
Planejamento de trajetória para estacionamento de veículos autônomos

Marcos Gomes Prado

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Planejamento de trajetória para estacionamento de veículos autônomos

Marcos Gomes Prado

Orientador: Prof. Dr. Denis Fernando Wolf

Dissertação apresentada ao Instituto de Ciências
Matemáticas e de Computação - ICMC-USP, como
parte dos requisitos para obtenção do título de Mestre
em Ciências - Ciências de Computação e Matemática
Computacional. *VERSÃO REVISADA*

USP – São Carlos
Abril de 2013

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

P896p Prado, Marcos Gomes
Planejamento de Trajetória para Estacionamento de
Veículos Autônomos / Marcos Gomes Prado; orientador
Denis Fernando Wolf. -- São Carlos, 2013.
62 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2013.

1. Planejamento de Trajetória. 2. Veículos
Autônomos. 3. Estacionamento de Veículo Autônomo. I.
Fernando Wolf, Denis, orient. II. Título.

À Deus e aos meus pais.

”Não sejas sábio a teus próprios olhos, teme ao SENHOR e aparta-te do mal. Isto será saúde para o teu âmago, e medula para os teus ossos.”

(Provérbios 3:7-8)

Agradecimentos

Em primeiro lugar, agradeço a Deus, aos meus pais e irmã pelo amor, apoio e dedicação durante todos os momentos.

Agradeço ao meu orientador Denis Fernando Wolf pela compreensão, apoio, orientação e por ter confiado em mim o desenvolvimento dessa dissertação.

Aos amigos do Laboratório de Robótica Móvel pela companhia em todos os momentos, ajuda, motivação e força nos estudos.

Agradeço à todos os amigos e familiares que de alguma maneira contribuíram para a conclusão deste trabalho.

Agradeço ao apoio financeiro da FAPESP.

Resumo

A navegação autônoma é um dos problemas fundamentais na área de robótica móvel. Esse problema vem sendo pesquisado nessa área por décadas e ainda apresenta um grande potencial para pesquisas científicas. A maior parte dos algoritmos e soluções desenvolvidas nessa área foi concebida para que robôs operem em ambientes estruturados. No entanto, outra questão de grande interesse para pesquisadores da área é a navegação em ambientes externos. Em ambientes não estruturados os veículos autônomos (robôs de grande porte) devem ser capazes de desviar de obstáculos, que eventualmente apareçam no caminho. Esta dissertação aborda o desenvolvimento de um sistema inteligente capaz de gerar e executar um planejamento de caminho para o estacionamento de veículos autônomos em ambientes semi-estruturados. O sistema é capaz de reconhecer vagas de estacionamento por meio de sensores instalados no veículo, gerar uma trajetória válida que o conduza até a vaga e enviar os comandos de esterçamento e aceleração que guiam o veículo pelo caminho gerado.

Palavras-chave: Planejamento de Trajetória, Veículos Autônomos, Estacionamento de Veículo Autônomo e Planejamento baseado em busca.

Abstract

Autonomous navigation is one of the fundamental problems in mobile robotics. This problem has been addressed for decades and still has great potential for scientific research. Most solutions and algorithms developed in this field is designed for robots that operate in structured environments. However, another issue of great interest to researchers in this area is autonomous navigation in outdoor environments. In partially structured environments autonomous vehicles (large robots) must be able to avoid obstacles that may arise along the way. This dissertation addresses the development of an intelligent system able to generate and run a path planning for parking of autonomous vehicles in semi-structured environments. The system is able to recognize parking lots using sensors installed in the vehicle, generate a valid path that leads up to the parking lot and send the steering commands and acceleration that to guide the vehicle to its goal point.

Keywords: Path planning, Autonomous vehicles, Parking of autonomous vehicles and Search-based planning.

Sumário

Agradecimentos	ix
Resumo	xi
Abstract	xiii
Lista de Figuras	xvii
Lista de Tabelas	xxi
1 Introdução	1
1.1 Motivação	1
1.2 Objetivos	2
1.2.1 Objetivo geral	2
1.2.2 Objetivos específicos	2
1.3 Justificativa	3
1.4 Organização da dissertação	4
2 Trabalhos Relacionados	5
2.1 Trabalhos em ambiente simulado	5
2.1.1 SEVA3D	5
2.1.2 Parking Simulator 2D	6
2.1.3 Estacionamento de um veículo autônomo baseado em RRT	8
2.2 Trabalhos em ambiente real	9
2.2.1 Driving4U	9
2.2.2 Boss	10
2.2.3 Junior 3 - <i>Autonomous Valet Parking</i>	11

2.3	Considerações finais	13
3	Fundamentação Teórica	15
3.1	Robótica móvel	15
3.2	Localização	16
3.3	Mapeamento	19
3.3.1	Grades de ocupação	20
3.4	Navegação	21
3.4.1	Planejamento baseado em algoritmos de busca	22
3.5	Modelo cinemático Ackerman	26
3.6	Considerações finais	27
4	Implementação	29
4.1	Sistema proposto	29
4.2	Ferramentas e Bibliotecas	30
4.2.1	ROS	30
4.2.2	Gmapping e Costmap_2D	31
4.2.3	SBPL	32
4.3	Modelagem do veículo	32
4.4	Fusão de dados dos sensores	33
4.5	Localização	34
4.6	Detecção de Vagas de Estacionamento	36
4.7	Planejamento	39
4.8	Controle	40
4.9	Considerações finais	42
5	Resultados	43
5.1	Estacionamento diagonal	43
5.2	Estacionamento perpendicular	44
5.3	Estacionamento paralelo	44
5.4	Definição de manobras válidas	48
5.5	Resultados simulados	48
5.6	Resultados preliminares em ambiente real	49
5.6.1	Planejamento com desvio de obstáculos	50
5.6.2	Estacionamento em vaga perpendicular	50
5.6.3	Estacionamento em vaga paralela	50
5.7	Considerações finais	52
6	Conclusão	55
6.1	Trabalhos futuros	56
Referências Bibliográficas		57

Lista de Figuras

1.1	Veículos do projeto CaRINA. (a) Veículo Elétrico. (b) Palio Adventure.	4
2.1	Localização dos sensores (Heinen <i>et al.</i> , 2005).	6
2.2	Ambiente Modelado (Heinen <i>et al.</i> , 2005).	7
2.3	Trajetória de Estacionamento Paralelo (Andrade, 2011).	7
2.4	Estágios de um planejamento de trajetória utilizando RRT bidirecional (LaValle e Kuffner, 2001).	8
2.5	Trajetórias para estacionamento de veículos autônomos (Han <i>et al.</i> , 2011). . . .	9
2.6	Veículo Inteligente - Projeto <i>Driving4u</i> (Honório <i>et al.</i> , 2010).	10
2.7	Variáveis consideradas no problema de estacionamento paralelo (Honório <i>et al.</i> , 2010).	10
2.8	Veículo Boss (Buehler <i>et al.</i> , 2009).	11
2.9	Junior 3 (Jeevan <i>et al.</i> , 2010).	12
2.10	Trajetória de estacionamento perpendicular do veículo Junior 3 (Jeevan <i>et al.</i> , 2010).	13
3.1	Exemplo do funcionamento da localização pelo método de Monte Carlo. (a) Conjunto de setas indicando as partículas do filtro espalhadas pelo mapa. (b) e (c) Concentração das partículas em área com maior probabilidade da real posição do robô. (d) Ilustra uma estimativa da posição do robô em uma única região.	18
3.2	Tipos de Mapa. (a) Mapa métrico. (b) Mapa topológico (Adaptada de (Beevers, 2004)).	20
3.3	Exemplo de mapeamento em grades de ocupação.	21
3.4	(a) Exemplo de movimentos primitivos e (b) Exemplo de uma malha de estados (ou <i>state lattice</i>) (Adaptada de (Likhachev e Ferguson, 2009)).	23

3.5	Algoritmo AD* (Likhachev <i>et al.</i> (2005)). (a) Função para gerar o melhor caminho. (b) Função principal.	25
3.6	Modelo Cinemático de Ackerman.	26
4.1	Modelo do sistema de estacionamento autônomo proposto.	30
4.2	Ambiente de simulação do ROS. (a) Simulador Gazebo. (b) Visualizador de ambientes 3D do ROS.	31
4.3	Modelo simulado do veículo. (a) Visão tridimensional do carro. (b) Representação cinemática em uma estrutura de árvore (Modelo URDF). (c) Veículo real.	33
4.4	Sensores lasers. (a) Hokuyo UTM-30LX. (b) Sick LMS 291.	34
4.5	Visão superior do veículo com a localização e representação dos sensores laser. (a) Localização dos três sensores laser. (b) Representação do laser virtual.	35
4.6	Visão de 360° dos sensores laser. Pontos vermelhos representam a detecção de obstáculos pelo laser virtual.	35
4.7	Diagrama das etapas do modelo de detecção de vagas.	37
4.8	Detecção de vagas de estacionamento. Os pontos em vermelho consistem nos obstáculos detectados pelo laser. Os retângulos em azul representam os espaços definidos como vagas de estacionamento. As setas indicam a posição e orientação de destino. (a) Vaga perpendicular. (b) Vaga paralela. (c) Vaga diagonal. .	38
4.9	Conjunto de movimentos primitivos. 14 ações de movimentos válidos para o veículo (7 movimentos para frente e 7 para traz).	40
4.10	Exemplo do planejamento de trajetória. (a) Trajetória gerada para manobra de estacionamento, ilustrada por uma linha preta. (b) Trajetória executada destacada em verde.	41
5.1	Sistema de estacionamento autônomo em ambiente simulado. (a) Mapeamento e detecção da vaga de estacionamento. (b) Planejamento de trajetória. (c) Execução da trajetória. (d) Finalização da manobra.	45
5.2	Sistema de estacionamento autônomo em ambiente simulado - Estacionamento perpendicular. (a) Mapeamento e detecção da vaga de estacionamento. (b) Planejamento de trajetória. (c) Execução da trajetória. (d) Finalização da manobra. .	46
5.3	Sistema de estacionamento autônomo em ambiente simulado - Estacionamento paralelo. (a) Mapeamento e detecção da vaga de estacionamento. (b) Planejamento de trajetória. (c) Execução da trajetória. (d) Finalização da manobra. .	47
5.4	Posicionamento do veículo em finalizações de manobras de estacionamento. (a) Veículo dentro da zona específica de estacionamento (manobra válida). (b) Veículo parcialmente dentro da zona de estacionamento (manobra inválida). .	48
5.5	Teste de planejamento de trajetória em ambiente real. (a) Trajetória em linha reta. Obstáculos destacados em vermelho e destino representado por um retângulo azul. (b) Replanejamento após detecção de obstáculos em rota de colisão. (c) Execução da nova trajetória. (d) Finalização da trajetória.	51

5.6	Teste de estacionamento em vaga perpendicular em ambiente ambiente real. (a) Busca por vaga perpendicular no lado direito do veículo. (b) Inicio da manobra de estacionamento. (c) Execução da trajetória encontrada. (d) Finalização da trajetória.	52
5.7	Teste de estacionamento em vaga paralela em ambiente ambiente real. (a) Busca por vaga paralela no lado direito do veículo. (b) Inicio da manobra de estacionamento. (c) Execução da trajetória encontrada. (d) Finalização da trajetória. . . .	53

Lista de Tabelas

4.1	Especificações do veículo	33
5.1	Validação das manobras de estacionamento	49

Introdução

Este capítulo apresenta a motivação desta dissertação, enfatiza a importância de seu desenvolvimento, aborda seus objetivos e apresenta uma visão sucinta dos demais capítulos.

1.1 Motivação

Segundo Thrun *et al.* (2005), a robótica é a ciência que percebe e manipula o mundo físico através de dispositivos computacionais. Para tal, são necessários mecanismos para perceber e atuar no ambiente. Sensores laser (*laser range finders*), câmeras de vídeo, GPS e unidades de medida inercial são exemplos de sensores que podem ser usados para a percepção do ambiente. Enquanto garras e rodas são atuadores utilizados pelo robô para interagir com o ambiente. Uma das áreas da robótica em destaque na última década é a robótica móvel autônoma, correspondendo ao grupo de sistemas robóticos com capacidade de locomoção e de operação de modo semi ou completamente autônomo (Carvalho e Kowaltowski, 2009).

A pesquisa em robótica móvel tem alcançado um progresso significativo nos últimos 20 anos. Parte dela se especializa em navegação autônoma, que é uma tarefa fundamental em robôs móveis (Thrun *et al.*, 2006). Nesses anos, diversas abordagens têm sido direcionadas para a navegação autônoma em ambientes externos, o que consiste em um problema interessante e complexo. Nestes ambientes uma das maiores preocupações é determinar as regiões navegáveis para que possa ser desempenhada uma navegação autônoma com segurança. Há algum tempo, a

tarefa dos robôs móveis tem sido estendida para veículos robóticos inteligentes. Uma aplicação dessa tecnologia é o desenvolvimento de sistemas de direção autônoma.

Os trabalhos relacionados aos veículos autônomos inteligentes têm crescido tanto no setor acadêmico como na indústria. Mapeamento de ambientes, localização, planejamento e controle são problemas complexos e totalmente ligados à pesquisa de veículos autônomos inteligentes. Diversos trabalhos, Markelic *et al.* (2011), Petrovskaya e Thrun (2009), Bonnifait *et al.* (2008) e Guizzo (2011), têm focado na navegação de veículos robóticos em ambientes urbanos. Competições tais como DARPA (2010a), DARPA (2010b) e ELROB (2011) têm impulsionado o estado da arte no controle de veículos autônomos.

Problemas que fizeram parte do desafio proposto pelo DARPA em 2007, como o problema de estacionar um veículo de forma autônoma, começam a chegar ao mercado. O sistema *park assist*, que pode ser encontrado em alguns veículos no mercado, auxilia o motorista na execução da manobra de estacionamento. Esse sistema, por meio de sensores localizados no veículo, é capaz de encontrar vagas paralelas e perpendiculares próximas e guiar o veículo até a mesma. Mas o controle de aceleração e troca de marchas ainda são da responsabilidade do condutor, sendo que a velocidade do carro deve permanecer abaixo de 7 km/h durante a manobra. O ambiente também é bastante estruturado. Para que uma vaga seja encontrada, outros veículos devem delimitar o espaço referente à mesma. Essas limitações garantem que novos trabalhos possam ser desenvolvidos para melhorar a resolução desse problema.

1.2 Objetivos

1.2.1 Objetivo geral

Desenvolvimento de um sistema que é capaz de planejar trajetórias viáveis de execução para veículos autônomos, tratando especificamente das manobras para estacionamento em vagas paralelas, perpendiculares e diagonais (45º).

1.2.2 Objetivos específicos

- Desenvolvimento de um sistema de percepção capaz de detectar obstáculos e identificar possíveis vagas de estacionamento
- Desenvolvimento de um sistema de mapeamento que construa um mapa 2D local do ambiente.

- Desenvolvimento de um sistema de planejamento de trajetória, tendo como base o mapa gerado pelos sensores, que seja capaz de guiar o veículo até uma determinada vaga encontrada.

1.3 Justificativa

O Departamento Nacional de Trânsito (DENATRAN) apresentou o número de veículos regularizados no país no final do ano de 2011 e a frota já ultrapassa 70 milhões de unidades, incluindo automóveis, caminhões, ônibus e outros automotores (DENATRAN, 2011). Em um intervalo de dez anos, o crescimento foi de aproximadamente 125% na frota do país. Entre alguns pontos negativos referentes a este crescimento pode ser citada a dificuldade de encontrar vagas livres para estacionamento, tanto públicas como privadas. Além disso, por muitas vezes existe a dificuldade de realização da manobra de estacionamento em determinadas vagas devido à dinâmica do ambiente. Espaço reduzido, alto fluxo de trânsito, baixa luminosidade e obstáculos dinâmicos podem trazer dificuldades ao condutor do veículo durante a realização da manobra citada.

Por meio de um sistema de estacionamento para veículos autônomos, que seja capaz de reconhecer vagas, planejar trajetórias válidas para a manobra e executar a mesma, pretende-se diminuir as dificuldades apresentadas referente à atividade de estacionar um veículo. Com isso, tempo e esforços serão economizados por parte do condutor, danos materiais e humanos podem diminuir, visto que os sensores impedirão que o veículo venha colidir com algum obstáculo. Segundo IBGE (2011), em 2009 a população brasileira com sessenta anos ou mais era de vinte e um milhões, com perspectiva de crescimento no índice percentual durante os próximos anos. O sistema proposto engloba o pensamento de aumento da mobilidade no país, um vez que pretende facilitar a realização de uma tarefa que muitas vezes se torna algo complexo para pessoas idosas.

Os trabalhos desenvolvidos neste mestrado fazem parte de um projeto maior que tem como objetivo a criação de veículos autônomos inteligentes para navegação em ambientes urbanos. O projeto CaRINA (Carro Robótico Inteligente para Navegação Autônoma) é administrado pelo Laboratório de Robótica Móvel (LRM) pertencente ao Instituto de Ciências Matemáticas e de Computação (ICMC), em parceria com o INCT-SEC (Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos). Atualmente, o projeto CaRINA dispõe de dois veículos para teste, sendo um veículo elétrico (Figura 1.1(a)) e um Palio Adventure 2011 (Figura 1.1(b)).



Figura 1.1: Veículos do projeto CaRINA. (a) Veículo Elétrico. (b) Palio Adventure.

1.4 Organização da dissertação

O restante do documento está organizado em cinco capítulos, cujos conteúdos se encontram estruturados abaixo:

- O Capítulo 2 apresenta trabalhos relacionados ao problema de estacionamento de veículos. Estão subdivididos em trabalhos simulados e em trabalhos com resultados em ambiente real.
- O Capítulo 3 descreve uma introdução à robótica móvel, apresentando os conceitos teóricos e aplicações dos sistemas robóticos, incluindo os algoritmos utilizados neste trabalho.
- O Capítulo 4 apresenta as ferramentas utilizadas e a implementação do sistema proposto.
- O Capítulo 5 apresenta os resultados alcançados deste trabalho, descrevendo os experimentos realizados.
- O Capítulo 6 como conclusão dessa dissertação, apresenta as contribuições do trabalho, além das perspectivas futuras.

Trabalhos Relacionados

Este capítulo faz o levantamento de alguns trabalhos relacionados e são apresentados divididos em dois grupos: trabalhos em ambiente simulado e em ambiente real.

2.1 Trabalhos em ambiente simulado

2.1.1 SEVA3D

O trabalho desenvolvido por Osório *et al.* (2001) apresenta o desenvolvimento do sistema SEVA (Simulador de Estacionamento de Veículos Autônomos), que é capaz de simular a tarefa de estacionamento de um veículo autônomo em uma vaga paralela. É utilizado um ambiente bidimensional (2D) para a simulação, onde o veículo é equipado com seis sensores do tipo infra-vermelho podendo ser controlado por meio de um autômato finito ou uma Rede Neural.

Devido às limitações de um ambiente 2D e presença de ruídos nos dados coletados por sensores infravermelhos serem praticamente nula, é proposto o desenvolvimento do sistema SEVA3D (Simulador de Estacionamento de Veículos Autônomos em um ambiente tridimensional) (Heinen *et al.*, 2005). Nessa nova abordagem é utilizado no veículo sensores do tipo Sonar (3D), localizados em pontos estratégicos (Figura 2.1), e a tarefa de estacionamento é controlada por um autômato finito baseado em regras.

