) Replanejamento. (d) Desvio de obstáculo. (e) Meta atingida.	74

LISTA DE TABELAS

3.1	Especificações do Pioneer 3-AT	30
3.2	Especificações do veículo	38

LISTA DE ABREVIATURAS

.....

ROS - **R**obot **O**perating **S**ystem

RMA - **R**obô **M**óvel **A**utônomo

AD* - **A**nytime **D**ynamic **A***

CaRINA - **C**arro **R**obótico **I**nteligente para **N**avegação **A**utônoma

RRT - **R**apidly-exploring **R**andom **T**rees

RDT - **R**apidly-exploring **D**ense **T**rees

RBPF - **R**ao-**B**lackwellized **P**article **F**ilter

DWA - **D**ynamic **W**indow **A**pproach

ICMC - **I**nstituto de **C**iências **M**atemáticas e **C**omputação

INCT-SEC - **I**nstituto **N**acional de **C**iência e **T**ecnologia em **S**istemas **E**mbarcados **C**ríticos.

RMRD - **R**obô **M**óvel de **R**odas **D**eslizantes

CG - **C**entro de **G**ravidade

ICC - **C**entro **I**nstatâneo de **R**otação

LRM - **L**aboratório de **R**obótica **M**óvel

SBPL - **S**earch-**B**ased **P**lanning **L**ibrary

AMCL - **A**daptive **M**onte **C**arlo **L**ocalization

SLAM - **S**imultaneous **L**ocalization **A**nd **M**apping

ICP - **I**terative **C**losest/**C**orresponding **P**oin

SUMÁRIO

Resumo	xi
Abstract	xiii
Lista de Figuras	xv
Lista de Tabelas	xix
Lista de Abreviaturas	xxi
1 Introdução	5
1.1 Organização do Texto	7
2 Revisão Bibliográfica	9
2.1 Mapeamento	9
2.1.1 Grade de Ocupação	10
2.2 Localização	11
2.3 Planejamento de Movimento	12
2.3.1 Espaço de Configurações	12
2.3.2 Planejamento Cinemático-Dinâmico	14
2.3.3 Planejamento Probabilístico de Movimento	15
2.3.4 Malha de Estados	17
2.3.5 Planejadores <i>Anytime</i> e <i>Incremental</i>	20
2.4 Método da Janela Dinâmica	25

3	Plataformas Robóticas	29
3.1	Pioneer 3AT	29
3.1.1	Modelo Cinemático	31
3.1.2	Modelo Dinâmico	34
3.2	Projeto CaRINA	37
3.2.1	Modelo Ackerman	40
4	Ferramentas e Bibliotecas	43
4.1	ROS	43
4.2	SBPL	45
4.2.1	SBPL LATTICE PLANNER	46
4.3	AMCL	47
4.4	GMAPPING	48
5	Metodologia	49
5.1	Modelagem Computacional	49
5.2	Navegação em ambientes internos	51
5.2.1	Mapeamento	51
5.2.2	Localização	52
5.2.3	Planejamento	53
5.2.4	Controle	54
5.3	Navegação em ambientes externos	56
5.3.1	Mapeamento	57
5.3.2	Localização	58
5.3.3	Planejamento	59
5.3.4	Controle	60
6	Resultados	63
6.1	Planejamento e Navegação em Ambientes Internos	63
6.1.1	Ambiente Simulado	64
6.1.2	Ambiente Real	66
6.2	Planejamento e Navegação em Ambientes Externos	68
6.2.1	Ambiente Simulado	68
6.2.2	Ambiente Real	72

7	Conclusão	75
7.1	Publicações Resultantes deste Trabalho	77
7.1.1	Aceitos	77
7.1.2	Submetidos	77
7.2	Trabalhos Futuros	77
	Referências	79

CAPÍTULO 1

INTRODUÇÃO

Um importante objetivo na área da robótica é a criação e o desenvolvimento de robôs autônomos. Tais robôs devem ser capazes de realizar tarefas sem a necessidade de intervenções humanas adicionais (Latombe, 1991). Nos últimos anos observou-se um aumento no grau de consciência com relação ao potencial de aplicações dos robôs autônomos. Os Robôs Móveis Autônomos (RMA) podem operar de modo semi ou completamente autônomo, sendo que o grau de autonomia de cada robô está associado à sua: capacidade de percepção (sensores que podem "ler" o ambiente onde ele atua); capacidade de agir (atuadores e motores capazes de produzir ações, tais como o deslocamento do robô no ambiente); robustez e inteligência (capacidade de lidar com as mais diversas situações, de modo a resolver e executar tarefas por mais complexas que sejam) (Wolf et al., 2009).

Em tarefas de navegação autônoma, o planejamento de movimento tem o propósito de determinar quais movimentos o robô deve realizar de forma que alcance posições ou configurações desejadas no ambiente sem que ocorram colisões com obstáculos (Latombe, 1991). No processo de planejamento utiliza-se informações sobre o ambiente no qual o robô está inserido, na forma de um mapa, por exemplo, juntamente com informações sobre o próprio robô, ou seja, seu modelo cinemático e dinâmico. Assim, dentro de um ambiente conhecido são determina-

das trajetórias livres de colisões que possam ser executadas pelo robô dadas suas restrições de movimento e características dinâmicas. Os métodos de planejamento de movimento também podem considerar formas de se lidar com incertezas no movimento, ou até mesmo com obstáculos dinâmicos não previstos no mapa (modelo do ambiente), e que são percebidos apenas durante a execução do movimento. Para veículos autônomos, por exemplo, é possível determinar o melhor caminho para se chegar a um local, ou então quais as manobras que o veículo deve fazer para estacionar em uma vaga entre dois outros veículos.

Diversos métodos de planejamento para navegação são apresentados de forma detalhada em Choset et al. (2005); Latombe (1991); LaValle (2006), dentre outros. Algumas das abordagens utilizadas para planejamento de movimento são: *roadmaps*, decomposição em células, campos potenciais, e planejamento baseado em amostragem. A escolha do método de planejamento influencia a maneira como ocorre a navegação e também como comportamentos reativos de navegação podem ser integrados no sistema de controle. Um exemplo de comportamento reativo é o desvio de obstáculos móveis que não são considerados no mapa no momento do planejamento.

É comum na robótica móvel simplificar o planejamento de movimento representando o robô pelas coordenadas (x, y) do seu centro e desconsiderando qualquer restrição de movimento. Entretanto a maioria dos robôs móveis possuem restrições cinemáticas não-holonômicas. Um exemplo desse tipo de restrição é a impossibilidade de um carro se mover na direção perpendicular a de suas rodas, precisando fazer uma série de manobras para alcançar uma determinada posição. Quando o planejamento não considera essas restrições, o caminho planejado pode servir como um guia para que o robô chegue até a posição final, mas nem sempre o robô poderá reproduzir exatamente o caminho planejado. Em alguns casos, dependendo do ambiente, da tarefa de navegação, e do robô, é preciso considerar além da posição do centro do robô, sua orientação, e as restrições não-holonômicas para se obter uma solução para o problema. Esse é o caso quando se deseja manobrar um veículo para estacioná-lo entre outros dois (estacionamento paralelo).

O presente trabalho propõe desenvolver um método de planejamento que permita planejar trajetórias para um robô móvel com restrições não-holonômicas, e que possibilite o desvio de obstáculos no momento da execução da trajetória planejada. As restrições não-holonômicas

desse tipo de robô, e seu modelo dinâmico são considerados para que sejam geradas trajetórias possíveis de serem realizadas. O método de planejamento é baseado em malha de estados (*Lattice State*). Esse método cria uma malha através da combinação de um conjunto de ações predefinidas que respeitam as restrições de movimento do robô com isso, o problema de planejamento de movimento se transforma em uma busca em um grafo. As soluções são encontradas usando o algoritmo de busca AD* (*Anytime Dynamic A**) que tem uma característica em que uma trajetória planejada inicialmente pode ser modificada ou melhorada em tempo de execução para que o robô evite a colisão com obstáculos não considerados no mapa na etapa de planejamento, e que são percebidos pelos sensores embarcados no robô. Associado ao planejamento de movimentos, é comum usar técnicas de desvio de obstáculos para garantir uma navegação segura pelo ambiente, assim, um controle para seguir trajetória é baseado no Método da Janela Dinâmica (*Dynamic Window Approach*) que cria um conjunto de velocidades que são possíveis de serem aplicadas no robô durante a navegação para garantir a segurança na realização da tarefa.

1.1 Organização do Texto

Este texto está organizado da seguinte maneira:

- **Capítulo 2:** Apresenta um estudo introdutório e sucinto sobre os principais problemas da robótica móvel: mapeamento, localização e planejamento de movimento. Esses conhecimentos são necessários para tarefa de navegação, cuja metodologia será mostrada no Capítulo 5. O planejamento de movimento, discute as abordagens de planejamento probabilístico e planejamento cinemático-dinâmico, bem como a utilização das Malhas de Estado (*Lattice State*) e dos algoritmos de busca em grafo. A ênfase é dada ao algoritmo AD*, usado aqui para planejar as trajetórias.
- **Capítulo 3:** Apresenta as plataformas robóticas utilizadas neste trabalho. A primeira plataforma robótica é um Pioneer 3AT, é um robô de pequeno porte da *MobileRobot* usado para o planejamento e navegação em ambientes internos (*indoor*). A segunda plataforma é um veículo autônomo CaRINA 1 (Carro Robótica Inteligente para Navegação Autônoma) desenvolvido pelo ICMC-USP com apoio do Instituto Nacional de Ciência e

Tecnologia em Sistemas Embarcados Críticos (INCT-SEC). Esta plataforma será usada para o planejamento e navegação em ambientes semi-estruturados urbanos (*outdoor*).

- **Capítulo 4:** Apresenta as principais ferramentas e bibliotecas utilizadas no desenvolvimento deste trabalho.
- **Capítulo 5:** Apresentada a metodologia usada neste trabalho. Essa metodologia está basicamente dividida em duas seções: Navegação em ambientes internos e Navegação em ambientes externos. Na Seção 5.2 discutiremos os métodos usados para o robô Pioneer 3AT e na Seção 5.3 os métodos usados no veículo autônomo CaRINA 1.
- **Capítulo 6:** Mostra os resultados obtidos na tarefa de navegação em ambientes reais e simulados das plataformas robóticas mostradas no Capítulo 3.
- **Capítulo 7:** Apresenta as conclusões obtidas no desenvolvimento deste trabalho, bem como as possíveis propostas para o melhoramento do mesmo.

CAPÍTULO 2

REVISÃO BIBLIOGRÁFICA

Este capítulo descreve de forma sucinta os fundamentos teóricos necessários para o desenvolvimento deste trabalho.

2.1 Mapeamento

A geração de mapas do ambiente é fundamental para a realização de diversas tarefas executadas por robôs móveis como: localização e planejamento de movimento. Dessa forma, é considerada uma das competências essenciais em robôs verdadeiramente autônomos (Thrun et al., 2005).

Para o processo de mapeamento, é essencial que o robô esteja equipado com sensores para o levantamento de dados sobre o ambiente. A imprecisão na informação obtida por sensores e a complexidade do ambiente são exemplos das dificuldades encontradas para a realização da tarefa de mapeamento. Em algumas circunstâncias o robô não tem conhecimento de sua localização, aumentando ainda mais a complexidade do problema. Neste caso, o robô deve gerar um mapa e ao mesmo tempo usá-lo para se localizar no ambiente (Durrant-Whyte e Bailey, 2006). Esse problema é conhecido como SLAM (*Simultaneous Localization and Mapping*).

Basicamente, dois tipos de mapas podem ser criados por robôs móveis: topológicos e métricos. Os mapas topológicos representam o ambiente por grafos, no qual os vértices representam regiões de interesse no ambiente e os arcos representam as vias que interligam essas regiões (Kuipers e Byun, 1991). Os mapas topológicos são bastante eficientes para se estimar trajetórias em um ambiente, mas apresentam uma representação muito pobre do ambiente físico (Wolf et al., 2009). Os mapas métricos representam os ambientes físicos em detalhes e também podem ser utilizados para se estimar trajetórias. Entre os mapas métricos, destaca-se a representação do ambiente utilizando-se grades de ocupação (*occupancy grids*) que foi introduzido em Moravec e Elfes (1985) e até hoje é uma das representações métricas mais utilizadas.

2.1.1 Grade de Ocupação

O método de grade de ocupação consiste em criar uma representação bidimensional, na qual o ambiente é dividido em pequenas unidades de espaço ou células, e para cada uma dessas células é calculada a possibilidade daquele espaço estar ocupado ou não. Este cálculo é realizado com base nas informações obtidas dos sensores. Normalmente, são utilizados sensores que detectam a distância entre o robô e os obstáculos (sonares e lasers) para o mapeamento de ambientes (Wolf et al., 2009).

Figura 2.1: Grade de Ocupação (Wolf et al., 2009).

A Figura 2.1 é um exemplo de um mapa construído utilizando esta abordagem. As áreas brancas representam os espaços livres, as áreas pretas representam os espaços com a presença de obstáculos, enquanto as áreas cinzas representam as áreas desconhecidas.

2.2 Localização

Conhecer a posição e orientação de um robô é essencial para muitas tarefas na área da robótica, dentre elas podemos citar tarefas de mapeamento e planejamento de movimento (Pires e Chaimowicz, 2012). Na robótica móvel, a questão relacionada com a determinação da posição de um robô é referenciada como problema de Localização (Pires e Chaimowicz, 2012).

Uma vez conhecida a posição do robô, novas posições podem ser obtidas por intermédio do acompanhamento do deslocamento efetuado pelo robô. Técnicas de *Dead Reckoning*, tais como odometria, têm sido amplamente utilizadas para o rastreamento da posição. O princípio dessas técnicas consiste no emprego de sensores que medem os parâmetros internos ao sistema, para estimar uma nova posição por meio da mensuração do deslocamento efetuado pelo robô. A incerteza inerente à utilização de sensores reais acarreta erros que acumulam com o tempo, principalmente quando o robô percorre longas distâncias.

Técnicas baseadas em métodos estatísticos têm sido amplamente utilizadas com o propósito de reduzir o erro acumulado entre a estimativa de um estado e o estado real do robô. Entre essas técnicas podemos citar a Localização de Monte Carlo (Fox et al., 1999) e filtros probabilísticos como o Método de Markov (Fox et al., 1999) e o Filtro de Kalman (Siegwart et al., 2004).

O Filtro de Kalman (Siegwart et al., 2004), por exemplo, diminui o erro na estimativa da posição por meio da observação de marcos (*landmarks*) que possuem localização conhecida em um mapa, o que viabiliza ajustar a crença que o robô possui com relação à sua pose atual.

A localização de Monte Carlo (Fox et al., 1999) consiste em manter várias hipóteses de localização no ambiente, onde cada hipótese é representada por uma partícula. Ao ser iniciado, o algoritmo distribui partículas pelo mapa do ambiente. Comparando os dados obtidos dos sensores com as informações disponíveis no mapa, é associado um peso para cada partícula. Este peso representa a chance dessa partícula representar a real localização do robô. As partículas com os menores pesos são descartadas enquanto partículas com os maiores pesos são replicadas.

Várias pesquisas têm explorado as potencialidades da utilização de um grupo de robôs para solucionar os mais diferentes tipos de tarefas (Cao et al., 1997). Dentre as vantagens de se utilizar sistemas de múltiplos robôs podemos citar a melhoria no desempenho e a possibili-

dade de divisão da tarefa entre os membros do grupo. O problema de localização ganha uma nova dimensão em sistemas multi-robóticos, passando a considerar as características coletiva e colaborativa do grupo. As informações de localização de cada membro do grupo tornam-se relevantes para o ajuste da qualidade da localização conjunta. Tem-se, então, uma estimativa de localização global que visa minimizar os erros locais associados a cada componente do sistema. Esse tipo de localização é chamado de Localização Cooperativa. Em Odakura (2006) tem-se um exemplo de localização de Markov aplicada a sistemas robóticos.

