Universidade Federal de Juiz de Fora
Faculdade de Engenharia
Pós-graduação em Engenharia Elétrica

Sistema de Navegação Semiautônomo para Robótica Móvel Assistiva Baseado em Movimentos de Cabeça e Comandos de Voz

Guilherme Marins Maciel

JUIZ DE FORA
JANEIRO, 2018

Sistema de Navegação Semiautônomo para Robótica Móvel Assistiva Baseado em Movimentos de Cabeça e Comandos de Voz

Guilherme Marins Maciel

Universidade Federal de Juiz de Fora

Faculdade de Engenharia

Engenharia Elétrica

Mestrado em Engenharia Elétrica

Orientador: Dr. Ivo Chaves da Silva Junior

Coorientador: Dr. André Luís Marques Marcato

JUIZ DE FORA

JANEIRO, 2018

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Maciel, Guilherme Marins.
Sistema de navegação semiautônomo para robótica móvel assistiva baseado em movimentos de cabeça e comandos de voz / Guilherme Marins Maciel. -- 2018.
99 f. : il.

Orientador: Ivo Chaves da Silva Junior
Coorientador: André Luís Marques Marcato
Dissertação (mestrado acadêmico) - Universidade Federal de Juiz de Fora, Faculdade de Engenharia. Programa de Pós Graduação em Engenharia Elétrica, 2018.

1. Cadeira de Rodas Inteligente. 2. Controle Compartilhado. 3. Robótica Assistiva. I. Silva Junior, Ivo Chaves da , orient. II. Marcato, André Luís Marques, coorient. III. Título.

Sistema de Navegação Semiautônomo para Robótica Móvel Assistiva Baseado em Movimentos de Cabeça e Comandos de Voz

Guilherme Marins Maciel

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO FACULDADE DE ENGENHARIA DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA.

Aprovada por:

Dr. Ivo Chaves da Silva Junior
Universidade Federal de Juiz de Fora

Dr. André Luís Marques Marcato
Universidade Federal de Juiz de Fora

Dr. Tales Pulinho Ramos
Instituto Federal do Sudeste de Minas Gerais

Dr. Leonardo Willer de Oliveira
Universidade Federal de Juiz de Fora

JUIZ DE FORA

22 DE JANEIRO, 2018

Resumo

O campo da robótica assistiva lida com as tecnologias voltadas para pessoas com imparidades físicas. Alguns deficientes possuem tetraplegia e não podem movimentar os membros inferiores e superiores mas, dependendo do grau da lesão, podem acionar as musculaturas do ombro e pescoço. Visando este grupo, o presente trabalho propõe uma arquitetura de sistema semiautônomo para cadeiras de rodas inteligentes (CRIs) baseada na plataforma *Robotic Operating System* (ROS), embarcado em uma *Raspberry Pi 3*, com ênfase em baixo custo e não necessidade de localização global. É apresentada uma proposta de interface homem-máquina baseada em comandos de voz, na qual uma central de controle utiliza uma Rede de Petri para configurar e administrar o sistema. As transições são disparadas por um conjunto de frases. Depois de cada evento, o usuário recebe uma frase de retorno através de alto-falantes. Para o sistema de navegação é utilizado um controle compartilhado de velocidade, em que, utiliza-se uma Unidade de Medição Inercial (IMU) para reconhecer o padrão de movimento desejado através de medições da inclinação da cabeça do usuário. Paralelamente, um algoritmo inteligente emprega uma câmera de profundidade para reconhecer os obstáculos nos arredores e atuar em conjunto no controle, de modo a aumentar a manobrabilidade e segurança. Uma metodologia de travessia de passagens estreitas de forma autônoma é uma outra técnica proposta. Nesta técnica duas extremidades de um acesso delgado são reconhecidas por meio da câmera de profundidade e um controlador não-linear realiza a tarefa. Nos resultados, as metodologias propostas foram analisadas através de modelos de CRIs em Matlab e na plataforma de simulação Gazebo. Para testes práticos, foi utilizado um robô diferencial Pionner-P3DX. Os resultados exibidos nesta dissertação se revelaram promissores, evidenciando a aplicabilidade do sistema.

Palavras-chave: Cadeira de Rodas Inteligente, Controle Compartilhado, Tecnologia Assistiva.

Abstract

The field of assistive robotics works with technologies aimed at people with physical impairments, some handicapped have quadriplegia and cannot move the lower and upper limbs, but, depending on the degree of injury, the shoulder and neck musculature can be used. Aiming at this group, the present work proposes a semiautonomous system architecture for smart wheelchairs based on the Robotic Operating System (ROS) platform, embedded in a Raspberry Pi 3, with emphasis on low cost and no need for global localization. A proposal for a human-machine-interface based on voice commands is presented, where the control center uses a Petri Net to configure and administer the system. Transitions are triggered by a set of phrases and after each event, the user receives a return phrase through speakers. For the navigation system, a shared control of velocity is used. An Inertial Measurement Unit (IMU) is applied to recognize the desired movement pattern through measurements of the user's head inclination. In parallel, an intelligent algorithm employs a depth camera to recognize obstacles around and work together in control, increasing maneuverability and safety. Another technique proposed is a methodology of autonomous narrow passages crossing, the depth camera recognizes the borders of thin access and a non-linear controller performs the task. In the results, the proposed methodologies were analyzed using simulated models in the Matlab and the Gazebo platform. For practical tests, a Pioneer-P3DX differential robot was used, the results presented in this dissertation were promising, evidencing the applicability of the system.

Keywords: Smart Wheelchair, Shared Control, Assistive Technology.

Agradecimentos

Primeiramente gostaria de agradecer aos meus orientadores Ivo Junior e André Marcato pela amizade e todo incentivo para meu desenvolvimento profissional. Aos colegas do Grupo de Robótica Inteligente (GRIn) e do Grupo de Otimização e Heurística Bioinspirada (GOHB) da UFJF pelo apoio e troca de conhecimentos, em especial, à Diego Carvalho e Fabrício Coelho. Aos professores e amigos Leonardo Olivi, Daniel Fernandes, Exuperry Costa, Leonardo Honório e Leonardo Willer, pelas contribuições e ensinamentos ao longo destes dois anos.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

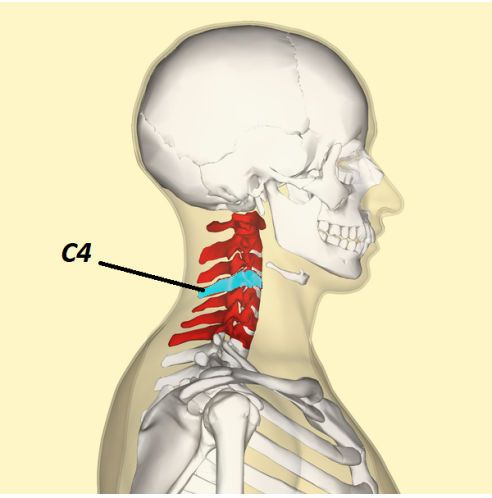
APF	Campos Potenciais Artificiais (Artificial Potential Fields)
CRI	Cadeira de Rodas Inteligente
EEG	Eletroencefalograma
EFK	Filtro de Kalman Estendido (Extended Kalman Filter)
EMG	Eletromiograma
HMI	Interface Homem-Máquina (Human Machine Interface)
IMU	Unidade de Medição Inercial (Inertial Measurement Unit)
KF	Filtro de Kalman (Kalman Filter)
RP	Rede de Petri
PNE	Portador de Necessidade Especial
ROS	Robot Operating System

1 Introdução

A robótica assistiva é a área da robótica que desenvolve recursos para ampliar habilidades funcionais de pessoas com deficiência. Pesquisas neste campo contribuem efetivamente para qualidade de vida dos Portadores de Necessidades Especiais (PNEs) (CAMPANELI; MESTRIA, 2013). De acordo com IBGE (2015), no cenário brasileiro de 2010, existiam cerca de 45,6 milhões de pessoas com algum tipo de deficiência, sendo que mais de 4,5 milhões com limitações físicas severas, e mais de 700 mil brasileiros incapazes de caminhar. O mesmo estudo constatou que cerca de 23% dos brasileiros possuem rendimentos inferiores ao salário mínimo, 50% das famílias sobreviviam com menos de $R\$1200,00$. Fica evidente a importância de tecnologias assistivas economicamente viáveis para a população.

Alguns deficientes apresentam tetraplegia, que se refere a perda sensitiva nos segmentos cervicais da coluna vertebral, comprometendo a função dos braços, bem como no tronco, pernas e órgão pélvicos. Dependendo do tipo da lesão na coluna cervical, normalmente abaixo do nível C4 (Figura 1.1), o indivíduo possui contrações visíveis e até mesmo movimentos normais da musculatura acima dos ombros, podendo movimentar sua cabeça (DITUNNO et al., 1994).

Figura 1.1: Segmentos da Coluna Cervical

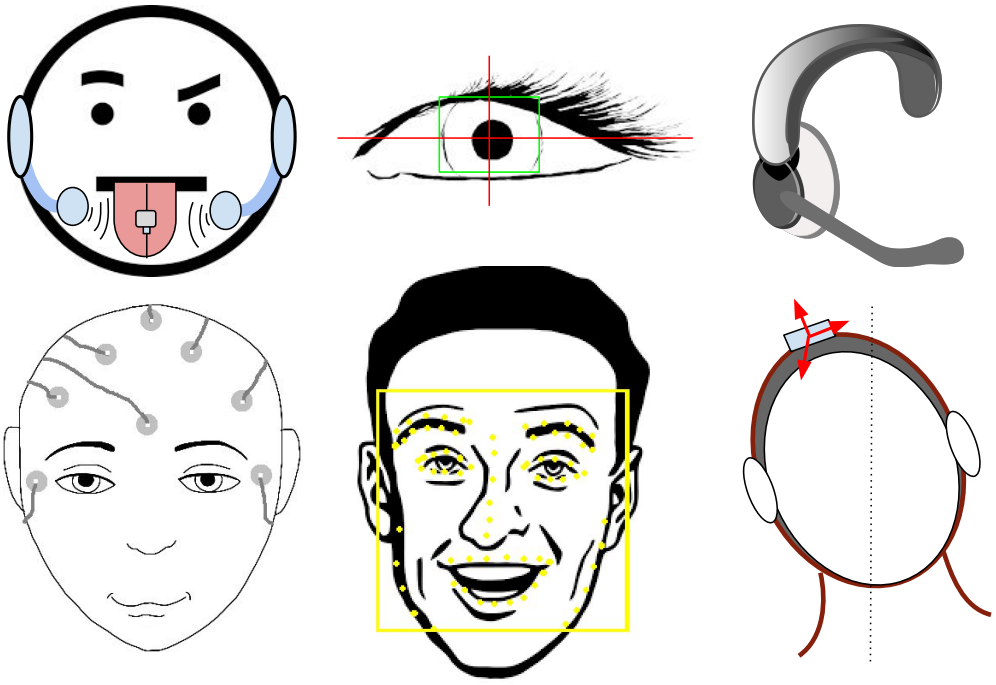


Fonte: (Wikimedia Commons, 2017)

Muitas pessoas com problemas motores podem utilizar as tradicionais cadeiras de rodas, manuais ou motorizadas, para se locomover de forma independente. Um segmento da comunidade deficiente, incluindo os tetraplégicos, é incapaz de utilizá-las. Para acomodar este segmento, pesquisas vêm usando tecnologias originais da robótica móvel para desenvolvimento de Cadeiras de Rodas Inteligentes (CRIs) (SIMPSON, 2005). Diferente das cadeiras motorizadas comuns, a CRI possui processamento, sensores e *interfaces* de comunicação.

Para usuários quadriplégicos são necessários controles não-manuais (*hands-free*), pode-se utilizar, por exemplo: Movimentos da língua, rastreamento dos olhos, comandos de voz, eletrograma, reconhecimento facial e inclinação de cabeça (PEREIRA, 2017), como apresentado na Figura 1.2.

Figura 1.2: Tecnologias *hands-free* para Robótica Móvel Assistiva



Fonte: Elaborada pelo autor

As CRIs são comumente utilizadas em sistemas de controle compartilhado, onde o usuário atua diretamente nas decisões de atuação por meio das *interfaces* de comunicação. Ao mesmo tempo, um sistema supervisor atua em paralelo, ajudando o usuário a alcançar seus objetivos em segurança (OLIVI et al., 2014).

Segundo Torkia et al. (2015), reclamações típicas de usuários de cadeiras de rodas

motorizadas incluem a dificuldade em subir rampas, movimentar em locais pequenos, atravessar portas e evitar obstáculos. Entre os pilares do sistema de robótica móvel assistiva pode-se destacar a tarefa de evitar colisões. Esta se torna um grande desafio ao se deparar com passagens estreitas, como portas, corredores e espaços entre móveis. Seja por controle humano ou robótico, esta tarefa é bastante delicada e difícil de ser executada de maneira confiável (BOUCHER et al., 2013), sendo um dos tópicos abordados neste trabalho.

1.1 Revisão da Literatura

A escolha da comunicação homem-máquina é essencial para o funcionamento da cadeira de rodas inteligente, uma vez que seu principal objetivo é dar autonomia de movimento para o cadeirante. Esta escolha considera o tipo de debilidade, para usuários tetraplégicos deve-se utilizar dispositivos "*handsfree*". Diversos trabalhos vêm apresentando tecnologias e metodologias para esse grupo.

Em Pereira (2017) é utilizado reconhecimento facial para atuar sobre as velocidades da CRI. Essa técnica utiliza algoritmos de visão computacional para reconhecer a face e cada um de seus componentes em uma imagem. Os autores realizam um estudo comparando duas categorias: A Joyface, que simula um *joystick* a partir de movimentos do rosto, e o Intelreal RealSense SDK que atua na cadeira a partir de expressões faciais. Apesar de demandar maior esforço físico, os voluntários consideraram que o controle realizando movimentos da cabeça é mais fácil e seguro do que o de expressões faciais.

Outra opção é utilizar os movimentos da língua, os autores de Kim et al. (2013) desenvolveram o sistema *Tongue Drive System*, no qual o participante utiliza um *piercing* metálico na língua. Sensores magnéticos, próximos a bochecha, reconhecem a posição do *pircing* para classificar uma ação de controle.

Apesar de estar sujeito a influência de som externo do ambiente, avanços na área de reconhecimento de fala vem aumentando a acurácia do controle de CRIs por voz. Nesta interface o usuário pronuncia frases para configurar e operar o sistema. Algoritmos de reconhecimento muito utilizados são os Hidden Markov Model e Redes Neurais Artificiais (PACNIK; BENKIC; BRECKO, 2005).

O reconhecimento da voz pode ser realizado por um hardware especializado, como em Chauhan et al. (2016), que utilizou o módulo eletrônico HM2007 configurado para reconhecer até 40 palavras, ou pode-se utilizar um programa de reconhecimento de voz embarcado, como em Naeem, Qadar e Safdar (2014) onde foi aplicado o *Google API* em uma *Raspberry Pi*.

Um sistema que vem ganhando espaço em tecnologias assistivas é a eletrografia, que utiliza sinais biológicos de origem elétrica. Os dois principais são a Eletromiografia (EMG) que utiliza sinais provenientes de contrações musculares e o Eletroencefalograma (EEG) que mede sinais vinculados ao cérebro. Um protocolo muito utilizado no EEG é o P300, que mede a reação do usuário a um estímulo externo que possui um significado desejado, um sinal de potencial positivo ocorre cerca de 300 milissegundos após o estímulo (ESCOLANO; ANTELIS; MINGUEZ, 2012).

Dependendo do grau da inaptidão do tetraplégico, pode-se utilizar de movimento do pescoço. Em Taher et al. (2016) é utilizado EEG e movimentos de cabeça por meio de um aparelho *Emotiv*, além de reconhecimento de voz para configurar a cadeira. Alguns trabalhos utilizam uma *Inertial Measurement Unit* (IMU), para medir a inclinação e os movimentos angulares da cabeça, em Rohmer et al. (2015) o usuário movimenta a cabeça para controlar o apontamento de um *laser* sobre o solo, uma câmera capta esta posição e o sistema faz uma navegação autônoma para este ponto. Pode-se também utilizar o IMU para atuar diretamente nas velocidades da CRI, utilizando o movimento de *pitch* para movimentar para frente/trás, e *roll* para direita/esquerda (BUREAU et al., 2007).

Independente do tipo e interface, tendo em vista a necessidade de missões autônomas na robótica móvel, é indispensável o uso de algoritmos eficientes de localização. A posição do robô não pode ser estimada de forma acurada quando é utilizado somente os sensores odométricos, visto que o erro de sua estimativa cresce de forma cumulativa. Existem diversos algoritmos para a localização de robôs móveis terrestres. Um dos muitos utilizados é o *Extended Kalman Filter* (EKF), que assume que o mapa possui uma coleção de referenciais, que ao serem observados geram uma estimativa da posição global. Utilizando a odometria do robô, corrigida pelas observações de referenciais globais, é realizado o rastreamento de sua posição (THRUN; BURGARD; FOX, 2005).

Existem diversos trabalhos envolvendo aplicação do *Kalman Filter* (KF) na localização de robôs móveis, a maior diferença entre eles está na maneira que a observação é realizada e como se utilizam sensores e padrões medidos do ambiente. A odometria apresenta boa acurácia em pequenos deslocamentos, por isso a mesma é utilizada na etapa de predição alimentando o modelo cinemático do robô diferencial.

Em Espinosa et al. (2011), o EKF é aplicado para que um robô siga a trajetória de um segundo robô líder. Foi utilizado um *laser scanner* para identificar um marcador tridimensional sobre o líder, estimar a posição em relação a este referencial e alimentar um controlador específico. Uma outra solução é espalhar *AR Codes* nas paredes de uma ambiente mapeado, compostos por imagens com códigos específicos e, com uma câmera, estimar a distância à este referencial e atualizar a estimativa de posição global do robô sempre que um for avistado (COELHO et al., 2017).

O KF é utilizado também como fusão sensorial, sendo possível utilizar observações vindas de diversos sensores e fundi-las para obter uma medição mais acurada. Em Ruiz et al. (2014) é desenvolvido um sistema híbrido de navegação para veículos terrestres. Os autores utilizam a fusão de GPS, bússola, odometria, câmera e *laser scanner*. Através da câmera e do laser são identificados objetos chave, que estão representados em um mapa digital, consumando um Método de Localização Referenciada As informações do GPS e da bússola complementam a observação deste sistema.

Após uma localização precisa e acurada, o robô está apto a realizar missões de forma autônoma. No contexto do controle compartilhado em CRIs algumas tarefas como: navegação autônoma, evitar obstáculos, estacionar, seguir paredes e atravessar por passagens estreitas podem ser bastante úteis para o usuário.