As principais características do simulador SEVA3D são:

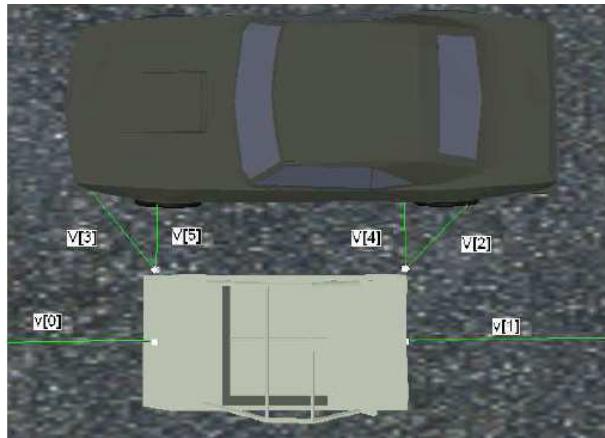


Figura 2.1: Localização dos sensores (Heinen *et al.*, 2005).

- Utiliza um ambiente tridimensional e sensores mais realistas, que simulam inclusive a presença de ruído;
- Realiza o estacionamento em uma vaga paralela independente da presença de outros carros estacionados (o sistema funciona mesmo não haja nenhum outro carro);
- É muito mais robusto do que o SEVA 2D em relação à distância entre o carro e o meio-fio, aceitando valores de 2 e 4 metros;
- Realiza o afastamento do veículo de forma automática se ele estiver muito próximo aos carros estacionados antes do início da manobra (distâncias menores que 30 cm em relação aos carros à direita);
- Permite que a manobra seja visualizada sob qualquer ângulo e posição do ambiente tridimensional.

A figura 2.2 ilustra a modelagem do ambiente utilizado para os testes e validação do sistema SEVA3D.

2.1.2 Parking Simulator 2D

Andrade (2011) apresenta um sistema inteligente de simulação para o estacionamento paralelo em veículos de passeio. O sistema desenvolvido é capaz de encontrar uma vaga partindo de uma determinada posição, planejar a trajetória necessária para a realização do estacionamento e controlar o veículo durante essa tarefa. A trajetória em S é adotada para a realização da manobra. Essa abordagem consiste em posicionar corretamente duas circunferências com um

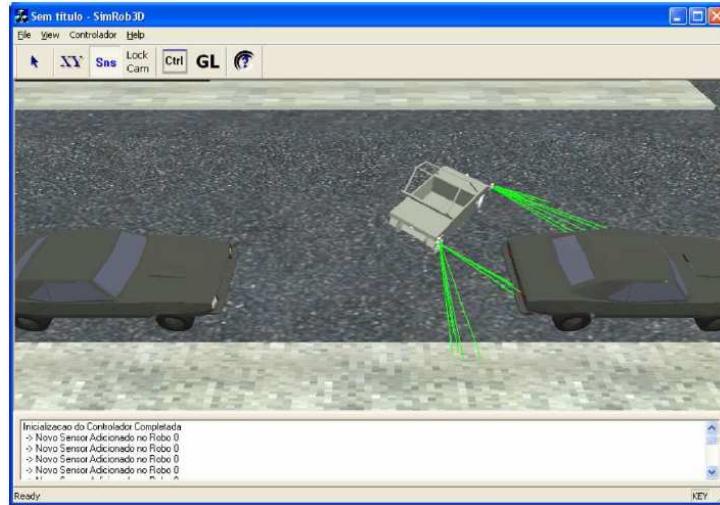


Figura 2.2: Ambiente Modelado (Heinen *et al.*, 2005).

raio de esterçamento do veículo, onde um caminho em forma de S será gerado por meio da união de dois arcos pertencentes às circunferências (Figura 2.3).

O controlador construído é baseado em aprendizado supervisionado utilizando Redes Neurais Artificiais (RNA). Os dados de entrada do controlador são provenientes de dois sensores do tipo laser, da odometria, de um sensor inercial, além da etapa da manobra em que o veículo se encontra. O mesmo foi treinado apenas para realização de manobras em vagas paralelas à direita.

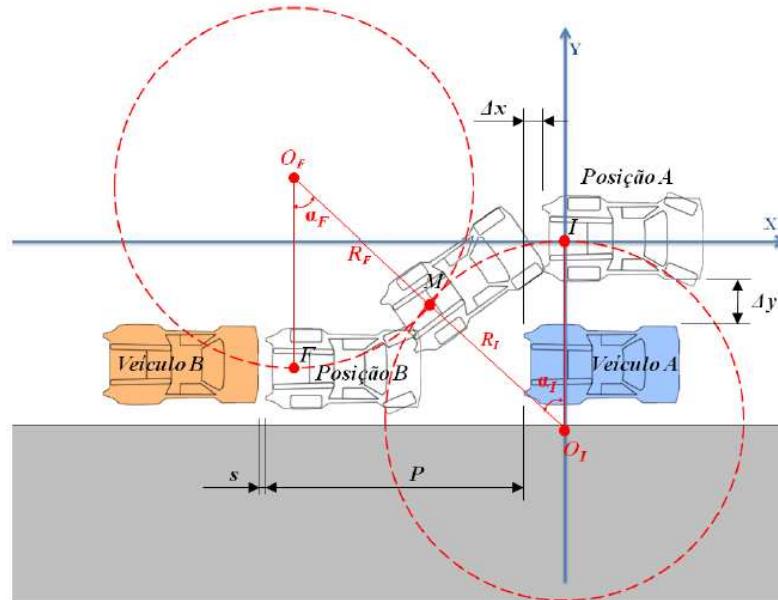


Figura 2.3: Trajetória de Estacionamento Paralelo (Andrade, 2011).

Os resultados obtidos por meio de simulação se mostraram muito satisfatórios, pois o veículo controlado por uma RNA conseguiu estacionar de forma correta em uma vaga paralela. Uma das propostas de trabalhos futuros é a validação em um veículo equipado com sensores do tipo laser e inercial.

2.1.3 Estacionamento de um veículo autônomo baseado em RRT

Em Han *et al.* (2011) é proposto um algoritmo baseado em árvores aleatórias de exploração rápida para a resolução do problema de estacionamentos de veículos autônomos. Diferente de outras abordagens onde o problema de estacionamento é dividido em três problemas distintos: paralelo, perpendicular e diagonal, Han propõem a resolução de forma unificada.

O planejador desenvolvido utiliza uma extensão de árvores aleatórias de exploração rápida (RRT) conhecido como RRT bidirecional (LaValle e Kuffner, 2001). Esse algoritmo faz uso de duas estruturas de árvores aleatórias para a geração de trajetórias. Diferente da abordagem clássica, onde apenas uma árvore é criada, são expandidas duas árvores: uma na região correspondente ao início da trajetória e a outra no objetivo. O caminho é encontrado quando as duas árvores se conectam no espaço de configuração, como pode ser visualizado na figura 2.4.

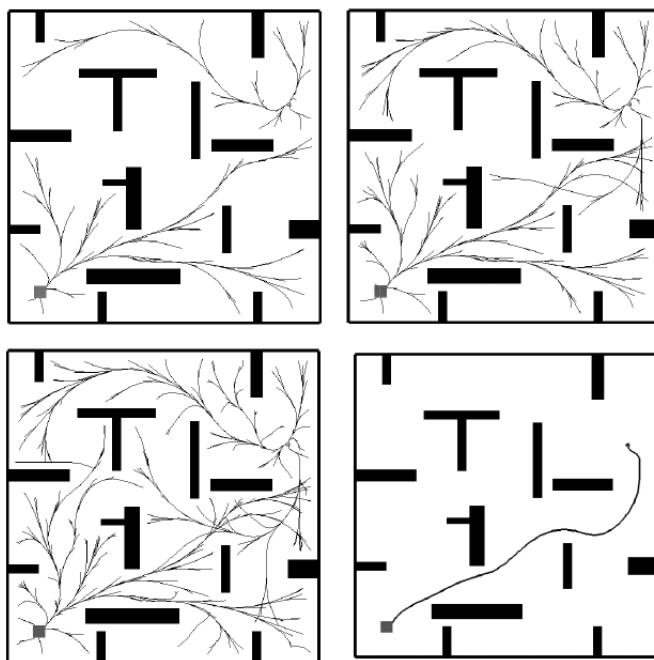


Figura 2.4: Estágios de um planejamento de trajetória utilizando RRT bidirecional (LaValle e Kuffner, 2001).

Um algoritmo de suavização é utilizado para retirar a redundância de pontos da trajetória retornada pelo RRT bidirecional. Sua implementação é uma modificação do algoritmo Ramer-Douglas-Peucker,

onde dada uma curva composta por segmentos de reta é encontrado uma curva similar com menos pontos (LaValle, 2006). O sistema desenvolvido foi testado apenas em ambiente simulado, mas os testes apresentaram êxito na resolução do problema nos três tipos de estacionamento: paralelo, perpendicular e diagonal (Figura 2.5).

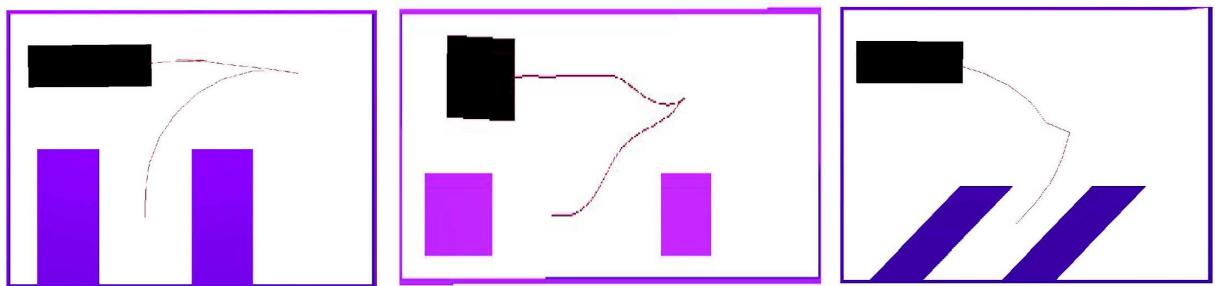


Figura 2.5: Trajetórias para estacionamento de veículos autônomos (Han *et al.*, 2011).

2.2 Trabalhos em ambiente real

2.2.1 Driving4U

O projeto *Driving4u*, de responsabilidade do Grupo de Automação e Tecnologia da Informação da UNIFEI (Universidade Federal de Itajubá), tem por objetivo a estruturação de uma plataforma de trabalho para pesquisa e desenvolvimento na área de Veículos Inteligentes. Essa plataforma é utilizada para validar a metodologia apresentada em Honório *et al.* (2010).

O trabalho apresenta uma metodologia baseada em lógica *fuzzy* aplicada em veículos autônomos. Essa metodologia tem o objetivo de possibilitar o aprendizado supervisionado de tarefas de navegação e realização de manobras em veículos inteligentes (Honório *et al.*, 2010). A proposta foi dividida em 3 fases. Inicialmente é criado um banco de dados contendo informações tanto do veículo como do ambiente em que o mesmo atua. A segunda fase consiste na indução de um sistema *fuzzy* a partir do banco de dados criado na fase anterior. Esse sistema leva em consideração as ações tomadas pelo motorista em diferentes situações durante a coleta de informação para a criação do bando de dados. A última fase é aplicação do Controlador *fuzzy* no veículo para a navegação.

Um dos testes foi o estacionamento em vaga paralela, realizado no veículo do projeto *Driving4u* (Figura 2.6), mostrando-se eficaz em ambientes reais.

O conjunto de regras para o controlador *fuzzy* foi extraído de um banco de dados contendo informações de manobras de estacionamento bem sucedidas. As variáveis de entrada para o



Figura 2.6: Veículo Inteligente - Projeto *Driving4u* (Honório *et al.*, 2010).

sistema foram: (θ) - ângulo do veículo em relação à vaga, X_A - distância do veículo até o limite da vaga e Y_D - distância do veículo até a guia (Figura 2.7).

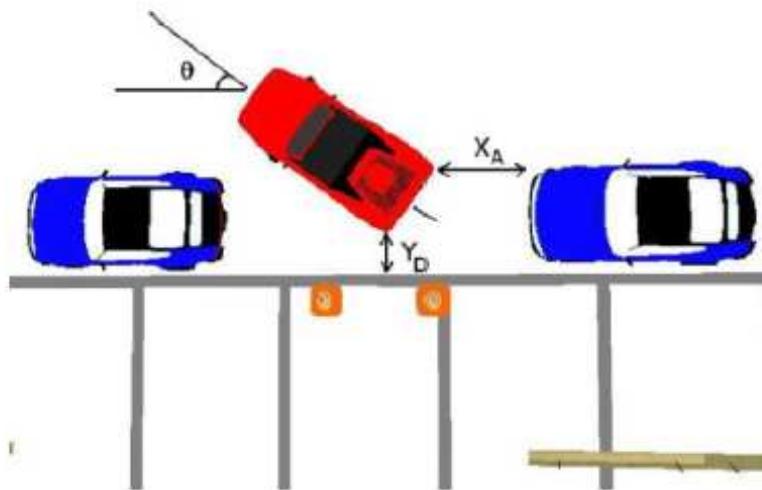


Figura 2.7: Variáveis consideradas no problema de estacionamento paralelo (Honório *et al.*, 2010).

2.2.2 Boss

O veículo autônomo denominado como Boss, tendo como principais desenvolvedores do projeto pesquisadores da universidade Carnegie Mellon, competiu e venceu o desafio proposto pelo

DARPA (*Defense Advanced Research Projects Agency*) em 2007 (Likhachev e Ferguson, 2009). Boss (Figura 2.8), dotado de sensores como: IMU (Unidade de medida inercial), laser, radar e câmera, foi capaz de percorrer 3000 quilômetros em ambiente urbano, incluindo o desafio do DARPA, onde problemas como navegar em vias com outros carros e estacionar eram algumas das tarefas.



Figura 2.8: Veículo Boss (Buehler *et al.*, 2009).

A abordagem para o planejamento de trajetória utilizada no projeto baseia-se na busca incremental dinâmica em uma malha de estados de espaço (ou *lattice state space*). Inicialmente o espaço de configuração do robô é discretizado por amostragem. Essa amostragem constitui na criação de uma malha contendo apenas movimentos válidos, segundo as restrições do veículo, dentro da área livre de colisão. Para a geração de um caminho entre dois pontos dentro dessa estrutura é usado um algoritmo de busca em grafo que é capaz de trabalhar em situações dinâmicas dentro de tempo predefinido. Utilizando essa abordagem o veículo foi capaz de estacionar em vagas perpendiculares.

2.2.3 Junior 3 - *Autonomous Valet Parking*

Junior 3 é o veículo autônomo desenvolvido pelo laboratório de pesquisas eletrônicas do grupo Volkswagen. Esse veículo baseia-se nos projetos Montemerlo *et al.* (2006) e Montemerlo *et al.* (2008), dando origem a terceira geração. Faz uso tanto do sistema de hardware como do módulo de software das plataformas anteriores, sendo umas das principais diferenças o sistema de sensoriamento, onde houve a diminuição dos sensores (Jeevan *et al.*, 2010). O projeto mais recente é adaptado com câmera, odômetro e dois lasers automotivos apenas (Figura 2.9).



Figura 2.9: Junior 3 (Jeevan *et al.*, 2010).

O trabalho de Jeevan *et al.* (2010) apresenta o desenvolvimento de um sistema inteligente de manobrista autônomo para grandes estacionamentos. O mapa do ambiente de atuação do veículo possui pontos de referência e é de conhecimento de sistema. Para a localização são utilizadas a odometria e a câmera para identificar os pontos de referência citados anteriormente, além de um filtro de Kalman. O detector de vagas é desenvolvido utilizando-se do conhecimento prévio do mapa em conjunto com os sensores lasers do veículo. O sistema é detalhado no trabalho (Stanek *et al.*, 2010).

O planejador de trajetória para o estacionamento perpendicular, diferente da abordagem utilizada na versão anterior baseada em um algoritmo de busca A* híbrido, faz uso de uma heurística bastante específica. Após a detecção de uma vaga livre o sistema constrói uma trajetória, que será realizada de ré pelo veículo, baseada em um conjunto de cinco segmentos. Esses segmentos são gerados levando em consideração o raio mínimo de curvatura do veículo, sua velocidade e a velocidade máxima de esterçamento. A trajetória é montada em forma de arco, possibilitando que a tarefa de estacionamento seja realizada com apenas uma manobra. Mas para que isso ocorra o veículo deve estar posicionado no ponto inicial da trajetória. Sendo assim, outra manobra é gerada por meio de uma função predefinida, que guia o carro até a posição onde dará início à manobra de ré, como é mostrado na Figura 2.10.

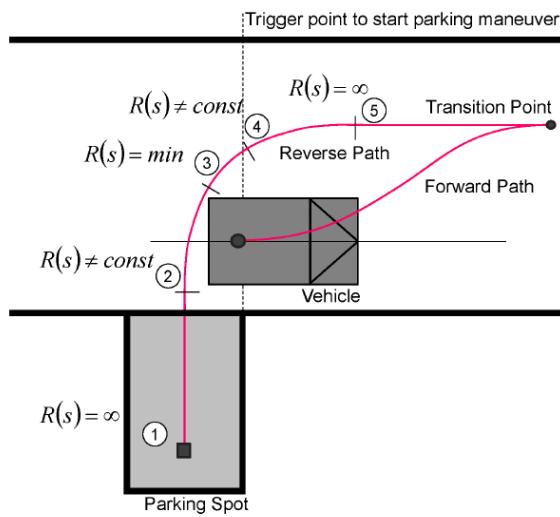


Figura 2.10: Trajetória de estacionamento perpendicular do veículo Junior 3 (Jeevan *et al.*, 2010).

2.3 Considerações finais

Este capítulo apresentou trabalhos que abordam o problema de planejamento de trajetória para estacionamento de veículos autônomos. Foram apresentadas abordagens diferentes para detectar uma vaga de estacionamento. Sensores do tipo laser e sonares 3D foram alguns exemplos utilizados para essa detecção. Para a criação das trajetórias também foram apresentados várias estratégias, como algoritmos baseado em aprendizado, sistemas de lógica *fuzzy*, métodos baseados em amostragem, além de abordagens que fazem uso de um conjunto limitado de movimentos. Como apresentado no capítulo anterior, a proposta deste trabalho inclui a detecção de três tipos de vagas (perpendicular, paralela e diagonal), além do planejamento de trajetória para a manobra de estacionamento.

Fundamentação Teórica

Este capítulo descreve de forma sucinta o que é a robótica móvel, dando ênfase aos robôs móveis autônomos. É apresentada uma introdução a alguns problemas da área, sendo: localização, mapeamento e navegação. Os quais montam o núcleo base do escopo deste trabalho. É apresentado também o modelo cinemático do veículo utilizado.

3.1 Robótica móvel

Bekey (2005) afirma que autonomia refere-se a sistemas capazes de operar em ambientes reais, sem qualquer forma de controle externo por longos períodos de tempo. Ele define um robô como sendo uma máquina que percebe, pensa e age. Desse modo, robôs devem ser dotados de sensores, que gerem dados para o mesmo através da percepção do ambiente, além de atuadores.

Um robô móvel é um dispositivo mecânico com uma base não fixa que age sob o controle de um sistema computacional, equipado com sensores e atuadores, os quais permitem interação com o ambiente (Bekey, 2005). Esses robôs, dotados de autonomia, podem realizar tarefas bastante específicas. Tarefas estas que possivelmente estejam relacionadas com mapeamento do ambiente, localização e navegação. E estas características serão abordadas nas próximas seções.

A robótica móvel é uma área de pesquisa que aborda diferentes linhas do conhecimento. Uma dessas linhas enfatiza o controle de veículos autônomos e semi-autônomos (Dudek e

Jenkin, 2000). Existem diversas áreas relacionadas, como: robótica convencional de manipuladores, inteligência artificial e visão computacional. Diversos trabalhos de pesquisa estudam veículos autônomos: (Zhang e Chen, 2006), (Dolgov *et al.*, 2010) e (Zhao *et al.*, 2011). Assim como os robôs que atuam em ambientes internos, os veículos autônomos que trabalham em ambientes externos têm de lidar com ambientes dinâmicos, imprecisão dos sensores e atuadores. Entretanto, muitas vezes os veículos autônomos são colocados em situações bem mais complexas.