2.3 Planejamento de Movimento

O objetivo central do planejamento de movimento é levar o robô de um estado atual a um estado final desejado, evitando colisões com os obstáculos presentes no ambiente durante o percurso. O planejamento deve sempre encontrar um caminho se ele existir, ou retornar que não existe uma solução. A definição apresentada anteriormente é muito ampla para o que de fato é uma junção de vários conceitos e técnicas. Na prática, deve-se conhecer o espaço de configurações do robô, escolher a técnica de planejamento a ser usada e, com a sua localização, realizar o controle do seguimento da trajetória planejada para a posição atual (mais detalhes podem ser encontrados em Choset et al. (2005)). Nesta seção, para o estudo sobre o planejamento de movimento, foi considerado que os ambientes de trabalho do robô são conhecidos e estáticos.

Esta é uma tarefa desafiadora e considerada um dos problemas fundamentais da robótica autônoma (Choset et al., 2005).

2.3.1 Espaço de Configurações

O espaço de configurações é um dos conceitos mais importantes no planejamento de movimento. Para criar o espaço de configurações de um robô é necessário conhecer seu espaço de trabalho, ou seja, o ambiente no qual o robô está inserido. O espaço de trabalho $\mathcal{W}$, é o espaço físico onde o robô se move, e pode ser definido tanto no espaço bidimensional $\mathbb{R}^2$ quanto no espaço tridimensional $\mathbb{R}^3$, isso depende do tipo de robô. No caso de robôs móveis que se deslocam apenas no plano, o espaço de trabalho é bidimensional, ou seja, definido no $\mathbb{R}^2$. Já para robôs

manipuladores e robôs móveis aéreos ou submarinos, o espaço de trabalho é tridimensional, ou seja, definido no $\mathbb{R}^3$.

Uma configuração, q , de um robô, $\mathcal{R}$, é definida como sendo a representação completa da posição de todos os seus pontos em relação a um sistema de coordenadas fixo no ambiente $\mathcal{F}_w$. O espaço de configurações $\mathcal{C}$, é então definido como sendo o conjunto de todas as configurações possíveis para este robô. Neste espaço o robô é definido como sendo um ponto, q . A dimensão do espaço de configurações é igual ao número de variáveis independentes utilizadas para representar uma configuração do robô (graus de liberdade do robô).

Dentro do contexto de espaço de configurações, pode-se definir dois conjuntos que auxiliam na formulação de problemas de planejamento de movimento: o espaço de configurações ocupado por obstáculos, denotado por $\mathcal{C}_{obs}$, e o espaço de configurações livre, denotado por $\mathcal{C}_{free}$.

A Figura 2.2, apresentada em Choset et al. (2005), mostra a diferença entre espaço de trabalho e espaço de configurações para um dos casos mais simples de se obter $\mathcal{C}_{obs}$, caso em que se considera um robô móvel de geometria circular se movendo no plano. O estado do robô é representado pela posição do centro geométrico. Deslizando o robô pelo contorno do obstáculo, a posição do centro geométrico define uma linha, Figura 2.2(b), que representa a fronteira de colisão entre robô e obstáculo, essa linha define a fronteira de $\mathcal{C}_{obs}$. Após essa operação o robô pode ser representado por um ponto. Na Figura 2.2(c) mostra a representação dessa linha de fronteira cuja região interior a essa linha é definido como sendo o espaço de configurações

Figura 2.2: Diferença entre espaço de trabalho e espaço de configurações.

ocupado por obstáculos, $\mathcal{C}_{obs}$ e a região para fora dessa linha é definida como sendo o espaço de configurações livre de obstáculos, $\mathcal{C}_{free}$. Realizando essa mesma operação para todos os obstáculos do ambiente, tem-se a representação completa de $\mathcal{C}_{obs}$ e conseqüentemente, $\mathcal{C}_{free}$.

2.3.2 Planejamento Cinemático-Dinâmico

De uma forma geral, os métodos de planejamento de movimento não consideram restrições de movimento cinemáticas e dinâmicas do robô. Entretanto muitos dos métodos permitem que na etapa de planejamento essas restrições possam ser consideradas. Em Esposito e Kumar (2002), por exemplo, modifica-se o método de planejamento baseado em funções potenciais de navegação para considerar as restrições não-holonômicas de um robô móvel do tipo carro. Um campo potencial dipolar é utilizado pois o gradiente negativo deste campo descreve curvas que respeitam a restrição não-holonômica desse tipo de robô. Uma vez que o robô se encontra alinhado ao gradiente do campo dipolar, o gradiente conduzirá o robô por uma trajetória possível de ser seguida. Em Tanner et al. (2001), um campo potencial dipolar também é usado para definir de forma adequada uma função de navegação para robôs não-holonômicos. Restrições dinâmicas no movimento também podem ser tratadas de forma especial. Em Conner et al. (2003) utiliza-se o método de decomposição do ambiente em células, e então define-se uma função de navegação dentro de cada uma das células de forma a considerar as restrições dinâmicas do robô. A velocidade do robô é controlada pela função de navegação dentro de cada célula de forma a se evitar descontinuidades no movimento quando o robô passa de uma célula para outra no ambiente.

Métodos de *roadmaps* em Choset e Burdick (2000) e métodos baseados em amostragem em (Kavraki et al., 1996; LaValle, 2006) retornam o caminho principal a ser seguido da posição inicial do robô até sua posição final e podem ser adaptados para considerar as restrições não-holonômicas do robô. Além disso, na execução do movimento, modificações locais desse caminho podem ser computadas para que se possa lidar com obstáculos imprevistos. Um exemplo disso é a abordagem adotada no sistema de navegação do veículo autônomo da Universidade de Stanford (Thrun et al., 2006), vencedor do *Darpa Grand Challenge* de 2005, cujo objetivo era atravessar um trecho do deserto americano. A partir da curva principal do caminho a ser seguido, eram computadas curvas alternativas para o veículo. Então, dependendo dos obstácu-

los encontrados, a melhor alternativa era escolhida a cada instante de forma a manter o veículo próximo ao percurso principal. A Figura 2.3 nos mostra esse fato.

Figura 2.3: Planejamento local de caminhos alternativos (Thrun et al., 2006).

2.3.3 Planejamento Probabilístico de Movimento

As restrições cinemáticas e as características dinâmicas do robô podem ser incluídas de forma natural no planejamento baseado em amostragem. Esse tipo de planejamento não utiliza uma construção explícita do espaço de configurações dos obstáculos, que para espaços de configuração de dimensão maior que 3 é computacionalmente muito custosa. Ao invés disso estes métodos realizam uma busca por caminhos livres no espaço de configurações utilizando uma amostragem desse espaço. Entretanto, por trabalharem com uma amostragem do espaço de configurações, estes métodos não podem ser considerados métodos completos de planejamento. Um método completo é capaz de fornecer uma solução para o problema de planejamento se ela existir, ou de dizer que a solução não existe caso contrário. Para os métodos de planejamento baseados em amostragem, utiliza-se um conceito mais fraco, e considera-se que esses métodos são probabilisticamente completos. Ou seja, conforme o tempo gasto na busca pela solução tende a infinito, a probabilidade do método retornar uma solução, se ela existir, tende a 1.

Os métodos baseados em amostragem podem ser classificados em métodos de busca única (*single query*), ou métodos de busca múltipla (*multiple queries*). Os métodos de busca única retornam uma única solução para uma configuração inicial e final fornecidas. O método também reporta falha caso não consiga encontrar uma solução dentro de um limite de tempo pré-estabelecido. Os métodos de busca múltipla investem um esforço significativo no pré-

processamento do espaço de configurações para construir uma estrutura de dados (uma *roadmap*, por exemplo) que é usada para tornar eficiente a busca por soluções. Nestes métodos, uma vez que esta estrutura é criada, considerando que não ocorrem mudanças no espaço de configurações, podem-se encontrar soluções para múltiplos pares de configurações iniciais e finais.

Como exemplo de métodos de busca única pode-se citar as *Árvore Aleatórias de Rápida Exploração* (*Rapidly-exploring Random Trees*, ou *RRTs*) e as *Árvore Densas de Rápida Exploração* (*Rapidly-exploring Dense Trees*, ou *RDTs*) (LaValle, 2006). Como exemplo de método de busca múltipla, pode-se mencionar *Roadmaps Probabilísticas* (*Probabilistic Roadmaps*, ou *PRMs*) (Kavraki et al., 1996).

No método de amostragem de busca única RRT, por exemplo, a partir da configuração inicial do robô constrói-se uma árvore de possíveis movimentos que explora o espaço de configurações até que um de seus ramos chegue na configuração final desejada. Esta árvore pode ser construída utilizando uma função de controle que descreve o movimento cinemático e dinâmico do robô. Assim, para uma dada configuração, a amostragem das próximas configurações (ramificações da árvore) é feita somente no espaço de configurações que o robô consegue alcançar, dada suas restrições cinemáticas e dinâmicas, a partir da configuração anterior (Vaz et al., 2010).

No contexto do desafio *DARPA Urban Challenge*, por exemplo, onde veículos autônomos precisavam se mover em um contexto urbano, era necessário planejar trajetórias para alcançar determinadas posições na cidade, e ao longo do caminho lidar com situações imprevistas, dentre elas, desviar de outros veículos e de obstáculos na pista observados pelos sensores durante a execução da trajetória. O método usado pela equipe campeã da competição faz o planejamento da trajetória do veículo usando uma malha com múltipla resolução do espaço de estados (Likhachev e Ferguson, 2009). Essa malha é uma discretização do espaço de configuração em um conjunto de estados e conexões entre esses estados, onde cada conexão representa um caminho possível de ser executado considerando a cinemática e dinâmica do robô. Como essa malha leva em consideração o modelo do robô, ela difere das grades convencionais onde os elementos do espaço de configuração são representados por quadrados conectados aos seus 4 ou 8 vizinhos. Usando malhas de estados o problema de planejamento pode então ser tratado como uma busca em um grafo, e devido a múltipla resolução da malha, trajetórias longas podem ser

planejadas usando uma malha com baixa resolução, e desvios locais de obstáculos podem ser tratados usando uma malha de resolução mais alta em um escopo menor.

2.3.4 Malha de Estados

Representação discreta de estados é um método bem sucedido para reduzir a complexidade computacional do planejamento. No entanto, no planejamento do movimento, tais representações discretas podem complicar a satisfação de restrições diferenciais que refletem a capacidade de manobra limitada de muitos veículos reais. Para este fim, (Pivtoraiko e Kelly, 2005) introduz um espaço de busca conhecido como malha de estados (*Lattice State*), que é a construção conceitual usada para formular uma busca de planejamento não-holonômico de movimento, como busca em grafo.

A malha de estados é obtida através de uma discretização do espaço de configurações em um conjunto de estados e conexão entre esses estados, onde cada conexão representa um caminho possível de ser executado levando em conta a cinemática e a dinâmica do robô (Knepper e Kelly, 2006; Likhachev e Ferguson, 2009; Pivtoraiko e Kelly, 2005). Como em um *grid*, a malha de estados converte o problema de planejamento em uma função contínua para gerar uma sequência de decisões a partir de alternativas distintas. Ao contrário de um *grid*, a malha estados não codifica nenhuma suposição padrão sobre como os estados estão conectados. Por outro lado, é comum supor que as células adjacentes em um *grid* estão conectados em um arranjo de 4 ou 8 conexões. Essa conectividade padrão não consegue captar as restrições não-holonômicas, deixando-as para serem consideradas heurísticamente no processo de otimização durante o planejamento. A malha de estados, no entanto, tem uma propriedade importante que cada conexão representa um caminho possível, gerando um esquema de conectividade que representa as restrições de mobilidade do robô, como resultado disso o planejador de movimento não perde tempo, para gerar, avaliar ou fixar planos inviáveis (Pivtoraiko e Kelly, 2005; Pivtoraiko et al., 2009).

A conectividade da malha é definida como sendo um subconjunto finito de ações (estados que incluem apenas os caminhos desde a origem para algum raio). Este subconjunto de ações é formado de pequenos caminhos primitivos que quando conectados entre si, podem gerar qualquer outro caminho na rede. Esses pequenos caminhos são chamados de movimentos primitivos

do robô e são construídos *offline* usando o modelo do veículo de modo que possam ser devidamente executados. A Figura 2.4 mostra um exemplo desses movimentos primitivos com 5 ações para frente, cada ação é um pequeno caminho possível de ser executado pelo robô.

Figura 2.4: Movimentos Primitivos. Cada estado S é um estado possível de ser alcançado pelo robô.

Para elucidar a ideia da malha de estados, na Figura 2.5 tem-se um exemplo da construção de uma rede 3D (x, y, θ) com cinco ações para frente e nenhuma para trás. A rede é construída a partir dos movimentos primitivos mostrados na Figura 2.4, sendo que cada estado S da rede é representado por uma dessas ações, esse conjunto de ações é então combinado para formar novos estados, assim é possível gerar caminhos maiores que também são possíveis de ser executados pelo robô. As ações de interseção com obstáculos são removidas (Likhachev e Ferguson, 2009).

Malha de Múltipla Resolução

Mesmo para um espaço de ação compacto, o planejamento de longos e complexos caminhos sobre uma malha de estados, pode ser custoso em termos de computação e memória. Uma observação importante, porém, é que geralmente existe uma ampla quantidade de caminhos suaves dinamicamente possíveis entre o veículo e seu destino, e é um desperdício de tempo e memória explorar todos eles. Para resolver esse problema, Likhachev e Ferguson (2009) utiliza uma abordagem de malha de estados de multirresolução, onde utiliza uma alta resolução no

Figura 2.5: Exemplo de uma malha

espaço de ação nas imediações do robô e da meta, e uma baixa resolução no espaço de ação em outros lugares, essa combinação resultante é chamada de malha de múltipla resolução. Com esta abordagem, pode-se aproveitar o máximo de benefício da representação de alta resolução com um esforço computacional não muito grande.

As malhas de alta e baixa resolução compartilham a mesma dimensionalidade e se encaixam perfeitamente, isto é, a malha de baixa resolução é um subconjunto da malha de alta resolução. Sejam dois conjuntos denotados por $\mathcal{A}_h$ e $\mathcal{A}_l$ como sendo respectivamente o conjunto de alta resolução e o conjunto de baixa resolução, assim uma característica importante para a malha de multi-resolução gerar caminhos resultantes da sobreposição das duas malhas é fazer $\mathcal{A}_l \subset \mathcal{A}_h$. A Figura 2.6 ilustra como o conjunto de ação de alta e baixa resolução são diferentes.

Figura 2.6: Malha de Múltipla Resolução

2.3.5 Planejadores *Anytime* e *Incremental*

Planejamento no mundo real envolve lidar com uma série de desafios não enfrentados em muitos domínios mais simples. Em primeiro lugar, o mundo real é dinâmico e inerente à incertezas, o que dificulta a obtenção de modelos precisos para o planejamento. Em segundo lugar, quando estamos operando no mundo real, muitas vezes, tem-se pouco tempo disponível para planejamento de suas próximas ações. Assim, a elaboração de planos ideais é inviável nesses cenários. Em vez disso, os agentes devem buscar os melhores planos que podem gerar dentro do tempo disponível (Likhachev, 2005; Likhachev et al., 2008, 2005).

Dado um espaço de busca (por exemplo, uma malha de estados) e uma função de custo associado a cada ação, precisamos de um método eficiente para buscar uma solução nesse espaço (Likhachev e Ferguson, 2009). Uma série de algoritmos de busca em grafos têm sido desenvolvidos para a geração de caminhos. O algoritmo A* e o algoritmo de Dijkstra são duas abordagens comumente utilizadas e amplamente estudadas que geram caminhos ideais. Esses algoritmos são muito eficientes. Na verdade, eles processam o número mínimo de estados possíveis, garantindo uma solução ótima quando nenhuma outra informação é fornecida (Likhachev et al., 2008).