Uma técnica clássica para navegar evitando colisões é o Campos Potenciais Artificiais (APF), a qual utiliza forças repulsivas virtuais entre os obstáculos e o robô, baseado-se na distância. É um algoritmo simples, mas apresenta mínimos locais e oscilação em locais estreitos (MOHANTY; PARHI, 2013). Para melhorar o método, Amiryan e Jamzad (2015) propõem um gerador de trajetória baseado em Rapidly-exploring Random Trees (RRT), que divide a trajetória em segmentos em torno dos obstáculos, suavizando-a. Em sua forma padrão, o robô é considerado um ponto adimensional móvel, em Seki et al.

(2008) o corpo é modelado em formato retangular, o campo repulsivo é calculado sobre a parte frontal e traseira e, com isso, ao se considerar a forma do robô, o método se tornou bem mais efetivo.

Para planejar um caminho por uma passagem estreita de forma discreta e autônoma é necessário um número grande de pontos para conseguir se obter uma rota segura. O trabalho Shi, Denny e Amato (2014) propõe uma metodologia baseada em RRT, que desencadeia o crescimento de amostras a partir de pontos em passagens estreitas até que a mesma esteja totalmente mapeada. Com isso a chance de obter uma rota segura cresce assiduamente.

Em Leishman, Horn e Bourhis (2010) é utilizado um *joystick* e uma interface gráfica. As passagens estreitas são identificadas agrupando leituras do *laser scanner* e reconhecendo os espaços vazios. Um algoritmo de visão computacional apresenta as passagens na tela e o usuário seleciona uma opção pela interface gráfica, assim uma trajetória discreta é planejada para a CRI segui-la. Nesse trabalho também é apresentado um algoritmo para seguir paredes de forma eficiente.

Tomari, Kobayashi e Kuno (2012) utiliza tanto um laser scanner, quanto uma câmera de profundidade (RGB-D) para reconhecer a geometria do ambiente e buscar rotas seguras. Um controle semiautônomo é realizado onde o usuário seleciona a direção que deseja partir do rastreamento dos olhos por um *webcam*, e juntamente o algoritmo Vector Field Histogran(VHF) busca a melhor direção para uma trajetória rápida e sem colisões.

O sistema de Boucher et al. (2013) foi desenvovido para CRIs hospitalares, sendo utilizado um *laser scanner* com 360 graus de cobertura e uma interface formada por uma tela iterativa, um *joystick* e um microfone. O sistema reconhece uma lista de frases pelo *software: "Dragon NaturallySpeaking"*. Há um sistema de localização preciso, rotinas de estacionar, passagens estreitas e seguidor de paredes são implementados baseados na localização no mapa conhecido, no qual essas regiões serão previamente reconhecidas. Todo o controle é feito por meio de *waypoints*.

Os autores de Wang, Chen e Liao (2013), que também utilizaram joytick e comandos de voz como interface, apresentam rotinas de navegação autônoma, evitar obstáculos,

estacionar e passagens estreitas, todos utilizando o algoritmo A*(A-estrela) sobre o ambiente discretizado. Para uma localização precisa é utilizado um *laser scanner* e proposto um modelo modificado de Filtro de Partículas. Nos resultados, os autores afirmam que tiverem problemas em atravessar portas, havendo colisões em alguns momentos.

Olivi et al. (2014) propõe uma metodologia baseada em campos vetoriais, a partir da leitura de um *laser scanner* 180 graus é calculado um campo vetorial sobre o robô que carrega informação sobre os obstáculos ao redor, que age de forma a limitar as velocidades da CRI. A navegação assistida através de controle por eletromiografia se tornou mais fácil em relação a evitar os obstáculos e passar por passagens estreitas, e a mesma metodologia foi implementada em controle autônomo.

A partir de uma câmera RGB-D, em Burhanpurkar et al. (2017) é realizado o mapeamento e a localização. As passagens estreitas são identificadas pela nuvem de pontos 3D, baseando-se em sua natureza planar e espaço vazio.

Na última década, diversos trabalhos vêm expondo novas arquiteturas para CRIs, com diferentes interfaces e inteligência computacional aplicadas ao controle compartilhado. A maioria delas utiliza de um *Laser Scanner*, e a plataforma ROS (Robotic Operating System). A Tabela 1.1 contém os principais trabalhos correlatos desta revisão.

Tabela 1.1: Trabalhos Correlatos

Trabalho	**Interface Homem-Máquina**	**Tarefas Robotizadas**	**Sensores Exteroceptivos**
Rofer, Mandel e Laue (2009)	Joystick Analógico, Head-Joystick	Evitar Colisões	Laser Scanner
Leishman, Horn e Bourhis (2010)	Joystick Analógico, Interface Gráfica	Passagens Estreitas, Seguidor de Parede	Laser Scanner, Câmera RGB
Ruzaij e Poonguzhali (2012)	Comando de Voz, Keypad	Evitar Colisões	Sonar
Tomari, Kobayashi e Kuno (2012)	Eye Tracking, Joystick Analógico	Evitar Colisões	Laser Scanner, Câmera RGB-D
Boucher et al. (2013)	Comando de Voz, Joystick Analógico, Interface Gráfica	Seguidor de Parede, Estacionar, Passagens Estreitas	Laser Scanner
Lopes, Pires e Nunes (2013)	EEG, Interface Gráfica	Navegação Autônoma, Evitar Colisões	Laser Scanner
Wang, Chen e Liao (2013)	Joystick Analógico, Comando de voz, Interface Gráfica	Evitar Colisões, Estacionar, Passagens Estreitas, Navegação Autônoma	Laser Scanner
Olivi et al. (2014)	EEG, EMG, Interface Gráfica	Navegação Autônoma, Evitar Colisões, Passagens Estreitas	Laser Scanner
Naeem, Qadar e Safdar (2014)	Comando de voz, Alto-falante	Evitar Colisões	Sonar
Burhanpurkar et al. (2017)	Joystick Analógico, Interface Gráfica	Navegação Autônoma, Passagens Estreitas, Evitar Colisões	Câmera RGB-D
Este Trabalho	Head-Keypad, Comando de voz, Alto-falante	Passagens Estreitas, Evitar Colisões	Câmera RGB-D

1.2 Objetivo e Estrutura do Trabalho

A principal contribuição desta dissertação está na proposta de arquitetura baseada em ROS para Controle Compartilhado em CRIs, com enfoque em usuários tetraplégicos. São utilizados pacotes consagrados e é desenvolvido outros nós em Python. Em comparação aos trabalhos correlatos, tem-se uma arquitetura de menor custo e todos os processos são independentes de localização global, não sendo necessário o conhecimento prévio do mapa ou mapear o ambiente. O sistema, criado para ser embarcado em uma Raspberry Pi 3, utiliza uma câmera RGB-D Asus XtionPRO como sensor exteroceptivo.

É proposto um sistema de navegação assistida baseado em movimentos de cabeça e campos vetoriais, com enfoque em evitar colisões, aumento de manobrabilidade e no reconhecimento dos padrões de cada usuário. A central de controle é baseada em Redes de Petri, na qual os eventos que configuram o sistema são inferidos por comando de voz do usuário. Foi proposto uma nova metodologia, baseado em EKF, de travessia por passagens estreitas para robótica móvel.

O presente trabalho foi divido em 7 capítulos. O Capítulo 2 tem como objetivos resumir os principais fundamentos teóricos e metodologias utilizados como instrumento para o desenvolvimento da dissertação. O Capítulo 3 apresenta a arquitetura completa do sistema, descrevendo toda a parte de *software* e *hardware* e a interação entre eles, além de explicar a central de controle baseada em Redes de Petri e controle por voz. Neste mesmo capítulo são descritos os materiais e o protótipo desenvolvido.

O Capítulo 4 se dedica ao sistema de navegação assistida, descrevendo a interface de movimentos de cabeça, como o treinamento de usuário e o reconhecimentos de padrões. Outro ponto explanado é o tipo de inferência realizada nas velocidades da CRI, explicando atuação do algoritmo inteligente baseado em campos vetoriais para evitar colisões e facilitar a navegação.

O Capítulo 5 apresenta a proposta de algoritmo de controle e planejamento para travessia por passagens estreitas, explicando a parte de reconhecimento das passagens, processo de localização do robô em relação à mesma e a proposta de um controlador para a tarefa. O Capítulo 6 contém os resultados simulados e experimentais. O Capítulo 7 é dedicado às conclusões e trabalhos futuros.

1.3 Publicações Decorrentes

Durante a execução desta pesquisa, alguns artigos em conferências em âmbito nacional e internacional foram aprovados.

1.3.1 Publicações Diretas

- MACIEL, G. M. et al. Planejamento e controle não linear para passagens estreitas em robótica móvel assistiva. XIII Simpósio Brasileiro de Automação Inteligente (SBAI), 2017.

- MACIEL, G. M. et al. Development of a control system for electric wheelchairs based on head movements. In: IEEE. Proceedings of SAI Intelligent Systems Conference (Intellisys), London, UK. [S.l.], 2017.

1.3.2 Publicações Correlatas

- COELHO, F. et al. Filtro de Kalman Estendido baseado em visão computacional e odometria aplicado à localização de robos móveis. In: . [S.l.]: XIII Simpósio Brasileiro de Automação Inteligente (SBAI), 2017.

1.3.3 Publicações Paralelas

- MACIEl, G. M. et al. Utilização de Técnicas Bioinspiradas para Controle de Manipuladores Robóticos . In: . [S.l.]: IEEE/IAS International Conference on Industry Applications. (INDUSCON), 2016.

- SANTOS, A. L. et al. Otimização da Operação de Sistemas de Bateria para Aplicações em Redes de Distribuição de Energia Elétrica. In: . [S.l.]: XIII Simpósio Brasileiro de Automação Inteligente (SBAI), 2017.

2 Fundamentos Teóricos

Este Capítulo apresenta um resumo das técnicas e metodologias utilizadas na elaboração do trabalho, constituindo a estrutura que servirá para os demais capítulos.

2.1 Robô de Rodas Diferencial

As CRIs são consideradas robôs que se enquadram na categoria diferencial. Este tipo de tração consiste em duas rodas idênticas e independentes sobre um mesmo eixo. Pode-se adicionar uma ou mais rodas do tipo *caster* [1] para garantir o equilíbrio estático.

O robô diferencial é não holonômico, em seu próprio referencial possui apenas liberdade de movimento linear em x e angular em z. Essas velocidades dependem da combinação entre as velocidades angulares da roda direita e esquerda $\dot{\phi}_r$ e $\dot{\phi}_l$, além do raio da roda r e a distância entre a roda e o centro b. Esses parâmetros podem ser visualizados na Figura 2.1.

Sabe-se que a velocidade linear das rodas são dadas por $v_{r,l} = r\dot{\phi}_{r,l}$ e a velocidade linear do centro do robô é a média das duas rodas:

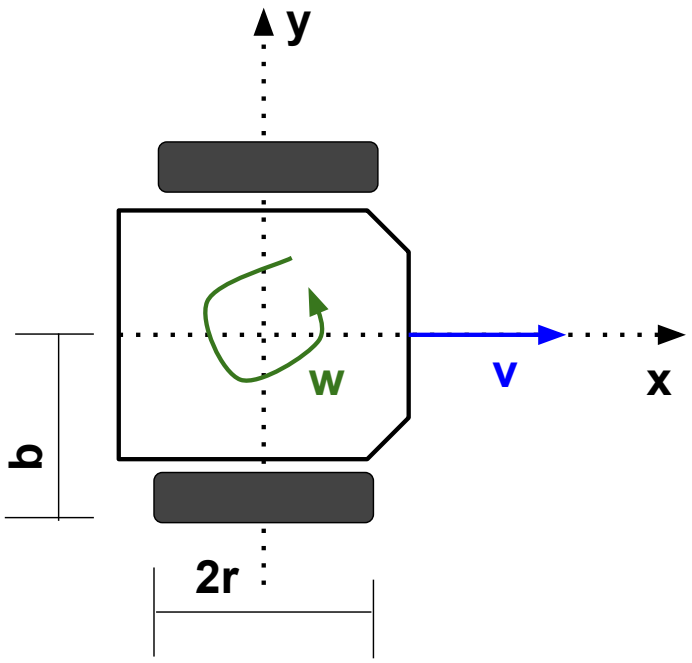
$$v = \frac{v_r + v_l}{2} = \frac{r}{2}(\dot{\phi}_r + \dot{\phi}_l) \quad \boldsymbol{m/s} \tag{2.1}$$

Considerando o centro do robô um eixo fixo, sabe-se que a velocidade linear da roda direita v_r contribui com uma velocidade angular no sentido anti-horário de $\omega_r = v_r/b$ e a roda esquerda no sentido horário de $\omega_l = -v_l/b$, combinando essas duas contribuições tem-se a velocidade angular do corpo do robô:

$$\omega = \frac{\omega_r + \omega_l}{2} = \frac{r}{2b}(\dot{\phi}_r + \dot{\phi}_l) \quad \boldsymbol{rad/s} \tag{2.2}$$

[1] Roda que permite movimentação em múltiplas direções, girando 360 graus sob uma carga.

Figura 2.1: Parâmetros Robô Diferencial



Fonte: Elaborada pelo autor

2.1.1 Modelagem Cinemática

Um robô diferencial se movendo sobre uma superfície plana, pode ser modelado sobre este referencial pela posição e orientação no instante k (NEGENBORN, 2003).

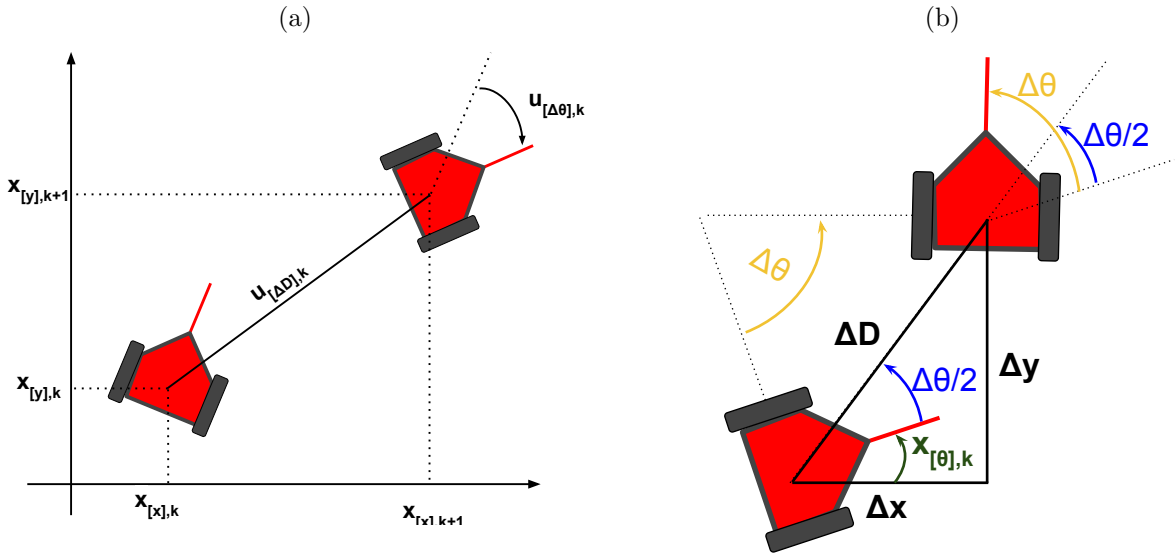
$$\boldsymbol{x}_k = \begin{bmatrix} x_{[x],k} \\ x_{[y],k} \\ x_{[\theta],k} \end{bmatrix} \tag{2.3}$$

Considera-se esta posição referente ao ponto médio entre as rodas tracionadas. O deslocamento relativo deste ponto entre o instantes k e $k+1$ é parametrizado pelo vetor $\boldsymbol{u}_k$, Equação 2.4, onde $u_{[\Delta D],k}$ é a distância entre o centro do robô nesses dois instantes, e $u_{[\Delta\theta],k}$ é a diferença angular. A Figura 2.2a apresenta a modelagem de um deslocamento graficamente.

$$\boldsymbol{u}_k = \begin{bmatrix} \Delta D_k \\ \Delta\theta_k \end{bmatrix} \tag{2.4}$$

Para computar as mudanças de localização, considera-se que o robô manteve

Figura 2.2: Parametrização de Deslocamento



Fonte: Elaborada pelo autor

suas velocidades linear e angular contantes entre instantes k e $k+1$. Assim, é realizado um trajeto circular de raio fixo sobre um centro de curvatura. Nesta situação pode-se considerar que a orientação da reta que liga o centro do robô nos instantes k e $k+1$ é a média da orientação do robô nestes dois momentos, com isto, a diferença angular entre o robô e esta reta é $\Delta\theta/2$, como mostrado na Figura 2.2b. Finalmente, chega-se na Equação (2.5), que modela a cinemática do robô diferencial.

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \begin{bmatrix} \Delta D_k \cos(x_{[\theta]k} + \Delta\theta_k/2) \\ \Delta D_k \sin(x_{[\theta]k} + \Delta\theta_k/2) \\ \Delta\theta_k \end{bmatrix} \tag{2.5}$$

Os robôs móveis comumente apresentam sistemas de odometria, onde, a partir de um sensor *encoder* em cada roda, é possível determinar os deslocamento angular de cada roda. De acordo com os aspectos construtivos, pode-se estimar ΔD *e* $\Delta\theta$ e utilizar a Equação (2.5) para localizar o robô em tempo real no espaço 2D. Infelizmente, este tipo de localização apresenta um erro integrativo, e está susceptível ao deslizamento dos pneus e erros construtivos.

2.1.2 Modelagem Dinâmica

Assim como robô diferencial, as CRIs costumam possuir drives de acionamento independentes para controlar as velocidades de rodas e computar a odometria. Este *drive* já possui um controlador em malha fechada para acionar um par independente de motores de corrente contínua, variando as tensões de entrada de acordo com as velocidades requisitadas. Assim pode ser feito um controle em alto nível, requisitando diretamente as velocidades linear e angular, e a malha intrínseca será responsável pelo controle em baixo nível.

Por considerarmos que CRI trabalha com velocidades relativamente baixas, e possui motores e acionamento robustos, é comum considerar que o tempo de respostas das velocidades é desprezível. São ignoradas as força e inércia envolvidas. Com isso, o modelo dinâmico pode ser descrito apenas pela Equação (2.6) (FAHIMI, 2008), onde a entrada de controle é exatamente as velocidades requisitadas, e os motores irão impor esta velocidade ao sistema em um tempo insignificante.