3.2 Localização

O problema de localização é de fundamental importância para a realização de tarefas na robótica móvel. Este problema consiste na estimativa das coordenadas de um robô em um quadro de referência externo a partir de dados dos sensores (Thrun *et al.*, 2005). Sabendo onde se encontra, por meio da localização, o robô poderá calcular uma trajetória para a realização de determinada tarefa e executá-la. Através de um sensor de odometria, que calcula o deslocamento do robô, é possível estimar sua posição a partir uma posição inicial conhecida por meio da transformação de coordenadas. As coordenadas locais retornadas pelo sensor são transformadas em coordenadas relativas ao ambiente (coordenadas globais) (Hata, 2010). Entretanto, utilizando apenas um odômetro essas transformações não estarão compatíveis com a localização real do robô, pois as leituras desse sensor são afetadas por ruídos.

Uma das soluções amplamente utilizadas é associar informações provindas de outros sensores ao odômetro para aumentar a precisão da localização (Thrun, 1998). A utilização de dados provenientes de mais de um sensor, como outros odômetros, sensor laser (*laser range finders*), IMU (Unidade de medição inercial), por exemplo, permitem uma estimativa mais precisa da localização, pois métodos onde leituras anteriores são comparadas com as leituras atuais podem ser utilizados.

Os problemas de localização podem ser classificados em três tipos, sendo caracterizados pelo conhecimento que está disponível inicialmente e pelo tempo de execução (Thrun *et al.*, 2005):

- Rastreamento de posição: Esse tipo de problema assume que a posição inicial do robô é conhecida. Dessa forma, a localização pode ser feita estimando os ruídos retornados pelos sensores durante a locomoção do robô e tentando corrigi-los. Considera-se que os erros dos sensores são pequenos e na maioria das vezes essa incerteza é representada por uma distribuição unimodal (por exemplo, distribuição normal). Assim, a posição correta

do robô é calculada, considerando este um problema local, pois as incertezas são locais e restritas às regiões próximas à posição real do robô.

- Localização global: Nesta abordagem a posição inicial não é conhecida. Assim a utilização de uma distribuição probabilística unimodal é ineficiente, visto que o robô pode estar em qualquer lugar do mapa. Diferentes abordagens podem ser utilizadas para calcular a posição inicial, além do problema de rastreamento de posição.
- Problema do robô raptado: Esse problema é uma variação do problema de localização global. Neste caso, a qualquer momento o robô pode ser transportado para outro local no mapa sem ser comunicado. Assim como nos casos anteriores, a posição real deverá ser calculada independente dos erros dos sensores. A importância prática deste problema pode ser vista como forma de avaliar a robustez de um algoritmo de localização global diante de falhas momentâneas no sensoriamento.

O tipo de ambiente também é um fator importante a ser considerado. Podem ser classificados como estáticos ou dinâmicos (Thrun *et al.*, 2005). Diferente do primeiro ambiente onde apenas o robô se movimenta, no ambiente dinâmico outros objetos também se movem, tornando o cálculo da localização mais complexo. Uma das técnicas utilizadas para resolver o problema da localização em ambientes dinâmicos é filtrar os dados retornados pelos sensores, tentando retirar as informações dos objetos dinâmicos.

O interesse pela robótica móvel tem crescido nas últimas décadas. Com isso, vários trabalhos têm focado no problema da localização. Alguns algoritmos, baseados em métodos estatísticos, são utilizados para estimar a posição de robôs. Podem ser destacados o método de Markov (Fox *et al.*, 1999b) e o Filtro de Kalman (Smith *et al.*, 1990).

O Filtro de Kalman, a fim de estimar o posicionamento do robô a partir dos diversos sensores previamente descritos, opera de forma a minimizar o erro entre a estimativa de um estado e o estado real do processo. A formulação tradicional do filtro de Kalman para localização de robôs móveis utiliza determinados pontos do ambiente que servem de referência para que a posição do robô seja estimada (Leonard e Whyte, 1991). Para representar cada uma das coordenadas da posição do robô o filtro de Kalman utiliza funções Gaussianas.

O método de Monte Carlo (Fox *et al.*, 1999a) também é utilizado para resolver o problema de localização. Um exemplo de sua utilização pode ser visualizado na figura 3.1. A imagem mostra o processo de localização de um robô em uma sala. A área branca do mapa é considerada navegável, a área cinza representa a não-navegável e as setas são as prováveis posições e orientações do robô segundo o método de Monte Carlo. O item (a) da figura 3.1 mostra várias setas espalhadas pela sala, indicando que a posição do robô é desconhecida. Cada partícula, ou seta,

possui um peso e esse valor é modificado durante o processo. Quando o robô se movimenta os dados de leitura dos sensores realimentam o sistema, ou seja, as leituras atuais dos sensores são passadas para o sistema. Dessa forma V é feito com que as partículas com menor probabilidade de ser a real posição do robô tenham seus pesos diminuídos e as com maior probabilidade tenham seus pesos aumentados. As partículas com pesos baixos têm maior chance de serem excluídas do conjunto, como visto nos itens (b) e (c) da figura 3.1, onde muitas partículas são excluídas e outras se concentram. O item (d) ilustra uma estimativa da posição do robô em uma única região, indicando a real posição do mesmo. Nesse caso ilustrado o robô encontrava-se inicialmente no centro da sala e se deslocou para o corredor.

Esse método pode ser utilizado tanto para o problema de rastreamento de posição como de localização global. Para que o mesmo seja capaz de resolver o problema do robô raptado, partículas aleatórias são adicionadas ao conjunto das demais partículas durante a execução do método.

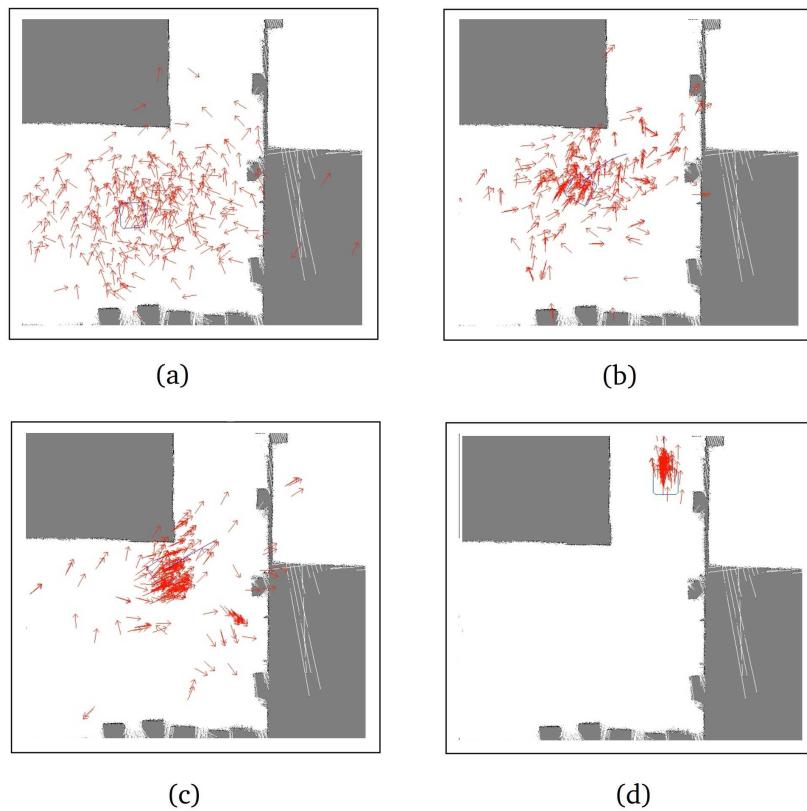


Figura 3.1: Exemplo do funcionamento da localização pelo método de Monte Carlo. (a) Conjunto de setas indicando as partículas do filtro espalhadas pelo mapa. (b) e (c) Concentração das partículas em área com maior probabilidade da real posição do robô. (d) Ilustra uma estimativa da posição do robô em uma única região.

3.3 Mapeamento

A geração de mapas do ambiente é considerada uma das competências essenciais de robôs verdadeiramente autônomos (Thrun *et al.*, 2005). Para a realização desta tarefa o robô deve ser capaz de construir um mapa do ambiente onde se encontra, apenas com as informações retornadas pelos seus sensores. Esta capacidade de mapeamento é considerada importante, visto que em muitas situações o robô não terá o conhecimento exato do local onde irá atuar. A credibilidade deste mapeamento deve ser levada em consideração, pois por intermédio do mapa retornado, o robô poderá se localizar e realizar sua tarefa de forma segura e precisa (Hata, 2010).

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 (do inglês: *Simultaneous Localization and Mapping*).

A estrutura dos mapas utilizada pelos robôs define sua classificação em dois tipos: topológicos ou métricos (Murphy, 2000). Alguns trabalhos, como o Simmons e Koenig (1995), utilizam uma abordagem híbrida unindo características de ambos. O mapa topológico é construído tendo como base pontos de referência conectados entre si (Lakemeyer e Nebel, 2003). Nesse caso, uma estrutura em grafo é utilizada para a representação do ambiente. As arestas conectam pontos de referência representando a existência de um caminho entre eles. Esses pontos, ou nós, por sua vez indicam locais específicos (portas ou cruzamento entre corredores, por exemplo) que podem ser identificados pelo robô.

Diferente da abordagem topológica (Figura 3.2(b)), o mapa métrico geralmente reproduz o ambiente por meio de uma grade, sendo um plano composto por células geometricamente iguais. Essas células representam áreas específicas do espaço e também armazenam a informação sobre seu estado, como a ausência ou não de obstáculos (Thrun, 1998). Um dos pontos positivos dessa representação é uma maior precisão na localização, visto que uma quantidade maior de informação é armazenada. Entretanto, o mapa topológico garante um planejamento de trajetória mais rápido levando em consideração a menor quantidade de dados em armazenamento (Congdon *et al.*, 1993). A Figura 3.2(a) mostra um tipo de mapa métrico em modelo CAD (do inglês: *computer-aided design*), onde as partes escuras representam área ocupada e a parte clara área livre de obstáculos.

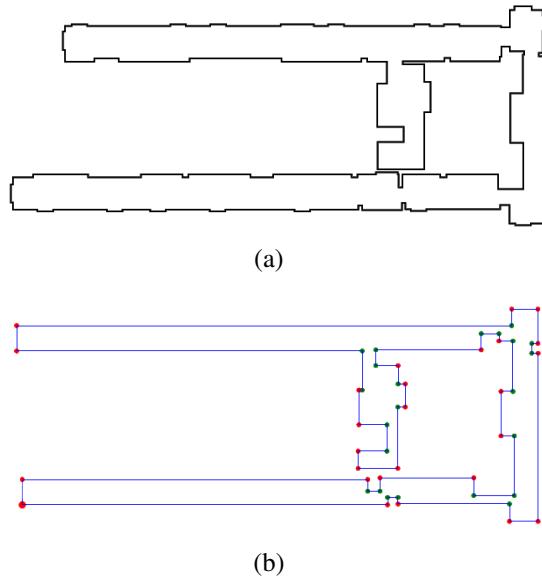


Figura 3.2: Tipos de Mapa. (a) Mapa métrico. (b) Mapa topológico (Adaptada de (Beevers, 2004)).

3.3.1 Grades de ocupação

O mapeamento por grade de ocupação tem por objetivo gerar mapas consistentes levando em consideração os dados ruidosos retornados pelos sensores (Thrun *et al.*, 2005). Essa abordagem métrica é uma das mais comuns e utilizadas segundo Thrun *et al.* (2000).

Consiste na criação de uma grade de células que se correlacionam com porções específicas do ambiente a ser representado. Essas células armazenam um valor binário que indica presença de obstáculos no ambiente ou um espaço livre de obstáculos. Assim, se a posição no mapa e um conjunto de medições são de conhecimento do robô, é possível calcular a probabilidade a posteriori de todas as células no mapa.

A Figura 3.3 é um exemplo de um mapa construído utilizando desta abordagem. A área clara representa espaço desocupado, enquanto a parte escura denota a presença de obstáculos.

Câmera estéreo, sonar e laser são tipos de sensores com a capacidade de dar suporte a criação de mapas em grades de ocupação. Thrun (2002) afirma que muitos trabalhos, utilizando os sensores citados, têm feito uso da abordagem de mapeamento em grade para representar ambientes, tanto em duas como em três dimensões. Além disso, alguns trabalhos tentam resolver dois problemas ao mesmo tempo: localização e mapeamento. Essa abordagem, como citada anteriormente, é conhecida como SLAM (do inglês: *Simultaneous Localization and Mapping*).

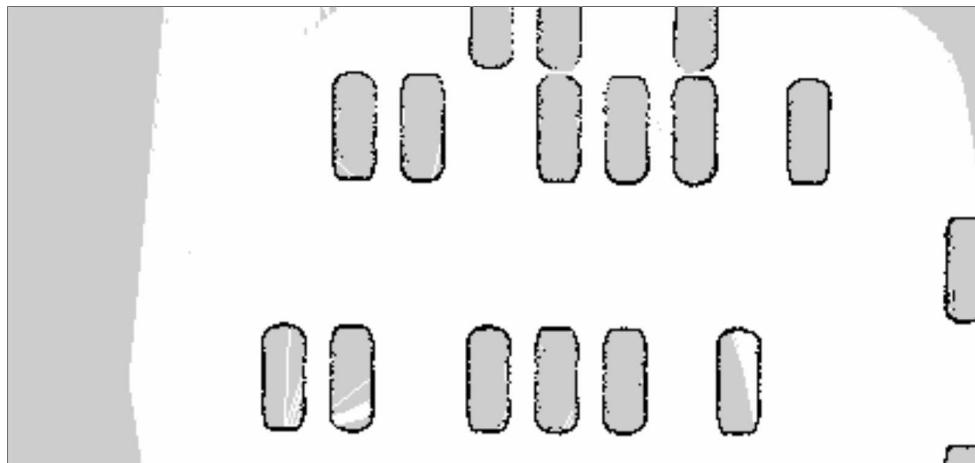


Figura 3.3: Exemplo de mapeamento em grades de ocupação.

3.4 Navegação

Os problemas citados anteriormente: mapeamento e localização estão diretamente ligados ao problema de navegação. Para um robô se deslocar de um estado inicial e alcançar outro determinado estado, os conhecimentos do ambiente como o mapa e a sua localização no mesmo são importantes. Várias abordagens são apresentadas por Choset *et al.* (2005) como técnicas de navegação para a robótica móvel. Muitos trabalhos, como Knepper e Mason (2011), Alves Barbosa de Oliveira Vaz *et al.* (2010) e Pivtoraiko e Kelly (2005), também têm focado em técnicas relacionadas com navegação para robôs móveis.

De um modo geral, as abordagens de planejamento de trajetórias não levam em consideração as características cinemáticas do robô. Entretanto, a inclusão das restrições de movimento para robôs não-holonômicos é fundamental para a criação de caminhos válidos.

Os algoritmos PRM (*Probabilistic Roadmaps*) e RRT (*Rapidly-exploring Random Tree*), propostos por Kavraki *et al.* (1998) e LaValle (1998), respectivamente, são aplicados em problemas de planejamento de trajetória utilizando técnicas baseadas em amostragem (Junior, 2008). Essa abordagem de planejamento, com intuito de diminuir a complexidade computacional na busca da solução do problema, gera amostras do espaço de configuração. As amostras são armazenadas em estruturas de dados, como árvores ou grafos, onde serão representadas por elementos como os nós. Esse nós mantêm uma relação entre si por meio de conexões, sendo estas as representantes dos caminhos entre as amostras do espaço.

O PRM pode ser construído como uma estrutura de grafo e essa construção é dividida em duas fases: aprendizado e questionamento (Thrun *et al.*, 2005). A primeira fase consiste na criação de um mapa de rotas. Inicialmente, o grafo que irá representar o PRM encontra-se

vazio. Então, repetidamente, uma configuração é amostrada do mapa, podendo pertencer a um espaço livre ou não (Thrun *et al.*, 2005). Esse passo é repetido até um número pré-definido de amostras do espaço livre de colisão serem adicionadas ao grafo. Só então o planejador local é chamado para conectar os nós vizinhos, verificando as colisões e montando um grafo de conectividade. Após a criação do grafo, representando o ambiente livre de colisões, inicia-se o segundo passo do algoritmo. Nesse passo, são passados para o PRM os pontos de início e de objetivo para a construção da trajetória. Os pontos (nós) passados são adicionados ao grafo e conectados aos nós geometricamente mais próximos. Desse modo, um algoritmo de busca como o A* ou o Dijkstra podem ser usados para encontrar a melhor rota dentro do espaço de configuração.

O algoritmo de RRT consiste na construção, em tempo real, de uma árvore de possíveis trajetórias, estendendo ramos para pontos de destino gerados aleatoriamente (LaValle, 1998). As Rapidly- Exploring Random Trees têm um grande potencial e podem ser utilizadas na solução de diversos problemas complexos, como abordado em (Lavalle *et al.*, 2000). A árvore gerada, recebendo como entrada os espaços com e sem obstáculos e um ponto de início (raiz da árvore), tenta cobrir a maior parte do espaço livre de obstáculos com caminhos possíveis, utilizando técnicas probabilísticas e informações parciais do ambiente. A quantidade de tentativas para expansão da estrutura é pré-definida, quanto maior o valor, maior será o espaço explorado e maiores as chances de encontrar o melhor caminho (LaValle, 2006). Mas, dessa forma, além do tempo de execução, a complexidade computacional irá aumentar.

3.4.1 Planejamento baseado em algoritmos de busca

O problema de planejamento da trajetória consiste principalmente em definir os pontos navegáveis ou não no ambiente que o robô está inserido. Para essa definição, devem ser descritas todas as configurações do ambiente que são livres ou não de obstáculos, reconhecendo o espaço de trabalho do robô. Porém, a complexidade computacional para a análise desse espaço e obtenção das configurações varia exponencialmente com sua dimensão, podendo torná-lo inviável (Frazzoli *et al.*, 2000).

Likhachev e Ferguson (2009) apresentam um algoritmo de planejamento de trajetória baseado em busca incremental para um ambiente discreto de estados. Dessa forma, o problema da complexidade computacional diminui. Esse espaço de busca foi proposto por Pivtoraiko e Kelly (2005), tendo como propósito servir de referência para um planejador de trajetória que codifica de forma eficiente apenas movimentos válidos de um robô.

A discretização de um espaço de busca proposto por Pivtoraiko e Kelly (2005), conhecida como Malha de Estados, fornece o ambiente para um planejamento não-holonômico de movi-

mento que faz uso de busca em grafo. O espaço de busca é formado por um conjunto de estados e suas conexões. Cada estado pode armazenar um conjunto de informações atualizadas do robô como posição, orientação ou velocidade. As conexões entre os estados constituem movimentos válidos de acordo a cinemática e dinâmica do robô.

O elemento básico dessa abordagem consiste em um conjunto limitado de movimentos básicos que o robô é capaz de realizar, podendo ser chamados de movimentos primitivos (Figura 3.4(a)).

A área livre de obstáculos do mapa é discretizada por meio da construção de um grafo que corresponde a Malha de Estados (ou *state lattice*). Essa malha, por onde o robô pode se movimentar, é criada por meio da união de vários movimentos primitivos, como apresentado na Figura 3.4(b).