Planejamento no mundo real, no entanto, costuma ser bem complexo para ser resolvido dentro de um tempo aceitável. Além disso, mesmo quando um plano ideal é encontrado, o modelo usado para representar o problema não é perfeito e mudanças podem ocorrer no ambiente, isso pode gerar certas discrepâncias em seu modelo durante a execução de seu plano. Em tais situações, o agente precisa atualizar o seu modelo e replanejar. Encontrar um plano ideal toda

vez que precisar replanejar faria o agente em execução parar muitas vezes e por muito tempo. Planejamento *Anytime* apresenta uma alternativa atraente para esse problema.

Algoritmos de planejamento *Anytime*, tentam encontrar o melhor plano que pode dentro do tempo disponível. Eles rapidamente retornam um caminho, que possivelmente, é um caminho sub-ótimo, e depois tentam melhorar este plano enquanto ainda tem tempo disponível. Além de ser capaz de cumprir os prazos de tempo, muitos destes algoritmos também tornam possível a combinação de planejamento e execução: enquanto o agente executa seu plano atual, o planejador trabalha para melhorar o plano. Um exemplo desse tipo de algoritmo pode ser visto em (Likhachev, 2005; Likhachev et al., 2005) e (Likhachev et al., 2008).

Uma outra classe de algoritmos conhecida como algoritmos de replanejamento também são eficazes. Eles usam os resultados de esforços de planejamento prévio para ajudar a encontrar um novo plano quando algum problema surge. Algoritmos tais como, *Dynamic A* (D*)* e *LifeLong Planning A* (LPA*)*, têm sido particularmente úteis para o replanejamento baseado em pesquisa heurística em inteligência artificial e robótica (Likhachev et al., 2008). Esses algoritmos realizam uma pesquisa A* para gerar uma solução inicial. Então, quando o modelo do ambiente é atualizado, reparam a sua solução reutilizando o máximo de informações das pesquisas anteriores. Com essa característica, eles podem ser mais eficientes do que o replanejamento a partir do zero cada vez que o modelo do ambiente muda. No entanto, estes algoritmos não têm a característica dos algoritmos *Anytime*, uma vez que a solução é encontrada, mesmo se mais tempo de planejamento é disponível, os algoritmos *Incrementais* não podem melhorar a solução. Esses algoritmos podem ser pré-configurados para procurar uma solução ótima ou para procurar uma solução delimitada por um fator de sub-otimização fixo.

O algoritmo A* e sua variação *Anytime* funcionam melhor quando o espaço de busca e, portanto, o meio ambiente, é conhecido a priori. No planejamento de movimento raramente é esse o caso, e o robô geralmente recebe informações atualizadas do ambiente através de seus sensores embarcados. Para lidar com as primeiras informações imperfeitas e ambientes dinâmicos, que a variação *Incremental* de A* foi desenvolvida.

Quando confrontados com o tempo limitado e ambientes dinâmicos pouco conhecidos, é extremamente útil ter um algoritmo de busca que seja *Incremental* e *Anytime*. O algoritmo *Anytime Dynamic A** desenvolvido por (Likhachev et al., 2005) é uma versão do algoritmo A*

que combina estas duas características em uma única abordagem e tem se mostrado muito eficaz para uma série de tarefas de planejamento de movimento.

*Anytime Dynamic A**

*Anytime Dynamic A** (AD*) explora a característica de A* que pode resultar na geração de soluções rápidas, ou seja, se uma heurística consistente é usada e multiplicada por um fator de inflação $\epsilon > 1$, então pode-se gerar uma solução muito mais rápida que se nenhum fator de inflação é usado (Likhachev e Ferguson, 2009), e o custo da solução gerada será na maioria das vezes o custo de uma solução ótima. AD* opera realizando uma série de pesquisas com fator de inflação decrescente, onde cada pesquisa reutiliza informações de pesquisas anteriores. Para lidar com informações atualizadas, AD* também toma emprestado as ideias do algoritmo *Incremental* e propaga essas informações atualizadas através das porções afetadas do espaço de busca. O algoritmo AD* é mostrado na Figura 2.7.

key(s)

```
01. if ( $g(s) > rhs(s)$ )
02.   return [ $rhs(s) + \epsilon h(s_{start}, s); rhs(s)$ ];
03. else
04.   return [ $g(s) + h(s_{start}, s); g(s)$ ];
```

UpdateState(s)

```
05. if  $s$  was not visited before
06.    $g(s) = \infty$ ;
07. if ( $s \neq s_{goal}$ )  $rhs(s) = \min_{s' \in Succ(s)} (c(s, s') + g(s'))$ ;
08. if ( $s \in OPEN$ ) remove  $s$  from  $OPEN$ ;
09. if ( $g(s) \neq rhs(s)$ )
10.   if  $s \notin CLOSED$ 
11.     insert  $s$  into  $OPEN$  with key( $s$ );
12.   else
13.     insert  $s$  into  $INCONS$ ;
```

ComputeorImprovePath()

```
14. while  $\min_{s \in OPEN} (key(s)) < key(s_{start})$  OR  $rhs(s_{start}) \neq g(s_{start})$ ;
15.   remove state  $s$  with the minimum key from  $OPEN$ ;
16.   if ( $g(s) > rhs(s)$ )
17.      $g(s) = rhs(s)$ ;
18.      $CLOSED = CLOSED \cup \{s\}$ ;
19.     for all  $s' \in Pred(s)$  UpdateState( $s'$ );
20.   else
21.      $g(s) = \infty$ ;
22.     for  $s' \in Pred(s) \cup \{s\}$  UpdateState( $s'$ );
```

Main()

```
01.  $g(s_{start}) = rhs(s_{start}) = \infty; g(s_{goal}) = \infty$ ;
02.  $rhs(s_{goal}) = 0; \epsilon = \epsilon_0$ ;
03.  $OPEN = CLOSED = INCONS = \emptyset$ ;
04. insert  $s_{goal}$  into  $OPEN$  with key( $s_{goal}$ );
05. ComputeorImprovePath();
06. publish current  $\epsilon$ -suboptimal solution;
07. forever
08.   if change in edge costs are detected
09.     for all edges  $(u, v)$  with changed edge costs
10.       Update the edge cost  $c(u, v)$ ;
11.       UpdateState( $u$ );
12.   if significant edge cost change were observed
13.     increase  $\epsilon$  or replan from scratch;
14.   else if  $\epsilon > 1$ 
15.     decrease  $\epsilon$ ;
16.   Move state from  $INCONS$  into  $OPEN$ ;
17.   Update the priorities for all  $s \in OPEN$  according to key( $s$ );
18.    $CLOSED = \emptyset$ ;
19.   ComputeorImprovePath();
20.   publish current  $\epsilon$ -suboptimal solution;
21.   if  $\epsilon = 1$ 
22.     wait for change in edge costs;
```

(a) Função para calcular ou melhorar o plano

(b) Função Principal

Figura 2.7: Algoritmo Anytime Dynamic A* (Likhachev et al., 2005).

A função principal primeiro define o fator de inflação ϵ para um valor suficientemente elevado ϵ_0 , para que um plano inicial de qualidade inferior possa ser gerado rapidamente (linhas 02 a 06, Figura 2.7(b)). Então, enquanto não houver mudanças nos custos das extremidades, a função principal vai diminuir ϵ e melhorar a qualidade da solução encontrada até que seja garantida a solução ótima, ou seja, $\epsilon = 1$ (linhas 14 a 20, Figura 2.7(b)). Toda vez que ϵ é decrementado, todos os estados inconsistentes são movidos da lista INCONS para a lista OPEN e a lista CLOSED fica vazia. Quando as mudanças nos custos de extremidade são detectados, há uma chance de que a solução atual não seja mais sub-ótima. Se as mudanças são substanciais, então elas podem ser computacionalmente caras para reparar a solução atual e recuperar ϵ , isso gera um caminho não ideal. Nesse caso, o algoritmo volta a incrementar o valor de ϵ de modo que uma solução menos ideal seja produzida rapidamente (linhas 12 a 13, Figura 2.7(b)). Aumentar o custo das extremidades pode levar alguns estados a se tornar estados pouco consistentes, esses estados precisam ser inseridos na lista OPEN com um valor de chave retornando o mínimo entre seu custo antigo e seu custo atual. Além disso, para garantir que os estados pouco consistentes propaguem seus novos custos para os seus vizinhos afetados, seus valores chave devem usar os valores da heurística não inflacionada. Isto significa que diferentes valores chave devem ser computados de estados pouco consistentes para estados muito consistentes (linhas 01 a 04, Figura 2.7(a)) (Likhachev et al., 2005).

A Figura 2.8 mostra navegação de um robô que se inicia na célula inferior direita (a posição do robô é indicada pela letra R). Usando o algoritmo AD*, uma trajetória é rapidamente gerada para a célula superior esquerda (indicada pela letra G). A medida que o robô se desloca, o algoritmo de planejamento tenta melhorar a solução encontrada decrementando o fator de inflação, denotado por ϵ . Quando novas observações são inseridas e o mapa do ambiente é atualizado, o algoritmo AD* é capaz de melhorar a sua solução atual incorporando estas novas informações.

Informações Heurísticas

O propósito de uma heurística é melhorar a eficiência da pesquisa, orientando-a em direções promissoras. Uma abordagem comum para a construção de uma heurística é usar os resultados de um problema de pesquisa simplificada (por exemplo, a partir de um problema de pesquisa de menor dimensão). Para uma seleção heurística apropriada, é importante analisar o problema

Figura 2.8: Exemplo de Navegação com AD* (Likhachev et al., 2005).

de pesquisa original e determinar os fatores fundamentais que contribuem para sua complexidade. No planejamento de movimento esta complexidade está normalmente relacionada com as restrições do robô e da natureza do ambiente.

Para lidar com a complexidade herdada pelas restrições do robô, uma heurística muito útil em geral é o custo de uma solução ótima através do espaço de busca assumindo um ambiente completamente vazio. Esta função heurística é muito bem informada quando se operando em ambientes escassos e pode ser uma boa aproximação do custo de um caminho real. Em ambientes com obstáculos, no entanto, esta função heurística pode subestimar os custos e induzir uma busca para regiões erradas do espaço de estado. Para lidar com a complexidade herdada da natureza do ambiente, não é prático calcular valores heurísticos para todas as configurações de ambiente, pois há um número infinito de possibilidades. No entanto, neste caso, pode-se resolver esse problema *online* através de uma busca simplificada do ambiente atual (busca 2D) e usar o resultado dessa pesquisa como uma heurística para guiar a busca original.

Cada uma destas funções heurísticas, tanto em relação ao mecanismo quanto em relação ao ambiente, tem benefícios fortes e complementares (Figura 2.9). Em vez de selecionar uma, é possível combinar as duas. Isso pode ser através da construção de uma nova heurística que, para cada estado s , retorna o valor $h(s) = \max(h_{fsh}(s), h_{2D}(s))$, onde $h_{fsh}(s)$ é o valor heurístico do estado s de acordo com a heurística de restrições mecânicas (heurística *freespace*) e $h_{2D}(s)$ é o valor de acordo com a heurística de restrição do ambiente (heurística 2D) (Likhachev e Ferguson, 2009).

Na Figura 2.9 tem-se um exemplo das informação fornecidas pelas heurísticas 2D e *freespace* mostrando a grande vantagem em usar uma heurística combinada.

Figura 2.9: Heurística de restrições mecânicas (linha contínua) e de restrições no ambiente (linha tracejada). As posições inicial e desejada do veículo são mostradas como retângulos azuis e vermelhos, respectivamente. (a) A heurística de restrições mecânicas perfeitamente informa quando não há obstáculos presentes no ambiente. (b) A heurística com restrição do ambiente pode proporcionar benefícios significativos quando existem obstáculos (Likhachev e Ferguson, 2009).

Para mais detalhes sobre os benefícios desta função heurística combinada e outras otimizações implementadas, incluindo o espaço de busca multi-resolução e como ele executa de forma eficiente convoluções e replanejamento, ver em (Ferguson et al., 2008; Likhachev e Ferguson, 2009).

2.4 Método da Janela Dinâmica

Realizar o controle de um veículo apenas com as informações vindas do planejamento de movimento não garante sua segurança. Isto ocorre principalmente em decorrência das mudanças no ambiente do robô e simplificações em seu espaço de configurações. Para contornar estes problemas, é comum associar técnicas de desvio de obstáculos às de planejamento de movimento para que o robô navegue em segurança e completude (Lima, 2010). Nesta seção serão discutidos os princípios de uma destas técnicas, o Método da Janela Dinâmica (DWA), para compor o planejamento e fazer o controle de suas ações na realização de uma tarefa.

O DWA foi inicialmente desenvolvido por (Fox et al., 1997) para realizar o controle de um robô não-holonômico do tipo Synchro-Drive em um ambiente com obstáculos, sem garantia

de completude de seu caminho. Seu princípio está na criação de um conjunto de velocidades aplicáveis como entradas de controle. O Conjunto é obtido por técnicas reativas locais de desvio de obstáculos, que levam em consideração as restrições cinemáticas e dinâmicas deste robô. No caso do Synchro-Drive, a entrada de controle é o par (v, ω) que representam suas velocidades linear e angular, respectivamente. As restrições cinemáticas são relacionadas à estrutura do robô e limitam o mapa de velocidades em:

$$V_s = \{(v, \omega) \mid v \in [v_{min}, v_{max}], \omega \in [\omega_{min}, \omega_{max}]\} \quad (2.1)$$

Para exemplificar o conjunto V_s , considere o situação mostrada na Figura 2.10. Nesse caso, as velocidades que compõem V_s são todos os pares internos ao limite indicado na Figura 2.11. As restrições dinâmicas (máximas acelerações possíveis) limitam localmente as velocidades atuais (v_a, ω_a) do robô, de acordo com a equação 2.2. Desta forma, são fornecidas as velocidades máximas e mínimas que o robô pode atingir em um intervalo de tempo Δt e o mapa de velocidades é reduzido para uma janela dinâmica V_d , que se altera a cada Δt . Na Figura 2.11 as velocidades que compõem V_d estão limitadas em vermelho.

$$V_d = \{(v, \omega) \mid v \in [v_a - \dot{v}\Delta t, v_a + \dot{v}\Delta t], \omega \in [\omega_a - \dot{\omega}\Delta t, \omega_a + \dot{\omega}\Delta t]\} \quad (2.2)$$

Figura 2.10: Instante de movimento do robô Synchro-Drive (Lima, 2010).