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} \cos(x_{[\theta]}(t)) & 0 \\ \sin(x_{[\theta]}(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{2.6}$$

Caso seja necessário pode-se, através de mecânica lagrangiana do corpo rígido conjuntamente com a dinâmica do motor de corrente contínua de imã permanente, obter um modelo mais condizente com a realidade, como apresentando em Costa (2017). As variáveis de estado envolvidas são representadas por:

$$\boldsymbol{x}(t) = [x_{[x]}(t), x_{[y]}(t), x_{[\theta]}(t), v(t), \omega(t), \dot{\phi}_r(t), \dot{\phi}_l(t)]^T \tag{2.7}$$

onde $x_{[x]}(t)$, $x_{[y]}(t)$ e $x_{[\theta]}(t)$ são as coordenadas cartesianas e orientação; $v(t)$ e $\omega(t)$ são as velocidades linear e angular respectivamente; $\dot{\phi}_r(t)$ e $\dot{\phi}_l(t)$ são as velocidades angulares, em radianos por segundo, das rodas direita e esquerda. As variáveis de entrada, $\boldsymbol{u} = [E_r(t), E_r(t)]^T$, são as tensões de entrada nos motores das rodas direita e esquerda respectivamente. Assim a dinâmica do robô diferencial pode se modelada pela Equação

(2.9).

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u} \tag{2.8}$$

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} v(t)\cos(\theta(t)) \\ v(t)\sin(\theta(t)) \\ \omega(t) \\ (\ddot{\phi}_r(t) + \ddot{\phi}_l(t))/2 \\ (\ddot{\phi}_r(t) - \ddot{\phi}_l(t))r/b \\ p_2\ddot{\phi}_l(t) + p_3\dot{\phi}_r(t) \\ p_2\ddot{\phi}_r(t) + p_3\dot{\phi}_l(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ p_1 & 0 \\ 0 & p_1 \end{bmatrix} \cdot \begin{bmatrix} E_r(t) \\ E_l(t) \end{bmatrix} \tag{2.9}$$

onde:

$$p_1 = \frac{K_t}{R(2a_1 + I_s)}, \quad p_2 = -\frac{a_2}{2a_1 + I_s}, \quad p_3 = -\frac{\frac{K_t K_w}{R} + \nu}{2a_1 + I_s}, \tag{2.10}$$

e:

$$a_1 = \frac{mr^2}{8} + \frac{I_{zz}r^2}{2b^2} + \frac{I_w}{2}, \quad a_2 = \frac{mr^2}{4} - \frac{I_{zz}r^2}{b^2} \tag{2.11}$$

As variáveis m e I_{zz} são a massa total do robô e o momento de inércia sobre o eixo z; I_w e I_s são os momentos de inércia referentes à roda e do eixo do robô; K_t e K_w são as constantes de torque e tensão do motor de corrente contínua; R é a resistência elétrica da armatura; ν é o coeficiente de atrito; r é o raio da roda e; b é a metade da distância entre as rodas do robô.

2.2 Filtro de Kalman

O Filtro de Kalman, proposto em Kalman et al. (1960), é um algoritmo recursivo utilizado para estimação de estados de sistemas dinâmicos. Originalmente foi desenvolvido para sistemas lineares, com observações acrescidas de ruído gaussiano. As variáveis de

estado $\boldsymbol{x}_k$ devem ser modelas em função do estado anterior, uma entrada de controle $\boldsymbol{u}_k$ e um ruído gaussiano $\boldsymbol{w}_k$ de média zero e covariância $\boldsymbol{Q}_k$.

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k) + \boldsymbol{w}_k \quad ; \quad \boldsymbol{w}_k(N(0, \boldsymbol{Q}_k)) \tag{2.12}$$

Este sistema está sendo observado por um ou mais sensores, que disponibilizam um vetor de medidas ruidosas $\boldsymbol{z}_k$. A relação matemática entre as medições e os estados se dá pela Equação (2.13), onde se atua um ruído $\boldsymbol{v}_k$ de média zero e covariância $\boldsymbol{R}_k$:

$$\boldsymbol{z}_k = h(\boldsymbol{x}_{k-1}) + \boldsymbol{v}_k \quad ; \quad \boldsymbol{v}_k(N(0, \boldsymbol{R}_k)) \tag{2.13}$$

No caso de uma relação linear entre estados e medidas, usa-se a matriz de observação $\boldsymbol{H}$ para modelar o sensor:

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k \quad ; \quad \boldsymbol{v}_k(N(0, \boldsymbol{R}_k)) \tag{2.14}$$

Como apresentado em Kim (2011) o filtro é dividido em duas etapas: Predição e Atualização.

2.2.1 Etapa de Predição

Tendo em vista um modelo matemático linear que simule o sistema real com fidelidade, obtém-se uma predição do estado atual (Equação 2.15). As matrizes $\boldsymbol{A}$ e $\boldsymbol{B}$ modelam o sistema com o modelo de transição de estado e com a entrada de controle, respectivamente.

$$\bar{\boldsymbol{x}}_k = \boldsymbol{A}_k\boldsymbol{x}_{k-1} + \boldsymbol{B}_k\boldsymbol{u}_k \tag{2.15}$$

Nesta etapa também é estimado a covariância do priori $\bar{\boldsymbol{P}}_k$, esta variável estima o quão acurado está o filtro, utilizando em seu cálculo a matriz de covariância ruído do processo $\boldsymbol{Q}_k$.

$$\bar{\boldsymbol{P}}_k = \boldsymbol{A}_k\boldsymbol{P}_{k-1}\boldsymbol{A}_k^T + \boldsymbol{Q}_k \tag{2.16}$$

$$\boldsymbol{Q}_k = \begin{bmatrix} \sigma^2_{w[1,1]} & \sigma^2_{w[1,2]} & \cdots & \sigma^2_{w[1,n]} \\ \sigma^2_{w[2,1]} & \sigma^2_{w[2,2]} & & \vdots \\ \vdots & & \ddots & \\ \sigma^2_{w[n,1]} & \cdots & & \sigma^2_{w[n,n]} \end{bmatrix} \tag{2.17}$$

2.2.2 Etapa de Correção

Após medidas de sensores que estão observando o sistema, é compensada a diferença entre elas e a predição de estados. Primeiramente calcula-se o erro residual através da Equação 2.18, chamado de valor de inovação.

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - \boldsymbol{H}\bar{\boldsymbol{x}}_k \tag{2.18}$$

Em seguida, computa-se o ganho ótimo de Kalman, que compara a confiabilidade atual das variáveis de estado em relação a medida real. Assim estima-se o estado ótimo, corrigindo as variáveis de estado do sistema e estimando a matriz a posteriori da covariância do erro do sistema.

$$\boldsymbol{S}_k = \boldsymbol{H}_k\bar{\boldsymbol{P}}_k\boldsymbol{H}_k^T + \boldsymbol{R}_k \tag{2.19}$$

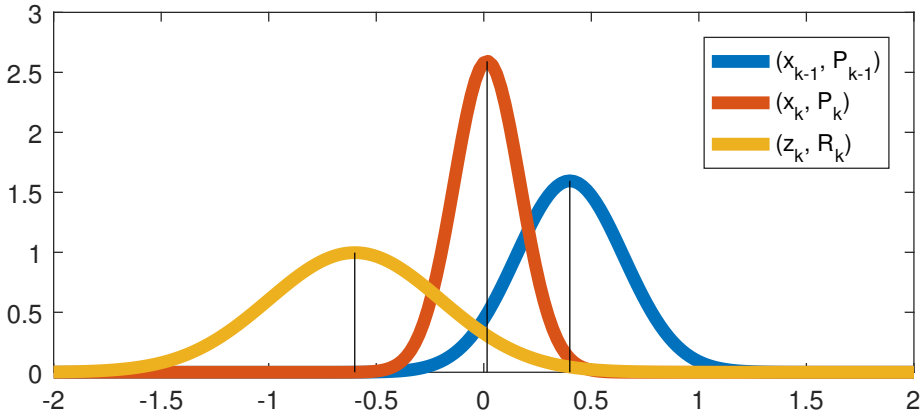
$$\boldsymbol{K}_k = \bar{\boldsymbol{P}}_k\boldsymbol{H}_k^T\boldsymbol{S}_k^{-1} \tag{2.20}$$

O estado a posteriori é dado pela Equação (2.21), o novo valor de $\boldsymbol{x}_k$ estará entre $\bar{\boldsymbol{x}}_k$ e $\boldsymbol{z}_k$, e o ganho $\boldsymbol{K}_k$ de acordo com a probabilidade envolvida no processo. A Equação (2.22) calcula o covariância a posteriori do sistema $\boldsymbol{P}_k$, onde $\boldsymbol{I}$ é a matriz identidade de mesma dimensão do estado. A Figura exemplifica o processo na forma unidimensional.

$$\boldsymbol{x}_k = \bar{\boldsymbol{x}}_k + \boldsymbol{K}_k\tilde{\boldsymbol{y}} \tag{2.21}$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\bar{\boldsymbol{P}}_k \tag{2.22}$$

Figura 2.3: Atualização do Filtro de Kalman



Fonte: Elaborada pelo autor

2.2.3 Filtro de Kalman Estendido

O Filtro de Kalman básico é utilizado para estimação de estados de sistemas lineares ruidosos. Muitos sistemas dinâmicos e sensores possuem comportamento não linear, mas que possuem um comportamento aproximadamente linear para pequenas variações dos estados (NEGENBORN, 2003).

Nestes sistemas nas Equações (2.12) e (2.13), as funções h e/ou f são não-lineares. Para aplicar o filtro, esses modelos são constantemente linearizados. Para predição do estado a priori e cálculo do erro residual utilizam-se as equações não lineares na íntegra, já para os demais cálculos há constantemente uma linearização das matrizes A_k e H_k através do cálculo dos jacobianos sobre o estado a priori.

$$\begin{aligned} \bar{\boldsymbol{x}} &= f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k) \\ A_k &= \left.\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\bar{x}_k} \\ \tilde{\boldsymbol{y}_k} &= \boldsymbol{z}_k - h(\bar{\boldsymbol{x}}) \\ H_k &= \left.\frac{\partial h(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\bar{x}_k} \end{aligned} \tag{2.23}$$

2.3 Navegação Assistida por Campos Vetoriais

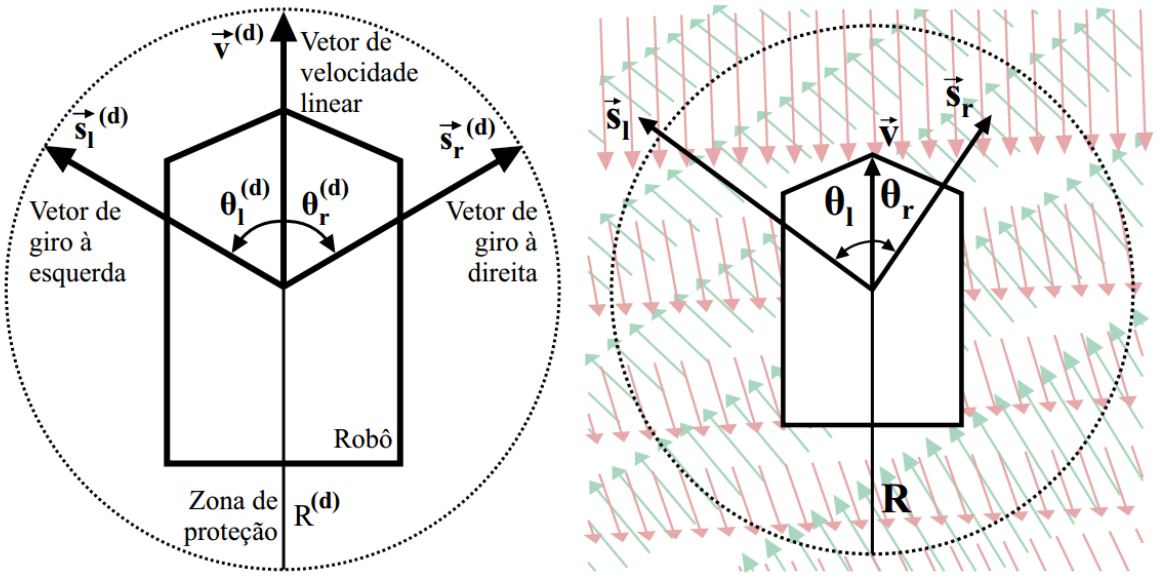
Esta Seção explica a navegação assistida por campos vetoriais (OLIVI et al., 2013), que serviu como base para o sistema de navegação proposto neste trabalho, a ser apresentado no Capítulo 4. Esta técnica foi originalmente proposta para uma interface

EEG P300, a qual o usuário pode selecionar entre 4 comandos: virar a esquerda, virar a direita, finalizar navegação e manter-se em frente sem giros. Como se trata de uma interface muito simples, com poucos comandos, a escolha da velocidade linear e qual ângulo de giro que a CRI deve rotacionar irá depender de qual dos quatro comandos o usuário selecionou, da disposição dos obstáculos e zonas seguras em relação ao robô.

A partir de um sensor *laser scanner* é realizado o mapeamento e localização do robô. De acordo com a posição dos obstáculos são calculados campos vetoriais que serão utilizados para estimar velocidades seguras para o robô. Esses campos podem ser repulsivos em relação a obstáculos e atrativos em relação às áreas seguras do mapa.

O robô possui um conjunto de vetores padrão (*default*) que geram o comportamento das velocidades: "vetor de velocidade linear" $\vec{v}^{(d)}$, "vetor de giro à esquerda" $\vec{s}_l^{(d)}$ e "vetor de giro à direita" $\vec{s}_r^{(d)}$. Além disto, tem-se a variável $R^{(d)}$, que é o raio de proteção. Esses vetores são alterados ao interagir com os campos vetoriais das imediações ao robô. São produzidos novos parâmetros que irão definir as velocidades finais, os vetores: $\vec{v}$, $\vec{s}_l$, $\vec{s}_d$ e R. A Figura 2.4 apresenta todos os parâmetros envolvidos. Existe um campo repulsivo, em vermelho, referente aos obstáculos, e um campo atrativo, em verde, relativo a zonas seguras no mapa.

Figura 2.4: Atuação dos Campos Vetoriais



Fonte:(OLIVI et al., 2014)

Neste sistema controlado por interface cerebral, ao iniciar a navegação, a CRI

assumirá valores de velocidade linear proporcional a $\vec{\boldsymbol{v}}$ e o usuário escolherá as direções do movimento, fazendo com que o sistema gire de $\theta_{r,l}$ ao ser requisitado tais movimentos.

2.3.1 Controle de Velocidade Linear

A partir de uma varredura de *laser scanner* e da localização atual da CRI $\boldsymbol{P} = [P_x,\ P_y,\ P_\theta]^T$ no mapa, obtêm-se as distâncias d dos obstáculos nas mediações do robô referente aos ângulos λ de cada uma das medidas. Para calcular o campo repulsivo $\vec{\boldsymbol{c}}$, é somada a contribuição dos M pontos que estão dentro de região delimitada pelo raio R, conforme Equação (2.24). A valor de β é um ganho a ser calibrado, que pode ser uma contante ou uma função polinomial relativa ao valor do ângulo da medida $\beta(\lambda_i)$, de forma a valorizar as medições laterais.

$$\vec{\boldsymbol{c}} = -\beta \sum_{i=1}^{M} \frac{R}{d_i} \begin{bmatrix} \cos(\lambda_i + P_\theta) \\ \sin(\lambda_i + P_\theta) \end{bmatrix} \tag{2.24}$$

A cadeira de rodas irá assumir a velocidade linear $\vec{\boldsymbol{v}}$ que é a soma entre o vetor padrão $\vec{\boldsymbol{v}}^{(d)}$ e a projeção do vetor repulsivo $\vec{\boldsymbol{c}}$ sobre o próprio vetor $\vec{\boldsymbol{v}}^{(d)}$, conforme Equação (2.25), que fará o robô reduzir sua velocidade com a proximidade de obstáculos à frente, e até mesmo poderá assumir velocidade negativa até se afastar o suficiente.

$$\vec{\boldsymbol{v}} = \vec{\boldsymbol{v}}^{(d)} + \left(\frac{\vec{\boldsymbol{c}}.\vec{\boldsymbol{v}}^{(d)}}{\vec{\boldsymbol{v}}^{(d)}.\vec{\boldsymbol{v}}^{(d)}} \right) \vec{\boldsymbol{v}}^{(d)} \tag{2.25}$$

2.3.2 Controle dos Ângulos de Giro

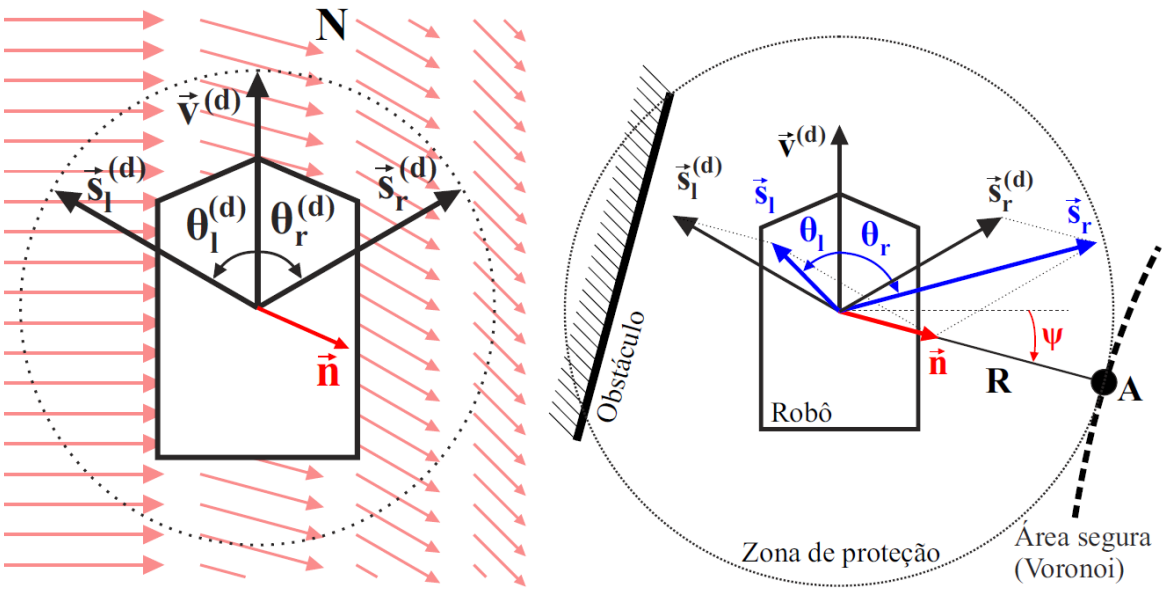
Os ângulos de giro são referentes ao deslocamento angular realizado pela CRI cada vez que o usuário solicitar a opção equivalente na interface. O objetivo do campo vetorial será alterar esses giros para facilitar as manobras realizadas pelo usuário, de forma a privilegiar o deslocamento para áreas seguras.

Contando com um sistema de mapas e localização, pode se obter o diagrama de Voronoi, que subdivide o espaço em células a partir de pontos fixos. Quando aplicado na robótica, os pontos fixos são os obstáculos e as bordas destas células são os locais mais distantes possíveis de todos os obstáculos, sendo regiões mais seguras para a trajetória do

robô.