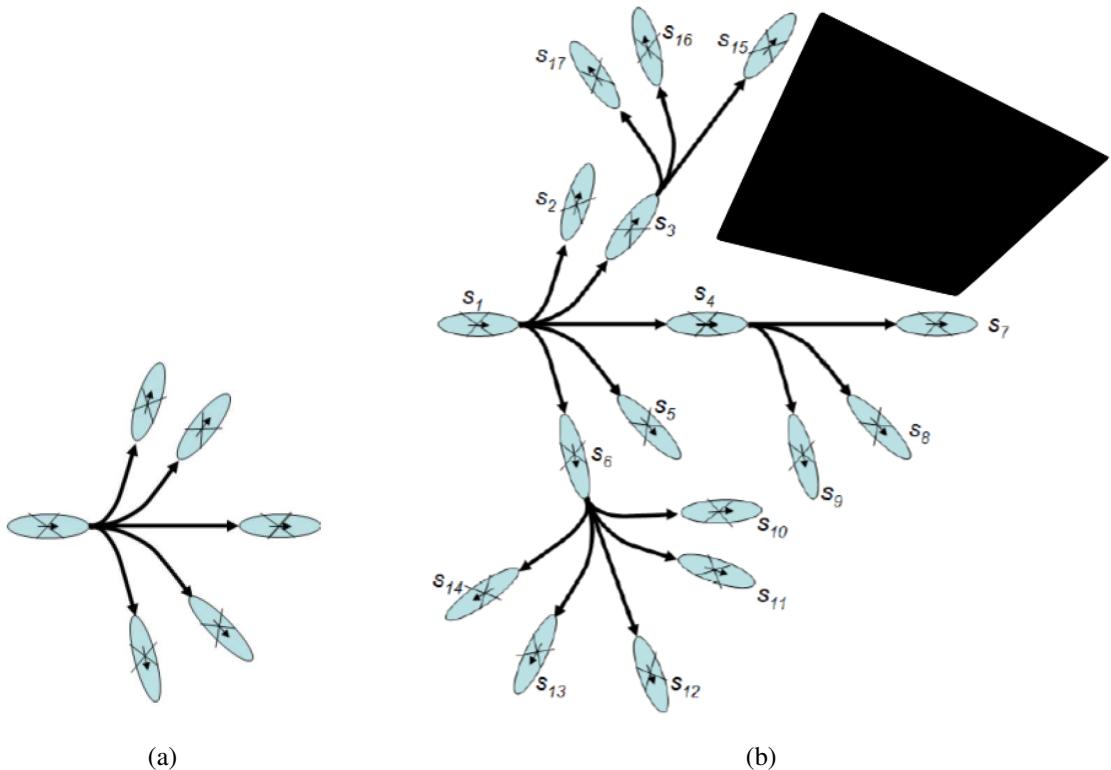


Figura 3.4: (a) Exemplo de movimentos primitivos e (b) Exemplo de uma malha de estados (ou *state lattice*) (Adaptada de (Likhachev e Ferguson, 2009)).

Esses movimentos primitivos são criados tendo como base as restrições de movimento do robô. Analisando a Figura 3.4(a) como exemplo, pode-se perceber que os movimentos primitivos

tivos ilustrados delimitam os movimentos do robô em apenas cinco. Essas cinco ações possíveis garantem a mudança de um estado para outro dentro do espaço de busca.

Cada um desses movimentos tem comprimento e custo de execução, que são levados em consideração pelo planejador. Devido o fato da Malha de Estados ser construída apenas de movimentos factíveis de serem realizados pelo robô, o planejador não precisa de preocupar com otimizações.

Após a criação do grafo, representando o espaço de atuação do robô, pode-se utilizar de algoritmos de busca para determinar a existência de um caminho entre dois nós (ou estados).

O algoritmo A* é capaz de encontrar o melhor caminho dentro de um ambiente discreto, como em uma abordagem de Malha de Estados. Esse algoritmo é considerado eficiente, pois consegue processar o menor número possível de estados para encontrar o melhor caminho. Entretanto, o algoritmo citado pode não ser eficiente em ambientes reais, onde existem restrições de tempo e conhecimento parcial do espaço de busca.

Em um ambiente real o espaço de atuação do robô pode variar, quando são levados em consideração os obstáculos dinâmicos, sendo necessário o replanejamento durante o tempo de execução. Essa abordagem se torna inviável para esses algoritmos, quando a variação ou o tamanho do ambiente impedem a resolução dentro de um tempo limitado.

Visando uma busca mais eficiente em ambientes complexos são desenvolvidas as abordagens incremental e *anytime*. A primeira tem como objetivo permitir o replanejamento quando ocorre alguma mudança no ambiente, enquanto a segunda abordagem permite encontrar caminhos sub-ótimos dentro de um intervalo de tempo predefinido. O algoritmo *Dynamic A** (*D**) (Stentz, 1995) por exemplo, faz uso da abordagem incremental. Ele realiza a primeira busca usando a abordagem A* e quando o espaço de busca é atualizado, consegue replanejar levando em consideração informações adquiridas durante os planejamentos anteriores.

O algoritmo ARA* (*Anytime A**) (Likhachev *et al.*, 2004) prioriza encontrar um caminho dentro de um tempo disponível. Inicialmente uma trajetória sub-ótima é retornada e enquanto existir tempo o replanejamento continua sendo realizado, buscando outra trajetória melhor.

Objetivando unir as duas abordagens, Likhachev *et al.* (2005) desenvolveram o algoritmo *Anytime Dynamic A** (*AD**), que é capaz de realizar buscas em ambientes dinâmicos e encontrar um caminho factível dentro das limitações de tempo. O AD* baseia-se na característica do algoritmo A* que pode resultar na criação de soluções rápidas, ou seja, se uma heurística consistente é utilizada e multiplicada por um fator de inflação $e > 1$, assim pode ser gerada uma solução mais rápida que se nenhum fator de inflação for usado (Likhachev e Ferguson, 2009). A partir da realização de uma série de pesquisas com fator de inflação decrescente, informações de pesquisas anteriores são reutilizadas. Baseado no D*, o AD* propaga as atualização das in-

formações através das porções afetadas do espaço de busca. A Figura 3.5 apresenta como esse algoritmo é implementado.

```

key( $s$ )
01. if ( $g(s) > rhs(s)$ )
02.   return [ $rhs(s) + \epsilon \cdot h(s_{start}, s); rhs(s)$ ];
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 ( $s \neq s_{goal}$ )  $rhs(s) = \min_{s' \in \text{succ}(s)}(c(s, s') + g(s'))$ ;
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 ( $\min_{s \in OPEN}(\text{key}(s)) < \text{key}(s_{start})$  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 all  $s' \in Pred(s) \cup \{s\}$  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 changes in edge costs are detected
09.     for all directed edges  $(u, v)$  with changed edge costs
10.       Update the edge cost  $c(u, v)$ ;
11.       UpdateState( $u$ );
12.     if significant edge cost changes were observed
13.       increase  $\epsilon$  or replan from scratch;
14.     else if  $\epsilon > 1$ 
15.       decrease  $\epsilon$ ;
16.     Move states 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 changes in edge costs;

```

(a)

(b)

Figura 3.5: Algoritmo AD* (Likhachev *et al.* (2005)). (a) Função para gerar o melhor caminho.
(b) Função principal.

Inicialmente (linhas 2 a 6, Figura 3.5(b)) o algoritmo define o fator de inflação ϵ para um valor suficientemente elevado, para que um plano inicial de qualidade inferior possa ser gerado rapidamente. Em seguida, enquanto não existir variação nos custos das extremidades, a função principal vai diminuir ϵ e melhorar a qualidade da solução encontrada até que seja garantida a solução ótima, $\epsilon = 1$ (linhas 14 a 20 Figura 3.5(b)). Os estados inconsistentes são movidos da lista $INCONS$ para a lista $OPEN$, sempre que ϵ é decrementado, deixando a lista $CLOSED$ vazia. Quando as variações nos custos de extremidade são detectados, existe a possibilidade de que a atual solução não seja mais sub-ótima. Se essas variações são substanciais, então 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 forma que uma solução menos ideal seja encontrada (linha 12 e 13, Figura 3.5(b)). Aumentar o custo das extremidades pode levar alguns estados a se tornarem pouco consistentes. Dessa forma, precisam ser adicionados 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. Isso mostra que diferentes valores chave devem ser computados de estados pouco consistente para estados muito consistentes (linhas 1 a 4, Figura 3.5(a)) (Likhachev *et al.*, 2005).

3.5 Modelo cinemático Ackerman

A geometria cinemática do sistema de esterçamento de um veículo não é um paralelogramo que produz ângulos de esterçamento iguais para ambas às rodas, mas sim um trapezóide que mais se aproxima da geometria de Ackerman, onde a roda interna tem um maior ângulo de esterçamento que a externa (Figura 3.6). O grau do atendimento da geometria de Ackerman no veículo tem pouca influência no comportamento direcional para altas velocidades, mas tem influência na auto centralização em manobras em baixas velocidades. Com o atendimento da geometria de Ackermann, também se verifica progressividade do torque de resistência em função do ângulo de esterçamento (Gillespie, 1992).

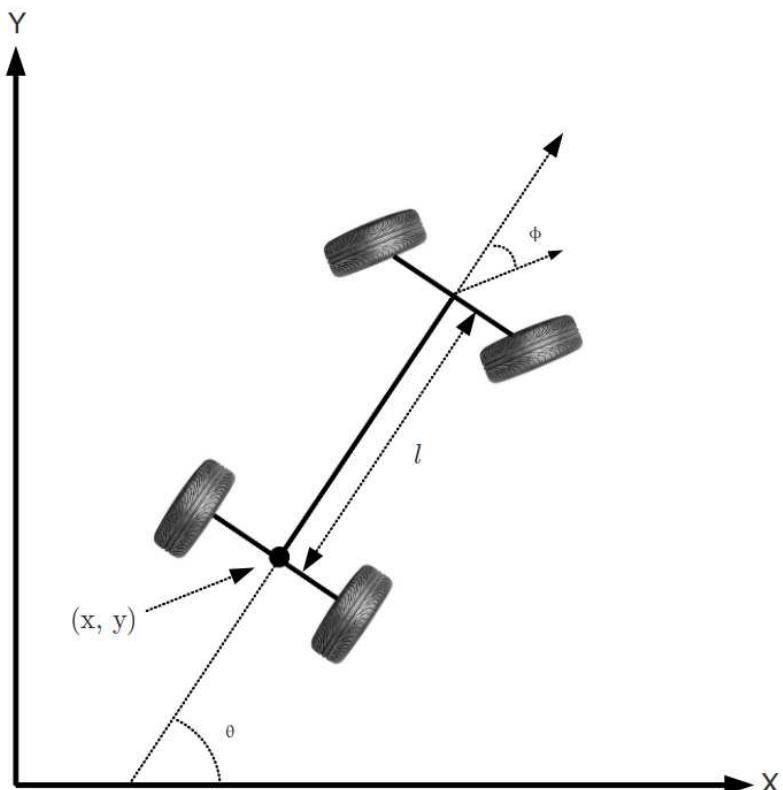


Figura 3.6: Modelo Cinemático de Ackerman.

A Figura 3.6 representa um veículo de quatro rodas, sendo que as duas rodas traseiras são mantidas por um eixo fixo e as dianteiras podem ser direcionadas por meio do giro da barra de direção. As coordenadas do veículo são representadas por: (x, y, θ) , onde x e y identificam o ponto médio do eixo traseiro e θ define sua orientação (ângulo em relação a direção de referência). A velocidade do veículo é representada por v . A distância entre os eixos e o ângulo de esterçamento médio das rodas dianteiras são representados por l e ϕ respectivamente. As equações (3.1 - 3.3) são utilizadas para descrever a posição do veículo, tendo como base sua velocidade, orientação e ângulo de esterçamento.

$$x = v \cos(\theta) \cos(\phi) \quad (3.1)$$

$$y = v \sin(\theta) \cos(\phi) \quad (3.2)$$

$$\theta = v \sin(\phi)/l \quad (3.3)$$

3.6 Considerações finais

Esse capítulo apresentou os principais problemas enfrentados na área de robótica móvel, bem como as abordagens utilizadas para a resolução dos mesmos. Foram apresentadas também as características do modelo cinemático utilizado. Tendo em vista a complexidade do problema proposto, optamos por utilizar o mapa métrico como forma de representar o ambiente de atuação do veículo. O sistema de sensoriamento adotado inclui sensores do tipo laser, sensor inercial, além de encoders que dão suporte à localização, mapeamento e planejamento. Essas abordagens utilizadas são apresentadas no capítulo seguinte.

Implementação

Este capítulo apresenta os métodos propostos, bem como o desenvolvimento do sistema de planejamento de trajetória para veículos autônomos. Neste capítulo são apresentadas as implementações dos subsistemas que compõem o sistema principal. O modelo simulado do veículo, a fusão de dados dos sensores, o sistema de localização, o detector de vagas e o sistema de planejamento são os tópicos principais deste capítulo.

4.1 Sistema proposto

O sistema de estacionamento autônomo proposto é formado por três fases: (I) detecção da vaga de estacionamento; (II) planejamento de trajetória; (III) e controle de esterçamento e aceleração para a manobra planejada. Essas etapas são ilustradas na figura 4.1.

A detecção da vaga é feita com a utilização de três sensores do tipo laser fixados no veículo. O planejador de trajetória trabalha sobre um espaço de busca discretizado que já obedece as restrições da cinemática do veículo. E o sistema de controle utilizado faz uso da abordagem de janela dinâmica (Fox *et al.*, 1997).

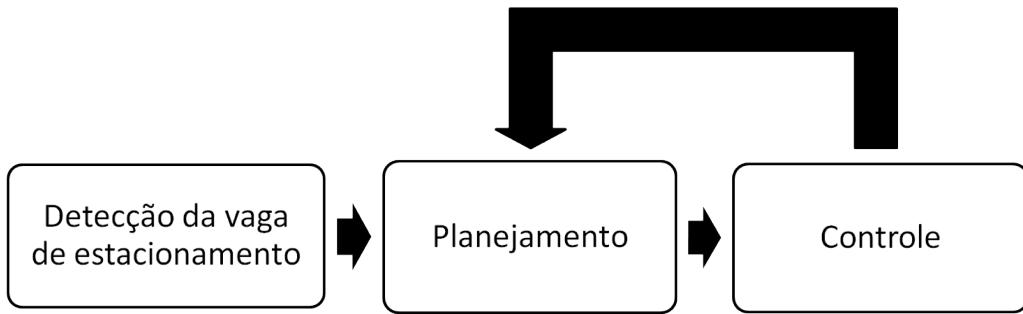


Figura 4.1: Modelo do sistema de estacionamento autônomo proposto.

4.2 Ferramentas e Bibliotecas

Essa Seção apresenta brevemente algumas ferramentas de software utilizadas no desenvolvimento deste trabalho.

4.2.1 ROS

ROS (do inglês *Robot Operating System*) é uma plataforma de desenvolvimento de software para robótica. Por meio de bibliotecas e ferramentas disponibilizadas pelo mesmo é possível construir, simular e controlar sistemas robóticos complexos.

Esse meta sistema operacional provê abstração de hardware, controle de dispositivos de baixo nível, troca de mensagens entre processos e gerenciamento de pacotes (ROS, 2012). O ROS utiliza a abordagem de comunicação em rede *peer-to-peer*, onde os processos se comunicam em pares por meio de troca de mensagens. Esses processos, por exemplo, podem representar: a odometria, o mapeamento ou as leituras dos sensores.

Essa plataforma é utilizada como base para a montagem da arquitetura do sistema desenvolvido. O ROS possibilitou a união dos subsistemas desenvolvidos, bem como uma estrutura que possibilitasse uma distribuição dos processos em diferentes computadores.

O ambiente Gazebo (Gazebo, 2012), é um simulador 3D compatível com a plataforma ROS que foi utilizado para testar e validar os sistemas desenvolvidos. Além do Gazebo (Figura 4.2(a)), o ROS possui um ambiente de visualização em 3D para robôs, que é o Rviz (Figura 4.2(b)). Esse ambiente disponibiliza a visualização de robôs e sua interação com o ambiente, como a representação das leituras dos sensores e criação de mapas.

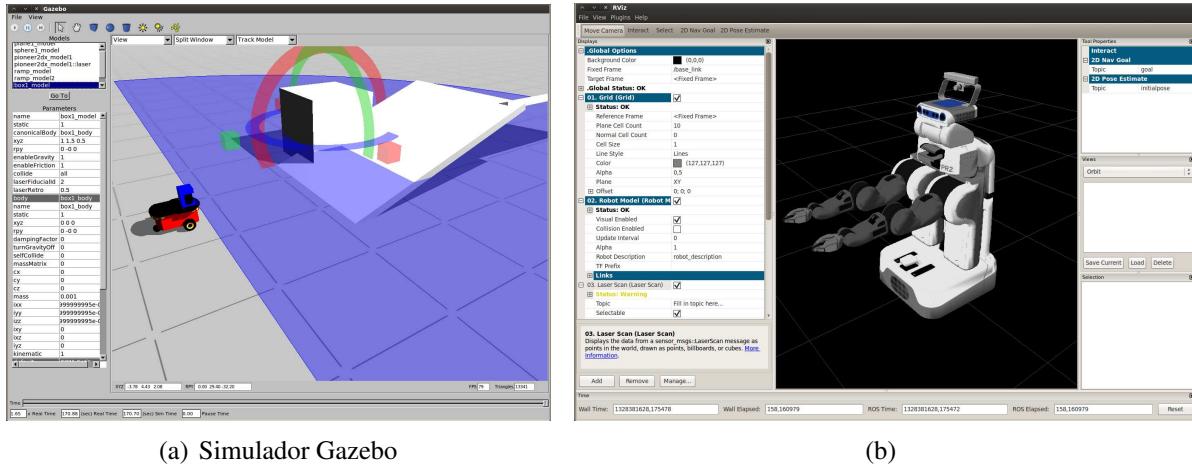


Figura 4.2: Ambiente de simulação do ROS. (a) Simulador Gazebo. (b) Visualizador de ambientes 3D do ROS.

4.2.2 Gmapping e Costmap_2D

As bibliotecas Gmapping e Costmap_2D, ambas suportadas pelo ambiente ROS, dão suporte à criação de mapas em duas dimensões. Por meio dos dados de leitura de um sensor laser somados com a odometria, é possível criar um mapa métrico.

A biblioteca Gmapping foi adaptada objetivando satisfazer as necessidades do problema proposto. Seu uso mais comum é para ambientes internos, onde existe a presença de muitos obstáculos próximos ao sensor. Em ambientes externos, como estacionamentos, a criação do mapa era inconsistente devido a classificação de zonas livres de obstáculos como área desconhecida, quando o sensor não detectava obstáculos na região de atuação dos sensores. Após as modificações, os resultados garantiram um mapa que satisfaz os requisitos do sistema de planejamento.

É criado também um mapa de custo local, onde as células próximas aos obstáculos têm seus valores maiores do que as demais. Por meio da Costmap_2d esse mapa local é criado e utilizado tanto pelo planejador de trajetória como pelo controlador. O objetivo é criar caminhos fora da rota dos obstáculos e manter o veículo neste caminho evitando colisões. Ambos os mapas possuem resolução de 0.1 metros, garantindo uma precisão suficiente para a resolução do problema em questão.

4.2.3 SBPL

Assim como a GMapping, vista na seção anterior, SBPL (do inglês *Search-based Planning Library*) é uma biblioteca integrada ao ambiente ROS. A SBPL armazena um conjunto de planejadores de movimento que trabalham com a abordagem de planejamento baseado em busca.

Essa biblioteca trata o problema de planejamento de trajetória utilizando a abordagem de algoritmos de busca, como apresentado na Seção 3.4.1. O mapa métrico construído por meio da biblioteca GMapping é empregado para montar a malha de estados que é utilizada pela SBPL.

4.3 Modelagem do veículo

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. Visando isto, um modelo do carro elétrico do projeto CaRINA foi construído em um projeto de iniciação científica. O modelo criado foi desenvolvido com base no ambiente de simulação Gazebo. Devido à complexidade do veículo, o modelo simulado continua sofrendo modificações com o intuito de ficar o mais próximo do real. Além de novos sensores, são modelados também características cinemáticas e dinâmicas no modelo simulado.