Nesta janela V_d são apenas consideradas velocidades que sejam seguras em relação aos obstáculos, tais que permitam ao Synchro-Drive parar sem se colidir. Isto é feito calculando-se as distâncias para os obstáculos vizinhos do robô quando este trafega com cada um dos pares de velocidades de V_d . A função $dist(v, \omega)$, estima a trajetória do robô com suas equações de

movimento e verifica a presença de obstáculos nesta trajetória. A menor distância entre o robô e os obstáculos é então retornada, assim, as velocidades admissíveis para o robô são:

$$V_a = \left\{ (v, \omega) \mid v \leq \sqrt{2dist(v, \omega)\dot{v}_b}, \omega \leq \sqrt{2dist(v, \omega)\dot{\omega}_b} \right\} \quad (2.3)$$

Figura 2.11: Exemplo dos conjuntos V_s , V_a , V_d e V_r , necessários para a aplicação do Método da Janela Dinâmica em um robô do tipo Synchro-Drive. Estes conjuntos foram obtidos por meio da leitura sensorial no ambiente da Figura 2.10, com os valores atuais das velocidades representados por (v_a, ω_a) . Em cinza escuro estão todos os pares de (v, ω) que implicam em um movimento inválido no robô, com possíveis colisões. A região destacada em verde são as velocidades válidas para a janela dinâmica. (Lima, 2010).

onde $\dot{v}_b$ e $\dot{\omega}_b$ são as acelerações de frenagem do robô. Estas velocidades estão representadas pela região em cinza claro e verde da Figura 2.11. O espaço resultante de busca V_r é então definido como a interseção dos demais espaços apresentados, como segue:

$$V_r = V_s \cap V_d \cap V_a \quad (2.4)$$

Ainda na Figura 2.11, estas velocidades foram marcadas com a cor verde. O par de velocidades final é escolhido segundo uma função objetivo $G(v, \omega)$. Esta função representa a soma ponderada de outras subfunções que medem o direcionamento (*heading*) do robô, sua velocidade (*velocity*) e a clareza (*dist*) do caminho em relação ao obstáculos, segundo:

$$G(v, \omega) = \alpha heading(v, \omega) + \beta dist(v, \omega) + \gamma velocity(v, \omega) \quad (2.5)$$

onde α , β e γ são os pesos destas subfunções. Na função objetivo $G(v, \omega)$, a subfunção

heading favorece velocidades que impliquem em uma orientação do robô mais próximas do seu destino. O segundo termo *dist* (mesmo utilizado para calcular as velocidades válidas V_a) é uma tentativa de maximizar o espaço entre o robô e seus obstáculos. Já a subfunção *velocity* prioriza as maiores velocidades de translação para realizar o movimento do robô quando estiver longe do destino e menores, caso contrário. O objetivo do DWA, portanto, é otimizar $G(v, \omega)$ a cada Δt de maneira que o robô trafegue em segurança até um ponto de destino e respeite as subfunções apresentadas.

Como o DWA é construído em torno de subfunções das velocidades de controle do robô, novas funções podem ser criadas para que o mesmo se adeque a tarefas mais específicas. Em Brock e Khatib (1999) por exemplo, duas funções foram adicionadas para permitir que o DWA também seguisse um planejamento de movimento global previamente realizado.

Os conceitos preliminares do DWA são apresentados para o robô Synchrono-Drive. No entanto, na literatura existem várias adaptações deste método para diferentes robôs e podem ser verificadas em Brock e Khatib (1999), Rebai et al. (2007) e Rebai e Azouaoui (2009).

CAPÍTULO 3

PLATAFORMAS ROBÓTICAS

Neste capítulo são descritas as plataformas robóticas utilizadas no desenvolvimento deste trabalho. A primeira plataforma robótica é um Pioneer 3AT, robô de pequeno porte da fabricante *Mobile Robots*, usado neste trabalho em tarefas de planejamento e navegação em ambientes internos (*indoor*). Como segunda plataforma, temos um veículo elétrico autônomo desenvolvido pelo Instituto de Ciências Matemáticas e Computação (ICMC/USP) com apoio do Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos (INCT-SEC). Esse veículo, chamado CaRINA (Carro Robótico Inteligente para Navegação Autônoma), será usado em tarefas de planejamento e navegação em ambientes semi-estruturados urbanos (*outdoor*).

3.1 Pioneer 3AT

O Pioneer 3AT é um robô fabricado pela *Mobile Robots* bastante conhecido e empregado em pesquisas na área de robótica móvel, podendo ser modelado como um robô móvel de rodas deslizantes (RMRD). É dotado de quatro rodas atuadas por dois motores, sendo que um motor é responsável por acionar as rodas do lado direito e outro motor aciona as rodas do lado esquerdo. A mudança de orientação do robô é feita ao se aplicar torques com valores distintos para cada

motor.

Todas as plataformas *Mobile Robots* possuem arquitetura cliente-servidor para controle do robô móvel. O servidor trabalha para gerenciar todos os detalhes do sistemas do robô em baixo nível, estes incluem, operar os motores, acionamento dos sonares, coleta de dados dos sonares e dos *encoders* das rodas e assim por diante, todos sob comando de um aplicativo cliente separado, como por exemplo ARIA (*Advanced Robotics Interface for Applications*) (ROBOTS, 2006).

Figura 3.1: Laser SICK embarcado ao Pioneer.

O robô é equipado com sensor laser SICK LMS 200 que permite um alcance máximo de 80 metros e uma abertura de 180°. A Figura 3.1 mostra o laser SICK embarcado ao robô, esse laser será responsável pelo sistema de percepção da plataforma.

A Tabela 3.1 mostra alguns parâmetros dessa plataforma, que por sua vez é mostrada na Figura 3.2.

Tabela 3.1: Especificações do Pioneer 3-AT

Parâmetro	Valores	Unidade
Largura	49.3	cm
Comprimento	50.1	cm
Altura	27.7	cm
Max. velocidade de translação	700	mm/seg
Max. velocidade de Rotação	140	deg/seg
Diâmetro da roda	220	mm
Carga máxima	40	kg
Peso	14	Kg

Mais detalhes sobre o Pioneer podem ser encontrados em (ROBOTS, 2006) ou na *homepage*

Figura 3.2: Robô Pioneer 3AT

<http://www.mobilerobots.com>.

A seguir, são desenvolvidos os modelos cinemático e dinâmico para o robô Pioneer 3-AT. A modelagem aqui apresentada está baseada em (Caracciolo et al., 1999; Kozlowski e Pazderski, 2004). Esses modelos também são apresentados em (Vaz, 2011).

3.1.1 Modelo Cinemático

A geometria do robô é mostrada na Figura 3.3, sendo que (X, Y) é o sistema de coordenadas inercial, (X', Y') o sistema de coordenadas fixo no corpo tendo como origem o centro de gravidade do robô CG . A coordenada do centro de gravidade do robô no sistema de coordenadas inerciais é (x, y) , e ψ é o ângulo entre o eixo X' e o eixo X . As coordenadas do centro instantâneo de rotação (CIR) são (x'_{CIR}, y'_{CIR}) . O parâmetro a corresponde à distância entre o centro de gravidade e o eixo de simetria da roda atuada frontal paralelo ao eixo Y' , b a distância entre centro de gravidade e o eixo de simetria paralelo ao eixo Y' da roda atuada traseira, c é a metade da distância entre as rodas ao longo do eixo Y' e r o raio das rodas.

No sistema de coordenada do corpo a velocidade linear do robô é dada por $\mathbf{v} = \begin{bmatrix} v_{x'} & v_{y'} & 0 \end{bmatrix}^T$, sendo $v_{x'}$ e $v_{y'}$ as componentes de $\mathbf{v}$ no eixo X' e Y' , e a velocidade angular é dada por $\omega = \begin{bmatrix} 0 & 0 & \dot{\psi} \end{bmatrix}$, sendo $\omega = \dot{\psi}$. As velocidades lineares e angulares de cada roda atuada são dadas por v_i e ω_i , sendo $i = 1, 2, \dots, 4$ a i -ésima roda atuada.

Figura 3.3: Geometria do robô móvel (Vaz, 2011).

As velocidades absolutas $\dot{x}$, $\dot{y}$ e $\dot{\psi}$ no sistema de coordenadas inercial são dadas por

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_{x'} \\ v_{y'} \\ \omega \end{bmatrix} \quad (3.1)$$

$$= R(\psi) \begin{bmatrix} v_{x'} \\ v_{y'} \\ \omega \end{bmatrix} \quad (3.2)$$

Diferenciando a equação acima em relação ao tempo tem-se

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\psi} \end{bmatrix} = R(\psi) \begin{bmatrix} \dot{v}_{x'} - \dot{\psi} v_{y'} \\ \dot{v}_{y'} - \dot{\psi} v_{x'} \\ \dot{\omega} \end{bmatrix} \quad (3.3)$$

$$= R(\psi) \begin{bmatrix} a_{x'} \\ a_{y'} \\ \dot{\omega} \end{bmatrix} \quad (3.4)$$

Diferentemente da maioria dos robôs móveis, a velocidade lateral do RMRD é geralmente diferente de zero, isso vem da estrutura mecânica do RMRD que faz o deslizamento lateral necessário para que o veículo mude de orientação, e quando a velocidade do robô $v_{y'} = 0$ não há deslocamento lateral. A velocidade angular ω e a velocidade lateral $v_{y'}$ ambas desaparecem quando o robô movimenta-se em linha reta, e o CIR vai para infinito ao longo do eixo Y' . Já em uma trajetória curva o CIR desloca x'_{CIR} ao longo do eixo X' . Se x'_{CIR} ultrapassar a base das rodas do robô, o veículo desliza drasticamente com perda de estabilidade de movimento. Assim, para completar o modelo cinemático do RMRD, a seguinte restrição não-holonômica foi introduzida em (Caracciolo et al., 1999).

$$v_{y'} = x'_{CIR}\dot{\psi}, \quad \text{com} \quad x'_{CIR} \in (-b, a) \quad (3.5)$$

No entanto, da equação 3.1 obtém-se:

$$v_{y'} = -\sin \psi \dot{x} + \cos \psi \dot{y} \quad (3.6)$$

Seja $\mathbf{q} = \begin{bmatrix} x & y & \psi \end{bmatrix}$ o estado do robô, substituindo a equação 3.6 em 3.5 e escrevendo na forma Pfaffian, tem-se:

$$\begin{bmatrix} -\sin \psi & \cos \psi & -x'_{CIR} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = A(\mathbf{q})\dot{\mathbf{q}} = 0$$

A partir da Figura 3.3, as velocidades $\dot{\mathbf{q}}$ podem ser expressas como:

$$\dot{\mathbf{q}} = S(\mathbf{q})\boldsymbol{\eta} \quad (3.7)$$

Onde

$$S(\mathbf{q}) = \begin{bmatrix} \cos \psi & -x'_{CIR} \sin \psi \\ \sin \psi & -x'_{CIR} \cos \psi \\ 0 & 1 \end{bmatrix},$$

$$\boldsymbol{\eta} = \begin{bmatrix} v_{x'} \\ \omega \end{bmatrix}.$$

$S(\mathbf{q})$ é uma matriz de posto completo, cujas colunas estão no espaço nulo de $A(\mathbf{q})$, ou seja

$$S^T(\mathbf{q})A^T(\mathbf{q}) = 0$$

É interessante observar que como a $\dim(\boldsymbol{\eta}) = 2 < \dim(\mathbf{q}) = 3$, a equação 3.7 descreve a cinemática de um robô sub-atuado, que é um sistema não-holonômico por causa da restrição apresentada na equação 3.5.

3.1.2 Modelo Dinâmico

A Figura 3.4 mostra as forças de tração $F_{x'i}$ que são sujeitas a forças resistivas longitudinais $R_{x'i}$. O Pioneer 3-AT possui dois motores, cada um responsável por acionar as rodas de um dos lados do robô. Então assume-se que a atuação das rodas é igual em cada lado, ou seja, $F_{x'1} = F_{x'3}$ e $F_{x'2} = F_{x'4}$, diminuindo assim o deslizamento longitudinal. As forças laterais $R_{y'i}$ que atuam sobre as rodas são consequências do deslizamento lateral. O momento resistivo M_r em volta do centro de massa, que se opõe ao momento M é induzido pelas forças $R_{y'i}$ e $R_{x'i}$.

Figura 3.4: Forças de tração e resistivas do robô móvel (Vaz, 2011).

Para um veículo, como mostrado na Figura 3.4, de massa m e de inércia I no centro de massa, as equações de movimento no sistema de coordenada do corpo são:

$$\begin{cases} ma_{x'} = 2F_{x'1} + 2F_{x'2} - R_{x'}, \\ ma_{y'} = -R_{y'}, \\ I\ddot{\psi} = 2c(F_{x'1} - F_{x'2}) - M_r. \end{cases} \quad (3.8)$$

Para representar as forças longitudinais $R_{x'}$, as forças laterais $R_{y'}$, e o momento resistivo M_r , deve-se considerar mg como a carga gravitacional, sendo m a massa do robô e g a aceleração da gravidade, que é dividida sobre as rodas do robô, e introduzir o modelo Coulomb de atrito para o contato das rodas com a superfície. Desse modo, desde que o robô tenha uma linha mediana longitudinal de simetria, obtém-se

$$R_{z'1} = R_{z'2} = \frac{b}{2(a+b)}mg,$$

$$R_{z'3} = R_{z'4} = \frac{a}{2(a+b)}mg.$$

Em velocidades baixas, a carga lateral transferida devido as forças centrífugas sobre percursos curvados pode ser desconsiderada. No caso de superfícies duras, pode-se assumir que o contato entre as rodas e o solo é retangular e que a carga vertical produz uma pressão uniformemente distribuída. Assim $R_{x'i} = f_r R_{z'i} \text{sgn}(v_{x'i})$ sendo que: f_r é o coeficiente resistivo de rolagem independente da velocidade e $\text{sgn}(\cdot)$ a função sinal. A força resistiva longitudinal total é:

$$\begin{aligned} R_{x'} &= \sum_{i=1}^4 R_{x'i} \\ &= \frac{f_r mg}{2} (\text{sgn}(v_{x'1}) + \text{sgn}(v_{x'2})), \end{aligned}$$

Introduzindo o coeficiente de atrito lateral μ , temos que $R_{y'i} = \mu R_{z'i} \text{sgn}(v_{y'i})$. Portanto a força lateral total é dada por

$$\begin{aligned} R_{y'} &= \sum_{i=1}^4 R_{y'i} \\ &= \mu \frac{mg}{a+b} (b \text{sgn}(v_{y'1}) + \text{sgn}(v_{y'3})), \end{aligned}$$

enquanto o momento resistivo será (Caracciolo et al., 1999):

$$M_r = \mu \frac{abmg}{a+b} (\text{sgn}(v_{y'1}) - \text{sgn}(v_{y'3})) + \frac{f_r cmg}{2} (\text{sgn}(v_{x'1}) - \text{sgn}(v_{x'2})).$$

utilizando-se notação matricial, o modelo dinâmico reescrito no sistema de coordenadas inerciais é

$$M\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}}) = E(\mathbf{q})\tau, \quad (3.9)$$

sendo

$$M = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix},$$

$$C(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} \cos \psi R_{x'} - \sin \psi R_{y'} \\ \sin \psi R_{x'} + \cos \psi R_{y'} \\ M_r \end{bmatrix},$$

$$E(\mathbf{q}) = \begin{bmatrix} \frac{\cos \psi}{r} & \frac{\cos \psi}{r} \\ \frac{\sin \psi}{r} & \frac{\sin \psi}{r} \\ \frac{c}{r} & -\frac{c}{r} \end{bmatrix},$$

$$\tau = \begin{bmatrix} \tau_L \\ \tau_R \end{bmatrix} = \begin{bmatrix} \tau_1 + \tau_3 \\ \tau_2 + \tau_4 \end{bmatrix}, \tau_i = r F_{x'i}.$$

Incluindo a restrição não-holonômica no modelo dinâmico (equação 3.9), como introduzido em (Caracciolo et al., 1999), tem-se:

$$M\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}}) = E(\mathbf{q})\tau + A^T(\mathbf{q})\lambda, \quad (3.10)$$

sendo λ o vetor de multiplicadores de Lagrange correspondente a equação 3.5. Para estimar as acelerações a partir do torque aplicado, devemos diferenciar a equação 3.7, substituir na equação 3.10 e eliminar λ , então teremos o seguinte resultado (Caracciolo et al., 1999):

$$\dot{\boldsymbol{\eta}} = (S^T M S)^{-1} S^T \left(E_\tau - M \dot{S} \boldsymbol{\eta} - C \right) \quad (3.11)$$

3.2 Projeto CaRINA

O projeto CaRINA¹. (Carro Robótico Inteligente para Navegação Autônoma), é um projeto desenvolvido no Laboratório de Robótica Móvel (LRM) do Instituto de Ciências Matemáticas e Computação (ICMC/USP) em parceria com o Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos (INCT-SEC) que visa o desenvolvimento de um veículo autônomo inteligente capaz de navegar em ambientes urbanos sem a necessidade de um condutor humano.