Ao invés de utilizar campos repulsivos, para controle de ângulos de giros é utilizado um campo atrativo, referente as regiões seguras pra trafegar no mapa. Para o cálculo do campo atrativo $\vec{n}$ é identificado o ponto $\boldsymbol{A}$ do diagrama de Voronoi mais próximo da atual localização do robô, como apresentado na Figura 2.5.

Figura 2.5: Cálculo Ângulos de Giro



Fonte:(OLIVI et al., 2014)

Uma vez que se sabe o ponto $\boldsymbol{A}$, computa-se o novo raio da zona de proteção, dado pela distância euclidiana entre $\boldsymbol{A}$ e $\boldsymbol{P}$, desde que a mesma seja maior que o valor padrão $R^{(d)}$, conforme Equação (2.26).

$$R = max(\sqrt{(A_x - P_x)^2 + (A_y + P_y)^2}, R^{(d)}) \tag{2.26}$$

Em seguida calcula-se o vetor atrativo utilizando o ângulo entre a pose do robô, o ponto $\boldsymbol{A}$ e o ganho α:

$$\psi = atan2(A_y - P_y, A_x - P_x) \tag{2.27}$$

$$\vec{n} = \alpha R \begin{bmatrix} \cos(\psi) \\ \sin(\psi) \end{bmatrix} \tag{2.28}$$

Finalmente o vetor atrativo é combinado com os vetores de giro padrão $\vec{s}_{r,l}^{(d)}$ e é computado os novos valores dos ângulos de giro.

$$\vec{s}_{r,l} = \vec{s}_{r,l}^{(d)} + \vec{n} \tag{2.29}$$

$$\theta_{r,l} = acos\left(\frac{\vec{s}_{r,l}.\vec{v}}{\parallel \vec{s}_{r,l} \parallel . \parallel \vec{v} \parallel}\right) \tag{2.30}$$

2.4 Redes de Petri

Neste trabalho será utilizada a ferramenta Rede de Petri (RP) para configurar e operar o sistema da cadeira de rodas, como será apresentado no Capítulo 3. Esta seção resume seus principais conceitos e foi baseada nas obras de Cardoso e Valette (1997), Neto et al. (2014) e Reisig (2013). Propostas por Carl Adam Petri, as Redes de Petri servem para modelar sistemas distribuídos discretos através de um grafo direcionado.

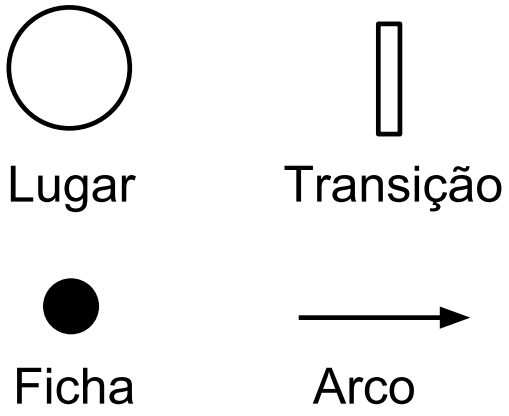
Em um sistema discreto as mudanças de estados ocorrem em instantes precisos, portanto, são observadas discretamente. As variáveis de estado podem variar bruscamente, sendo calculadas com base nos estados anteriores, sem ter que obrigatoriamente considerar o tempo. Neste contexto, as RPs se aplicam a um grande número de processos.

2.4.1 Estrutura Básica

Uma RP é formada por 4 elementos básicos, conforme Figure 2.6:

- **Lugares (*Places*)**: Graficamente um *place* é representado por um círculo. Sempre modelam um componente passivo, podendo representar uma condição, um estado parcial ou um conjunto de recursos.
- **Fichas (*Tokens*)**: Normalmente são representadas como pontos pretos dentro dos *Places*. Indicam que a condição e estado associados são verificados e que recursos estão disponíveis.
- **Transições**: Representadas por retângulos ou barras, são um modelo ativo e estão associados a ocorrência de eventos, mudanças de estados. Ao ser habilitada, uma

Figura 2.6: Elementos Básicos de uma RP



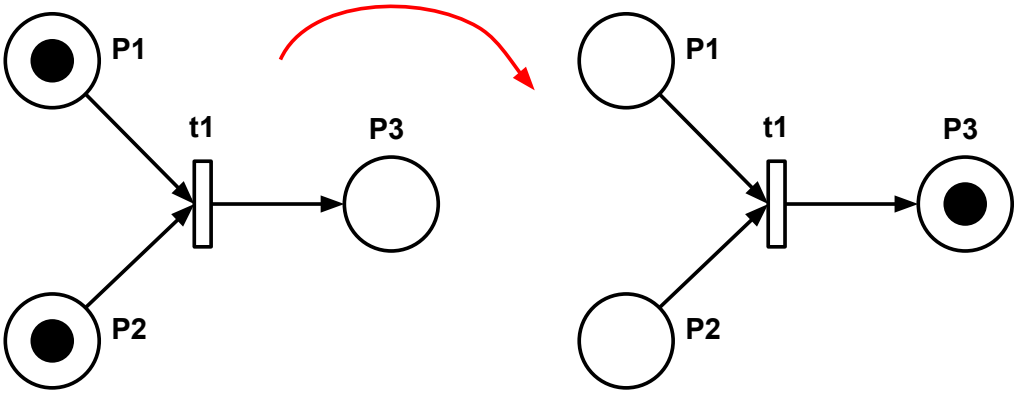
Fonte: Elaborada pelo autor

transição consome *tokens* de cada uma dos *places* de entrada e produz nós de saída. A condição desse disparo é a existência de fichas suficientes em cada posição de entrada.

- **Arcos**: Graficamente como setas, conectam os lugares e as transições. Possuem uma variável peso, que define o número de fichas que a transição irá consumir ou fornecer para o respectivo lugar.

A Figura 2.7 mostra um exemplo de evento, habilitado pela transição $t1$.

Figura 2.7: Exemplo de Evento

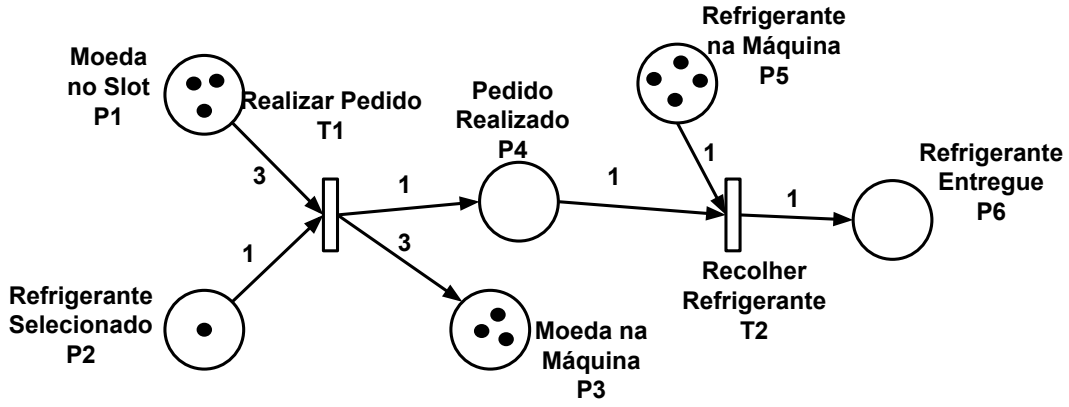


Fonte: Elaborada pelo autor

2.4.2 Formulação Matricial

Para implementação prática de uma RP pode-se utilizar a formulação matricial do sistema, que será explicada a partir de um exemplo. Considere uma RP que administra uma máquina de refrigerantes hipotética em que cada produto custa 3 moedas. Está representada em seu estado inicial na Figura 2.8, no qual já foram colocadas 3 moedas no *slot* e selecionado o refrigerante desejado.

Figura 2.8: Exemplo de Aplicação



Fonte: Elaborada pelo autor

Primeiramente modela-se o sistema pela Matriz de Incidência Combinada $\boldsymbol{C}$, onde as linhas representam os *places* e as colunas as transições. Cada coluna informa quantas *tokens* vão entrar ou sair de cada lugar quando a transição correspondente é ativada, para o exemplo:

$$\boldsymbol{C} = \begin{bmatrix} -3 & 0 \\ -1 & 0 \\ 3 & 0 \\ 1 & -1 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \tag{2.31}$$

O estado do sistema é dado pelo vetor $\boldsymbol{M}_k$, que contém a quantidade de fichas em cada lugar no instante k, no exemplo o estado inicial é:

$$\boldsymbol{M}_0 = \begin{bmatrix} 3 & 1 & 3 & 0 & 4 & 0 \end{bmatrix}^T \tag{2.32}$$

Outro vetor a se considerar é a matriz de transições $\boldsymbol{u}_k$, onde cada linha refere-se a uma transição, informando quais transições estão habilitadas e serão disparadas. Para que esteja habilitada, todos os lugares que antecedem a transição devem conter o número necessário de *tokens*. A matriz de transição inicial do exemplo é dada por:

$$\boldsymbol{u}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{2.33}$$

Em cada iteração, a dinâmica da rede é atualizada pela Equação de Estado:

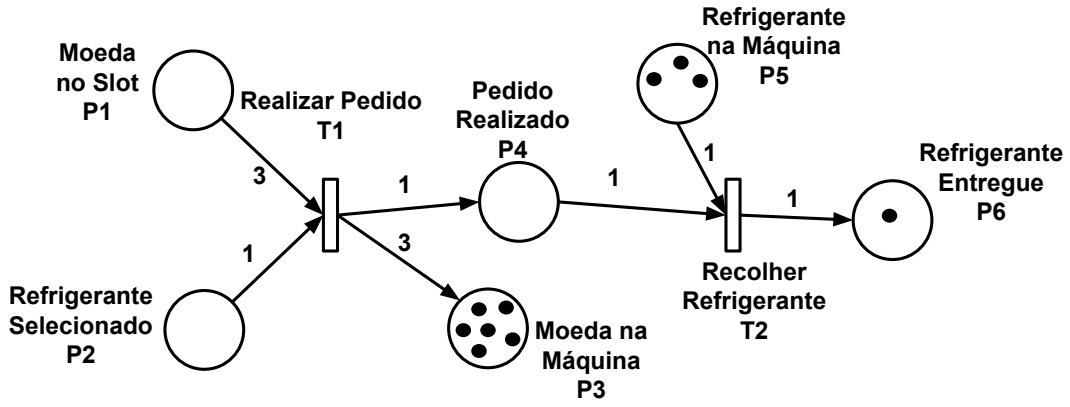
$$\boldsymbol{M}_{k+1} = \boldsymbol{M}_k + \boldsymbol{C}\boldsymbol{u}_k \tag{2.34}$$

No exemplo em questão haverá duas transições em sequência, finalizando como apresentado na Figura 2.9.

$$\boldsymbol{M}_1 = \begin{bmatrix} 3 \\ 1 \\ 3 \\ 0 \\ 4 \\ 0 \end{bmatrix} + \begin{bmatrix} -3 & 0 \\ -1 & 0 \\ 3 & 0 \\ 1 & -1 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 6 \\ 1 \\ 4 \\ 0 \end{bmatrix} \tag{2.35}$$

$$\boldsymbol{M}_2 = \begin{bmatrix} 0 \\ 0 \\ 6 \\ 1 \\ 4 \\ 0 \end{bmatrix} + \begin{bmatrix} -3 & 0 \\ -1 & 0 \\ 3 & 0 \\ 1 & -1 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 6 \\ 0 \\ 3 \\ 1 \end{bmatrix} \tag{2.36}$$

Figura 2.9: Rede Final



Fonte: Elaborada pelo autor

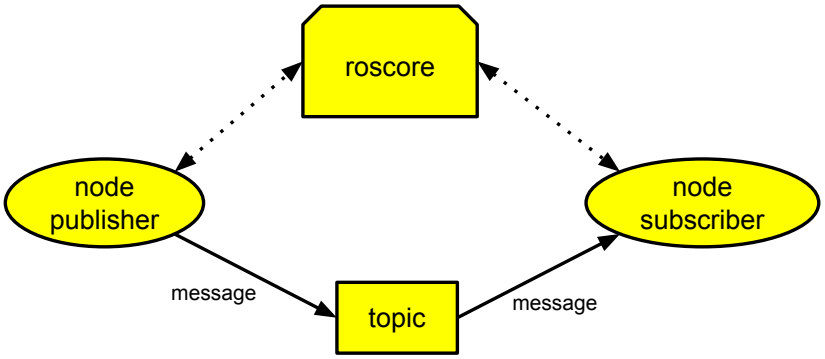
Após as transições, as moedas foram armazenadas na máquina e um refrigerante foi entregue. Caso forem inseridas novamente 3 moedas no *slot* e um refrigerante for selecionado, o processo será replicado enquanto houver estoque.

2.5 Robot Operating System

Neste trabalho faz-se o uso da plataforma Robot Operating System (ROS). O ROS é um *open-source framework* para desenvolver *softwares* para robôs, sendo uma coleção de bibliotecas, ferramentas e convenções com o objetivo de simplificar a criação de aplicações robóticas (QUIGLEY; GERKEY; SMART, 2015). Fornece abstração de *hardware*, *drivers*, visualizadores, trocador de mensagens, gerenciamento de pacotes entre outros.

O grande destaque do ROS é o paralelismo, a arquitetura descentralizada. Consiste em vários programas de computador conectados um ao outro, podendo estar sendo executados em múltiplos computadores. Estão continuamente trocando mensagens, essas mensagens viajam diretamente de um programa para outro. Documentação, tutoriais e downloads podem ser encontrados em: (Wiki ROS, 2017).

Figura 2.10: Estrutura ROS



Fonte: Elaborada pelo autor

2.5.1 Estrutura Básica

Nós

Um nó é um processo que faz o uso do *framework* ROS. Cada nó é um processo executando uma determinada rotina. Eles são processados paralelamente e se comunicam utilizando tópicos.

Roscore

Sempre em uma rede ROS haverá uma máquina mestra. Nesta máquina estará rodando o comando "roscore", que inicia todos os serviços que habilitam a comunicação entre os nós, gerenciando a comunicação TCP/IP.

Tópico

Tópico é o meio em que uma mensagem é trocada entre dois nós. O mesmo utiliza o protocolo *publish/subscriber* onde o nó *publisher* insere a mensagem de dados neste tópico e os nós *subscribers* leem a mesma.

Mensagem

As mensagens são os dados trocados através dos tópicos. Existem inúmeros tipos, desde uma simples variável inteira, até uma estrutura de dados complexa.

3 Desenvolvimento do Projeto

3.1 Introdução

O trabalho propõe uma arquitetura de software e hardware para CRIs para ser utilizada por usuários tetraplégicos. O usuário configura o modo de operação utilizando comando de voz, o sistema manda mensagens de áudio requisitando informações e confirmando a configuração, este processo será apresentado na Seção 3.3.

Devido o alto custo de um *laser scanner*, mesmo com um campo de visão inferior, optou-se pela utilização de uma câmera de profundidade como sensor exteroceptivo, fazendo uma varredura dos obstáculos a frente. Essas distâncias serão utilizadas no controle compartilhado, apresentados nos Capítulos 4 e 5. O foco é obter um sistema relativamente barato e de baixo custo computacional, não sendo necessário conhecimento prévio do ambiente através de mapas.

3.2 Descrição do Hardware e Software

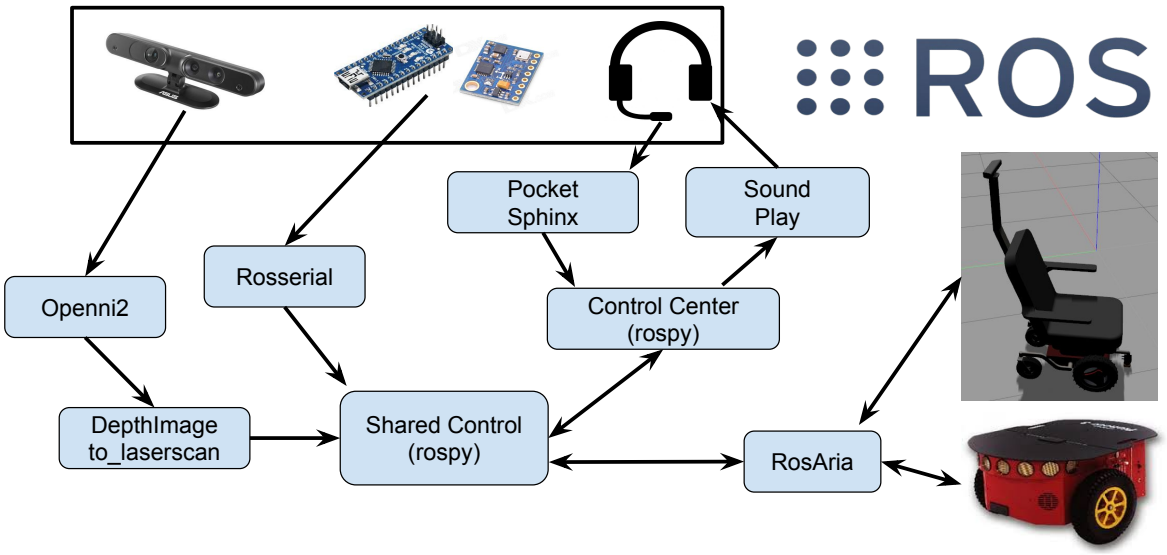
Esta seção apresenta uma descrição das arquiteturas de *software* e *hardware*. A Figura 3.1 contém um diagrama com as interações, a partir de uma arquitetura ROS, entre os nós e periféricos, que serão explicados individualmente. Todos os pacotes utilizados neste trabalho estão documentados e disponíveis em Wiki ROS (2017).

3.2.1 Pacotes e Programas

Openni2

Este pacote é utilizado para ler os dados da câmera RGB-D da Asus e disponibilizá-los de maneira estruturada em tópicos da plataforma ROS. No caso de um sensor Kinect é recomendado usar o pacote *freenect*.

Figura 3.1: Arquitetura do Sistema



Fonte: Elaborada pelo autor

Depthimage_to_laserscan

Neste trabalho utiliza-se, por questões econômicas, uma câmera de profundidade em vez de um *laserscanner*. Para tanto, faz-se o uso do pacote ROS *depthimage_to_laserscan* que emula uma varredura 2D a partir de uma imagem de profundidade.