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 baseado em um formato de XML específico. O URDF (do inglês *Unified Robot Description Format*) é o formato utilizado pelo ROS para descrever seus robôs. Esse modelo é descrito como um conjunto de elos e juntas. Um elo pode representar uma parte específica do robô, como uma roda ou um sensor, enquanto uma junta equivale à ligação entre dois elos. Várias características como inércia e área de colisão podem ser definidas através desse modelo. Usando como base o modelo inicial foi implementado o modelo URDF, que continua em desenvolvimento.

As principais medidas e restrições cinemáticas são representadas no modelo simulado, o mais próximo do modelo real não holonômico. A Figura 4.3(a) mostra o modelo do carro elétrico visto no ambiente Rviz e a Figura 4.3(c) ilustra o veículo real. O formato URDF provê suporte ao ambiente Gazebo, por intermédio da inclusão de alguns *plugins* no próprio arquivo XML. Além da simulação, durante a execução do sistema no robô real, o modelo URDF pode representá-lo e ser visualizado pelo ambiente Rviz. A Figura 4.3(b) ilustra o modelo do veículo elétrico, similar a uma estrutura de árvore, em sua relação de elos e juntas, destacando suas orientações e posições em relação a um nó (elo) principal.

As dimensões do veículo são apresentadas na Tabela 4.1.

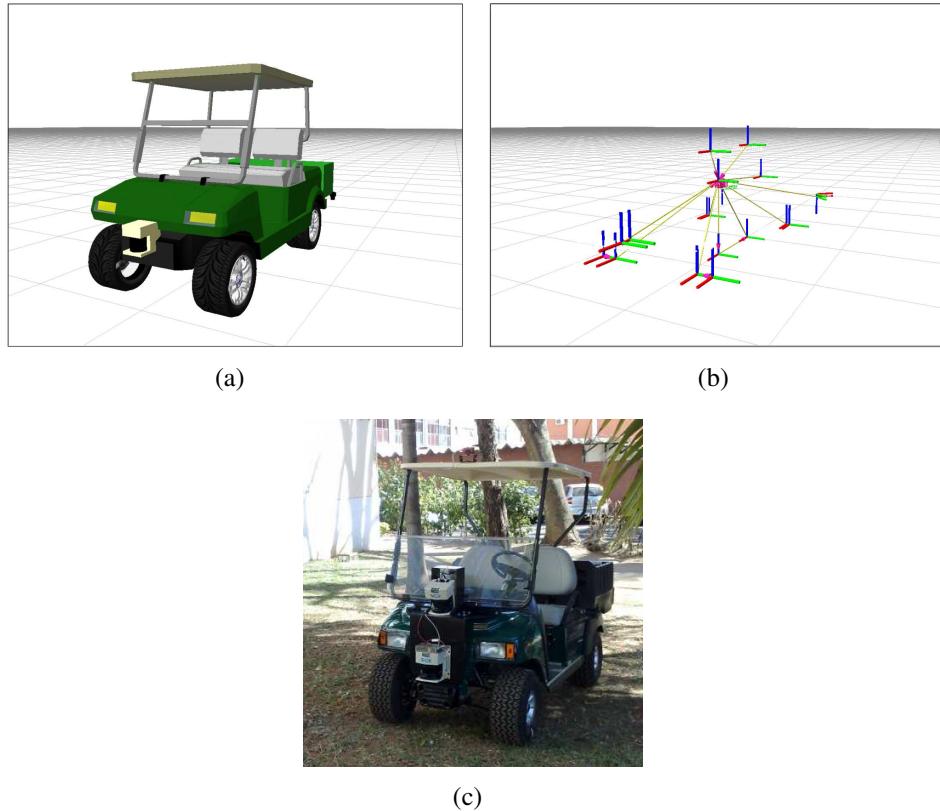


Figura 4.3: Modelo simulado do veículo. (a) Visão tridimensional do carro. (b) Representação cinemática em uma estrutura de árvore (Modelo URDF). (c) Veículo real.

Tabela 4.1: Especificações do veículo

Parâmetros	Valores	Unidade
Largura	1.35	m
Comprimento	2.57	m
Distância entre eixos	1.65	m
Distância entre as rodas	0.9	m
Diâmetro da roda	0.48	m
Raio mínimo de rotação	2.7	m

4.4 Fusão de dados dos sensores

Uma vez que um único sensor geralmente pode perceber apenas informações parciais e limitadas do ambiente, múltiplos sensores, sendo eles distintos ou não, deverão ser capazes de fornecer informações de forma integrada de diferentes pontos de vista (Xiong, 2002).

A fusão de sensores é uma forma eficiente de diminuir incertezas causadas por ruídos provenientes dos mesmos. Em alguns casos também aumenta o campo de percepção do robô. Neste

trabalho é utilizada a combinação dos dados de três sensores do tipo laser (*laser range finders*). Localizado na parte frontal do veículo encontra-se um sensor laser do modelo SICK LMS 291 (Figura 4.4(b)) e na parte traseira encontra-se dois sensores do modelo Hokuyo UTM-30LX (Figura 4.4(a)). 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°.



Figura 4.4: Sensores lasers. (a) Hokuyo UTM-30LX. (b) Sick LMS 291.

Com o intuito de facilitar os trabalhos com os sensores, foi gerado um laser virtual a partir dos três sensores reais. As informações sobre os sensores foram modeladas no arquivo URDF, podendo ser interpretadas pelo ROS. Os dados dos três sensores foram organizados em uma nuvem de pontos, montando um ambiente de duas dimensões. Em seguida, 360 pontos dessa nuvem são 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 4.5(b)) com capacidade de visão de 360° e alcance de 30 metros, devido o fato dos sensores estarem posicionados em locais específicos do veículo, como visto na Figura: 4.5(a).

Nesse trabalho o laser virtual é utilizado para a detecção de obstáculos, identificação de vagas de estacionamento, localização e mapeamento. A figura 4.6 ilustra a visão do sensor gerado pela fusão dos três sensores reais, detectando obstáculos ao redor do veículo. Os pontos na figura identificam a presença desses obstáculos.

4.5 Localização

A localizaçao do veículo é formada a partir da união de dois sistemas de localização. O primeiro sistema estima a posição baseado em dois *encoders*, um localizado em uma das rodas dianteiras

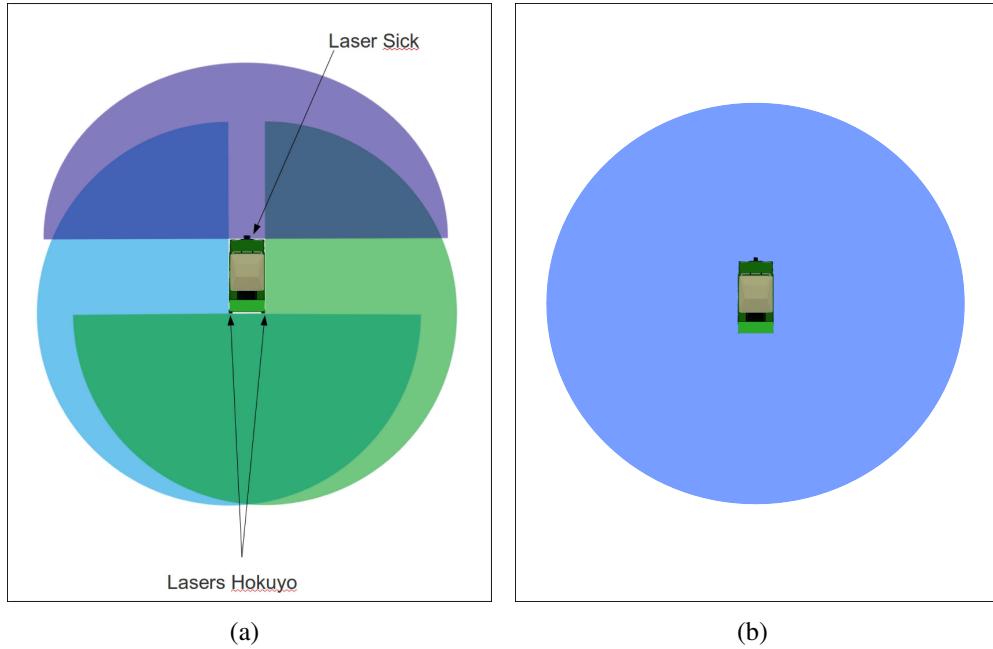


Figura 4.5: Visão superior do veículo com a localização e representação dos sensores laser. (a) Localização dos três sensores laser. (b) Representação do laser virtual.

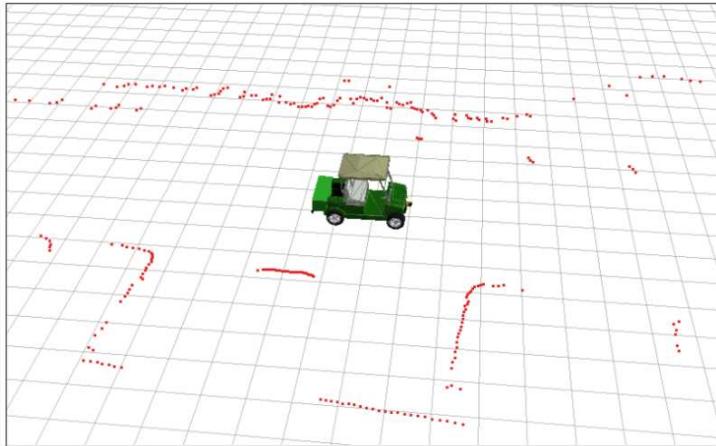


Figura 4.6: Visão de 360° dos sensores laser. Pontos vermelhos representam a detecção de obstáculos pelo laser virtual.

e outro localizado no barramento de direção. Foi utilizado também uma IMU (do inglês *Inertial Measurement Unit*) posicionada no centro do eixo traseiro para melhorar a estimativa da orientação. Este primeiro sistema calcula a odometria baseando-se no modelo Ackerman. O segundo sistema utiliza os três lasers presentes no veículo. A fusão dos dados desses sensores laser, apresentada na seção anterior, é utilizada pelo sistema de odometria baseado no algoritmo ICP (do inglês *Iterative Closest/Corresponding Point*) (Censi, 2008). O princípio básico deste

algoritmo é encontrar uma transformação entre duas nuvens de pontos. Em nosso trabalho, ele realiza um casamento de pontos dos lasers durante o deslocamento do veículo, possibilitando estimar sua posição. Ambos os sistemas de odometria são sujeitos a erros, que podem ser causados por imprecisão ou ruído dos sensores e 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 é utilizado um filtro de Kalman estendido (Meeussen, 2012), que calcula uma posição mais precisa do veículo utilizando o resultado dos dois sistemas de localização.

4.6 Detecção de Vagas de Estacionamento

O sistema de estacionamento autônomo deve ser capaz de detectar um espaço e classificá-lo como uma possível vaga de estacionamento. Para isso, as dimensões das vagas são predefinidas e o veículo utiliza o conjunto de sensores do tipo laser para estimar a posição desses espaços baseado nos obstáculos presentes no ambiente. Inicialmente o problema foi montado com as seguintes especificações:

- Ambiente de duas dimensões;
- Ambiente específico de estacionamento, onde os obstáculos são veículos, paredes ou colunas;
- Três tipos de vaga:
 - Perpendicular;
 - Paralela/Baliza;
 - Diagonal (45 graus).
- Possíveis vagas apenas dos lados direito ou esquerdo do veículo;
- Obstáculos alinhados com a zona navegável.

Com essas especificações o laser virtual citado anteriormente é capaz de detectar obstáculos que servirão de base para estimar a localização da vaga. O ambiente de busca limita-se em um local específico de estacionamento, onde qualquer obstáculo próximo ao veículo pode ser utilizado como referência para a estimativa das posições das vagas. Dessa forma, os obstáculos detectados são classificados por proximidade e por tamanho.

O sistema de detecção inicialmente agrupa os feixes de laser vizinhos que possuem uma diferença de tamanho pequena predefinida, considerando esses grupos como possíveis referências para a busca das vagas. Após esse agrupamento é feita uma filtragem para eliminar obstáculos distantes e também obstáculos pequenos (poucos feixes agrupados). Em seguida são feitas varreduras com o laser nas proximidades dos obstáculos classificados como válidos, buscando um espaço livre, correspondente a uma vaga de estacionamento. Esse espaço é definido de acordo com o tipo de vaga buscado, sendo que o sistema tem suporte a três tipos, como citado anteriormente. Ao encontrar um espaço livre, uma posição de destino é definida e o sistema de planejamento será responsável por gerar um caminho até a mesma.

As etapas do modelo de detecção proposto são ilustradas na figura 4.7.

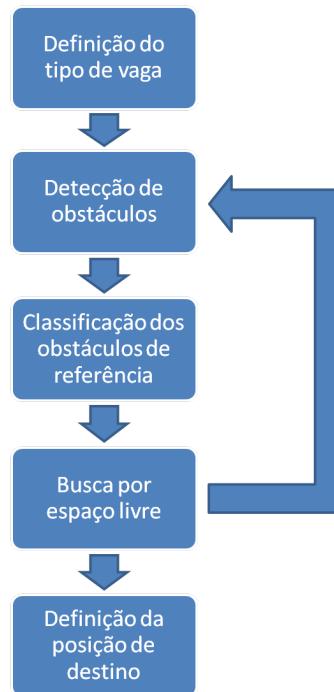


Figura 4.7: Diagrama das etapas do modelo de detecção de vagas.

A Figura 4.8 ilustra a detecção das vagas de estacionamento. Os pontos vermelhos são os obstáculos identificados por meio do laser. A Figura 4.8(a) ilustra a detecção de uma vaga perpendicular, que esta sendo representada por um retângulo azul. A Figura 4.8(b) mostra o sistema detectando uma vaga paralela enquanto o veículo se desloca próximo à mesma. Na Figura 4.8(c) o sistema procura por uma vaga diagonal do lado direito do veículo. Alguns obstáculos são detectados e após classificar um como referência, é feito uma busca por um espaço livre em suas proximidades. A vaga é encontrada e um ponto de destino, representado por uma seta na imagem, é gerado baseado no tipo de vaga buscada.

Os obstáculos detectados pelo laser nas figuras 4.8(a) e 4.8(b), são veículos reais, enquanto na figura 4.8(c) a coleta foi realizada em um ambiente simulado.

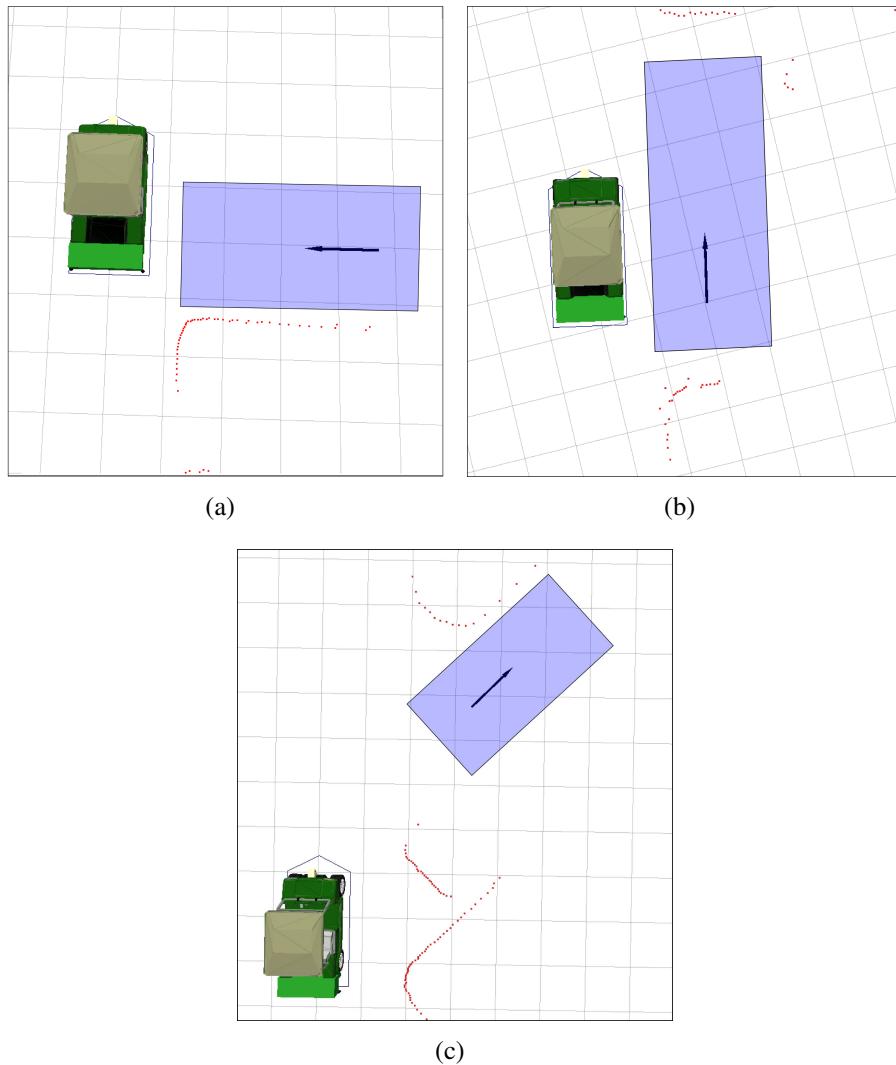


Figura 4.8: Detecção de vagas de estacionamento. Os pontos em vermelho consistem nos obstáculos detectados pelo laser. Os retângulos em azul representam os espaços definidos como vagas de estacionamento. As setas indicam a posição e orientação de destino. (a) Vaga perpendicular. (b) Vaga paralela. (c) Vaga diagonal.

As vagas perpendiculares e diagonais utilizadas possuem 2.1 metros de largura e 4.2 metros de comprimento. Essas dimensões são referentes a uma vaga de estacionamento de pequeno porte. Enquanto a vaga paralela possui um comprimento maior para possibilitar a execução da manobra. Neste trabalho utilizamos um comprimento mínimo de 5 metros.

Para cada um dos três tipos de estacionamento tratados neste trabalho é definida uma referência diferente como ponto de parada após a detecção da vaga. A busca de vagas diagonais

limitou-se na região frontal do veículo, assumindo que o mesmo deve realizar o estacionamento apenas com manobras frontais. Durante a busca o veículo se desloca em linha reta com uma velocidade constante. Logo após detectar um espaço livre de obstáculos para o estacionamento em vaga diagonal, o sistema de planejamento é acionado com o veículo em repouso.

Na detecção da vaga perpendicular, após a definição do ponto de destino, o veículo continua se deslocando até que a distância entre o centro da vaga e o centro de referência do mesmo seja maior que seu raio mínimo de curvatura ($l/\tan(\phi)$), sendo l a distância em metros entre os eixos e ϕ o ângulo de esterçamento. Essa métrica é usada com o objetivo de que o planejador encontre uma rota mais suave apenas com movimentos de ré.

O ponto de início para o planejamento da trajetória em vaga paralela é definido levando em consideração o comprimento mínimo requerido para esse tipo de vaga. Após a detecção da vaga o veículo se posiciona a 5 metros do início da mesma, seguindo em linha reta. A partir dessa posição o sistema de planejamento consegue construir um caminho até o ponto de destino.

4.7 Planejamento

Definir os melhores caminhos entre dois pontos que um robô deve se deslocar consiste no trabalho do planejador de trajetória. O algoritmo utilizado neste trabalho é descrito por Likhachev e Ferguson (2009) e apresentado na Seção 3.4.1. Os motivos que levaram a escolha desse algoritmo foram (I) a sua capacidade de atuar em um ambiente parcialmente conhecido; (II) o seu modelo de espaço de busca que discretiza o ambiente de acordo as limitações cinemática do veículo; (III) e a sua capacidade de encontrar trajetórias válidas dentro de um tempo predefinido. Essas características destacadas tornam-no viável para atuar em ambientes reais, visto que as mesmas estão presentes nesse ambiente.