O CaRINA 1 é um veículo elétrico Club Car modelo Carryall 232, adaptado para o controle computacional de velocidade e esterçamento, e instrumentado com sensores para localização e percepção. Ele atinge uma velocidade máxima de 30km/h e tem autonomia de aproximadamente 10hs. O esterçamento é realizado com um motor DC acoplado na barra de direção, um *encoder* óptico e um controlador. A aceleração é controlada com um microcontrolador Arduino que atua diretamente no módulo de controle do veículo. A velocidade e a odometria são obtidas através de *encoders* ópticos (Fernandes et al., 2012).

O CaRINA I está equipado com diversos tipos de sensores que são utilizados na navegação do veículo. Alguns dos sensores usados atualmente são: GPS, unidade inercial, sensores laser (LIDAR) e câmeras monoculares e estéreo. Alguns desses sensores embarcados no veículo podem ser vistos na Figura 3.5.

Figura 3.5: Sensores embarcados no CaRINA 1.

A Tabela 3.2 mostra alguns parâmetros dessa plataforma que por sua vez é mostrada na

¹Mais detalhes sobre o projeto CaRINA podem ser encontrados na *homepage* do LRM - Laboratório de Robótica Móvel no endereço: <http://www.lrm.icmc.usp.br/?page=projetos&projeto=carina>

Figura 3.6.

Tabela 3.2: Especificações do veículo

Parâmetro	Valores	Unidade
Largura	1.35	m
Comprimento	2.57	m
Distância entre eixos	1.65	m
Distância entre as rodas	0.90	m
Diâmetro da roda	0.48	m
Raio mínimo de curvatura	2.70	m
Peso	521.6	Kg

Figura 3.6: Veículo Autônomo CARina 1

Neste trabalho, o sistema de percepção do veículo CaRINA 1 é obtido através da combinação dos dados de três sensores laser (*Laser Range Finders*). Localizado na parte frontal do veículo encontra-se um sensor laser do modelo SICK LMS 291 (Figura 3.5(a)) e na parte traseira encontra-se dois sensores do modelo Hokuyo UTM-30LX (Figura 3.5(b)). O laser frontal possui um alcance de 80 metros e uma abertura de 180° . Enquanto que os lasers localizados na parte traseira do veículo possuem alcance de 30 metros e abertura de 270° (Prado, 2013). A Figura 3.7(a) mostra a posição dos lasers no veículo.

Com o objetivo de facilitar o trabalho com os sensores laser, (Prado, 2013) propôs a fusão desse sistema em uma única fonte de informações, assim os dados de cada sensor foram or-

ganizados em uma única nuvem de pontos, montando um ambiente de duas dimensões. Em seguida, 360 pontos dessa nuvem foram associados a uma estrutura onde é considerado uma única origem para todos esse pontos. Dessa forma o sistema considera a existência de apenas um laser (Figura 3.7(b)) com capacidade de visão de 360° e alcance de 30 metros.

A Figura 3.8 mostra a detecção de obstáculos pelo veículo CaRINA 1 a partir do laser 360° .

Figura 3.7: Sistema de percepção do CaRINA 1 (Prado, 2013). (a) Posicionamento e alcance de cada sensor laser, (b) Fusão dos sensores laser

Figura 3.8: Visão de 360° dos sensores laser. Pontos vermelhos representam a detecção de obstáculos pela fusão dos sensores laser.

A seguir apresentamos a modelagem desse veículo, que é baseada na geometria Ackerman.

3.2.1 Modelo Ackerman

Diferentes modelos aproximados para um veículo podem ser encontrados em (De Luca et al., 1998; Rebai et al., 2007). Estes modelos dependem, principalmente, de sua tração e do quão real deve ser a descrição da direção de movimento do veículo. A tração, que pode ser dianteira ou traseira, interfere no sentido da velocidade fornecida ao carro pelo torque das rodas. A descrição da direção interfere em como será o movimento do carro em curvas (Lima, 2010). Como cada uma das rodas frontais, quando direcionadas para uma curva, estará seguindo um caminho que corresponde à parte de uma circunferência, o raio dessa curva depende de quão esterçadas estão as rodas. Deste modo, a roda externa ao centro de curvatura percorre uma distância maior do que a roda interna, portanto as rodas, quando esterçadas, têm que estar com uma inclinação tal que ambas formem um ângulo de 90° , com uma linha que una o centro de curvatura ao centro da roda. Consequentemente, as rodas necessitam de ângulos diferentes uma em relação à outra para atenderem a este requisito. O modelo mais comum para aproximar a direção é o modelo de Ackerman (Choset et al., 2005), o qual aproxima o carro a uma bicicleta, com a direção final sendo a média dos ângulos das duas rodas frontais.

Figura 3.9: Modelo Cinemático

A Figura 3.9 representa um veículo de quatro rodas. Por simplicidade no modelo, assumiremos que as duas rodas de cada eixo (dianteiro e traseiro) foram unidas em uma única roda situada no ponto médio de seu respectivo eixo, sendo que a roda no eixo da frente pode ser esterçada enquanto a roda do eixo traseiro possui orientação fixa (De Luca et al., 1998).

As coordenadas generalizadas que representam esse sistema são $\mathbf{q} = (x, y, \theta, \delta)$, onde x e y identificam o ponto médio do eixo traseiro, θ define sua orientação (ângulo em relação a direção de referência) e δ é o ângulo de esterçamento.

Este sistema está sujeito à duas restrições não-holonômicas, uma para cada roda:

$$\begin{aligned} \dot{x}_f \sin(\theta + \delta) - \dot{y}_f \cos(\theta + \delta) &= 0 \\ \dot{x} \sin(\theta) - \dot{y} \cos(\theta) &= 0 \end{aligned} \quad (3.12)$$

onde (x_f, y_f) , representa o ponto médio no eixo dianteiro. Usando a restrição de corpo rígido dada por:

$$\begin{cases} x_f = x + L \cos(\theta) \\ y_f = y + L \sin(\theta) \end{cases} \quad (3.13)$$

podemos reescrever a primeira linha da equação 3.12 da seguinte forma:

$$\dot{x} \sin(\theta + \delta) - \dot{y} \cos(\theta + \delta) - \dot{\theta} L \sin(\delta) = 0$$

onde L é a distância entre os eixos do veículo.

Escrevendo a matriz de restrições na forma Pfaffian, tem-se:

$$A(q) = \begin{bmatrix} \sin(\theta + \delta) & -\cos(\theta + \delta) & -L \sin(\delta) & 0 \\ \sin(\theta) & -\cos(\theta) & 0 & 0 \end{bmatrix}$$

Assim, para um veículo de tração traseira, o modelo cinemático é expresso por: (De Luca et al., 1998)

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} \cos \theta \\ \sin \theta \\ \frac{\tan \delta}{L} \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_2 \quad (3.14)$$

As entradas deste modelo são: a velocidades linear das rodas dianteiras v_1 e a variação angular do esterçamento destas rodas v_2 . O modelo apresentado considera que o veículo move-se em trajetórias circulares definidas pelo centro de curvatura instantâneo (ICC), com o raio definido por $\rho = \frac{L}{\tan \delta}$. Neste caso o veículo possuirá uma velocidade angular definida por:

$$\omega = \frac{v}{\rho} \quad (3.15)$$

Para o caso de $\rho = \infty$, a velocidade angular será igual a zero e o veículo se moverá em linha reta. A partir do modelo cinemático apresentado é possível descrever o movimento do veículo em situações ideais.

CAPÍTULO 4

FERRAMENTAS E BIBLIOTECAS

Este capítulo apresenta uma descrição das principais ferramentas e bibliotecas utilizadas no desenvolvimento deste trabalho.

4.1 ROS

Robot Operating System (**ROS**) é o nome do *framework* para operações com robôs que será utilizado durante este projeto. O principal argumento por trás do uso de um sistema como o **ROS** é que ele fornece abstração de hardware, drivers de dispositivos, bibliotecas, visualizadores, simuladores, transmissão de mensagens e gerenciamento de pacotes.

A comunicação **ROS** é baseada na topologia *peer-to-peer*, isto significa que os nós se comunicam aos pares, e cada nó necessariamente não precisa saber da existência dos demais nós. A topologia *peer-to-peer* exige algum tipo de serviço mestre para que os processos possam ser encontrados por outros durante a execução. Na sua essência, **ROS** fornece uma estrutura para os processos separados (nós) e também os meios para comunicação entre esses nós (Quigley et al., 2009).

De uma forma geral o funcionamento do **ROS** pode ser fundamentado em quatro elementos

derivados da comunicação *peer-to-peer*:

- **Nós:** executável com uma única tarefa.
- **Mensagens:** estrutura de dados com a finalidade de servir de base para a comunicação entre os nós;
- **Tópicos:** definem a forma através da qual os nós enviam as mensagens. Os nós enviam suas mensagens para um tópico e caso algum outro necessite dessa informação ele obtém essa informação nesse mesmo tópico. Podendo um tópico se comunicar com vários nós e um nó se comunicar com vários tópicos;
- **Serviços:** são estruturas de dados para comunicação direta entre dois nós.

Um nó é um programa executável que leva os dados de sensores do robô e passa para os outros nós. A informação que é passada de um nó para outro é chamada de mensagem. Mensagens sempre viajam através de portas especiais chamadas tópicos. Um nó que envia mensagens em um tópico é chamado de *publisher* e o nó receptor tem que subscrever o tópico para receber essa mensagem, por isso é chamado de *subscriber*. Todos os nós relacionados são combinados em pacotes que podem ser facilmente compilados e portados para outros computadores. ROS suporta várias linguagens de programação incluindo *C++*, *Python*, *Octave*, *LISP*.

Tem-se utilizado o ambiente **ROS** em muitas aplicações de robótica, como o programa de robôs pessoais da Willow Garage e do projeto STAIR (*Stanford Artificial Intelligence Robot*) da Universidade de Stanford.

Um sistema robótico completo contém muitos nós funcionando juntos, sendo que cada nó é responsável por uma determinada tarefa. A Figura 4.1 mostra um exemplo de uma aplicação robótica. Cada balão representa um nó em funcionamento, cada um desses nós, publicam ou subscrevem uma mensagem através de um tópico específico e as setas indicam o sentido de comunicação.

Os ambiente de simulação STAGE (simulador 2D) (Gerkey et al., 2003) e GAZEBO (Simulador 3D) (Hsu e Koenig, 2012) são compatíveis com a plataforma **ROS** e são utilizados para testar e validar os sistemas desenvolvidos. Além do STAGE (Figura 4.2(a)) e GAZEBO (Figura 4.2(b)), o **ROS** possui um ambiente de visualização em 3D para robôs, chamado **RVIZ** (Figura 4.2(c)). Esse ambiente disponibiliza a visualização de robôs e sua interação com o ambiente,

Figura 4.1: Diagrama de troca de informações entre nós para uma determinada aplicação robótica.

como a representação das leituras dos sensores e criação de mapas.

(a) Simulador Stage

(b) Simulador Gazebo

(c) Visualização RVIZ

Figura 4.2: Ambiente de simulação do **ROS**. (a) Simulador STAGE. (b) Simulador GAZEBO (c) Visualizador de ambientes 3D do ROS.

4.2 SBPL

SBPL (*Search-based Planning Library*) é uma biblioteca integrada ao ambiente **ROS** que implementa um conjunto genérico de planejadores de movimento baseados em busca em grafo. Essa biblioteca foi desenvolvida pela University of Pennsylvania com colaboração da Willow Garage e está bem descrita em Likhachev (2009). Atualmente essa biblioteca contém as seguintes implementações:

- ARA* - Versão *Anytime* do A*;
- AD* - Versão *Anytime* e *Incremental* do A*;
- R* - Versão Randomizada do A* (híbrido entre pesquisas determinísticas e planejamento ba-

seado em amostragem);

Essas implementações são utilizadas em ambientes de planejamento 2D (x, y) , que leva em conta somente a posição do robô, e 3D (x, y, θ) , que além da posição, também considera o orientação do robô. Na Figura 4.3 tem-se a representação de um grafo 2D para planejamento de movimento baseado em busca.

Figura 4.3: Representação em grafo para planejamento 2D

Mais detalhes dessa biblioteca, bem como sua aplicação, podem ser encontrados em (Likhahev, 2009; ROS, 2013a).

4.2.1 SBPL LATTICE PLANNER

Utilizado como o planejador global, esse nó do ambiente **ROS**, encapsula a biblioteca **SBPL** para gerar um caminho a partir da posição atual do robô para um objetivo desejado. Caminhos são gerados através da combinação de uma série de *movimentos primitivos* (movimentos que atendem a cinemática e a dinâmica do robô) que são passados para o planejador assim como outras informações do robô. O planejamento é, portanto, feito em três dimensões (x, y, θ) , resultando em caminhos suaves que levam em conta a orientação robô, o que é especialmente importante se o robô tem restrições não-holonômicos. Os caminhos são encontradas usando o planejador AD*. A Figura 4.4 mostra um exemplo da geração de um caminho utilizando malhas

de estados.

Figura 4.4: Representação em grafo baseado em malha de estados para planejamento de movimento 3D (Likhachev, 2009).

A abordagem de malhas de estados pode ser utilizada em vários tipos de plataformas para geração de planos seguros em diferentes aplicações. Um exemplo disso pode ser visto em (Scholz et al., 2011) que usa malha de estados em um robô manipulador (PR2) em tarefas de abrir portas. Outro exemplo pode ser visto em (Phillips e Likhachev, 2011) que utiliza malhas de estados para gerar planos para a tarefa de *cart pushing* com robô manipulador.

Outras informações podem ser obtidas em (ROS, 2013b).

4.3 AMCL

AMCL (*Adaptive Monte Carlo Localization*), biblioteca integrada ao ambiente **ROS**, é um sistema de localização probabilística que é derivada da localização de Markov (Fox et al., 1999). Este sistema é utilizado para localização de um robô que se move em ambientes 2D. Essa abordagem usa um filtro de partículas para acompanhar a posição de um robô em um mapa conhecido. Os algoritmos para implementação desse sistema estão bem descritos em Thrun et al. (2005).

AMCL estima a posição do robô no mapa através de dados fornecidos pelo sensor laser e das transformações dos sistemas de coordenadas existentes (Figura 4.5). **AMCL** inicializa seu filtro de partículas de acordo com os parâmetros fornecidos, levando em conta o tipo de

Figura 4.5: Estimação da posição pelo AMCL

odometria existente no robô.

4.4 GMAPPING

Mapeamento é o processo de criação de um modelo do ambiente em torno do robô usando seus sensores. O mapa é então usado para localização e navegação. GMAPPING é uma biblioteca integrada ao ambiente **ROS** que implementa o (RBPF) *Rao-Blackwellized particle filter* (Grisetti et al., 2005) para resolver o problema de mapeamento e localização simultâneos (SLAM).¹

Recentemente RBPF's foram introduzidos como meios eficazes para resolver o problema mapeamento e localização simultâneos (SLAM) problema. Esta abordagem utiliza um filtro de partículas em que cada partícula carrega um mapa individual do ambiente. Assim, a questão chave é a forma de reduzir o número de partículas. A partir de técnicas adaptativas pode-se reduzir o número de partículas em um RBPF para gerar mapas de grade de ocupação.

¹Fonte: <http://openslam.org/gmapping.html>

CAPÍTULO 5

METODOLOGIA

Este capítulo apresenta a metodologia para implementação desse trabalho. Essa metodologia foi dividida em duas seções para um melhor entendimento do método proposto. Essas duas seções são: navegação em ambientes internos e navegação em ambientes externos. Como mencionado no capítulo 3 o robô Pioneer 3AT foi utilizado para navegação em ambientes internos e o veículo CaRINA 1 para navegação em ambientes externos.