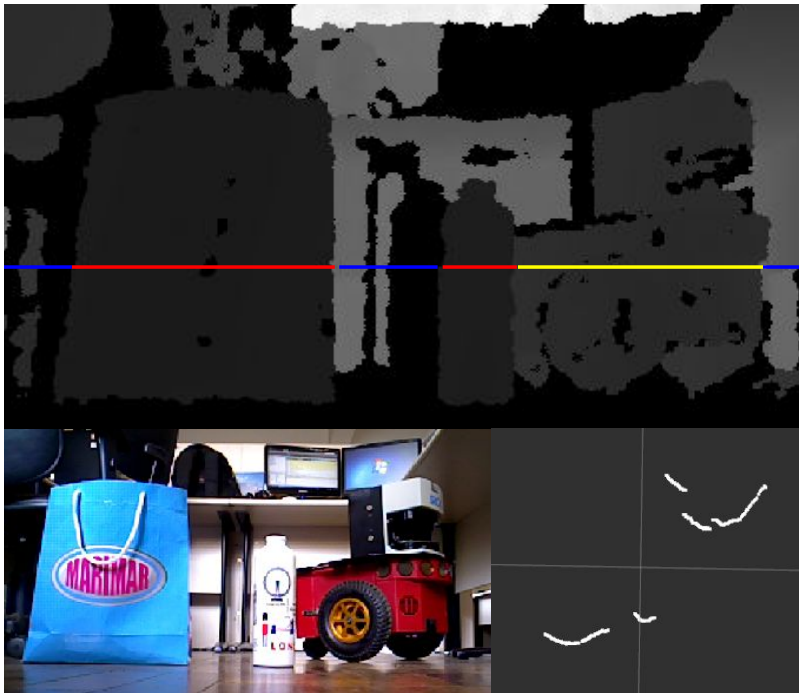
A Figura 3.2 contém um cenário de aplicação, apresentando uma imagem RGB e a respectiva imagem de profundidade. A reta representa o local que estão sendo estimadas as distâncias e a cor de acordo com as mesmas. Abaixo, o feixe de *laser scan* 2D simulado pelo pacote.

Utilizando este pacote na câmera escolhida para o trabalho, obtêm-se 640 medidas entre -0.5 e 0.5 radianos aproximadamente. Esta discretização é mais que o suficiente para os objetivos do trabalho, por isso, para melhorar o custo computacional dos algoritmos dependentes deste sensor, este pacote foi reestruturado para fornecer 72 medidas, com distribuição angular uniforme. Outra alteração realizada, para aumentar a confiabilidade, foi saturar em 4 metros as leituras acima de 4 metros ou medidas NaN (*not a number*).

Pocketsphinx

O reconhecimento de fala é realizado pelo pacote *Pocketsphinx* adaptado para ROS, usando *gstreamer* e uma interface em *Python*. O pacote requer um modelo de

Figura 3.2: DepthImage to LaserScan



Fonte: Elaborada pelo autor

linguagem e um arquivo dicionário. Neste trabalho utilizou-se o modelo padrão em língua inglesa. O programa não precisa de conexão à internet.

Para ser usuário da ferramenta, basta criar um arquivo de texto *.dic* contendo a composição fonética de todas as palavras que se deseja utilizar no projeto, além de criar um arquivo de texto *.lm* das frases de até três palavras que o algoritmo tentará vincular à entrada de voz e retornar como saída. A Figura 3.3 contém exemplos:

Figura 3.3: Exemplo Pocketsphinx

```
FULL    F UH L
HALF    HH AE F
MANUAL  M EY N Y UW AH L
MODE    M OW D
SPEED   S P IY D
STOP    S T OW P
```

```
\1-grams:
-1.9912 FULL -0.2921
-3.7350 MANUAL -0.3483
-4.5099 MODE -0.5291
-3.1126 STOP -0.2647

\2-grams:
-0.3010 FULL SPEED 0.0000
-0.3010 HALF SPEED 0.0000
-2.0492 MANUAL MODE 0.0000
```

Fonte: Elaborada pelo autor

Sound_play

Este pacote ROS permite a execução de vários tipos de arquivos de áudio. Além disso, há a conversão de texto para fala, que será utilizada para fazer a interface com o

usuário por meio de alto-falantes.

Rosserial

Rosserial é um protocolo utilizado para disponibilizar dados provenientes de uma comunicação serial em tópicos da plataforma ROS. Entre as bibliotecas está a *rosserial_arduino*, que faz com que uma placa de prototipagem Arduino publique dados diretamente em um tópico. Neste trabalho esse pacote é usado para disponibilizar em tempo real os dados da IMU na rede.

RosAria

Realiza a interface ROS para inúmeros tipos de robôs móveis, entre as funções destacam-se a leitura de sensores odométrico e a atuação em alto nível nas velocidades do robô. Inclusive neste trabalho, é utilizada em um robô Pioneer-P3DX para avaliação experimental.

Rospy

A biblioteca *Rospy* é utilizada como a interface de programação em Python para a plataforma ROS. Foi utilizada para a criação do nó "Shared Control"onde estão aplicadas as metodologias propostas nos Capítulos 5 e 6, além do nó "Control Center"apresentado na Seção 3.3, que administra todo o sistema através da interface de voz.

3.2.2 Materiais

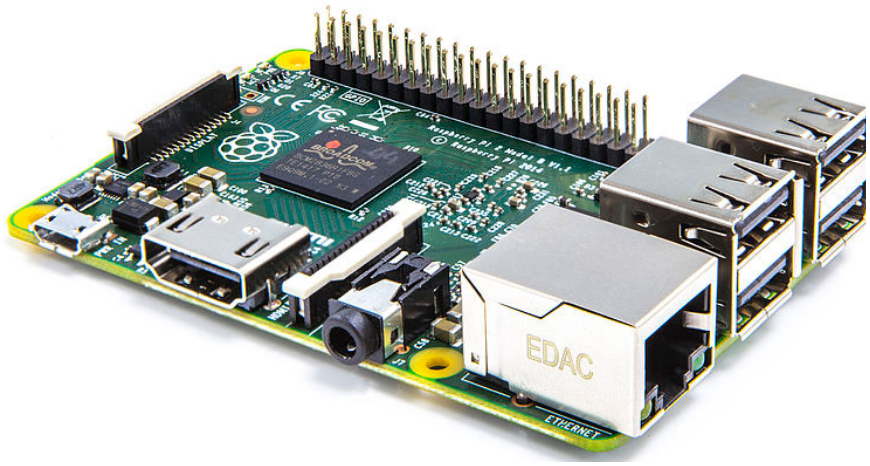
Raspberry Pi 3 - B

O processamento principal está embarcado em uma *Raspberry PI 3 - B*. Nela está contido um cartão *microSD* de 16gb onde está instalado o sistema operacional *Ubuntu Mate* e a versão *Kinect* do ROS. Todos os pacotes utilizados, além dos programas de controle compartilhado, estão neste dispositivo.

A placa conta com um processador 1.2GHz 64-bit quad-core ARM Cortex-A53, wifi, blueetooth e 4 portas USB. Nas portas USB estará conectado um microcontrolador,

uma câmera de profundidade, uma placa de som externa e o *drive* de acionamento do robô.

Figura 3.4: Raspberry Pi 3 - model B



Fonte: (Wikimedia Commons, 2017)

Asus XtionPRO

A Asus XtionPRO live (Figura 3.5) contém uma câmera RGB e uma de profundidade. Possui um campo de visão horizontal de 58 graus, e consome menos que 2 watts de energia, sendo que é a alimentação e comunicação são pela mesmo entrada USB. A câmera de profundidade tem boa precisão quando os objetos estão entre 0.45 e 4 metros. Deve ser posicionada o mais atrás possível para uma melhor observação dos obstáculos a frente da cadeira, neste trabalho será considerado que a XtionPRO ficará sobre o encosto das costas da CRI acima da cabeça do usuário.

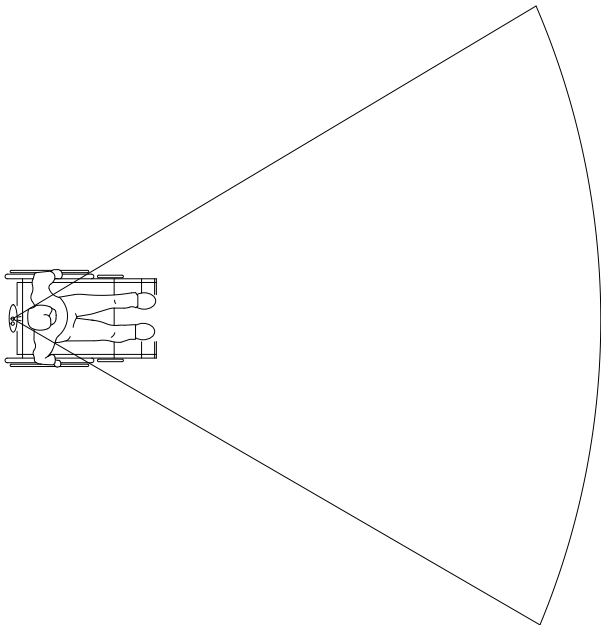
Figura 3.5: Asus XtionPRO

Fonte: Elaborada pelo autor

Com a aplicação dos pacotes *Oppeni2* e *Depthimage_to_laserscan* sobre esta câmera

tem-se um campo de visão conforme a Figura 3.6. Para as metodologias propostas neste trabalho é preciso que toda a parte frontal da cadeira esteja sobre esse campo.

Figura 3.6: Campo de Visão



Fonte: Elaborada pelo autor

Headset Gamer

Como é necessária a utilização de microfone e alto-falante para a interface, optou-se em utilizar um *headset* (Figura 3.7), já servindo de suporte para a IMU no topo da cabeça do usuário na confecção do protótipo, os fios do *headset* foram trançados junto com da rede I2C da IMU.

Adaptador de Áudio 3D SOUND

Devido à falta de entradas e saídas de áudio na Raspiberry PI 3, utilizou-se uma placa de som externa (Figura 3.8) que converte os dois pinos analógicos de entrada e saída de áudio em uma digital USB.

Figura 3.7: Headset

Fonte: Elaborada pelo autor

Figura 3.8: Adaptador de Áudio USB

Fonte: Elaborada pelo autor

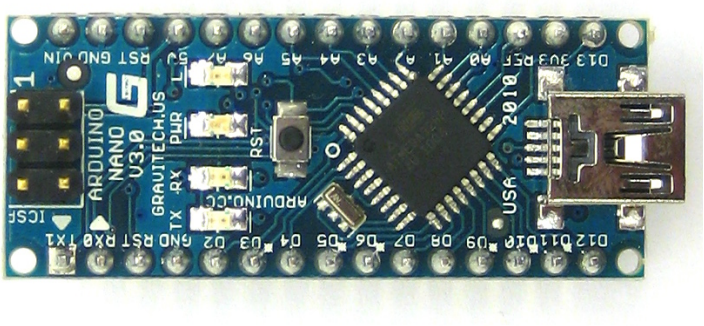
Arduino Nano

A função da plataforma de prototipagem Arduino Nano, Figura 3.9, é realizar a leitura da IMU através do protocolo I2C para obter os ângulos de Euler da cabeça do usuário. No próprio Arduino é processado um Filtro de Kalman, reduzindo ruídos destes ângulos para depois enviá-los para a Raspberry serialmente. O microcontrolador é um ATmega32 de 16MHz.

GY-521

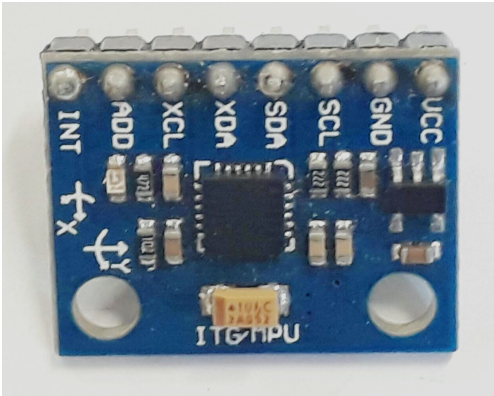
A IMU utilizada é uma GY-521, Figura 3.10 com o módulo MPU-6050, que contém acelerômetro e giroscópio, ambos com 3 eixos de medição.

Figura 3.9: Arduino Nano



Fonte: Elaborada pelo autor

Figura 3.10: GY-521



Fonte: Elaborada pelo autor

Regulador de Tensão LM2596

Para a alimentação utilizou-se um regulador de tensão LM2596, Figura 3.11. A tensão de saída pode ser ajustada entre 1.5 e 35V, tendo como entrada 3.2 a 40V, com isto aproveitam-se as baterias do próprio robô (normalmente 12V) para alimentar a *Raspberry* e demais periféricos. Pode suprir uma carga de 3A.

Figura 3.11: LM2596

Fonte: Elaborada pelo autor

Devido à proposta de baixo custo, é apresentada na Tabela 3.1 uma estimativa do orçamento do projeto para uma pessoa física no Brasil. Os valores foram obtidos a partir

de produtos novos no aplicativo MercadoLivre, em dezembro de 2017, não incluindo o frete e o preço da cadeira de rodas motorizada.

Tabela 3.1: Orçamento

Componente	Preço
Raspberry Pi 3	R$160
Asus XtionPRO	R$400
Headset Multilaser	R$35
Adaptador 3D Sound	R$13
Arduino Nano	R$17
IMU GY-521	R$ 14
Regulador LM2596	R$ 10
Total	R$ 649

3.3 Central de Comando Baseado em Voz

Neste trabalho utiliza-se uma HMI de áudio, na qual o usuário e o módulo de inteligência computacional interagem através de frases sonoras. Essa interação é feita por um sistema distribuído discreto baseado em Redes de Petri e é responsável por configurar os modos de operação da CRI. A Figura 3.12 apresenta a rede de Petri proposta em seu estado inicial, onde há apenas uma *token* no estado "*Parked*", em seguida será explicada toda a arquitetura da rede.

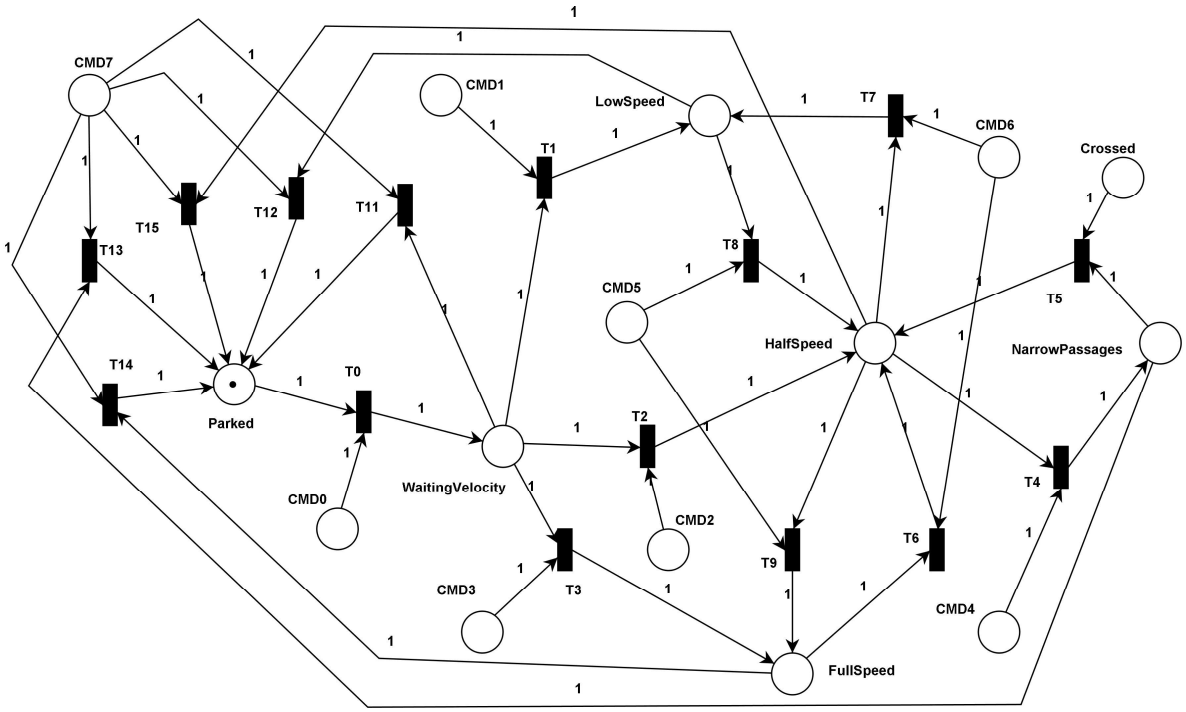
Este procedimento foi implementado em *Python*, correspondendo ao Nó "*Control Center*", fazendo a leitura constante do tópico de reconhecimento de voz do pacote *Pocketsphinx* e disponibiliza o estado atual da rede para ser utilizado pelo nó *"Shared Control"*.

3.3.1 Estados de Operação

Durante a operação, o sistema pode assumir seis estados e apenas um destes locais conterá a *token*. De acordo com o atual estado será configurado o modo de operação da cadeira de rodas, intervindo diretamente nas velocidades, são eles:

- Parked: Mantém a cadeira parada independente de qualquer variável, é o estado inicial do sistema e pode ser alcançado de qualquer ponto da rede.

Figura 3.12: Rede de Petri - Estado Inicial



Fonte: Elaborada pelo autor

- WaitingVelocity: Mantém a cadeira parada até o usuário selecionar o modo de velocidade desejado.

- LowSpeed: O usuário controla diretamente a cadeira pelos movimentos de cabeça com velocidades mínimas, ideal para manobras delicadas como estacionar a CRI sob a mesa de jantar.

- HalfSpeed: Com velocidade média, é ideal para ambientes internos, o usuário controla a cadeira por meio do controle compartilhado, que será apresentado no Capítulo 4, ajudando a navegar em ambientes com obstáculos.

- FullSpeed: Com o mesmo controle compartilhado, esta categoria é ideal para ambientes abertos, onde o usuário pode assumir velocidades mais altas.

- NarrowPassage: Neste estado, caso seja identificada uma passagem estreita, a cadeira irá realizar uma travessia completa utilizando o algoritmo do Capítulo 5. Ao final da missão, uma *token* é colocada no local "Crossed", retornando o sistema para o estado "HalfSpeed".

3.3.2 Estados de Comando

Quando o usuário realiza um comando de voz pré estabelecido, uma *token* é acrescentada artificialmente em determinado lugar, podendo acionar uma transição de estados. Estes locais estão nomeados como "CMD0-7", e cada um é ativado por um arranjo específico de frases do usuário como mostrado na Tabela 3.2. Para estes locais da rede, pode ser feita uma analogia a sensores, observando o reconhecimento de voz.

Tabela 3.2: Frases de Ativação

Local	**Frases de Ativação**
CMD0	"manual", "manual mode"
CMD1	"low", "low speed"
CMD2	"half", "half speed"
CMD3	"full", "full speed"
CMD4	"narrow", "narrow passage"
CMD5	"more speed", "faster"
CMD6	"less speed", slower"
CMD7	"stop", "abort"

Vale a pena salientar que apenas um estado de comando vai estar com *token*, e se essa *token* não acarretar uma transição na mesma iteração, ela será excluída. Isso garante que a dinâmica da rede seja em tempo real com as entradas de fala do usuário, e que não ocorra uma transição indesejada fornecida por um comando no passado.

3.3.3 Transições de Estados

Uma transição vai depender do atual estado de operação e do atual estado de comando. Depois de cada transição o sistema retorna uma frase pelo alto-falante, confirmando uma escolha ou requisitando novos comandos, estas frases estão apresentadas na Tabela 3.3.