O planejador de trajetória é construído usando a biblioteca SBPL_Lattice_Planner que está presente no ambiente ROS e encapsula a biblioteca SBPL. Essa abordagem trabalha sobre um ambiente discretizado construído por meio de um conjunto de movimentos primitivos válidos. Esse conjunto de movimentos foi criado baseado no modelo cinemático do veículo.

O conjunto de movimentos primitivos que o planejamento faz uso é apresentado na figura 4.9. Para a sua construção utilizamos as equações do modelo cinemático Ackerman, apresentadas na Seção 3.5, discretizando valores de velocidade e esterçamento. Como ilustrado, esse conjunto é composto por quatorze movimentos, sendo sete movimentos para a frente e sete para trás. Além disso, os conjuntos de movimentos podem ser combinados entre si em 24 ângulos diferentes, garantindo um maior espaço de busca.

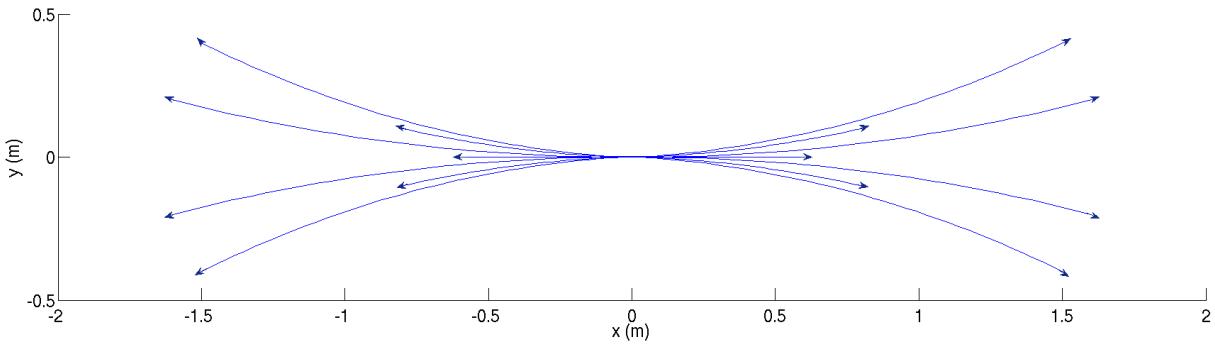


Figura 4.9: Conjunto de movimentos primitivos. 14 ações de movimentos válidos para o veículo (7 movimentos para frente e 7 para traz).

Neste trabalho a combinação dos movimentos primitivos sobre o mapa de ocupação gera uma malha de estados, onde cada estado formado pelas coordenadas (x, y, θ) representa a posição e a orientação. O mapa métrico gerado por meio do laser virtual, bem como o conjunto de movimentos do veículo, possuem uma resolução de 0.1m.

Após a discretização do espaço de busca é definido um ponto de destino através do detector de vagas de estacionamento. Deste modo, o planejador busca um caminho dentro dessa malha de estados utilizando o algoritmo AD*. Em seguida o sistema de controle, tendo conhecimento da trajetória criada, envia para o veículo os comando de esterçamento e aceleração.

A Figura 4.10 ilustra uma trajetória criada pelo sistema de planejamento e a trajetória realizada pelo veículo. Em 4.10(a) a figura mostra uma trajetória, representada por uma linha preta, que guia o veículo até dentro da vaga. Nessa figura o veículo é representado por um retângulo. A Figura 4.10(b) ilustra o veículo após ter realizado a manobra, destacando em verde a real trajetória cumprida. O veículo permaneceu próximo do caminho desejado se afastando não mais do que 0.1m.

Durante a execução da trajetória o mapa de custo é atualizado de acordo as leituras dos sensores. Com essa abordagem, tanto o planejamento como o controle evitam regiões próximas aos obstáculos, devido ao elevado custo nessa regiões. Como citado anteriormente, mesmo durante a etapa de controle o planejador é capaz de criar novas trajetórias de acordo as mudanças do ambiente.

4.8 Controle

Todas as ações de controle enviadas ao veículo para seguir a trajetória e desviar de obstáculos são obtidas localmente através de uma amostragem do seu espaço de controle. Essa abordagem

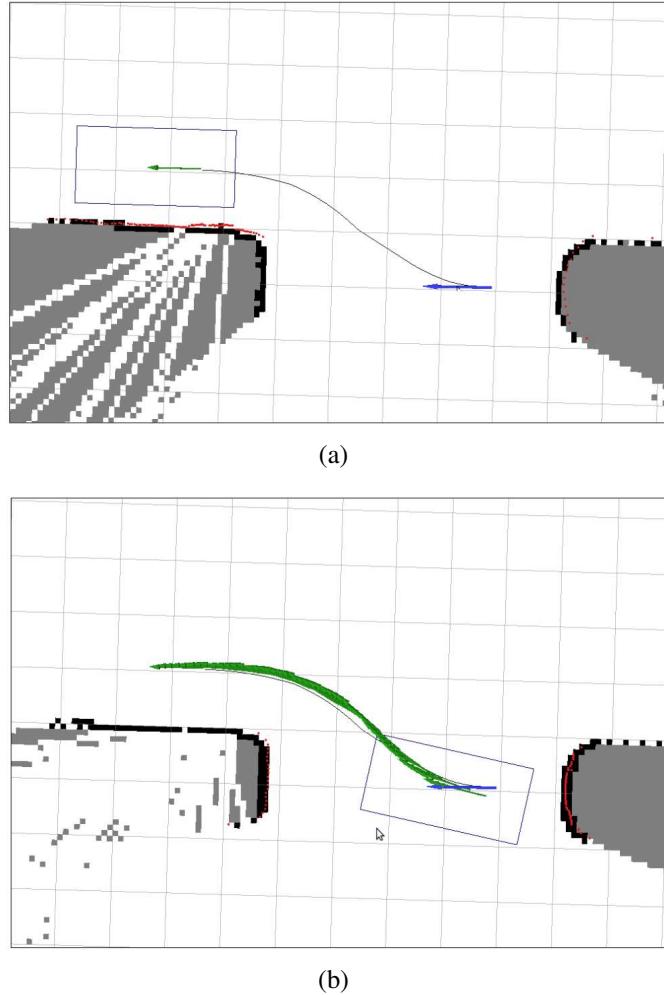


Figura 4.10: Exemplo do planejamento de trajetória. (a) Trajetória gerada para manobra de estacionamento, ilustrada por uma linha preta. (b) Trajetória executada destacada em verde.

cria trajetórias locais para o veículo através de simulações do seu próximo estado (para frente ou para trás) baseadas em DWA (Dynamic Window Approach), onde uma amostra de velocidade é aplicada durante um curto período de tempo com o objetivo de prever o que aconteceria ao veículo se essa amostra fosse mantida (Fox *et al.*, 1997). Dentre as várias trajetórias simuladas, aquela que apresentar o menor custo será enviada ao veículo, sendo que as simulações que colidem com obstáculos são eliminadas. As trajetórias resultantes dessas simulações usam uma métrica que incorpora proximidade de obstáculos, proximidade com a meta e a proximidade com a trajetória global. Esse sistema de controle é disponibilizado pela plataforma ROS e foi adaptado por Magalhães (2013) para funcionar na plataforma deste trabalho.

4.9 Considerações finais

Este capítulo apresentou os subsistemas implementados. Foram descritas as informações sobre a modelagem do veículo simulado usado nos testes, bem como as plataformas de software que deram suporte ao desenvolvimento do sistema. Foi apresentada também a fusão dos dados dos lasers, que são utilizados para detecção de obstáculos, mapeamento, localização e detecção de vagas. A abordagem para a detecção dos três tipos de vaga foi apresentada descrevendo suas principais características.

O sistema de planejamento descrito utiliza a combinação de malhas de estados com o algoritmos AD* possibilitando a geração de trajetórias para a manobra de estacionamento. A partir da união dos subsistemas implementados foi montado o sistema proposto neste trabalho. Capaz de gerar um mapa métrico do ambiente, encontrado uma determinada vaga de estacionamento, construir uma trajetória e guiar o veículo até o destino.

Resultados

Este capítulo apresenta os principais resultados desse trabalho. Como mencionado anteriormente, o escopo dessa dissertação inclui mapeamento, localização e navegação, tendo como foco principal o último. Dessa forma, tivemos a oportunidade de pesquisar e implementar sistemas referentes às principais áreas da robótica móvel. A navegação, como foco principal do trabalho, necessita de sistemas de mapeamento e localização, sendo assim, os resultados englobam a fusão de sensores, mapeamento de ambiente em duas dimensões, sistema de localização, além de um sistema de detecção de vagas de estacionamento e o planejamento de trajetória.

5.1 Estacionamento diagonal

O sistema foi testado em ambiente simulado, onde os obstáculos para referência às vagas de estacionamento eram constituídos por outros veículos. A Figura 5.1 nos mostra o sistema de estacionamento autônomo em funcionamento. O lado esquerdo de cada figura ilustra o ambiente simulado, enquanto o lado oposto, a visualização do ambiente na perspectiva dos sensores. A Figura 5.1(a) ilustra o veículo próximo a uma possível vaga utilizando o sistema de mapeamento. A área branca denota zona navegável, a parte preta indica presença de obstáculos, a área cinza é configurada como desconhecida e a verde faz parte de um mapa de custo local. Essa zona verde corresponde a uma área de custo elevado, por estar próximo de obstáculos. Dessa

forma, tanto o sistema de planejamento quanto o controle evitam essa área. Na Figura 5.1(b) podemos perceber que o sistema de detecção conseguiu identificar uma vaga de estacionamento que é destacada na imagem por uma seta azul. A informação de localização da vaga é utilizada pelo sistema de planejamento de trajetória como ponto de destino. Na mesma imagem é ilustrado o caminho gerado pelo planejador. A Figura 5.1(c) exibe o veículo percorrendo a trajetória traçada e marcando pelo caminho sua odometria. A figura também ilustra que o sistema de mapeamento continua trabalhando, diminuindo as regiões de incerteza e atualizando o mapa local de custo. A última Figura 5.1(d) ilustra a conclusão com sucesso da tarefa de estacionamento autônomo. É interessante ressaltar que no ambiente simulado o sistema de localização foi composto apenas pela nuvem de pontos criada por meio dos sensores lasers.

5.2 Estacionamento perpendicular

A Figura 5.2 ilustra uma manobra em vaga perpendicular definida como válida. Inicialmente o veículo se desloca em linha reta buscando uma vaga do lado direito (Figura 5.2(a)). Após localizar um espaço para estacionamento o veículo se posiciona levando em consideração o raio mínimo de curvatura como distância entre o ponto de referência do veículo e o ponto de destino. Finalizada essa etapa o planejador encontra um caminho até a vaga encontrada (Figura 5.2(b)). A Figura 5.2(c) mostra o sistema de controle atuando sobre o veículo gerando comandos de esterçamento e aceleração para manter o veículo na rota planejada. A Figura 5.2(d) ilustra a manobra finalizada com o veículo dentro da vaga.

5.3 Estacionamento paralelo

A Figura 5.3 ilustra as etapas do estacionamento em vagas paralelas. Inicialmente é buscado um espaço mínimo correspondente a uma vaga, como visto na Figura 5.3(a). Após a localização do ponto de destino pelo detector de vagas e a conclusão do posicionamento para início da manobra, o planejador constrói a trajetória necessária para realização do estacionamento (Figura 5.3(b)). As Figuras 5.3(c) e 5.3(d) ilustram o veículo entrando na vaga e concluindo a trajetória, respectivamente. Nos testes de estacionamento em vagas paralelas o sistema conseguiu uma melhor performance quando o obstáculo de referência (outro veículo estacionado) estava posicionado entre 0.5 e 1.5 metros da lateral do veículo.

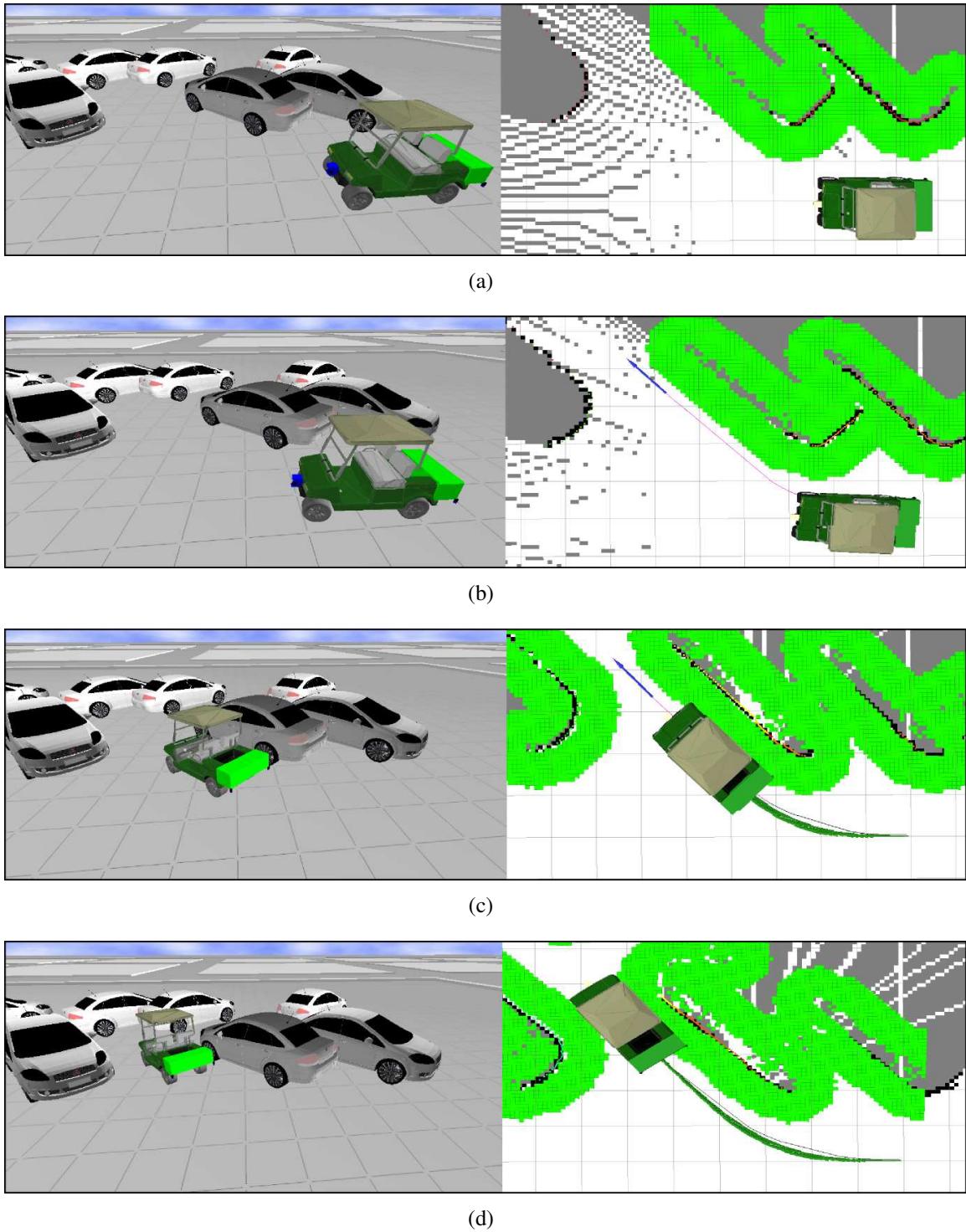


Figura 5.1: Sistema de estacionamento autônomo em ambiente simulado. (a) Mapeamento e detecção da vaga de estacionamento. (b) Planejamento de trajetória. (c) Execução da trajetória. (d) Finalização da manobra.

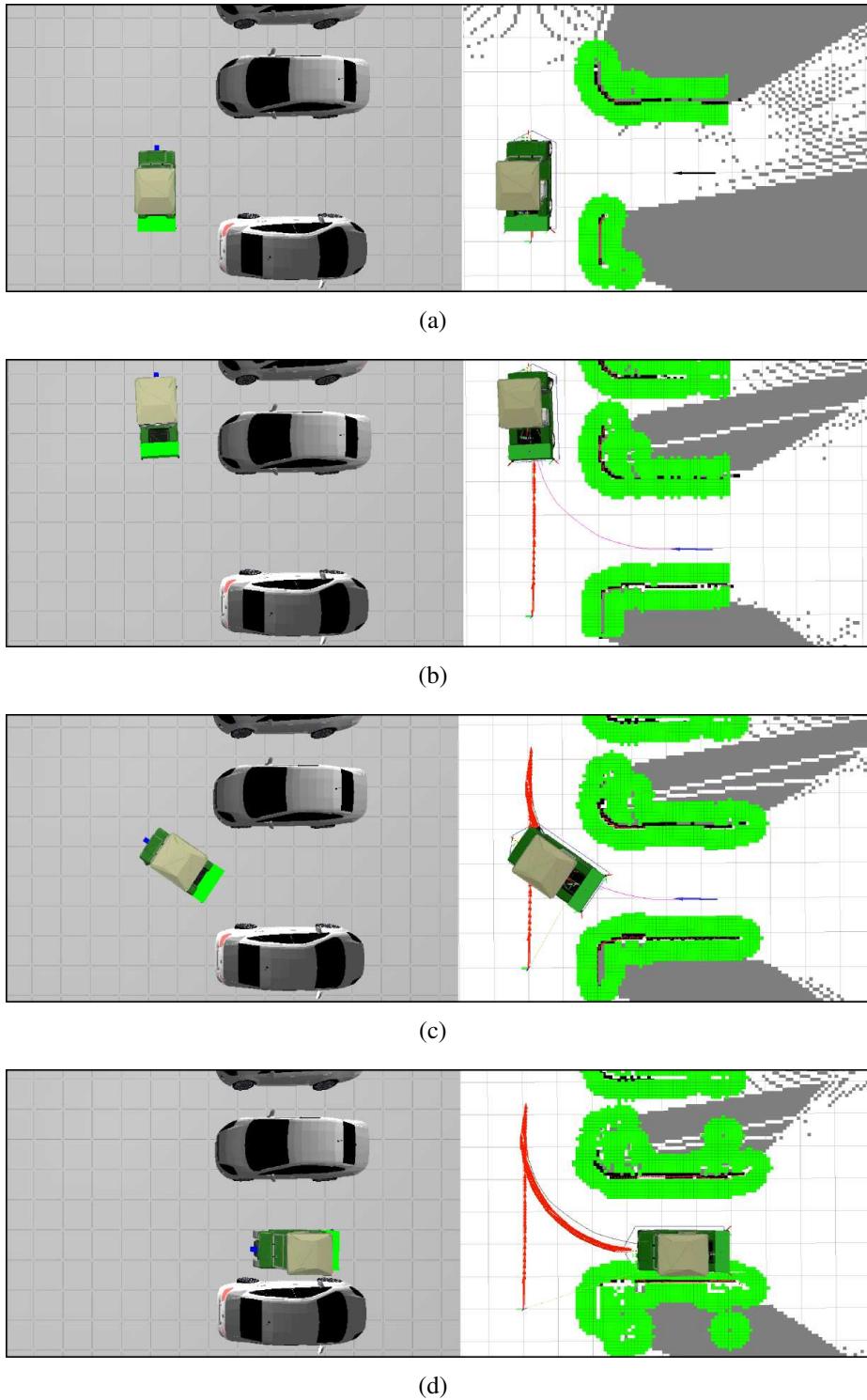


Figura 5.2: Sistema de estacionamento autônomo em ambiente simulado - Estacionamento perpendicular. (a) Mapeamento e detecção da vaga de estacionamento. (b) Planejamento de trajetória. (c) Execução da trajetória. (d) Finalização da manobra.