Como mencionado anteriormente, a navegação segura de um robô autônomo em um ambiente, seja ele interno ou externo, pode ser realizada de diversas formas.

5.1 Modelagem Computacional

A simulação é uma etapa importante em um projeto de desenvolvimento de sistemas robóticos, além da garantia de segurança durante os testes, tempo e esforços são economizados em comparação com testes reais (Prado, 2013). Para se criar um modelo para simulação é necessário criar um modelo simplificado da realidade, implementando este em um ambiente computacional. O modelo deve integrar os elementos e características que são mais relevantes para o funcionamento do sistema robótico (Wolf et al., 2009).

O ambiente **ROS**, como já citado, provê suporte tanto de visualização como simulação de modelos robóticos. Para isso, faz-se necessário a criação de modelos baseados em um formato XML (*eXtensible Markup Language*) específico. O URDF (*Unified Robot Description Format*) é o formato utilizado pelo **ROS** para descrever seus robôs. Os modelos são descritos como um conjunto de *links* e *joints*. Um *link* pode representar uma parte específica do robô, como uma roda ou um sensor, enquanto um *joint* equivale à ligação entre dois *links*. Várias características como inércia e área de colisão podem ser definidas através desses modelos.

As principais medidas e restrições cinemáticas são representadas nos modelos simulados. As Figuras 5.1(a) e 5.2(a) mostram os modelos simulados das plataformas no ambiente **RVIZ**, enquanto as Figuras 5.1(c) e 5.2(c) ilustram as plataformas reais. O formato URDF provê suporte ao ambiente Gazebo (Hsu e Koenig, 2012) por intermédio da inclusão de alguns *plugins* no próprio arquivo XML. As Figuras 5.1(b) e 5.2(b) ilustram os modelos das plataformas, similares a uma estrutura de árvore, em sua relação de *links* e *joints*, destacando suas orientações e posições em relação a um nó (*link*) principal Prado (2013).

Figura 5.1: Representação tridimensional da plataforma Pioneer 3AT. (a) Modelo 3D. (b) Modelo em estrutura de árvore. (c) Pioneer 3AT.

Figura 5.2: Representação tridimensional da plataforma CaRINA 1 (Prado, 2013). (a) Modelo 3D. (b) Modelo em estrutura de árvore. (c) CaRINA 1.

5.2 Navegação em ambientes internos

A navegação em ambientes internos será conduzida de acordo o esquema mostrado na Figura 5.3 usando o robô móvel Pioneer 3AT. Este esquema mostra todas as etapas necessárias para navegar de forma segura pelo ambiente.

Figura 5.3: Navegação com o Pioneer

5.2.1 Mapeamento

Na etapa de mapeamento (Figura 5.3) para navegação em ambientes internos, o robô possui um conhecimento prévio do modelo do ambiente que é passado na forma de um mapa métrico estático. Esse mapa mostrado na Figura 5.4, é a representação do andar térreo do Laboratório de Sistemas Inteligentes (LASI-EESC/USP) em uma grade de ocupação, e foi criado a partir dos dados coletados no ambiente através de uma navegação teleoperada usando o Pioneer 3AT.

Figura 5.4: Mapa de ambiente em grade de ocupação.

Na navegação teleoperada, armazenou-se: informações obtidas pelo sensor laser, o posicionamento do robô e as transformações entre os quadros de coordenadas existentes no modelo

do robô. Com esse conjunto de dados, foi possível criar o mapa usando o GMAPPING descrito na Seção 4.4. O mapa criado possui uma resolução de $2.5cm/pixel$, esse valor representa o tamanho das células da grade.

5.2.2 Localização

Na etapa localização (Figura 5.3) o robô precisa encontrar sua posição no mapa para garantir uma navegação segura em ambientes internos. Encontrar a posição real do robô usando apenas fontes de odometria pode não garantir a segurança da navegação devido aos erros acumulados, quanto mais o robô se desloca, maior é a incerteza sobre sua posição real. Abordagens probabilísticas representam uma forma de corrigir esse problema como visto na Seção 2.2.

Figura 5.5: Localização de Monte Carlo no Pioneer

A partir de informações do sensor laser, é possível diminuir a incerteza sobre a posição real do robô. Na Figura 5.5 temos a utilização do **AMCL** para rastrear a posição do robô no mapa obtido na Seção anterior.

A Figura 5.5(a) mostra as partículas espalhadas de acordo com o tipo de odometria do robô, isso representa a incerteza sobre sua posição inicial. A medida que o robô se desloca e novas observações são obtidas (Figuras 5.5(b) e 5.5(c)), é realizada uma comparação entre as observações feitas por cada partícula e as observações do próprio robô. As partículas cujas observações mais se aproximam com as obtidas pelo robô recebem um peso elevado e são mantidas, enquanto as demais são descartadas. Após percorrer uma certa distância, o filtro converge para uma região que tem uma maior probabilidade de ser a posição real do robô (Figura 5.5(d)).

5.2.3 Planejamento

Na etapa de planejamento (Figura 5.3), objetivasse determinar quais movimentos o robô deve realizar para que alcance posições desejadas no ambiente sem que ocorram colisões com obstáculos.

O planejamento de movimento foi baseado no método de malha de estados (Pivtoraiko e Kelly, 2005) descrito na Seção 2.3.4. Essa abordagem foi escolhida, devido ao fato de poder incorporar as restrições cinemáticas juntamente com o modelo dinâmico do robô. Além disso, possui a sua capacidade de encontrar trajetórias válidas rapidamente e melhorar essa solução encontrada quando ainda tiver tempo disponível.

O SBPL LATTICE PLANNER que cria a malha de estados 3D¹ (x, y, θ) através da combinação de movimentos primitivos do robô. Esse movimentos primitivos são pequenos movimentos possíveis de serem realizados pelo robô e são passados como entrada para o SBPL LATTICE PLANNER assim como outras características do robô.

Geração dos movimentos primitivos

Os movimentos primitivos foram obtidos utilizando técnicas de integração numérica para os modelos apresentados na Seção 3.1, modelo cinemático (Equação 3.7) e modelo dinâmico (Equação 3.11), assim pode-se obter estados possíveis de serem alcançados pelo robô.

A partir do estado inicial $\mathbf{q} = (0, 0, 0)$, da entrada de controle $\mathbf{u} = (v, \omega)$ e do tempo de integração Δt , é possível estima o estado futuro $\mathbf{q}_{n+1}$.

As integrações numéricas foram realizadas utilizando o método Runge-Kutta de quarta ordem. Abaixo tem-se o procedimento utilizado para a integração numérica da Equação 3.7.

$$\begin{aligned}\dot{\mathbf{q}} &= f(\mathbf{q}, \mathbf{u}, t), \mathbf{q}(t_0) = \mathbf{q}_n \\ \mathbf{q}_{n+1} &= \mathbf{q}_n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\ t_{n+1} &= t_n + \Delta t.\end{aligned}\tag{5.1}$$

¹Em Likhachev e Ferguson (2009) a malha de estados é 4D (x, y, θ, v) velando a velocidade em consideração além da posição e orientação do veículo

onde:

$$\begin{aligned} k_1 &= f(\mathbf{q}_n, \mathbf{u}_n)_{t_n}, \\ k_2 &= f\left(\mathbf{q}_n + \frac{k_1}{2}, \mathbf{u}_n\right)_{t_n + \frac{\Delta t}{2}}, \\ k_3 &= f\left(\mathbf{q}_n + \frac{k_2}{2}, \mathbf{u}_n\right)_{t_n + \frac{\Delta t}{2}}, \\ k_4 &= f(\mathbf{q}_n + k_3, \mathbf{u}_n)_{t_n + \Delta t}. \end{aligned}$$

e f é a equação de transição de estado.

A Figura 5.6, mostra um conjunto de 6 ações obtidas utilizando a Equação 5.1 para um tempo $t = 3s$. Além dessas 6 ações, ainda são permitidas as ações de rotação para esquerda e rotação para a direita. Como o Pioneer 3AT possui sensor laser apenas na frente, as manobras de ré são desconsideradas. Assim o conjunto de movimentos primitivos é composto de 8 ações para cada um dos 16 ângulos de orientação discretizados². Esse conjunto foi utilizado para planejar as trajetórias para a navegação em ambientes internos mostradas no Capítulo 6.

Figura 5.6: Primitivas de Movimento para o Pioneer

5.2.4 Controle

Controlar um robô autônomo é permitir que este realize seu movimento previamente planejado de forma segura (Lima, 2010).

²A Figura 5.6 mostra as ações possíveis de serem realizadas pelo robô quando este encontra-se com orientação $\theta = 0^\circ$, as ações para os demais ângulos são obtidas pela rotação deste conjunto.

Na seção 2.4 foi apresentado o método da janela dinâmica para o controle de um robô tipo Synchro-Drive. Esse método também foi utilizado na etapa de controle (Figura 5.3) para o robô Pioneer 3AT, mas para isso, algumas adaptações no método apresentado em (Fox et al., 1997) foram feitas para atender as necessidades de navegação.

Considerando o par (v, ω) como entrada de controle para o robô Pioneer 3AT, o espaço de velocidade criado pelo DWA é quadrado, sendo definido como na Equação 2.2. Para que o robô possa seguir a trajetória planejada e replanejar essa trajetória quando um obstáculo é encontrado, a função objetivo foi definida pela soma ponderada das seguintes subfunções: (I) $d_{path}(v, \omega)$ que representa a proximidade com o plano global obtido pela malha de estados dentro da janela de navegação, (II) $d_{goal}(v, \omega)$ que representa a proximidade em relação a meta estabelecida³ e (III) $d_{obs}(v, \omega)$ que representa a proximidade com os obstáculos dentro da janela de navegação.

A função objetivo retorna o custo da aplicação da ação (v, ω) em um tempo t e é dada por:

$$O(v, \omega) = \alpha d_{path}(v, \omega) + \beta d_{goal}(v, \omega) + \gamma d_{obs}(v, \omega) \quad (5.2)$$

Onde α , β e γ são as ponderações das subfunções⁴.

Para cada par (v, ω) do espaço de velocidade, cria-se uma trajetória cinemática para o robô. Essa trajetória é resultante da aplicação dessa entrada de controle por um determinado tempo⁵. Dentre essas trajetórias, aquela que retornar o menor custo na função objetivo, a ação correspondente (v, ω) será de fato aplicada ao robô por um instante de tempo Δt , passado esse tempo um novo espaço de velocidade é criado e novamente são feitas as verificações.

As ações do controle são tomadas baseadas nas observações do robô obtidas na janela de navegação que é mostrada na Figura 5.7. Esse janela é um mapa de custo local (*local costmap*) de tamanho (2 metros X 2 metros) cujo centro encontra-se no próprio centro do robô, isso significa que a janela se desloca juntamente com o robô. Na Figura 5.7 a janela está representada em *azul*, quando obstáculos entram nessa janela (*em vermelho*), cria-se uma região de segurança

³Quando a meta estabelecida encontra-se fora da janela de navegação, considera-se como meta o último ponto do caminho global dentro da janela.

⁴Os valores utilizados foram: $\alpha = 0.8$, $\beta = 0.4$ e $\gamma = 0.01$. Para esse valores, observou-se o seguimento da trajetória global planejada.

⁵Essa tarefa representa uma simulação dos estados futuros do robô obtidas através do seu modelo cinemático quando essa entrada de controle é aplicada por um tempo $t \gg \Delta t$.

Figura 5.7: Janela de navegação para o Pioneer.

(em verde) de 30cm em volta desses obstáculos. Essa região de segurança possui custo um inflacionado⁶ enquanto os obstáculos possuem um custo impossível, isso é uma garantia para que o planejador mostrado na Seção 5.2.3 evite essas regiões.

5.3 Navegação em ambientes externos

A navegação em ambientes externos será realizada de acordo com o esquema mostrado na Figura 5.8. Este esquema mostra todas as etapas necessárias para navegar de forma segura pelo ambiente usando o veículo CaRINA 1.

Essa Seção, mostra as adaptações feitas nas etapas da Seção 5.2 para utilização do veículo CaRINA 1 na tarefa de navegação em ambientes externos.

Figura 5.8: Navegação com o CaRINA

⁶As ações de controle que levam o robô para essas regiões são ações mais custosas na função objetivo.

5.3.1 Mapeamento

O mapeamento de ambientes externos é uma tarefa difícil de ser realizada devido a complexidade desses ambientes. Além disso, muitos dos algoritmos e técnicas de mapeamento são aplicáveis somente aos robôs que trafegam em ambientes internos (Hata, 2010).

Para navegação em ambiente externo utilizou-se técnicas de SLAM (Localização e Mapeamento Simultâneo). Utilizando o SLAM é possível criar um mapa e localizar-se nesse mapa ao mesmo tempo. O mapa é obtido a medida que o veículo se desloca e suas observações são representadas em uma grade de ocupação.

A Figura 5.9 mostra o ambiente simulado utilizado para navegação. Este ambiente é uma representação de um parque de estacionamento. Utilizando o GMapping um mapa do ambiente é obtido a medida que o veículo trafega pelo estacionamento. Esse mapa é uma grade de ocupação com resolução $10\text{cm}/\text{pixel}$ e está sendo construído em tempo real.

Para a tarefa de navegação, uma pequena adaptação em GMapping foi realizada para que este aceitasse metas que estão fora do janela de navegação do veículo. Isto é, assumir que inicialmente o mapa é totalmente navegável.

Figura 5.9: Ambiente simulado no Gazebo

5.3.2 Localização

O Sistema de localização do CaRINA 1 é formado a partir da combinação de duas abordagens para estimar a posição do veículo. A primeira abordagem estima a posição baseado em dois *encoders*, um localizado em uma das rodas dianteiras e outro localizado no barramento de direção, e uma IMU (*Inertial Measurement Unit*) posicionada no centro do eixo traseiro para melhorar a estimação da orientação. Esta primeira abordagem calcula a odometria baseando-se no modelo Ackerman. A segunda abordagem utiliza os três lasers presentes no veículo. Os dados desses três lasers são aplicados no algoritmo ICP (*Iterative Closest/Corresponding Point*) (Censi, 2008) para estimar a posição do veículo. O princípio básico deste algoritmo é encontrar uma transformação entre duas nuvens de pontos. Neste trabalho, ele realiza um casamento de pontos dos lasers durante o deslocamento do veículo, possibilitando assim, estimar sua posição.

Figura 5.10: Estimativa de posição obtida com o filtro de Kalman

Em ambas as abordagens, a estimação da posição do veículo está sujeita a erros que podem ser causados por imprecisão ou ruído dos sensores, como também por irregularidades do terreno. Esses erros se propagam enquanto o veículo se move pelo ambiente, aumentando a incerteza sobre sua localização. Com o objetivo de diminuir essa imprecisão, utilizou-se um filtro de Kalman estendido (*Extended Kalman Filter*) para calcular uma posição mais precisa do veículo utilizando como entrada os resultados obtidos nas duas abordagens.

A Figura 5.10 mostra a estimação da posição do veículo através das duas abordagens apresentadas e a estimativa da posição a partir do filtro de Kalman estendido que utiliza as estimativas das duas abordagens como entradas do filtro.

5.3.3 Planejamento

O planejamento de movimento (Figura 5.8) para navegação em ambientes externos foi também baseado em malhas de estados, assim como apresentado na Seção 5.2.3.

Assim como no Pioneer 3AT, os movimentos primitivos para veículo CaRINA 1 foram obtidos através de técnicas de integração numérica aplicadas ao modelo apresentado na Seção 3.2.

A partir do estado inicial $\mathbf{q} = (0, 0, 0, 0)$, da entrada de controle $\mathbf{u} = (v, \delta)$ e do tempo de integração Δt , estima-se o estado futuro $\mathbf{q}_{n+1}$.