A modelagem deste sistema está apresentada na Figura 3.13 por meio da matriz de incidência combinada, contendo a dinâmica de *tokens* para cada transição e as condições de cada transição, sendo utilizada para as equações de mudança de estado como apresentado na Seção 2.4. Em resumo, este sistema parte do estado estacionado, o usuário requisita o controle manual e em seguida escolhe a velocidade desejada. Durante a navegação é possível trocar essas velocidades ou pedir uma travessia autônoma em uma passagem

Tabela 3.3: Frases de FeedBack

Transição	**Frases de Feedback**
T0	“choose velocity type”
T1, T7	“low speed”
T2, T6, T8	“half speed”
T3, T9	“full speed”
T4	“entering in a narrow passage”
T5	“narrow passage finished”
T10, T11, T12, T13, T14	“you are parked”

Figura 3.13: Matriz de Incidência Combinada

	T0	T1	T10	T11	T12	T13	T14	T2	T3	T4	T5	T6	T7	T8	T9
CMD0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CMD1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0
CMD2	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0
CMD3	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0
CMD4	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0
CMD5	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	-1
CMD6	0	0	0	0	0	0	0	0	0	0	0	-1	-1	0	0
CMD7	0	0	-1	-1	-1	-1	-1	0	0	0	0	0	0	0	0
Crossed	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0
FullSpeed	0	0	0	0	0	0	-1	0	1	0	0	-1	0	0	1
HalfSpeed	0	0	-1	0	0	0	0	1	0	-1	1	1	-1	1	-1
LowSpeed	0	1	0	0	-1	0	0	0	0	0	0	0	1	-1	0
NarrowPassages	0	0	0	0	0	-1	0	0	0	1	-1	0	0	0	0
Parked	-1	0	1	1	1	1	1	0	0	0	0	0	0	0	0
WaitingVelocity	1	-1	0	-1	0	0	0	-1	-1	0	0	0	0	0	0

Fonte: Elaborada pelo autor

estreita. Independente do estado atual o usuário pode dar um comando para parar a cadeira retornando-a ao estado inicial.

4 Sistema de Navegação Assistida

Este capítulo descreve o sistema de controle compartilhado de navegação, que está implementado no nó "Shared Control"na arquitetura ROS proposta.

4.1 Interface Head-Keypad

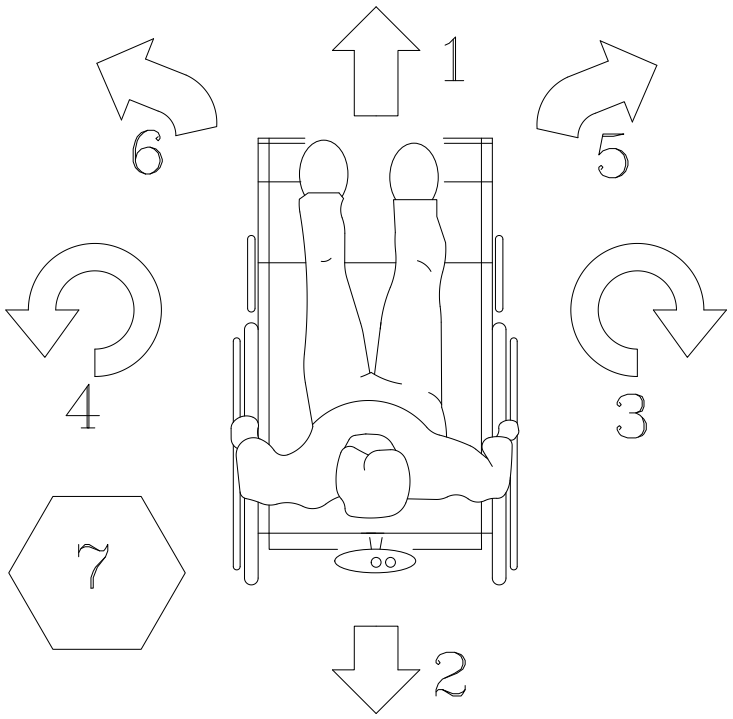
Na literatura encontram-se trabalhos que utilizam a interface de movimentos de cabeça que opera simulando um *Joystick* Analógico (*Head-Joystick*). Os valores das velocidades estão diretamente ligados as magnitudes dos ângulos de inclinação da cabeça, e o usuário deve ajustar constantemente o posicionamento da cabeça durante a navegação.

Neste trabalho é proposto o *Head-Keypad*. Esta HMI tem como objetivo classificar as atuais posições da cabeça do usuário para realização de movimentos na CRI, trabalhando de forma discreta, como se estivesse pressionando botões de um teclado. A justificativa está relacionada ao conforto e facilidade, o usuário deve apenas inclinar sua cabeça levemente na direção desejada e a inteligência envolvida no controle compartilhado irá decidir o valor das velocidades. Com isso haverá uma redução severa de esforço físico. O usuário possui sete classes de atuação, conforme Figura 4.1 e Tabela 4.1.

Tabela 4.1: Classes de Atuação

Classe	**Atuação na CRI**
1	Movimento puramente linear para frente
2	Movimento puramente linear para trás
3	Movimento puramente angular para direita
4	Movimento puramente angular para esquerda
5	Movimento curvilíneo para direita
6	Movimento curvilíneo para esquerda
7	Parado

Figura 4.1: Classes de Atuação



Fonte: Elaborada pelo autor

4.1.1 Sensoriamento

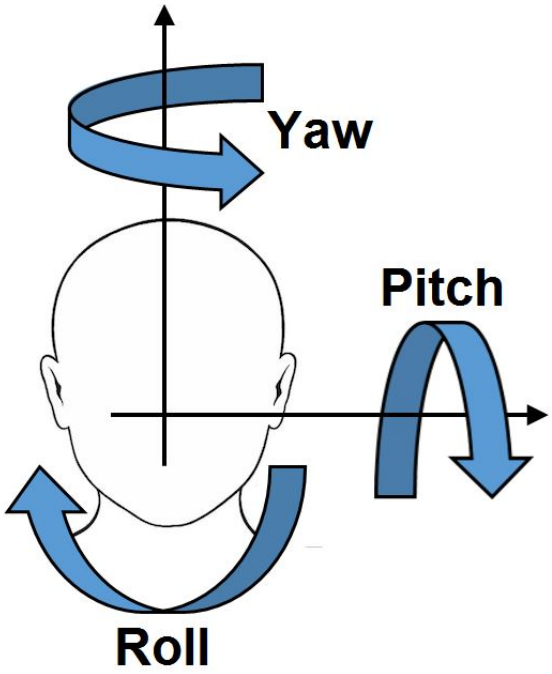
Para realizar a classificação da atual posição da cabeça do usuário são utilizados os ângulos de Euler através da IMU acoplada no topo da cabeça. Os dados amostrados correspondem aos valores dos Ângulos de Euler $roll_h$ e $pitch_h$ do *frame* da cabeça, correspondendo as rotações nos eixos x e y.

A IMU precisa estar devidamente calibrada, fornecendo os ângulos de Euler de forma acurada. O usuário deve iniciar o sistema na posição neutra. Neste momento é computado o *frame* inercial, tornando a posição neutra do usuário a origem do novo sistema. A inclinação da cabeça é computada neste referencial. Cada usuário possui uma inclinação natural da cabeça, e a IMU pode estar posicionada de diferentes maneiras. Este procedimento é utilizado tanto no treinamento quanto na operação.

4.1.2 Reconhecimento de Padrões

Como cada usuário dispõe de uma condição física única, deve-se realizar um treinamento de forma individual. Para cada classe, o usuário deve posicionar levemente

Figura 4.2: *Frame* e Rotações da Cabeça



Fonte: (MACIEL et al., 2017a)

a cabeça de uma determina maneira, como mostrado na Tabela 4.2 e Figura 4.3.

Tabela 4.2: Posicionamento da Cabeça

Classe	**Posição da Cabeça**
1	frente
2	trás
3	direita levemente para trás
4	esquerda levemente para trás
5	direita e para frente
6	esquerda e para frente
7	neutra

No treinamento é pedida a realização de diversas transições aleatórias, cinco vezes para cada classe, já que os ângulos podem sofrer variações para mesma categoria em diferentes momentos. O desejado é obter a dispersão probabilística dos dados no plano $roll - pitch$, no total são 35 movimentações. Utiliza-se giroscópio presente na IMU para reconhecer os transitórios de forma a coletar dados apenas com a cabeça parada. Em cada movimento coletam-se 100 pontos, totalizando 3500 dados que formam o conjunto do treinamento. Em conformidade com estudo em Maciel et al. (2017a), comparando as técnicas de classificação: Distância Euclidiana, Distância de Mahalanobis e Redes Neurais Artificiais para a presente aplicação, a Distância de Mahalanobis apresentou o

Figura 4.3: Posicionamento da Cabeça

Fonte: Elaborada pelo autor

melhor custo benefício em relação à acurácia e complexidade.

O treinamento consiste em calcular o vetor média $\boldsymbol{\mu}_n$ e matriz de covariância $\boldsymbol{S}_n$ de cada uma das 7 classes. Ao entrar um dado a ser classificado, composto por $\boldsymbol{x} = [roll_h, pitch_h]^T$ em radianos, são calculadas as sete distâncias de Mahalanobis, conforme Equação (4.1). O dado pertencerá à classe de menor distância.

$$D_{M[n]}(\boldsymbol{x}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu_n})^T \boldsymbol{S_n^{-1}} (\boldsymbol{x} - \boldsymbol{\mu_n})} \tag{4.1}$$

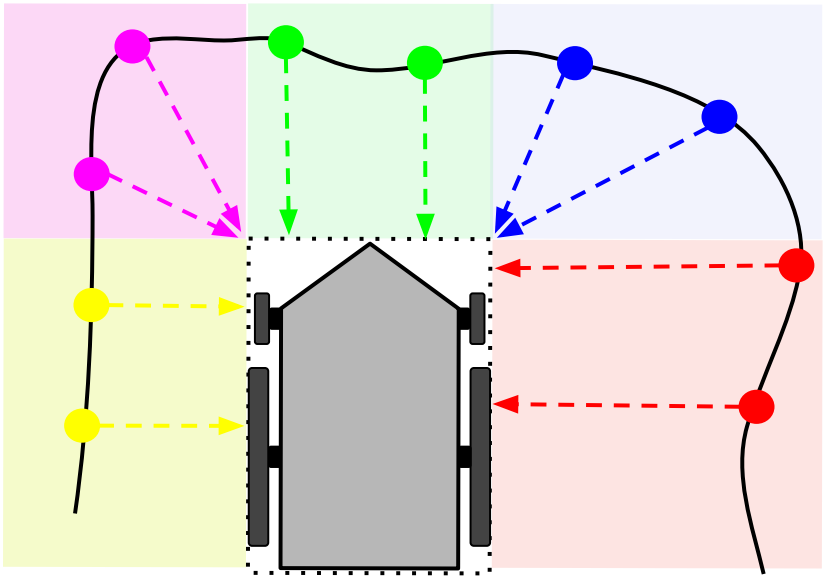
4.2 Campos Vetoriais

O controle compartilhado é baseado em campos vetoriais. Estes campos carregam informações dos obstáculos detectados próximos do robô, atuando juntamente com os comandos provenientes do *Head-Keypad* nas decisões de velocidade angular e linear. A metodologia utilizada é uma adaptação da proposta de navegação assistida de Olivi et al. (2014) que foi apresentada na Seção 2.3, algumas alterações foram propostas visando a melhora no desempenho para a HMI utilizada, e sua aplicação, sem a necessidade de mapeamento e localização. Como, por exemplo, o fato de não haver mapas, não permitiu a utilização do Diagrama de Voronoi do ambiente.

Cada obstáculo detectado nos arredores contribui com um vetor repulsivo. Ao contrário do algoritmo APF, visando uma melhor atuação do campo, o robô não é considerando um ponto adimensional. São consideradas as dimensões da cadeira de rodas e a mesma é modelada topologicamente como um corpo retangular, sendo que toda estrutura deve estar contida neste retângulo.

Os pontos de ação das forças repulsivas oriundas dos obstáculos serão computados sobre as arestas do retângulo. A distância e direção para o cálculo do campo consideram o ponto mais próximo ao obstáculo que pertence a alguma das arestas.

Figura 4.4: Regiões e Direções dos Obstáculos



Fonte: Elaborada pelo autor

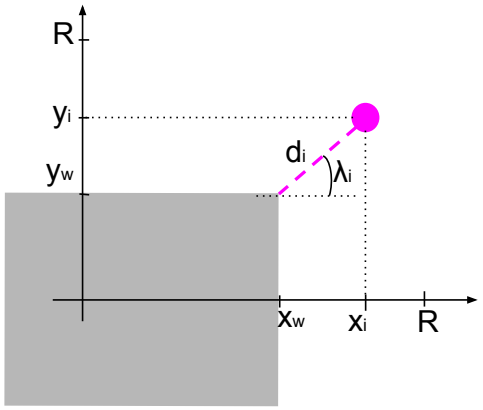
Desta maneira, conforme apresentado na Figura 4.4, os obstáculos podem ser

divididos em 5 regiões de acordo com sua localização em relação ao retângulo: (i) lateral direita, (ii) diagonal direita, (iii) frontal, (iv) diagonal esquerda e (v) lateral esquerda, correspondendo as regiões: vermelha, azul, verde, magenta e amarela respectivamente.

4.2.1 Cálculo do Campo

O cálculo dos campos é realizado utilizando o *frame* do robô. Primeiramente deve-se possuir as coordenadas cartesianas $[x_i, y_i]$ dos obstáculos observados. As leituras que estão a uma distância superior a R da superfície do retângulo serão descartadas. Sendo y_w a metade da largura da CRI e x_w a distância entre a origem do *frame* à frente, calculam-se os cinco campos. A Figura 4.5 contém um exemplo. Os campos são calculados baseados nas distâncias do obstáculo à aresta d_i e angulação λ_i, a Tabela 4.3 apresentá os limites das regiões, além dos cálculos da distância e orientação.

Figura 4.5: Distância e Direção do Campo Inferido por um Obstáculo



Fonte: Elaborada pelo autor

Tabela 4.3: Cálculo das Distâncias e Direções dos Campos

Região	**Limite** x_i	**Limite** y_i	d_i	λ_i
1	$x_i \leq x_w$	$y_i < -y_w$	$-y_i - y_w$	$-\pi/2$
2	$x_i > x_w$	$y_i < -y_w$	$\sqrt{(x_i - x_w)^2 + (y_i + y_w)^2}$	$atan2(y_i + y_w, x_i - x_w)$
3	$x_i > x_w$	$-y_w \leq y_i \leq y_w$	$x_i - x_w$	0
4	$x_i > x_w$	$y_i > y_w$	$\sqrt{(x_i - x_w)^2 + (y_i - y_w)^2}$	$atan2(y_i - y_w, x_i - x_w)$
5	$x_i \leq x_w$	$y_i > y_w$	$y_i - y_w$	$\pi/2$

Com as distâncias e ângulos, calcula-se o campo repulsivo $\vec{C}$ por meio da Equação 4.2, onde β é um ganho de acordo com o ângulo do obstáculo e duas contantes, a_1 e a_2, de forma de valorizar obstáculos laterais, Equação 4.3.

$$\vec{C} = \sum_{i=1}^{i_{max}} \frac{-\beta}{d_i} \begin{bmatrix} \cos(\lambda_i) \\ \sin(\lambda_i) \end{bmatrix} \tag{4.2}$$

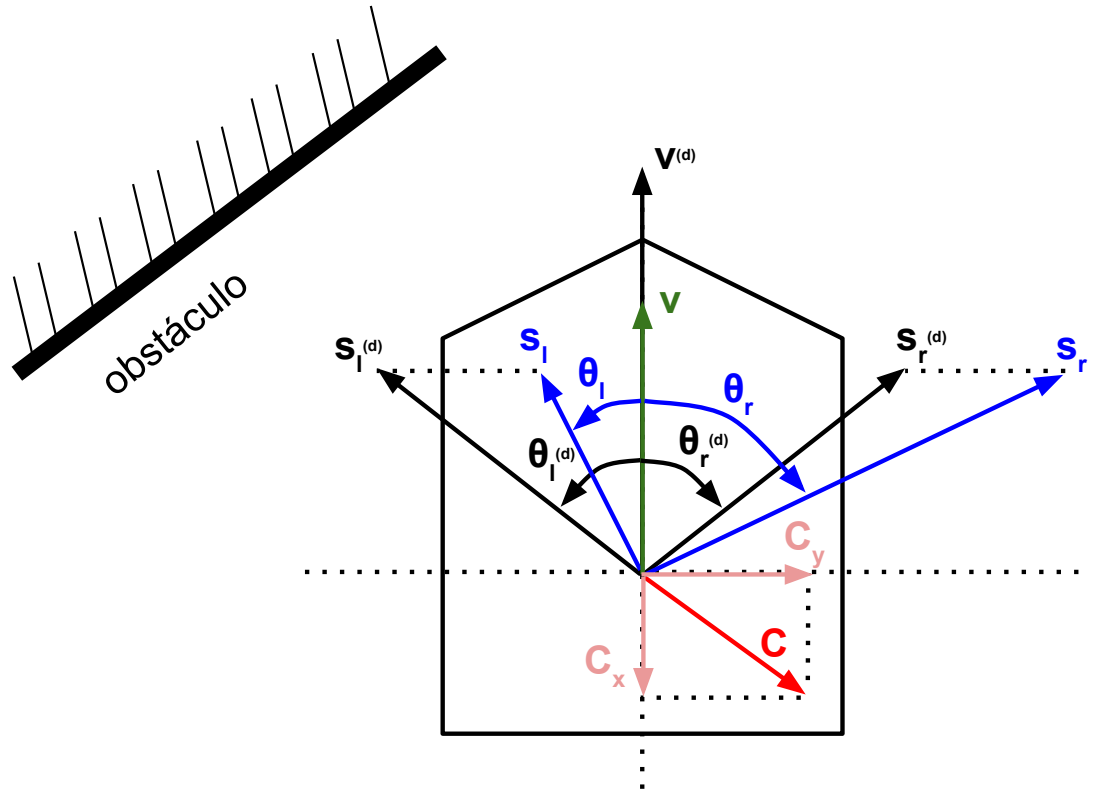
$$\beta = a_1 \lambda_i^2 + a_2 \tag{4.3}$$

4.2.2 Atuação do Campo

O robô possui vetores naturais que comandam o comportamento quando não há nenhum campo repulsivo. São eles o de velocidade $\vec{\boldsymbol{v}}^{(d)}$ e os de giro à direita e esquerda, $\vec{\boldsymbol{s}}_r^{(d)}$ e $\vec{\boldsymbol{s}}_l^{(d)}$. Devido à ausência de um Diagrama de Voronoi, algumas mudanças foram propostas para adaptação da técnica de (OLIVI et al., 2013).