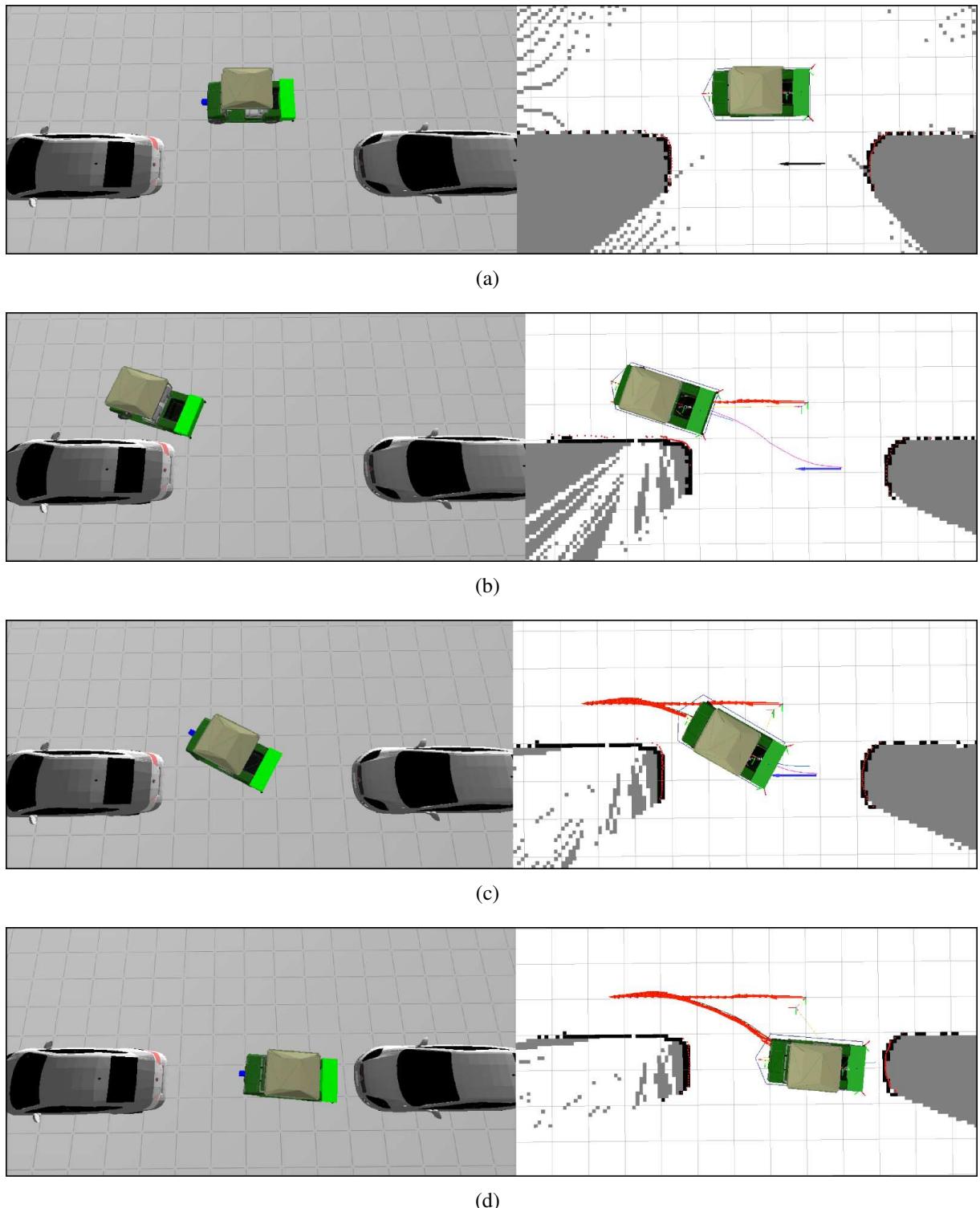


Figura 5.3: Sistema de estacionamento autônomo em ambiente simulado - Estacionamento paralelo. (a) Mapeamento e detecção da vaga de estacionamento. (b) Planejamento de trajetória. (c) Execução da trajetória. (d) Finalização da manobra.

5.4 Definição de manobras válidas

Para definir uma manobra como válida levamos em consideração as dimensões da vaga de estacionamento, bem como as do veículo. Dessa forma, classificamos como inválidas aquelas que após sua finalização parte do veículo permaneceu fora dos limites da vaga. A Figura 5.4 ilustra dois exemplos do posicionamento do veículo após a finalização de uma manobra de estacionamento. A área delimitada pelo retângulo azul corresponde ao espaço de estacionamento. Na Figura 5.4(a) todo o veículo se encontra dentro do espaço de estacionamento, garantindo assim uma manobra válida. A Figura 5.4(b) mostra um exemplo de manobra que não obteve sucesso em sua finalização, pois parte do veículo permaneceu fora do espaço definido como válido para o estacionamento.

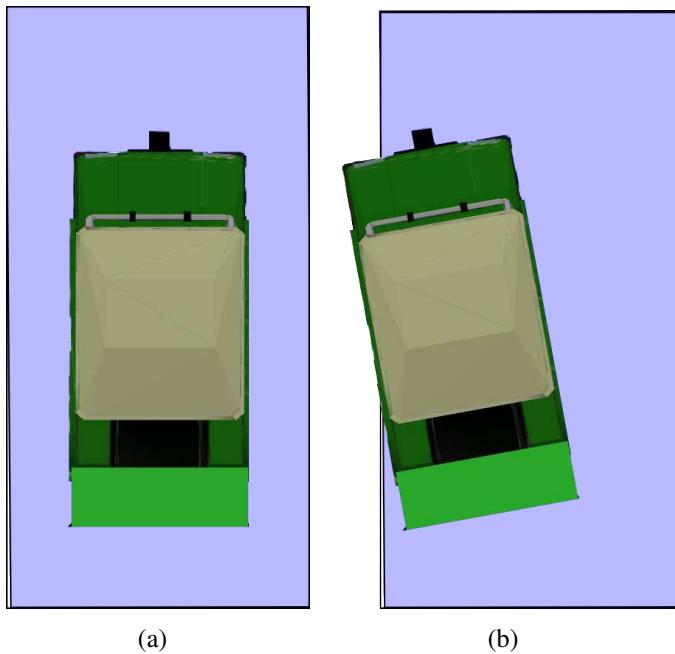


Figura 5.4: Posicionamento do veículo em finalizações de manobras de estacionamento. (a) Veículo dentro da zona específica de estacionamento (manobra válida). (b) Veículo parcialmente dentro da zona de estacionamento (manobra inválida).

5.5 Resultados simulados

Foram analisadas um total de 30 manobras para a validação do sistema. Baseado na métrica apresentada na Seção 5.4 o sistema de planejamento desenvolvido apresentou um resultado satisfatório de 83%. Esse valor corresponde ao percentual de manobras de estacionamento

finalizadas com sucesso. Dentre as 30 manobras analisadas, sendo 10 de cada tipo de vaga, as manobras em vagas paralelas e perpendicular obtiveram 80% de sucesso, enquanto as manobras em vagas diagonais alcançaram 90% de acerto (Tabela 5.1).

Tabela 5.1: Validação das manobras de estacionamento

Tipo de vaga	Manobras válidas	Manobras inválidas	Total
Perpendicular	8	2	10
Paralela	8	2	10
Diagonal	9	1	10

Em todas as 30 manobras analisadas o sistema de planejamento foi capaz de construir uma trajetória consistente. As manobras classificadas como inválidas podem ter sido afetadas pelo erro do sistema de localização em conjunto com as limitações do sistema de controle. Visto que o controle faz uso de um conjunto discretizado de velocidades, causando em alguns momentos uma leve fuga do caminho global. Tanto a manobra em vagas paralelas como em vagas perpendiculares podem ser consideradas mais complexas do que a manobra em vaga diagonal. A última pode ser realizada a partir de uma curva mais suave, devido sua orientação em relação ao posicionamento inicial do veículo. Dessa forma, o sistema de localização permanece mais consistente. Enquanto que a manobra em vaga perpendicular necessita de, no mínimo, uma curva de 90°. E a manobra em vaga paralela é composta por duas curvas.

O espaço de atuação do veículo foi discretizado. O mapa utilizado possui resolução de 0.1 metro e o conjunto de movimentos primitivos trabalha com a discretização da orientação em 0.26 radianos. O sistema conseguiu garantir, nas manobras válidas, que o ponto de referência do veículo sempre permanecesse dentro de um limite máximo de 0.2m de distância do ponto de destino e com uma diferença menor do que 0.26 radianos entre orientação do veículo e a orientação desejada.

5.6 Resultados preliminares em ambiente real

Em ambiente real foi testado o sistema de detecção de vagas. Baseados nos sistemas de localização e percepção, também implementados para o trabalho proposto, foi possível fazer a detecção de vagas perpendiculares e paralelas, destacadas nas figuras 4.8(a) e 4.8(b). O sistema de planejamento também foi testado em um ambiente onde o veículo deveria percorrer um determinado caminho e desviar de um obstáculo. Foram realizados também testes de estacionamento em vagas perpendiculares e paralelas. Como o veículo ainda não possui o controle de velocidade, a aceleração ficou sob a responsabilidade do condutor. Durante a realização do teste o veículo foi

capaz de percorrer o trajeto proposto e desviar de um obstáculo detectado durante a execução da trajetória.

5.6.1 Planejamento com desvio de obstáculos

A Figura 5.5 mostra um teste do sistema de planejamento realizado em ambiente real. O lado esquerdo de cada figura ilustra o veículo real e do lado oposto pode ser visualizado a trajetória e os obstáculos apresentados pelo ambiente Rviz (Seção 4.2.1). A Figura 5.7 ilustra uma trajetória gerada em linha reta com o destino representado por um retângulo azul. Durante o percurso o sistema de percepção detecta obstáculos, representados em vermelho, na região da trajetória. O planejador imediatamente encontra uma nova rota desviando dos obstáculos 5.6. Na Figura 5.5(c) percebemos uma leve fuga da trajetória logo após o replanejamento. Em seguida o veículo retorna ao trajeto desejado. A Figura 5.5(d) ilustra o veículo chegando ao destino, concluindo assim a sua tarefa.

5.6.2 Estacionamento em vaga perpendicular

A Figura 5.6 apresenta um teste do sistema de planejamento realizado em ambiente real tratando o problema de estacionamento em vagas perpendiculares. A Figura 5.6(a) ilustra o veículo fazendo a busca por uma vaga perpendicular do lado direito, a qual está delimitada por dois obstáculos: um outro veículo e uma árvore. Após a identificação da vaga o planejador encontra uma trajetória até o destino final e o veículo se posiciona para iniciar essa manobra (Figura 5.6(b)). Na Figura 5.6(c) é ilustrado a execução da manobra para dentro da vaga. Em seguida o veículo é ilustrado chegando ao destino (Figura 5.6(d)) , concluindo assim o estacionamento em vaga perpendicular.

5.6.3 Estacionamento em vaga paralela

A Figura 5.7 apresenta um teste do sistema de planejamento realizado em ambiente real tratando o problema de estacionamento em vagas paralelas. A Figura 5.7(a) destaca o veículo fazendo a busca por uma vaga paralela do lado direito, a qual está delimitada por duas caixas devidamente posicionadas. A altura e o tamanho das caixas foram suficientes para que os sensores classificassem como obstáculos válidos. Após a identificação da vaga o planejador encontra uma trajetória até o destino final e o veículo se posiciona para iniciar essa manobra (Figura 5.7(b)). Na Figura 5.7(c) é ilustrado a execução da manobra para dentro da vaga. Em seguida o veículo é ilustrado chegando ao destino (Figura 5.7(d)) , concluindo assim o estacionamento em vaga paralela.

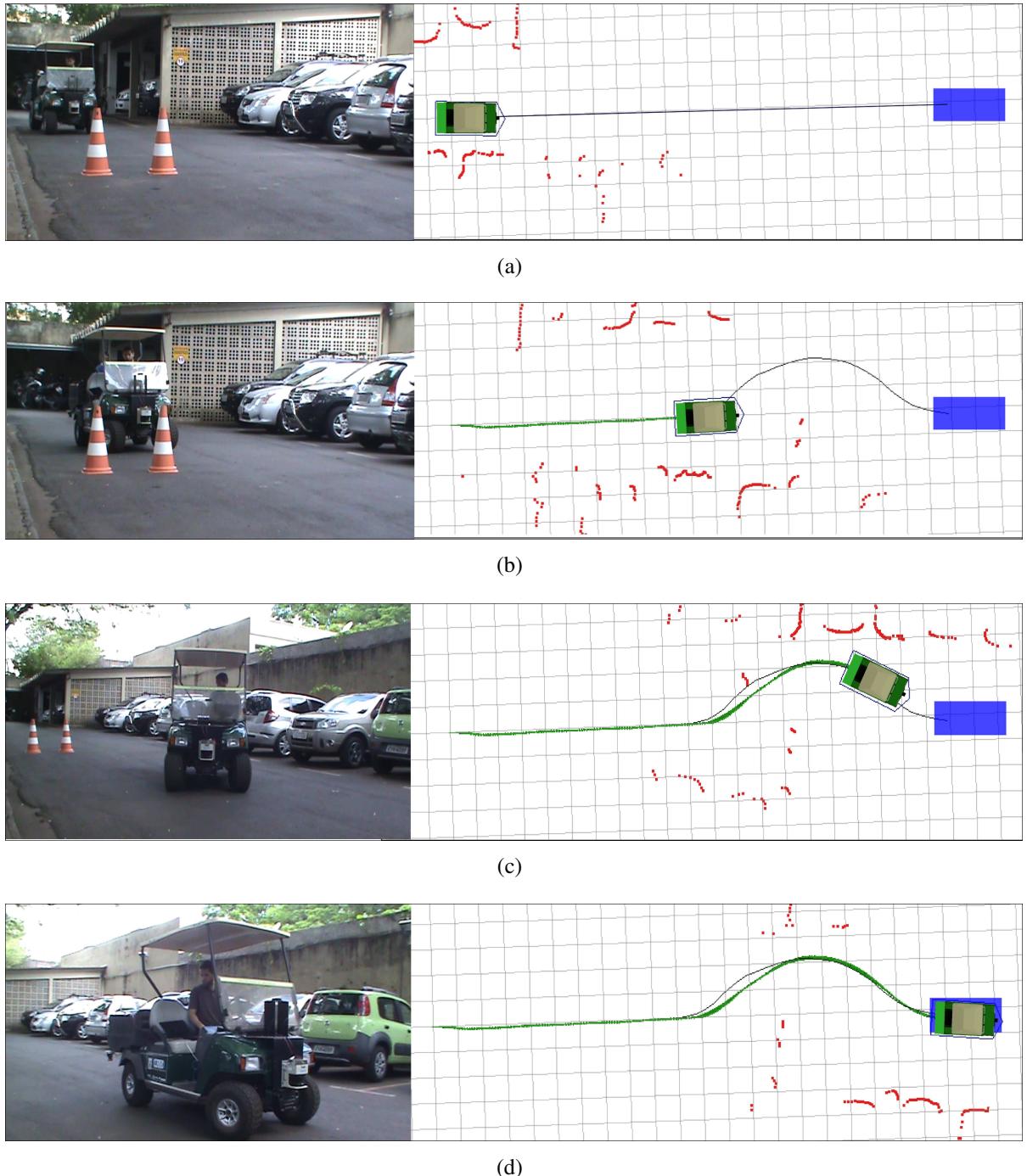


Figura 5.5: Teste de planejamento de trajetória em ambiente real. (a) Trajetória em linha reta. Obstáculos destacados em vermelho e destino representado por um retângulo azul. (b) Replanejamento apóis detecção de obstáculos em rota de colisão. (c) Execução da nova trajetória. (d) Finalização da trajetória.

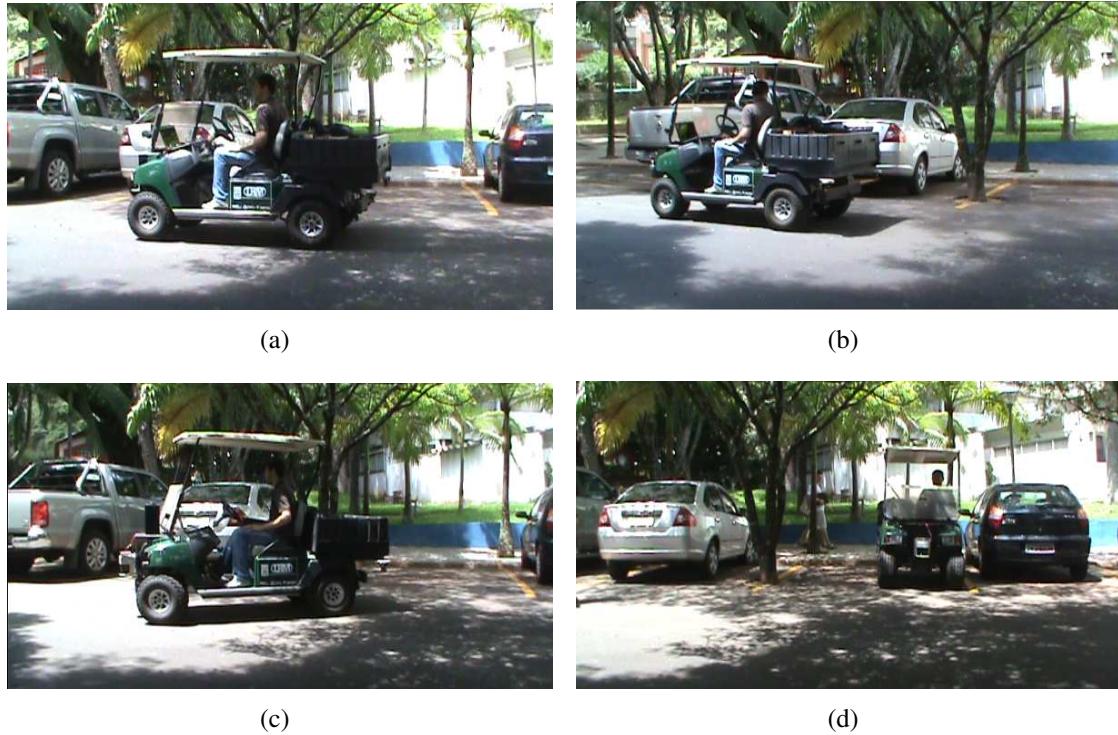


Figura 5.6: Teste de estacionamento em vaga perpendicular em ambiente real. (a) Busca por vaga perpendicular no lado direito do veículo. (b) Início da manobra de estacionamento. (c) Execução da trajetória encontrada. (d) Finalização da trajetória.

5.7 Considerações finais

O capítulo apresentou os resultados do sistema desenvolvido ilustrando as manobras de estacionamento realizadas em ambiente simulado, além dos primeiros testes dos subsistemas individualmente em ambiente real. O sistema conseguiu realizar 83% de manobras válidas. O objetivo, em trabalho futuros, é aumentar essa confiabilidade. Para isso, os sistemas de localização e controle continuam sendo aprimorados.