A Figura 5.11 mostra o conjunto de movimentos primitivos obtidos usando a Equação 5.1 aplicado ao modelo cinemático do veículo (Equação 3.14) para um tempo $t = 3s$. Esse conjunto é composto de 14 ações (7 ações para frente e 7 ações para trás) para cada dos 24 ângulos de orientações discretizados⁷.

Figura 5.11: Primitivas de movimento CaRINA 1.

Para gerar o conjunto de ações mostradas na Figura 5.11 os valores discretizados das entradas de controle $\mathbf{u} = (v, \delta)$ utilizadas na integração numérica foram $v = [-1.0, -0.3, -0.1, 0.1, 0.3, 1.0]$ em m/s , $\delta = [-\pi/6, -\pi/8, -\pi/12, 0, \pi/12, \pi/8, \pi/6]$ para um tempo de integração $\Delta t = 0.01s$.

Os resultados do planejamento de movimento em ambientes externos serão mostrados no Capítulo 6.

⁷A Figura 5.11 mostra as ações possíveis de serem realizadas pelo veículo quando este encontra-se com orientação $\theta = 0^\circ$, as ações para os demais ângulos são obtidas pela rotação deste conjunto.

5.3.4 Controle

A etapa de controle (Figura 5.8) para navegação em ambientes externos foi baseada no controle mostrado na Seção 5.2.4, porém algumas adaptações foram feitas para adequar esse tipo de controle para a mudança de plataforma.

O CaRINA 1 é um veículo elétrico de tração traseira, neste sentido, uma contribuição importante foi apresentada em Rebai et al. (2007), que aplicou DWA a um veículo autônomo com tração nas rodas traseiras.

A janela dinâmica foi definida para a velocidade linear v e angular $\omega = \dot{\theta}$ do carro, semelhante ao Synchro-Drive (Fox et al., 1997). Porém, ω não é uma velocidade diretamente aplicável no carro, pois ela depende não linearmente do ângulo de esterçamento do volante δ e da velocidade v , segundo a Equação 5.3, extraída do modelo cinemático do carro (Equação 3.14).

$$\omega = \dot{\theta} = v \frac{\tan \delta}{L} \quad (5.3)$$

O ângulo de esterçamento é limitado por δ_{max} ⁸. Isto nos leva a seguinte equação:

$$v \frac{\tan(-\delta_{max})}{L} \leq \omega \leq v \frac{\tan(\delta_{max})}{L} \quad (5.4)$$

Esta relação, aplicada aos limites definidos pela Equação 2.1, retorna um espaço de velocidades V_s triangular⁹.

Assim a função objetivo da Equação 5.2, passa a ser representada como mostrada abaixo, onde a ação de controle fica sendo par (v, δ) .

$$O(v, \delta) = \alpha d_{path}(v, \delta) + \beta d_{goal}(v, \delta) + \gamma d_{obs}(v, \delta) \quad (5.5)$$

A janela de navegação utilizada na navegação em ambientes externos é mostrada na Figura

⁸Para o veículo CaRINA 1 tem-se $\delta_{max} = 30^\circ$.

⁹O espaço de velocidades V_s é triangular pelo fato de ω ser não-linearmente dependente de v e δ . como mostra a Equação 5.4.

Figura 5.12: Janela de navegação para o CaRINA 1.

5.12 e possui um tamanho de (10 metros X 10 metros) centrada no ponto médio do eixo traseiro do veículo.

CAPÍTULO 6

RESULTADOS

Este capítulo apresenta os principais resultados obtidos nesse trabalho. Os experimentos foram realizados em ambientes simulado e real em tarefas de navegação interna e externa.

Na seção 6.1 tem-se os resultados obtidos para o planejamento de movimento baseado em malhas de estados para o robô Pioneer 3AT, e na Seção 6.2 tem-se os resultados obtidos na tarefa de navegação autônoma do veículo CaRINA 1.

6.1 Planejamento e Navegação em Ambientes Internos

Como mencionado anteriormente, os resultados de planejamento de movimento e navegação em ambientes internos foram obtidos utilizando o Pioneer 3AT em ambientes simulado e real. No âmbito da navegação em ambientes internos, os principais problemas encontrados estão relacionados à passagens estreitas e desvio de obstáculos.

6.1.1 Ambiente Simulado

A simulação foi conduzida no ambiente **STAGE**¹ utilizando o mapa mostrado na Figura 5.4 como representação do mundo simulado. A visualização do ambiente simulado e de todas as ações do robô é feita através do **RVIZ**².

Os resultados obtidos na tarefa de navegação executada pelo Pioneer 3AT são mostrados na Figura 6.1. Nesta tarefa, o robô navega seguindo a trajetória planejada desviando dos obstáculos encontrados pelo caminho que não foram mapeados.

A Figura 6.1(a) mostra o mundo simulado pelo **STAGE**, onde estão indicados os obstáculos que não foram considerados durante a tarefa de mapeamento (ver Seção 5.2.1) assim como a meta a ser atingida. Para navegar do ponto inicial até a meta estabelecida (estrela azul), é preciso planejar uma trajetória livre de obstáculos. Na Figura 6.1(b) tem-se a visualização do ambiente feita pelo **RVIZ**. Essa figura mostra a trajetória que foi planejada usando as malhas de estados (ver Seção 5.2.3) e a inicialização do filtro de partículas para fazer a localização do robô (ver Seção 5.2.2). As partículas são espalhadas ao redor do robô devido a incerteza na sua posição inicial.

Quando a trajetória é planejada, o robô começa a receber as ações de controle (velocidades linear e angular) necessária para manter-se no plano traçado. Durante a navegação, o robô precisa desviar dos possíveis obstáculos que podem aparecer no caminho. Assim, quando um obstáculo não mapeado (Figura 6.1(c) e Figura 6.1(e)) é encontrado, o planejador gera um novo caminho livre da sua posição atual até a meta (Figura 6.1(d) e Figura 6.1(f)), de modo a desviar do obstáculo encontrado.

Na Figura 6.1(g) mostra o robô atravessando uma passagem estreita e na Figura 6.1(h) mostra o instante quando a meta é atingida. A linha em verde representa o caminho percorrido pelo robô do ponto inicial ao ponto de destino.

¹Simulador 2D integrado ao ambiente ROS. <http://www.ros.org/wiki/stage>

²Visualizador 2D/3D integrado ao ambiente ROS. <http://www.ros.org/wiki/rviz>

Figura 6.1: Navegação em ambiente interno simulado. (a) Mundo simulado **STAGE**. (b) Planejamento de trajetória. (c) Detecção de obstáculo (I). (d) Replanejamento (I). (e) Detecção de Obstáculo (II). (f) Replanejamento (II). (g) Passagem estreita. (h) Meta atingida.

6.1.2 Ambiente Real

Os experimentos reais foram conduzidos no andar térreo do Laboratório de Sistemas Inteligentes (LASI/EESC-USP). Nesses experimentos, a plataforma de trabalho recebe comandos de velocidades (linear e angular) via rede TCP/IP entre a plataforma e um computador base, ambos utilizando sistema **ROS**.

Os resultados experimentais são apresentados na Figura 6.2, que mostra o planejamento de movimento e navegação da posição inicial robô para a posição de destino (estrela azul). O principal objetivo é validar os resultados obtidos na simulação (Figura 6.1), ou seja, fazer o robô navegar pelo ambiente até a posição de destino desviando de possíveis obstáculos que possam ser encontrados.

Na Figura 6.2(a) temos o plano inicial traçado do ponto inicial do robô para o ponto de destino (estrela azul). O mapa do ambiente é visualizado pelo **RVIZ**, e no canto inferior direito tem-se a visualização do robô real no momento de execução do caminho planejado.

Na Figura 6.2(b) temos uma passagem estreita (porta da sala), no canto inferior direito tem-se o instante em que o robô passa pela porta que possui 90cm de largura, visto que o robô possui 49,3cm de largura (ver Tabela 3.1). Quando algum obstáculo é encontrado, o planejador deve replanejar sua trajetória a fim de evitar desse obstáculo. A Figura 6.2(c) mostra o instante em que um obstáculo não mapeado é encontrado (caixa de papelão). O planejador considera o replanejamento quando o obstáculo se encontra a 1 metro³ de sua posição atual. Quando o robô percebe que não é possível seguir o caminho, gera um novo caminho a partir desse ponto até a posição de destino (Figura 6.2(d)). A linha em verde representa o caminho real percorrido pelo robô Pioneer 3AT.

³A janela de navegação usando para ambiente internos tem (2 m X 2 m), assim as observações são incluídas quando estas se encontram a 1 metro à frente do robô.

Figura 6.2: Navegação em ambiente interno real. (a) Planejamento de trajetória. (b) Passagem estreita. (c) Detecção de obstáculo. (d) Replanejamento. (e) Desvio de obstáculo. (f) Meta atingida.

6.2 Planejamento e Navegação em Ambientes Externos

Como mencionado anteriormente, os resultados para o planejamento de movimento e navegação em ambientes externos foram obtidos utilizando o veículo CaRINA 1 em ambientes simulado e real.

Quando um plano global é fornecido pelo sistema de planejamento, o veículo precisa atingir a meta estabelecida evitando todos os obstáculos em seu caminho de forma segura. Para navegação autônoma, o veículo deve perceber o ambiente no qual está inserido e interagir com ele.

6.2.1 Ambiente Simulado

Nos testes realizados em ambiente simulado, o veículo não possui nenhuma outra informação sobre o ambiente, além daquelas informações inicialmente obtidas através das observações contidas na janela de navegação (ver Figura 5.12). A medida que o veículo se desloca, novas observações do ambiente são incluídas na janela, e com essas observações, é possível criar uma representação em grade de ocupação (Occupancy Grid) do ambiente usando o GMAPPING.

Como mostrado na Seção 5.3.1, o GMAPPING foi adaptado para considerar um ambiente totalmente navegável no início do planejamento de trajetória. Essa modificação foi necessária para que o sistema pudesse receber metas que estão fora da janela de navegação do veículo.

O ambiente simulado foi mostrado na Seção 5.3.1, este ambiente é uma representação tridimensional de uma área de estacionamento contendo vários outros veículos criada no **GAZEBO**⁴ para realizar as tarefas de navegação e planejamento de movimento.

Três testes foram realizados nesse ambiente simulado: (I) Estacionamento Perpendicular, (II) Estacionamento Diagonal e (III) Estacionamento Paralelo. Em todos os testes o veículo navega pelo ambiente desviando dos obstáculos e termina sua ação executando a tarefa de estacionamento.

⁴Simulador 3D integrado ao ambiente ROS. <http://www.ros.org/wiki/gazebo>

Estacionamento Perpendicular

Os resultados obtidos na tarefa de estacionamento perpendicular podem ser vistos na Figura 6.3. Na Figura 6.3(a) tem-se a indicação da posição inicial do veículo no ambiente simulado. O objetivo deste teste é traçar uma trajetória livre de obstáculos que conduza o veículo até a vaga indicada (estrela azul). O plano inicial planejado pelo sistema pode ser visto na Figura 6.3(b). Observa-se que o plano inicial foi obtido a partir das observações da janela de navegação no instante de planejamento, visto que esse caminho inicial já considera os obstáculos que estão dentro da janela.

Durante a navegação, o veículo vai recebendo mais informações⁵ sobre o ambiente a medida que novas observações são incluídas na janela. A Figura 6.3(c) mostra o instante que o veículo percebe que o plano inicial traçado está em rota de colisão com um obstáculo. Quando este obstáculo é detectado um novo plano é traçado (Figura 6.3(d)) para desviar desse obstáculo. A Figura 6.3(e) mostra o veículo seguindo a nova trajetória planejada e a navegação termina quando o robô entra na vaga perpendicular (Figura 6.3(f)).

Estacionamento Diagonal

No teste de estacionamento diagonal (45°) o veículo precisa percorrer um grande distância até atingir a vaga. Como o veículo não possui outras informações sobre o ambiente além daquelas obtidas na janela de navegação, planejar grandes trajetórias sem muitas informações do ambiente pode representar uma tarefa bastante difícil.

Os resultados deste teste podem ser vistos na Figura 6.4. A Figura 6.4(a) mostra o posicionamento inicial do veículo no mundo simulado onde o veículo precisa alcançar a vaga indicada (estrela azul) para fazer o estacionamento diagonal.

A Figura 6.4(b) mostra a trajetória inicial planejada. Durante a navegação, obstáculos são encontrados como mostra as Figuras 6.4(c) e 6.4(e). Para desviar desses obstáculos o planejador planeja novas trajetórias para o veículo como podem ser visto nas Figuras 6.4(d) e 6.4(f), a navegação termina quando o veículo entra na vaga diagonal (Figura 6.4(b)).

⁵As observações que são incluídas na janela são utilizadas para criar o mapa do ambiente.

Figura 6.3: Tarefa de estacionamento perpendicular. (a) Representação inicial da posição inicial do veículo no GAZEBO. (b) Planejamento de movimento. (c) Detecção de obstáculo. (d) Replanejamento. (e) Seguimento de trajetória (f) Estacionamento

Figura 6.4: Tarefa de estacionamento diagonal. a) Representação inicial da posição inicial do veículo no GAZEBO. (b) Planejamento de movimento. (c) Detecção de obstáculo (I). (d) Replanejamento (I). (e) Detecção de Obstáculo (II). (f) Replanejamento (II). (g) Estacionamento na vaga.

Estacionamento Paralelo

No teste de estacionamento paralelo, o veículo precisa navegar pelo ambiente e estacionar-se paralelamente em relação a outro veículo do próprio ambiente simulado. Diferente dos testes anteriores, este teste foi realizado de modo que as manobras executadas foram realizadas com velocidade negativas, o significa dizer que o veículo se desloca para trás (marcha ré).

Na Figura 6.5 tem-se os resultados obtidos nesse teste. A posição inicial do veículo no mundo simulado é mostrada na Figura 6.5(a). A ideia deste teste foi validar o sistema de controle fazendo o veículo navegar pelo ambiente em marcha ré ate atingir a meta estabelecida (estrela azul). A trajetória inicial planejada pode ser vista na Figura 6.5(b). O veículo segue essa trajetória planejada até o instante que um obstáculo é encontrado (Figura 6.5(c)). Quando isso acontece o sistema replaneja uma nova trajetória para que o veículo possa desviar desse obstáculo (Figura 6.5(d)) e atingir a meta estabelecida, como visto na Figura 6.5(e).

6.2.2 Ambiente Real

O sistema de planeamento também foi testado num ambiente real onde o veículo precisou navegar pelo ambiente evitando os obstáculos que foram encontrados ao longo do caminho(Figura 6.6). O lado esquerdo de cada figura, mostra o veículo real no momento de execução da tarefa e do lado direito mostra os caminhos planejados. Neste teste o condutor foi responsável pela aceleração, devido ao fato do veículo ainda não possuir um controle de velocidade.

O ponto de destino é representado pela estrela azul na Figura 6.6. O plano inicial é mostrado na Figura 6.6(a), Embora a aceleração seja sendo feita manualmente, tentou-se manter o mais constante possível. Ao longo do caminho planejado, o sistema de percepção detecta obstáculos (Cones). Esses obstáculos estão representados em vermelho e indicados pela letra **(A)** na (Figura 6.6(b)). Com a detecção desse obstáculo, o planejador rapidamente encontra uma nova rota para evitar esses obstáculos indicada pela letra **(B)** na Figura 6.6(c). Na Figura 6.6(d) temos um pequeno desvio da trajetória planejada. Nesse momento o sistema de controle envia comando para o veículo seguir a trajetória. Finalmente, a Figura 6.6(e) mostra o veículo chegando ao destino, completando assim a sua tarefa.

Figura 6.5: Manobra com marcha Ré. (a) Representação inicial da posição inicial do veículo no GAZEBO. (b) Planejamento de movimento. (c) Detecção de obstáculo. (d) Replanejamento. (e) Meta atingida.