O campo repulsivo $\vec{\boldsymbol{C}}$ vai interagir com os vetores naturais, obtendo os novos vetores de velocidade linear $\vec{\boldsymbol{v}}$ e de giro $\vec{\boldsymbol{s}}_r$ e $\vec{\boldsymbol{s}}_l$, vide Figura 4.6.

Figura 4.6: Atuação do Campo Repulsivo



Fonte: Elaborada pelo autor

Sabe-se que o campo repulsivo é composto pelas componentes ortogonais no *frame* do robô $\vec{C} = [C_x,\ C_y]^T$, onde $\vec{C}_x$ agirá sobre o vetor de velocidade linear e $\vec{C}_y$, sobre os

vetores de giro. Para o novo vetor de velocidade linear há:

$$\vec{v} = \vec{v}^{(d)} + \vec{C}_x \tag{4.4}$$

Estando paralelo ao eixo x, $\vec{v} = [v_x,\ 0]^T$, a velocidade linear assumirá um valor proporcional a v_x, o qual é calculado pela Equação 4.5:

$$v_x = \left\|\vec{v}^{(d)}\right\| + C_x \tag{4.5}$$

Como exibido na Equação 4.6, o vetor $\vec{C}_y$ será somado aos vetores de giro *default* e com os novos $\vec{s}_r$ e $\vec{s}_l$, são calculados os ângulos de giro pela Equação 4.7.

$$\vec{s}_{r,l} = \vec{s}_{r,l}^{(d)} + \vec{C}_y \tag{4.6}$$

$$\theta_{r,l} = acos\left(\frac{\vec{s}_{r,l}.\vec{v}}{\|\vec{s}_{r,l}\| \,.\, \|\vec{v}\|}\right) \tag{4.7}$$

Nota-se que a distribuição dos obstáculos ao redor influencia a magnitude dos vetores, de modo a facilitar os desvios de obstáculos. O vetor de velocidade linear $\boldsymbol{v}$ se reduz com a proximidade de obstáculos a frente do robô. Os ângulos de giro vão aumentar ou diminuir em conformidade com a distribuição lateral dos obstáculos.

4.3 Controle Compartilhado de Velocidade

Estando previamente selecionado o modo manual pelo comando de voz, uma atuação nas velocidades da CRI será realizada por meio da atual classe de decisão relacionada ao movimento da cabeça juntamente com os campos vetoriais calculados. Diferente do APF, os obstáculos não são soberanos nas decisões de trajetória. A direção do movimento é dada pelo usuário e os campos vetoriais atuam modificando a magnitude das velocidades, de forma a tornar navegação manual mais fácil e segura.

Optou-se por não impedir completamente as colisões. Se a CRI estiver muito próxima de um obstáculo as velocidades serão reduzidas a valores mínimos, de forma que a colisão ocorrerá caso seja a vontade do usuário. O objetivo do método é facilitar

a navegação impedindo colisões involuntárias. As velocidades finais de atuação da CRI dependem (i) do modo de operação selecionado pelo usuário por voz, (ii) do atual padrão da inclinação da cabeça e (iii) dos valores de v_x e θ_r e θ_l obtidos pelo campo repulsivo $\vec{C}$.

Vale a pena salientar que a CRI jamais irá realizar um movimento não esperado pelo usuário mesmo com campos repulsivos intensos, apenas reduzirá a velocidade a valores mínimos. Por exemplo, caso o usuário deseje um movimento linear à frente, mesmo que haja um obstáculo muito próximo de modo que $\| \vec{v}^{(d)} \| - C_x < 0$, a cadeira jamais irá assumir uma velocidade negativa. O mesmo acontece quando campos laterais intensos mudem o sentido de $\theta_{r,l}$, a curva não será executada no sentido contrário do desejado.

Tabela 4.4: Velocidades de Atuação

Classe	Modo	Velocidade Linear	Velocidade Angular
↑ (1)	low speed	v_{min}	0
	half speed	$max(v_{min},\ \alpha_1 v_x)$	0
	full speed	$max(v_{min},\ \alpha_2 v_x)$	0
↓ (2)	-	$-v_{min}$	0
↷ (3)	low speed	0	$-\omega_{min}$
	half speed	0	$min(-\omega_{min},\ \alpha_3 \theta_r)$
	full speed	0	$min(-\omega_{min},\ \alpha_4 \theta_r)$
↶ (4)	low speed	0	ω_{min}
	half speed	0	$max(\omega_{min},\ \alpha_3 \theta_l)$
	full speed	0	$max(\omega_{min},\ \alpha_4 \theta_l)$
↗ (5)	low speed	v_{min}	$-\omega_{min}$
	half speed	$max(v_{min},\ \alpha_5 v_x)$	$min(-\omega_{min},\ \alpha_6 \theta_r)$
	full speed	$max(v_{min},\ \alpha_7 v_x)$	$min(-\omega_{min},\ \alpha_8 \theta_r)$
↖ (6)	low speed	v_{min}	ω_{min}
	half speed	$max(v_{min},\ \alpha_5 v_x)$	$max(\omega_{min},\ \alpha_6 \theta_l)$
	full speed	$max(v_{min},\ \alpha_7 v_x)$	$max(\omega_{min},\ \alpha_8 \theta_l)$
□ (7)	-	0	0

A Tabala 4.4 apresenta as velocidades para todos os casos do sistema. No modo "*low speed*" a cadeira só assume valores mínimos, já que este modo é apenas para manobras delicadas, como estacionar em um local pequeno. Para os modos "*half speed*" e "*full speed*" as velocidades serão de acordo com ganhos α_n e o campo vetorial $\vec{C}$. São utilizadas funções de máximo e mínimo, que garantirão que a cadeira respeitará as intensões de direção de movimento do usuário independente do campo repulsivo.

Para Classe 2, referente ao movimento retilíneo para trás, independente do modo

de velocidade, a cadeira assume uma velocidade pequena, já que esta situação não é segura para um cadeirante, pela baixa visibilidade e falta de sensoriamento. O objetivo desta classe é apenas sair de um local apertado, como por exemplo, retirar-se de uma mesa. A Classe 7 mantém a cadeira parada, forçando as velocidades a serem nulas.

5 Passagens Estreitas

Passagens estreitas são um grande desafio para o controle de CRIs, mesmo com o algoritmo de navegação compartilhada, apresentado no capítulo anterior. Para isso, este trabalho propõe um sistema de controle de movimento autônomo que reconhece passagens estreitas em ambientes desconhecidos e as atravessa sem recorrer a técnicas de localização global e mapeamento. Apesar do foco trabalho estar dirigido as CRIs, a metodologia pode ser aplicada a qualquer robô diferencial devidamente sensoreado. A técnica foi implementada em *python* dentro do nó "Shared Control"na arquitetura proposta.

5.1 Reconhecimento de Passagens

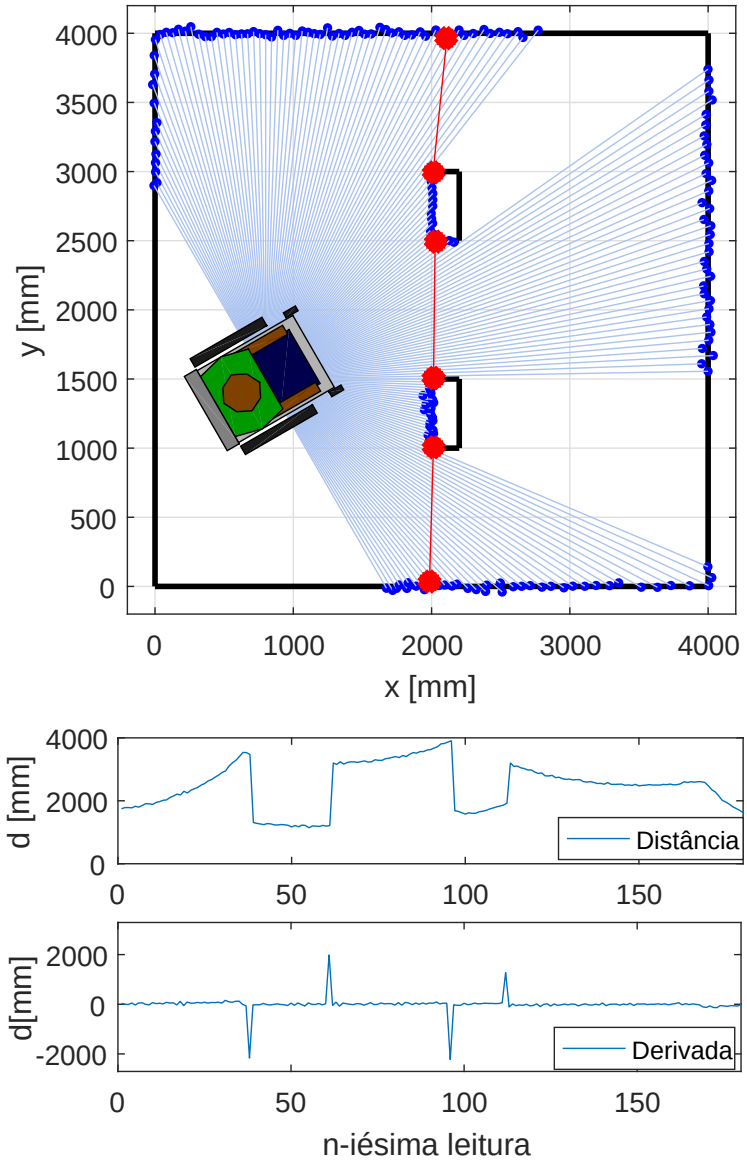
Para a realização da metodologia proposta é necessário um sensor do tipo *RangeFinder* para realizar uma varredura bidimensional dos obstáculos à frente e no mesmo plano do robô, resultando em um conjunto de pontos que contém a distância e o ângulo dos mesmos em relação ao sensor. É uma tarefa que pode ser realizada utilizando um *laser scanner* ou, como neste trabalho, a partir de uma câmera de profundidade. Sonares não seriam adequados já que é necessário um número razoável de medidas *Range Finder*.

Cada passagem será estimada por dois pontos, que constituem seus extremos, direito e esquerdo respectivamente. O objetivo do algoritmo é estabelecer, caso haja passagens à frente, quais das medidas n de uma varredura no sentido anti-horário do sensor podem ser consideradas bordas.

Primeiramente, identificam-se possíveis limites através da detecção de alta taxa de variação da distância em relação aos obstáculos vizinhos, isto porque uma passagem será constituída de pelo menos um ponto de descontinuidade. A Figura 5.1 contém um exemplo em uma região com 3 passagens: a do centro é formada por dois pontos de alta variação e as demais por apenas um. Abaixo são apresentados sinais relacionando às distâncias e suas referentes derivadas.

Se a n-ésima leitura é uma descontinuidade positiva, significa que pode ser o

Figura 5.1: Análise dos Sinais de Distância

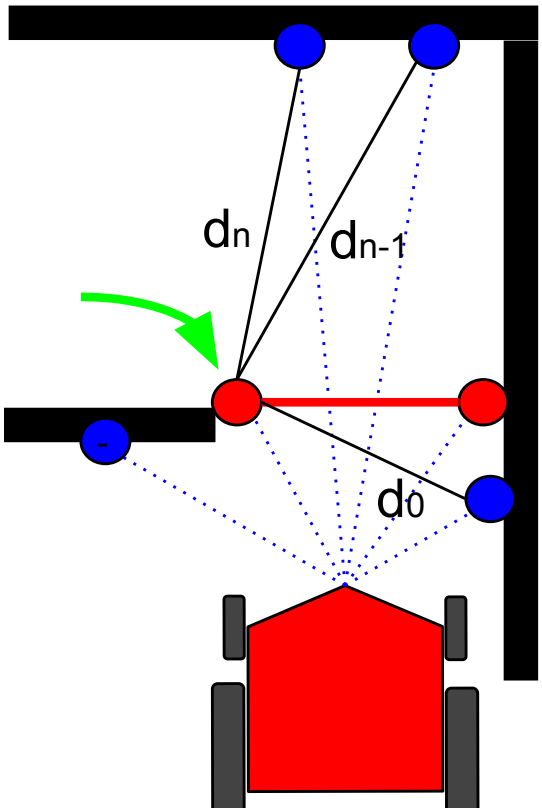


Fonte: (MACIEL et al., 2017b)

início de um acesso. Sendo assim, calcula-se a distância deste ponto a todos os seguintes, de $n+1$ até n_{max}. A leitura com a menor distância será considerada o extremo final. Analogamente, se a descontinuidade é negativa, implica em um provável final de passagem e, então, calcula-se a distância a todos os pontos anteriores, de 0 até $n-1$ e novamente forma-se o par com o ponto de menor distância.

Caso a passagem possua dois pontos de descontinuidade, a mesma será estimada duas vezes, portanto se descarta a que possuir maior tamanho, já que o par que forma a menor fenda é o mais crítico. A Figura 5.2 ilustra todo processo, em uma situação de descontinuidade negativa.

Figura 5.2: Reconhecimento da Passagem



Fonte: Elaborada pelo autor

Por último deve-se avaliar as passagens que foram estimadas. O objetivo desta etapa é rejeitar os pares que não satisfazem as condições necessárias para uma travessia segura, os que foram estimados de forma equivocada pelas limitações envolvidas na etapa anterior ou que simplesmente devem ser ignorados no contexto da missão. São avaliados e rejeitados os pares em algumas situações:

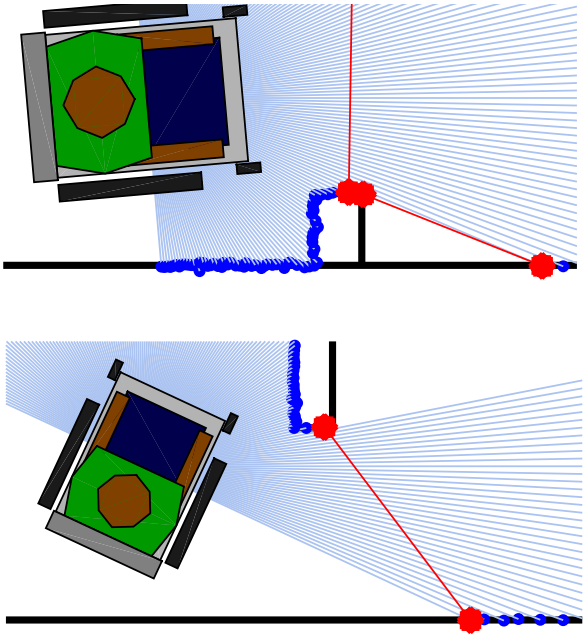
- Largura da passagem: rejeitam-se as pequenas em relação à largura do robô, ou tão grandes que não é necessária uma travessia autônoma.

- Distância ao robô: define-se um raio em volta do robô em que os acessos que estão por fora sejam desconsiderados, passagens distantes são mais susceptíveis a uma estimação equivocada. Evita-se que a identificação de passagens adjacentes que podem comprometer a missão enquanto o robô realiza uma travessia.

- Composição: Rejeitam-se passagens em que não há uma diferença angular aceitável entre as leituras dos dois extremos, ou que são formadas por leituras próximas dos limites do campo de visão sensor, ou seja, da primeira e a última medida da

varredura. A Figura 5.3 demonstra as situações que serão evitadas neste caso.

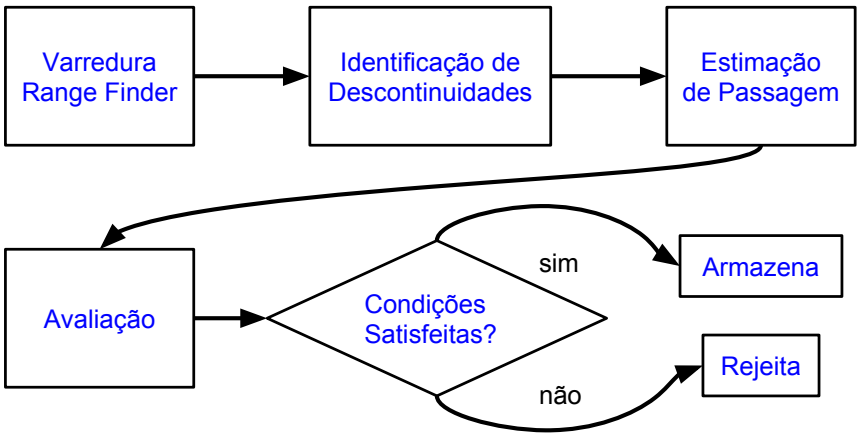
Figura 5.3: Exemplo de Reconhecimentos Enganosos



Fonte: Elaborada pelo autor

A Figura 5.4 apresenta o fluxograma resumindo todas as etapas do algoritmo proposto.

Figura 5.4: Fluxograma do Reconhecimento de Passagens

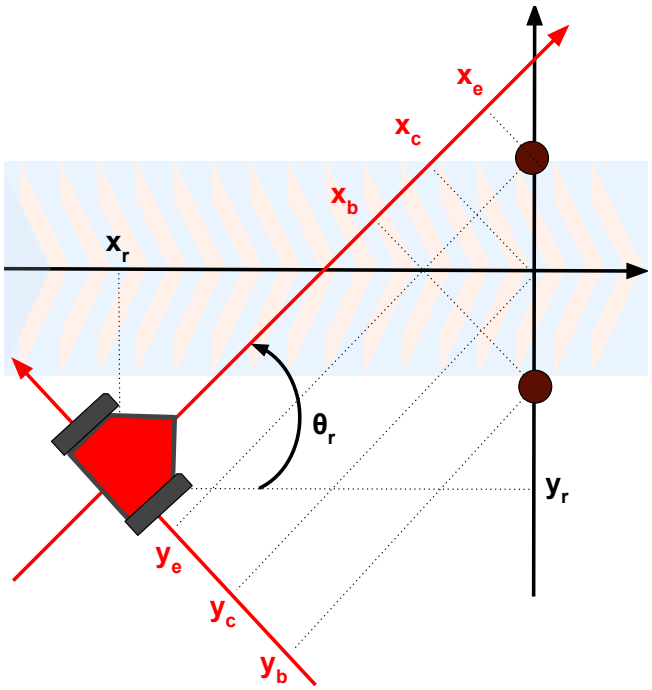


Fonte: Elaborada pelo autor

5.2 Modelagem e Localização

Após o reconhecimento da passagem estreita, tem-se a distância e os ângulos de seus pontos de início e fim em relação ao referencial do robô, d_b, θ_b, d_e *e* θ_e respectivamente. Faz-se necessária a criação de um novo sistema de coordenadas em relação ao local da passagem, cuja origem é o ponto médio das duas extremidades, ou seja, o centro da passagem. O objetivo final é parametrizar a localização do robô em relação ao *frame* da passagem, dada pelas coordenadas cartesianas x_r e y_r e a orientação θ_r. A Figura 5.5 ilustra os sistemas de coordenadas e variáveis envolvidos nesta localização.