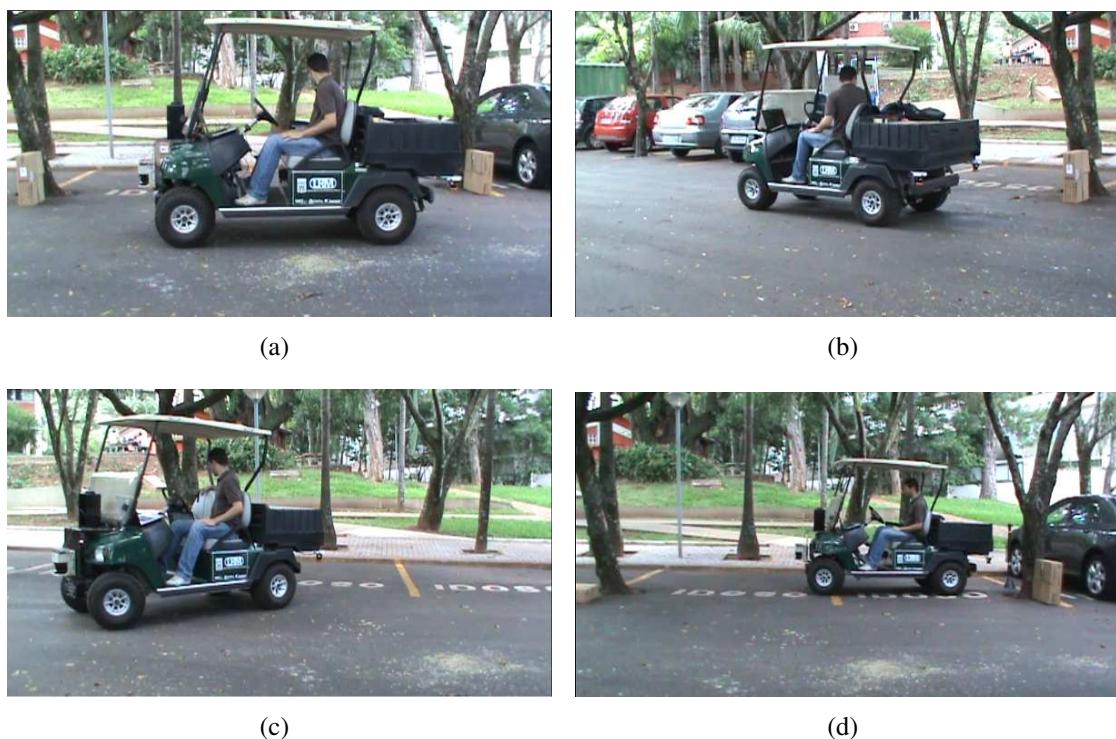


Figura 5.7: Teste de estacionamento em vaga paralela em ambiente real. (a) Busca por vaga paralela no lado direito do veículo. (b) Início da manobra de estacionamento. (c) Execução da trajetória encontrada. (d) Finalização da trajetória.

Conclusão

O sistema de trânsito enfrenta todos os dias diferentes problemas. O aumento do número de veículos, a imprudência dos condutores e o alto índice de acidentes são alguns exemplos. Inúmeros trabalhos têm sido propostos para melhorar a qualidade de vida no trânsito e a comunidade acadêmica, além da indústria, são participantes ativos na realização desses trabalhos. Uma das abordagens que estão sendo postas em prática é a integração de sistemas inteligentes com veículos. O intuito é que o veículo seja capaz de interagir com o ambiente, por meio de sensores, e tomar decisões em situações específicas. Em alguns casos a presença do motorista poderia ser dispensada. Essa autonomia poderá ajudar pessoas portadoras de deficiência e idosos, por exemplo, a se deslocarem em ambientes de trânsito de forma segura.

Esta dissertação apresenta um sistema de planejamento de trajetória para estacionamento de veículos autônomos, utilizando um planejador baseado em malha de estados em conjunto com o algoritmo AD*. Essa abordagem garante gerar trajetórias válidas em ambientes parcialmente conhecidos dentro de um tempo predefinido. Características importantes para atuação em ambientes reais.

Inicialmente foram pesquisados os principais tipos de sensores aplicados na robótica móvel. Levando em consideração as características de alta frequência para captura dos dados e a elevada faixa de alcance, optamos pelo uso do sensor laser. Neste trabalho esse sensor auxilia nos sistemas de localização, mapeamento e detecção de vagas. Foram analisados também os principais métodos de planejamento de trajetória. Dentre os métodos estudados a abordagem

apresentada por Likhachev e Ferguson (2009) se destacou. O método proposto discretiza o ambiente de atuação criando uma malha de estados composta apenas com movimentos válidos do robô. O planejamento é feito utilizando o algoritmo AD* (*Anytime Dynamic A**).

O sistema implementado é dividido em três etapas: (I) localização da vaga de estacionamento, (II) planejamento de trajetória e (III) controle. Baseado nas informações do laser um espaço classificado como vaga é detectado. Após essa etapa o sistema de planejamento é responsável por gerar um caminho a partir da posição atual do veículo até dentro da vaga de estacionamento. A etapa final é a execução do controle do veículo, onde comandos de velocidade e esterçamento são enviados ao veículo objetivando manter o mesmo na trajetória gerada.

O ROS (*Robot Operating System*) foi utilizado como plataforma base para o desenvolvimento do sistema e garantiu eficiência e robustez na integração dos subsistemas implementados. Essa ferramenta provê um grande suporte ao desenvolvimento de sistemas robóticos, incluindo simulação, conexão com diversos sensores e controle de atuadores.

Esse trabalho contribui na resolução do problema de planejamento de trajetória para estacionamento de veículos autônomos, atuando em três tipos de vaga: paralela, perpendicular (90°) e diagonal (45°). Várias abordagens tratam da tarefa de estacionamento autônomo, tanto no campo acadêmico como na indústria, mas ainda não validaram uma abordagem que consiga atuar de forma unificada nos três modelos de vaga em ambientes reais. Os produtos comerciais geralmente atuam em situações bastante específicas, limitando tipos de vaga e necessitando de intervenção humana em alguns momentos, por exemplo.

Esse trabalho também contribuiu para o projeto de Laboratório de Robótica Móvel do ICMC-USP, que tem com proposta o desenvolvimento de um veículo autônomo. O objetivo é que esse veículo seja capaz de navegar em ambientes urbanos sem a necessidade de um motorista. Dessa forma, o sistema de estacionamento desenvolvido poderá fazer parte do sistema de navegação desse veículo.

6.1 Trabalhos futuros

Dentre os trabalhos futuros inclui o teste do sistema implementado no veículo real. Para isso, está sendo implementado o sistema de controle de velocidade para a plataforma CaRINA, algo essencial para validação do sistema. Além disso, pretende-se aprimorar o sistema de detecção de vagas objetivando aumentar sua robustez. Podem ser realizados também testes com o veículo real em vagas menores para validar o sistema de controle.

Referências Bibliográficas

- ANDRADE, K. O. *Sistema neural reativo para o estacionamento paralelo com uma única manobra em veículos de passeio.* Dissertação de mestrado, Escola de Engenharia de São Carlos - Universidade de São Paulo, São Carlos-SP., 2011.
- BEEVERS, K. Topological mapping and map merging with sensing-limited robots. v. 2004, n. May, 2004.
Disponível em <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.117.4778&rep=rep1&type=pdf>
- BEKEY, G. A. *Autonomous robots: From biological inspiration to implementation and control.* MIT Press, 2005.
- BONNIFAIT, P.; JABBOUR, M.; CHERFAOUI, V. Autonomous navigation in urban areas using gis-managed information. *International Journal of Vehicle Autonomous Systems*, v. 6, n. 1-2, p. 83–103, 2008.
- BUEHLER, M.; IAGNEMMA, K.; SINGH, S., eds. *The darpa urban challenge: Autonomous vehicles in city traffic, george air force base, victorville, california, usa*, v. 56 de *Springer Tracts in Advanced Robotics*, Springer, 2009.
- CARVALHO, A. C. P. L. F.; KOWALTOWSKI, T. *Atualizações em informática.* PUC-RIO; Bento Gonçalves: SBC, 2009.
- CENSI, A. An ICP variant using a point-to-line metric. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, 2008.
Disponível em <http://purl.org/censi/2007/plicp>
- CHOSET, H.; LYNCH, K. M.; HUTCHINSON, S.; KANTOR, G.; BURGARD, W.; KAVRAKI, L. E.; S, T. *Principles of robot motion theory, algorithms, and implementations.* Mit Press, 625 p., 2005.

- CONGDON, C. B.; HUBER, M.; BATES, C.; MARC, C.; KORTENKAMP, D.; BIDLACK, C.; COHEN, C.; HUFFMAN, S.; KOSS, F.; RASCHKE, U.; WEYMOUTH, T.; KONOLIGE, K.; MYERS, K.; SAFFIOTTI, A.; DANIELA, E. R. Carmel vs. flakey: A comparison of two robots. 1993.
- DARPA Darpa grand challenge. <Http://www.darpa.mil/grandchallenge04/index.htm>, Acesso em 07 de outubro, 2010a.
- DARPA Darpa urban challenge. <Http://www.darpa.mil/grandchallenge>, Acesso em 07 de outubro, 2010b.
- DENATRAN Denatran. <Http://www.denatran.gov.br/frota.htm>, Acesso em 16 de janeiro, 2011.
- DOLGOV, D.; THRUN, S.; MONTEMERLO, M.; DIEBEL, J. Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research*, 2010.
- DUDEK, G.; JENKIN, M. *Computational principles of mobile robotics*. Cambridge University Press, 280 p., 2000.
- DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localisation and mapping (slam): Part i the essential algorithms. *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, v. 2, p. 2006, 2006.
- ELROB Elrob - european land robot. <Http://www.elrob.org/>, Acesso em 18 de junho, 2011.
- FOX, D.; BURGARD, W.; DELLAERT, F.; THRUN, S. Monte carlo localization: Efficient position estimation for mobile robots. In: *IN PROC. OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI)*, 1999a, p. 343–349.
- FOX, D.; BURGARD, W.; THRUN, S. The dynamic window approach to collision avoidance. *Robotics Automation Magazine, IEEE*, v. 4, n. 1, p. 23 –33, 1997.
- FOX, D.; BURGARD, W.; THRUN, S. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, v. 11, p. 391–427, 1999b.
- FRAZZOLI, E.; DAHLEH, M. A.; FERON, E. Real-time motion planning for agile autonomous vehicles. *AIAA JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS*, v. 25, p. 116–129, 2000.
- GAZEBO Simulador gazebo. <Http://gazebosim.org/about.html>, Acesso em 09 de fevereiro, 2012.
- GILLESPIE, T. D. *Fundamentals of vehicle dynamics*, v. 400. Society of Automotive Engineers, 1992 p., 1992.
Disponível em <http://www.sae.org/technical/books/R-114>

- GUIZZO, E. How google's self-driving car works. <Http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works>, Acesso em 16 de fevereiro de 2012, 2011.
- HAN, L.; DO, Q. H.; MITA, S. Unified path planner for parking an autonomous vehicle based on rrt. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on Robotics and Automation*, 2011, p. 5622 –5627.
- HATA, A. *Mapeamento de ambientes externos utilizando robôs móveis*. Dissertação (mestrado em ciência da computação e matemática computacional), Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo, São Carlos-SP, 2010.
- HEINEN, M. R.; OSÓRIO, F. S.; HEINEN, F. J. Estacionamento de um veículo de forma autônoma simulado em um ambiente tridimensional realístico. *XIV Seminário de Computação (SEMINCO) - FURB*, 2005.
- HONÓRIO, L. M.; LUIZ L. G. VERMMA AND, L. M. G.; VIDIGAL, M. Uma metodologia para aprendizado supervisionado aplicada em veículos inteligentes. *XVIII Congresso Brasileiro de Automática*, 2010.
- IBGE Ibge - instituto brasileiro de geografia e estatística. <Http://www.ibge.gov.br>, Acesso em 17 de janeiro, 2011.
- JEEVAN, P.; HARCHUT, F.; MUELLER-BESSLER, B.; HUHNKE, B. Realizing autonomous valet parking with automotive grade sensors. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, p. 3824 –3829.
- JUNIOR, J. *Um algoritmo baseado em Árvores aleatórias de exploração rápida para navegação de robôs móveis*. Dissertação (mestrado em mecatrônica), Universidade Federal da Bahia., 2008.
- KAVRAKI, L.; KOLOUNTZAKIS, M.; LATOMBE, J.-C. Analysis of probabilistic roadmaps for path planning. *Robotics and Automation, IEEE Transactions on*, v. 14, n. 1, p. 166 –171, 1998.
- KNEPPER, R. A.; MASON, M. T. Improved hierarchical planner performance using local path equivalence. In: *International Conference on Intelligent Robots and Systems (IROS)*, 2011.
- LAKEMEYER, G.; NEBEL, B., eds. *Exploring artificial intelligence in the new millennium*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.
- LAVALLE, S. M. Rapidly-exploring random trees: A new tool for path planning. In, v. 129, n. 98-11, p. 98–11, 1998.
Disponível em <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Rapidly-exploring+random+trees:+A+new+tool+for+path+planning#0>
- LAVALLE, S. M. *Planning algorithms*. Cambridge, U.K.: Cambridge University Press, available at <http://planning.cs.uiuc.edu/>, 2006.

- LAVALLE, S. M.; KUFFNER, J. J. Randomized kinodynamic planning. *The International Journal of Robotics Research*, v. 20, n. 5, p. 378–400, 2001.
Disponível em <http://ijr.sagepub.com/cgi/doi/10.1177/02783640122067453>
- LAVALLE, S. M.; KUFFNER, J. J.; JR. Rapidly-exploring random trees: Progress and prospects. In: *Algorithmic and Computational Robotics: New Directions*, 2000, p. 293–308.
- LEONARD, J. J.; WHYTE, D. H. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, v. 7, n. 3, 1991.
- LIKACHEV, M.; FERGUSON, D. Planning long dynamically feasible maneuvers for autonomous vehicles. *Int. J. Rob. Res.*, v. 28, p. 933–945, 2009.
Disponível em <http://dl.acm.org/citation.cfm?id=1577179.1577184>
- LIKACHEV, M.; FERGUSON, D.; GORDON, G.; STENTZ, A. T.; THRUN, S. Anytime dynamic a*: An anytime, replanning algorithm. In: *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2005.
- LIKACHEV, M.; GORDON, G.; THRUN, S. Ara*: Anytime a* with provable bounds on sub-optimality. In: *IN ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 16: PROCEEDINGS OF THE 2003 CONFERENCE (NIPS-03)*, MIT Press, 2004.
- MAGALHÃES, A. C. *Planejamento cinemático-dinâmico de movimento com desvio local de obstáculos*. Dissertação de mestrado, Escola de Engenharia de São Carlos - Universidade de São Paulo, São Carlos-SP., 2013.
- MARKEVIC, I.; KJAER-NIELSEN, A.; PAUWELS, K.; JENSEN, L. B. W.; CHUMERIN, N.; VIDUGIRIENE, A.; TAMOSIUNAITE, M.; HULLE, M. V.; KRUERGER, N.; ROTTER, A.; WOERGOETTER, F. The driving school system: Learning automated basic driving skills from a teacher in a real car. *International Journal of Vehicle Autonomous Systems*, 2011.
- MEEUSSEN, W. Robot pose ekf. Http://www.ros.org/wiki/robot_pose_ekf/, Acesso em 04 de outubro, 2012.
- MONTEMERLO, M.; BECKER, J.; BHAT, S.; DAHLKAMP, H.; DOLGOV, D.; ETTINGER, S.; HAENNEL, D.; HILDEN, T.; HOFFMANN, G.; HUHNKE, B.; JOHNSTON, D.; KLUMPP, S.; LANGER, D.; LEVANDOWSKI, A.; LEVINSON, J.; MARCIL, J.; ORENSTEIN, D.; PAEFFGEN, J.; PENNY, I.; PETROVSKAYA, A.; PFLUEGER, M.; STANEK, G.; STAVENS, D.; VOGT, A.; THRUN, S. Junior: The stanford entry in the urban challenge. *J. Field Robot.*, v. 25, p. 569–597, 2008.
Disponível em <http://dl.acm.org/citation.cfm?id=1405647.1405651>
- MONTEMERLO, M.; THRUN, S.; DAHLKAMP, H.; STAVENS, D. Winning the darpa grand challenge with an ai robot. In: *In Proceedings of the AAAI National Conference on Artificial Intelligence*, 2006, p. 17–20.
- MURPHY, R. R. *An introduction to ai robotics*. MIT Press, 2000.

- ALVES BARBOSA DE OLIVEIRA VAZ, D.; INOUE, R.; GRASSI, V. Kinodynamic motion planning of a skid-steering mobile robot using rrt*. In: *Robotics Symposium and Intelligent Robotic Meeting (LARS), 2010 Latin American*, 2010, p. 73 –78.
- OSÓRIO, F. S.; HEINEN, F. J.; FORTES, L. Controle inteligente de veículos autônomos: Automatização do processo de estacionamento de carros. *X Seminário de Computação (SEM-INCO) - FURB*, 2001.
- PETROVSKAYA, A.; THRUN, S. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots Journal*, v. 26, n. 2-3, p. 123–139, 2009.
- PIVTORAIKO, M.; KELLY, A. Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. In: *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, 2005, p. 3231 – 3237.
- ROS Ros - robot operating system. [Http://www.ros.org/](http://www.ros.org/), Acesso em 09 de fevereiro, 2012.
- SIMMONS, R.; KOENIG, S. Probabilistic robot navigation in partially observable environments. In: *In Proceedings of IJCAI-95*, IJCAI, Inc, 1995, p. 1080–1087.
- SMITH, R.; SELF, M.; CHEESEMAN, P. Autonomous robot vehicles. cap. Estimating uncertain spatial relationships in robotics, New York, NY, USA: Springer-Verlag New York, Inc., p. 167–193, 1990.
Disponível em <http://dl.acm.org/citation.cfm?id=93002.93291>
- STANEK, G.; LANGER, D.; MULLER-BESSLER, B.; HUHNKE, B. Junior 3: A test platform for advanced driver assistance systems. In: *Intelligent Vehicles Symposium (IV), 2010 IEEE*, 2010, p. 143 –149.
- STENTZ, A. The focussed d* algorithm for real-time replanning. In: *In Proceedings of the International Joint Conference on Artificial Intelligence*, 1995, p. 1652–1659.
- THRUN, S. Bayesian landmark learning for mobile robot localization. *Machine Learning*, v. 33, p. 41–76, 1998.
- THRUN, S. Learning occupancy grids with forward sensor models. *Autonomous Robots*, v. 15, p. 111–127, 2002.
- THRUN, S.; BURGARD, W.; FOX, D. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA: IEEE, 2000.
- THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic robotics*. MIT Press, 2005.
- THRUN, S.; MONTEMERLO, M.; DAHLKAMP, H.; STAVENS, D.; ARON, A.; DIEBEL, J.; FONG, P.; GALE, J.; HALPENNY, M.; HOFFMANN, G.; LAU, K.; OAKLEY, C.; PALATUCCI, M.; PRATT, V.; STANG, P.; STROHBAND, S.; DUPONT, C.; JENDROSSEK, L. E.; KOELEN, C.; MARKEY, C.; RUMMEL, C.; NIEKERK, J. V.; JENSEN, E.; ALESSANDRINI, P.; DAVIES, G. B. B.; ETTINGER, S.; KAEHLER, A.; NEFIAN, A.; MAHONEY, P.

- Stanley, the robot that won the darpa grand challenge. *Journal of Field Robotics*, v. 23, p. 661–692, 2006.
- XIONG, N. Multi-sensor management for information fusion: issues and approaches. *Information Fusion*, v. 3, n. 2, p. 163–186, 2002.
Disponível em <http://linkinghub.elsevier.com/retrieve/pii/S1566253502000556>
- ZHANG, S.; CHEN, G. A new model-free fuzzy logic controller for truck-parking. *Sixth International Conference on Intelligent Systems Design and Applications*, p. 49–54, 2006.
- ZHAO, P.; CHEN, J.; MEI, T.; LIANG, H. Dynamic motion planning for autonomous vehicle in unknown environments. *Intelligent Vehicles Symposium (IV), 2011 IEEE*, p. 284–289, 2011.