(a)

(b)

(c)

(d)

(e)

Figura 6.6: Navegação com o veículo CaRINA 1 em ambiente real. (a)Planejamento de movimento. (b) Detecção de obstáculos. (c) Replanejamento. (d) Desvio de obstáculo. (e) Meta atingida.

CAPÍTULO 7

CONCLUSÃO

Este trabalho apresenta a utilização de malhas de estados como método de planejamento de movimento cinemático-dinâmico. Este método possui a capacidade de incorporar as restrições cinemáticas de robôs móveis, bem como suas característica dinâmicas no momento do planejamento de movimento. Outra característica importante que levou a escolha das malhas de estado, foi o fato de gerar soluções rapidamente através de uma heurística inflacionada e melhorar essas soluções quando ainda tiver tempo disponível. Este fato é importante porque em sistemas reais, os robôs precisam tomar decisões rapidamente para garantir a segurança na execução de determinadas tarefas. Um exemplo disto é a tarefa de desvio de obstáculos.

A utilização de um sistema altamente integrado, como o **ROS** foi de grande valia no desenvolvimento das etapas deste trabalho, pois se trata de uma ferramenta que oferece grande suporte no desenvolvimento e aplicações robóticas, e vem sendo cada vez mais utilizada em diversos trabalhos na área.

Este trabalho foi inicialmente desenvolvido para a plataforma Pioneer 3AT em tarefas de navegação em ambientes internos com desvio de obstáculos. Devido aos bons resultados obtidos com essa plataforma, resolve-se estender as ideias aplicadas ao Pioneer 3AT para um veículo autônomo CaRINA desenvolvido no Instituto de Ciências Matemáticas e Computação

ICMC/USP, para tarefa de navegação em ambientes externos semi-estruturados.

As principais mudanças para adaptação da nova plataforma foram realizadas na etapa de controle DWA (Dynamic Window Approach). Devido as equações que descrevem a cinemática e dinâmica das plataformas serem diferentes, isso modificou a maneira de como a janela dinâmica foi calculada. A velocidade angular do veículo está relacionada não-linearmente como a velocidade linear e o ângulo de esterçamento da rodas, e não representa uma entrada de controle diretamente aplicável ao veículo, isso cria um espaço de velocidade triangular ao invés de quadrado como no Pioneer 3AT.

Os movimentos primitivos, usados para compor as malhas de estado, foram gerados a partir da integração numérica de ambos os modelos dos veículos para entradas de controle discretizadas. Observou-se que para um conjunto composto por uma grande quantidade de movimentos primitivos, a demanda de tempo necessária para a criação das malhas também era elevada. Embora nenhum estudo tenha sido realizado para avaliar a quantidade de primitivas que demandasse o menor tempo para a construção da malha, a escolha dessa quantidade foi baseada em observações puramente experimentais. Uma vez que a navegação foi iniciada, é desejável que o robô tome decisões rapidamente para que a navegação ocorra de maneira segura. O efeito de um número elevado de primitivas foi bem evidenciado quando o robô precisou replanejar sua trajetória para desviar dos obstáculos. Com um numero elevado de primitiva o tempo de replanejamento também foi elevado. Com isso, o numero de primitivas de movimento foi escolhido de forma a atender as necessidades estabelecidas de modo que o tempo gasto para a criação das malhas fosse aceitável.

Os resultado obtidos na navegação em ambientes internos e externos validam a utilização do método proposto. O projeto CaRINA é um projeto em pleno desenvolvimento no Laboratório de Robótica Móvel (LRM-ICMC/USP), este trabalho contribuiu no desenvolvimento dessa plataforma.

7.1 Publicações Resultantes deste Trabalho

7.1.1 Aceitos

- André C. Magalhães, Marcos G. Prado, Valdir Grassi Jr and Denis F. Wolf; **Autonomous Vehicle Navigation in Semi-Structured Urban Environment**, 2013, IFAC Intelligent Autonomous Vehicles Symposium.

7.1.2 Submetidos

- Leandro C. Fernandes, Jefferson R. Souza, Gustavo Pessin, Patrick Y. Shinzato, Daniel Sales, Caio Mendes, Marcos Prado, Rafael Klaser, André C. Magalhães, Daniel Pigatto Kalinka C. Branco, Valdir Grassi Jr., Fernando S. Osório and Denis F. Wolf; **CaRINA Intelligent Robotic Car: Architectural Design and Implementations**, 2013, Journal of Systems Architecture.

- Marcos G. Prado, André C. Magalhães, Denis F. Wolf and Valdir Grassi Jr; **Detecção de Vagas e Estacionamento Autônomo de Veículos**, 2013, XI Simpósio Brasileiro de Automação Inteligente (SBAI).

7.2 Trabalhos Futuros

Uma abordagem que também foi estudada foi a criação de malhas de estados que consideram além da posição e orientação do robô, a velocidade (Likhachev e Ferguson, 2009) ou raio de curvatura (Pivtoraiko et al., 2009). Outra abordagem também estudada foi a criação de malhas de múltiplas resoluções (Likhachev e Ferguson, 2009), as tarefas que precisam ser tomadas com uma maior precisão dos movimentos podem ser tomadas com a malha de alta resolução enquanto que para tarefas mais simples as malhas de baixa resolução se aplicam melhor.

Essas duas abordagens apresentadas podem ser futuramente integradas ao sistema proposto neste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

- Brock, O. e O. Khatib (1999). **High-Speed Navigation Using the Global Dynamic Window Approach.** In *IEEE International Conference on Robotics & Automation (ICRA)*, Volume 1, Detroit, Michigan, pp. 341–346.
- Cao, Y. U., A. S. Fukunaga, e A. Kahng (1997). **Cooperative Mobile Robotics: Antecedents and Directions.** *Autonomous Robots* 4, 7–27.
- Caracciolo, L., A. D. Luca, e S. Iannitti (1999). **Trajectory Tracking Control of a Four-Wheel Differentially Driven Mobile Robot.** In *IEEE International Conference on Robotics & Automation (ICRA)*, Volume 4, Detroit, MI, pp. 2632–2638.
- Censi, A. (2008). **An ICP Variant Using a Point-to-line Metric.** In *IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA.
- Choset, H. e J. W. Burdick (2000). **Sensor Based Exploration: The Hierarchical Generalized Voronoi Graph.** *International Journal of Robotics Research* 19, 96–125. MIT Press.
- Choset, H., K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, e S. Thrun (2005). **Principles of Robot Motion: Theory, Algorithms, and Implementations.** Press MIT.
- Conner, D. C., A. A. Rizzi, e H. Choset (2003). **Composition of Local Potential Functi-**

- ons for Global Robot Control and Navigation.** In *IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, Volume 3, Las Vegas, Nevada, pp. 3546–3551.
- De Luca, A., G. Oriolo, e C. Samsom (1998). *Robot Motion Planning and Control*, Chapter Feedback Control of a Nonholonomic Car-like Robot, pp. 277–342. Springer.
- Durrant-Whyte, H. e T. Bailey (2006). **Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms.** *IEEE Robotics and Automation Magazine* 13, 99–110.
- Esposito, J. e V. Kumar (2002). **A method for modifying closed loop motion plans to satisfy unpredictable dynamics constraints at run time.** In *IEEE International Conference on Robotics & Automation (ICRA)*, Volume 2, Washington, DC, pp. 1691–1696.
- Ferguson, D., T. M. Howard, e M. Likhachev (2008). **Motion Planning in Urban Environments.** *Journal of Field Robotics* 25 (11-12), 939–960.
- Fernandes, L. C., J. R. Souza, P. Y. Shinzato, G. Pessin, C. C. T. Mendes, F. S. Osório, e D. F. Wolf (2012). **Intelligent Robotic Car for Autonomous Navigation: Platform and System Architecture.** In *IEEE II Brazilian Conference on Critical Embedded Systems (CBSEC)*, Campinas, SP, pp. 12–17.
- Fox, D., W. Burgard, F. Dellaert, e S. Thrun (1999). **Monte Carlo Localization: Efficient Position Estimation for Mobile Robots.** In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI)*., Orlando, Florida, pp. 343–349.
- Fox, D., W. Burgard, e S. Thrun (1997). **The Dynamic Window Approach to Collision Avoidance.** *Robotics Automation Magazine IEEE* 4(1), 23–33.
- Fox, D., W. Burgard, e S. Thrun (1999). **Markov Localization for Mobile Robots in Dynamic Environments.** *Journal of Artificial Intelligence Research* 11, 392–427.
- Gerkey, B. P., R. T. Vaughan, e A. Howard (2003). **The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor System.** In *Proc. of 11th International Conference on Advanced Robotics*, Comibra, Portugal, pp. 317–323.
- Grisetti, G., C. Stachniss, e W. Burgard (2005). **Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling.** In *Proc.*

-
- of the IEEE International Conference on Robotics and Automation (ICRA)*, Palau de Congressos de Catalunya, Barcelona, Spain, pp. 2432–2437.
- Hata, A. Y. (2010). **Maapeamento de Ambientes Externos Utilizando Robôs Móveis**. Dissertação de mestrado, Universidade de São Paulo. Instituto de Ciências Matemáticas e Computação - ICMC/USP.
- Hsu, J. e N. Koenig (2012). **Gazebo**. In *ROS Developer Conference (ROSCon)*, St. Paul, Minnesota.
- Kavraki, L. E., P. Svestka, J.-C. Latombe, e M. H. Overmars (1996). **Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces**. *IEEE Transaction on Robotics and Automation* 12, 566–580.
- Knepper, R. A. e A. Kelly (2006). **High Performance State Lattice Planning Using Heuristic Look-Up Tables**. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, Beijing, pp. 3375–3380.
- Kozlowski, K. e D. Pazderski (2004). **Modeling and Control of a 4-wheel Skid-Steering Mobile Robot**. *International Journal of Applied Mathematics and Computer Science* 14(4), 477–496.
- Kuipers, B. e Y. T. Byun (1991). **A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations**. *Journal of Robotics and Autonomous Systems* 8, 47–63.
- Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer, Boston.
- LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press.
- Likhachev, M. (2005). *Search-based Planning for Large Dynamic Environments*. Tese de Doutorado, School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213.
- Likhachev, M. (2009). *Search-Based Planning with Motion Primitives*. Carnegie Mellon University.
- Likhachev, M. e D. Ferguson (2009). **Planning Long Dynamically-Feasible Maneuvers for Autonomous Vehicles**. *The International Journal of Robotics Research* 28(8), 933–945.

- Likhachev, M., D. Ferguson, G. Gordon, A. Stentz, e S. Thrun (2008). **Anytime Search in Dynamic Graphs**. *Artificial Intelligence Journal (AIJ)* 172(14), 1613–1643.
- Likhachev, M., D. I. Ferguson, G. J. Gordon, A. Stentz, e S. Thrun (2005). **Anytime Dynamic A*: An Anytime, Replanning Algorithm**. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, Monterey, USA, pp. 262–271.
- Lima, D. A. d. (2010). **Navegação Segura de um Carro Autônomo Utilizando Campos Vetoriais e o Método da Janela Dinâmica**. Dissertação de Mestrado, Universidade Federal de Minas Gerais.
- Moravec, H. e A. E. Elfes (1985). **High Resolution Maps from Wide Angle Sonar**. In *IEEE International Conference on Robotics and Automation (ICRA)*, Volume 2, St. Louis, Missouri, pp. 166–121.
- Odakura, V. V. V. A. (2006). **Localização de Markov para multirrobo cooperativos**. Dissertação de Mestrado, Escola Politécnica, Universidade de São Paulo.
- Phillips, M. e M. Likhachev (2011). **SIPP: Safe Interval Path Planning for Dynamic Environments**. In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, pp. 5628–5635.
- Pires, A. G. e L. Chaimowicz (2012). **Localização Cooperativa, Descentralizada e Baseada em Marcos Dinâmicos em Enxames de Robôs Móveis**. In *Workshop on MSc Dissertation and PhD Thesis in Robotics*, Fortaleza, CE.
- Pivtoraiko, M. e A. Kelly (2005). **Generating Near Minimal Spanning Control Sets for Constrained Motion Planning in Discrete State Spaces**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Alberta, Canadá, pp. 3231 – 3237.
- Pivtoraiko, M., R. A. Knepper, e A. Kelly (2009). **Differentially Constrained Mobile Robot Motion Planning in State Lattices**. *Journal of Field Robotics* 26(3), 308–333.
- Prado, M. G. (2013). **Planejamento de Trajetória para Estacionamento de Veículos Autônomos**. Dissertação de mestrado, Instituto de Ciências Matemáticas e Computação, Universidade de São Paulo.

- Quigley, M., K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, e A. Y. Ng (2009). **ROS: An Open-Source Robot Operating System**. In *Proc. of ICRA Workshop on Open Source Software*, Kobe, Japão.
- Rebai, K. e O. Azouaoui (2009). **Bi-steerable Robot Navigation Using a Modified Dynamic Window Approach**. In *IEEE International Conference on Robotics and Automation (ICRA)*, Sharjah, pp. 1–6.
- Rebai, K., O. Azouaoui, M. Benmami, e A. Larabi (2007). **Car-like Robot Navigation at High Speed**. In *IEEE International Conference on Robotics and Biomimetics*, Sanya, China, pp. 2053–2057.
- ROBOTS, M. (2006). **Pioneer 3: Operations Manual**. Mobile Robots. Disponível em: http://www.ist.tugraz.at/_attach/Publish/Kmr06/pioneer-robot.pdf
Acessado em: 10/12/2012.
- ROS (2013a). **SBPL**. <http://www.ros.org/wiki/sbpl> Acessado em: 20/02/2013.
- ROS (2013b). **SBPL Lattice Planner**. http://www.ros.org/wiki/sbpl_lattice_planner Acessado em: 29/01/2013.
- Scholz, J., S. Chitta, B. Marthi, e M. Likhachev (2011). **Cart Pushing with a Mobile Manipulation System: Towards Navigation with Moveable Objects**. In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, pp. 6115–6120.
- Siegwart, R., I. R. Nourbakhsh, e D. Scaramuzza (2004). **Introduction to Autonomous Mobile Robots** (2 ed.). Scituate, MA, USA: Bradford Company.
- Tanner, H. G., S. Loizou, e K. J. Kyriakopoulos (2001). **Nonholonomic Stabilization with Collision Avoidance for Mobile Robots**. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Volume 3, Maui, HI, pp. 1220–1225.
- Thrun, S., W. Burgard, e D. Fox (2005). **Probabilistic Robotics**. Press MIT.
- Thrun, S., M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen,

P. Alessandrini, G. Bradski, B. D. S. Ettinger, A. Kaehler, A. Nefian, e P. Mahone (2006). **Stanley: The Robot that Won the DARPA Grand Challenge**. *Journal of Field Robotics* 23(9), 661–692.

Vaz, D. A. B. O. (2011). **Planejamento de Movimento Cinemático-Dinâmico para Robôs Móveis com Rodas Deslizantes**. Dissertação (mestrado em sistemas dinâmicos), Escola de Engenharia de São Carlos, Universidade de São Paulo. Disponível em: <http://www.teses.usp.br/teses/disponiveis/18/18153/tde-02012012-144022/>. Acesso em: 03/02/2013.

Vaz, D. A. B. O., R. S. Inoue, e V. Grassi (2010). **Kynodynamic Motion Planning of a Skid-Steering Mobile Robot Using RRTs**. In *Proceedings of Latin American Robotics Symposium and Intelligent Robotic Meeting (LARS)*, Sao Bernardo do Campo, Brazil.

Wolf, D. F., E. V. Simões, F. S. Osório, e O. Trindade Jr (2009). **Robótica Inteligente: Da Simulação às Aplicações no Mundo Real**. In *Jornadas de Atualização em Informática - JAI (SBC JAI)*, pp. 1–51.