Figura 5.5: Sistemas de Coordenadas



Fonte: (MACIEL et al., 2017b)

Inicialmente calculam-se os pontos de início, meio e fim da passagem, dados por $[x_b, y_b]^T$, $[x_c, y_c]^T$ e $[x_e, y_e]^T$ em relação ao *frame* do robô. Com estes três pontos é computada a localização em relação à passagem, conforme Equação (5.1), na qual se tem a priori apenas a distância e os ângulos dos extremos da passagem através do procedimento da Seção 5.1.

$$
\begin{aligned}
x_b = d_b\cos(\theta_b) \qquad &\text{e} \qquad y_b = d_b\sin(\theta_b) \\
x_e = d_e\cos(\theta_e) \qquad &\text{e} \qquad y_e = d_e\sin(\theta_e) \\
x_c = \frac{x_b + x_e}{2} \qquad &\text{e} \qquad y_c = \frac{y_b + y_e}{2} \\
\theta_r = -atan2(y_e - y_b, x_e - x_b) + \frac{\pi}{2} & \\
x_r = -x_c\cos(\theta_r) + y_c\sin(\theta_r) & \\
y_r = -x_c\sin(\theta_r) - y_c\sin(\theta_r) &
\end{aligned}
\tag{5.1}
$$

5.2.1 Aplicação do Filtro de Kalman Estendido

Para obtenção de uma localização mais estável é aplicado o EKF com o objetivo de atenuar os ruídos aleatórios e incertezas inerentes ao processo de estimação da localização proposto. São reduzidos os efeitos causados por: (i) desvios de leitura do sensor *RangeFinder* e (ii) variações que ocorrem na estimação realizada devido a discretização angular das medidas dos obstáculos durante a varredura.

Alimentado pela odometria do robô, uma outra função do filtro é estimar a posição mesmo quando, após o início da travessia, o robô pare de observar a passagem. É uma situação comum a passagem sair do campo de visão durante a manobra. A amplitude deste campo influencia diretamente no desempenho da observação, por exemplo: um sensor *Laser Scanner* SICK LMS-291 com 180 graus observará a passagem com maior eficiência do que uma câmera de profundidade Kinect com 60 graus de cobertura.

Do momento inicial até o instante em que o robô tenha atravessado completamente a passagem, o EKF fará o rastreamento *online* da posição. O algoritmo foi implementado conforme apresentado em Negenborn (2003).

Modelos de Sistema e Medição

As variáveis de estado do sistema parametrizam a localização atual do robô em relação à passagem. As variáveis de estados estão exibidas na Equação (5.2).

$$\boldsymbol{x}_k = \begin{bmatrix} x_{[x],k} \\ x_{[y],k} \\ x_{[\theta],k} \end{bmatrix} \tag{5.2}$$

A dinâmica do sistema, Equação (5.3), é atualizada a partir de um deslocamento linear e uma variação angular em um determinado período, conforme Equação (5.4). Estas duas entradas devem ser computadas através de sensores odométricos.

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) = \begin{bmatrix} f_x(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) \\ f_y(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) \\ f_\theta(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1} \end{bmatrix} \tag{5.3}$$

$$\boldsymbol{u}_k = \begin{bmatrix} u_{[\Delta D],k} \\ u_{[\Delta \phi],k} \end{bmatrix} \tag{5.4}$$

É assumido que as fontes de ruído do sistema são independentes umas da outras. Com isso os elementos não-diagonais da matriz de covariância do modelo podem ser desprezados, como exibido na Equação (5.5).

$$\boldsymbol{Q} = \begin{bmatrix} \sigma^2_{\omega_{[x]}} & 0 & 0 \\ 0 & \sigma^2_{\omega_{[y]}} & 0 \\ 0 & 0 & \sigma^2_{\omega_{[\theta]}} \end{bmatrix} \tag{5.5}$$

Apesar do algoritmo de localização de passagens ser fruto indireto de leituras de sensor *range finder*, por questões de simplicidade, é considerado que se possui um sensor de estado completo, que retorna diretamente as medidas das variáveis de estado, como exibido na (5.6). Na prática, está sendo considerado um sensor virtual, que combina toda a informação do processo de reconhecimento de passagens real. É assumido um modelo perfeito de medição, e novamente as variáveis medidas são independentes. Com isso os termos não diagonais da matriz e covariância do processo são anulados $\boldsymbol{R} = diag\Big(\sigma^2_{z[x]},\ \sigma^2_{z[y]},\ \sigma^2_{z[\theta]}\Big)$.

$$\boldsymbol{z}_k = \begin{bmatrix} z_{[x],k} \\ z_{[y],k} \\ z_{[\theta],k} \end{bmatrix} = h(\boldsymbol{x}_k) = \boldsymbol{x_k} \tag{5.6}$$

Etapa de Predição do Vetor de Estados

A etapa de predição consiste em, a partir das informações dos incrementos linear e angular ΔD_k *e* $\Delta\phi_k$ vindas da odometria, fazer a predição do próximo estado do sistema. Conforme visto na Seção 2.1, o modelo cinemático do robô diferencial é dado por:

$$\bar{\boldsymbol{x}}_k = \boldsymbol{x}_{k-1} + \begin{bmatrix} \Delta D_k \cos(x_{\theta,k-1} + \Delta\theta_k/2) \\ \Delta D_k \sin(x_{\theta,k-1} + \Delta\theta_k/2) \\ \Delta\theta_k \end{bmatrix} \tag{5.7}$$

Ainda na predição, é estimada a covariância dos erros a priori pela Equação (5.10). Para isso é calculado, primeiramente, o jacobiano de (5.3) para o modelo (5.7), chegando na Equação (5.9).

$$\boldsymbol{A}_k = \begin{bmatrix} \frac{\partial f_x}{\partial x_{[x]}} & \frac{\partial f_x}{\partial x_{[y]}} & \frac{\partial f_x}{\partial x_{[\theta]}} \\ \frac{\partial f_y}{\partial x_{[x]}} & \frac{\partial f_y}{\partial x_{[y]}} & \frac{\partial f_y}{\partial x_{[\theta]}} \\ \frac{\partial f_\theta}{\partial x_{[x]}} & \frac{\partial f_\theta}{\partial x_{[y]}} & \frac{\partial f_\theta}{\partial x_{[\theta]}} \end{bmatrix} \tag{5.8}$$

$$\boldsymbol{A}_k = \begin{bmatrix} 1 & 0 & -\Delta D_k \sin(x_{\theta,k-1} + \Delta\theta_k/2) \\ 0 & 1 & \Delta D_k \cos(x_{\theta,k-1} + \Delta\theta_k/2) \\ 0 & 0 & 1 \end{bmatrix} \tag{5.9}$$

$$\boldsymbol{P}_k = \boldsymbol{A}_k \boldsymbol{P}_{k-1} \boldsymbol{A}_k^T + \boldsymbol{Q} \tag{5.10}$$

Etapa de Correção do Vetor de Estados

A etapa de correção minimiza a propagação do erro da predição do vetor de estados ao longo do tempo, que é baseada somente no uso de um modelo cinemático. Sendo assim, a diferença entre predição do modelo simulado e a medição de observação é compensada.

A matriz de observação $\boldsymbol{H}_k$ é o jacobiano do modelo de observação $h(\boldsymbol{x}_k)$. Como foi considerado um sensor de estado completo, $\boldsymbol{H}_k$ resulta em uma matriz identidade de ordem 3. Finalmente a Equação (5.11) computa a matriz de ganhos de Kalman, com a mesma é corrigido o vetor de estados e a covariância do erro a posteriori pelas Equações (5.12) e (5.12) respectivamente.

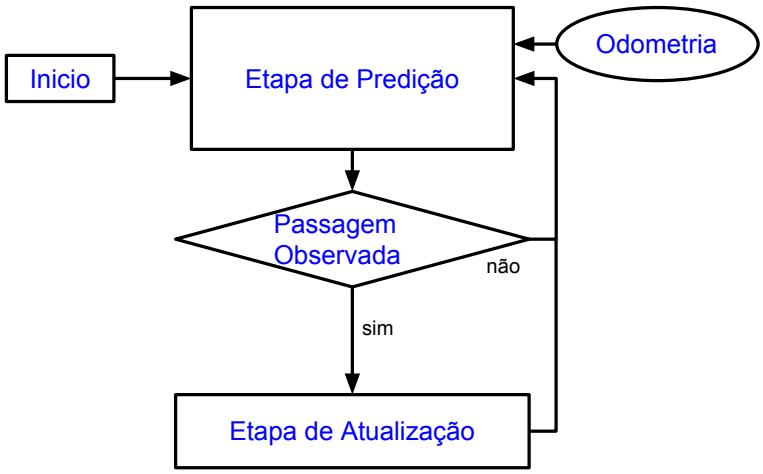
$$\boldsymbol{K}_k = \boldsymbol{P}_k \boldsymbol{H}_k^T (\boldsymbol{H}_k \boldsymbol{P}_k \boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1} \tag{5.11}$$

$$\boldsymbol{x}_k = \bar{\boldsymbol{x}}_k + \boldsymbol{K}_k (\boldsymbol{z}_k - \boldsymbol{H}_k \boldsymbol{x}_k) \tag{5.12}$$

$$\bar{\boldsymbol{P}}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_k \tag{5.13}$$

Durante uma missão de travessia de passagem, a predição é executada continuamente, enquanto a correção ocorre somente quando há atualização das medidas dos sensores, como apresentado na Figura 5.6 .

Figura 5.6: Fluxograma do EKF



Fonte: Elaborada pelo autor

5.3 Projeto do Controlador

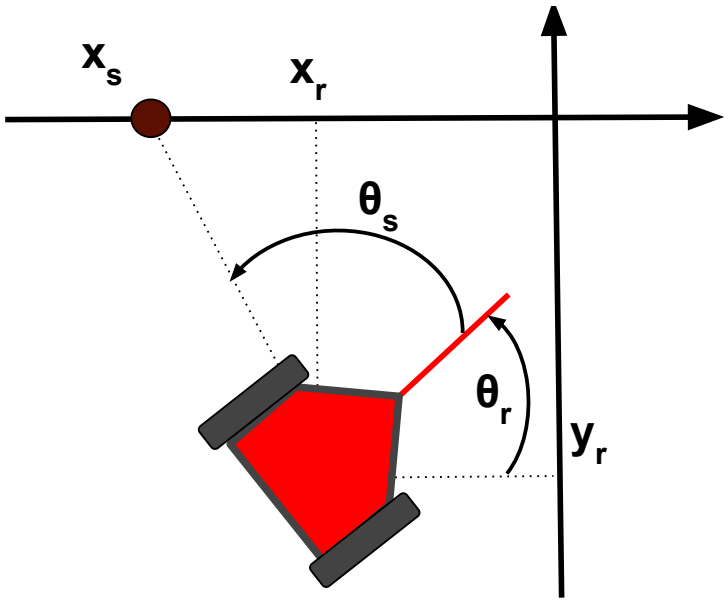
O *frame* da passagem estreita é utilizado para a localização local do robô até que a travessia seja completada. Assim, torna-se desnecessário o conhecimento do mapa detalhado do ambiente para esta finalidade, fazendo-se dispensável o uso de algoritmos de

localização global e mapeamento, reduzindo o custo computacional total. Para atravessar uma passagem com segurança, o robô deve percorrer a reta central formada pelo eixo x do *frame* da passagem estreita, passando o mais perto possível por sua origem com um ângulo de apontamento adequado.

Algoritmos seguidores de retas clássicos, como apresentados em Corke (2011), são ineficazes para o caso, já que não consideram o momento e o ponto em que o robô intercepta a reta, algo que é necessário para garantir a travessia pela passagem em segurança. O controlador deve ser capaz de (i) levar o robô até próximo à reta coincidente com o eixo x do *frame* da passagem estreita, e (ii) utilizá-la como referência de navegação. Quanto mais cedo ocorre a interceptação dessa reta pelo robô, maior a confiabilidade da manobra.

Assim foram adicionadas duas variáveis a este sistema de coordenadas: o ponto de segurança x_s e o ângulo do robô a este ponto $\theta_s = atan2(-y_r, x_s - x_r) - \theta_r$, como apresentado na Figura 5.7. Este ponto é uma referência segura para interceptar a reta e facilitar manobras quando o robô está inicialmente mal posicionado.

Figura 5.7: Entradas de Controle



Fonte: (MACIEL et al., 2017b)

Esta dissertação propõe um controlador de navegação não linear, que age diretamente nestas velocidades em alto nível, ambas controladas de forma independente. Os sinais de controle são ambos divididos em três parcelas, cada uma com uma função específica no processo, como exposto na Equação (5.14).

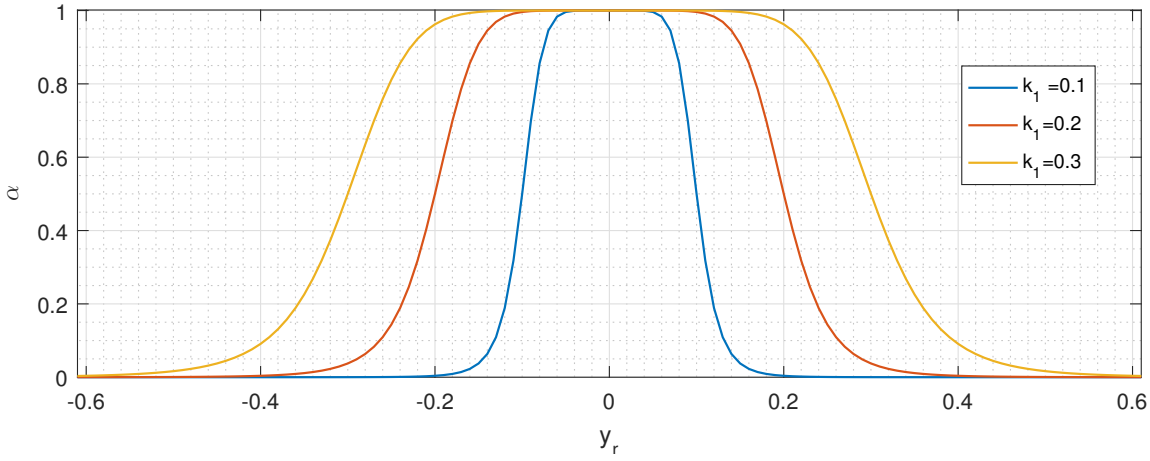
$$\begin{aligned} v &= v_1 + v_1 + v_3 \\ \omega &= \omega_1 + \omega_2 + \omega_3 \end{aligned} \tag{5.14}$$

A missão tem dois objetivos: (i) aproximar o robô do ponto de segurança quando o mesmo estiver longe, para preparar a manobra de travessia e (ii) seguir a reta de referência formada pelo eixo x, que passa exatamente no meio da passagem estreita. O controle poderia ser realizado em duas etapas sequenciais com dois controladores distintos, havendo um chaveamento entre eles ao chegar ao ponto de segurança. Mas, para uma obter um trajeto mais rápido e suave, optou-se por utilizar apenas um controlador, e introduziu-se a variável α, Equação (5.15), com objetivo de controlar o esforço de controle durante a travessia.

$$\alpha = \frac{1}{1 + \left(\frac{y_r}{k_1}\right)^8} = \frac{k_1^8}{k_1^8 + y_r^8} \tag{5.15}$$

A variável α é uma função sino generalizada de grau 8, dependente direta da distância do robô à reta central e pelo ganho k_1. Esta função apresenta saturação em 0 para locais afastados da reta central, aproximadamente ($|y_r| > 2k_1$), e em 1 em locais próximos ($|y_r| < 0.5k_1$), havendo uma transição suave entre essas regiões, vide Figura 5.8.

Figura 5.8: Variável de Controle α



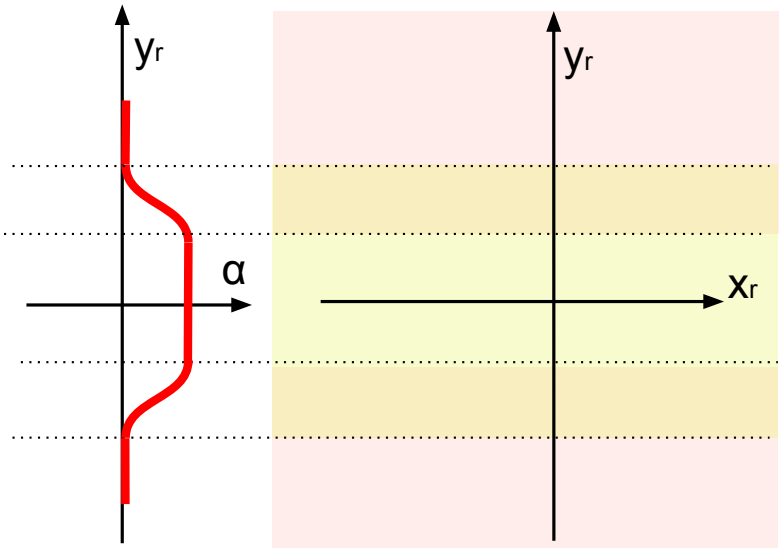
Fonte: Elaborada pelo autor

No cálculo das parcelas da Equação (5.14) haverá a multiplicação por α ou por seu complementar $(1 - \alpha)$, administrando os esforços do controlador de acordo com a

localização atual do robô. Portanto, há três regiões, sendo uma delas de interseção, onde todas as parcelas terão atuação notável, como apresentado na Figura 5.9.

Figura 5.9: Organização das Regiões de Atuação por α

.



Fonte: Elaborada pelo autor

Cada parcela do controlador tem um objetivo. As leis de controle v_1 e ω_1 são atuantes na região mais distante da reta central, sendo responsáveis em aproximar o robô desta reta, ω_1 é o responsável em orientá-lo, corrigindo o erro em θ_s. Já v_1 age reduzindo a velocidade a medida que se aproxima da reta ou quando o robô está mal orientado em relação ao ponto de segurança.

$$\omega_1 = k_2(1-\alpha)\theta_s \tag{5.16}$$

$$v_1 = k_3(1-\alpha)(y_r \cos(\theta_s/2))^2 \tag{5.17}$$

À medida que se aproxima do eixo x do *frame* da passagem, $\alpha \to 1$, faz com que $\{v_1, \omega_1\} \to \{0, 0\}$. Desta maneira, as parcelas v_2, v_3, ω_2 e ω_3 tornam-se predominantes. Por precaução, a velocidade v_2 decresce com a aproximação à passagem ou caso o robô esteja mal orientado em relação à direção da mesma. Já v_3 é uma pequena velocidade que vai suprir v_2 em momentos que tenda a zero, diminuindo o tempo da missão.

As parcelas ω_2 e ω_3 atuam como seguidores de reta, sendo que ω_2 corrige o erro em y_r e mantém o robô próximo a reta central. Já ω_3 é responsável por corrigir o erro de orientação θ_r para apontar o robô em direção ao acesso.

$$\omega_2 = -k_4 \alpha y_r \cos(\theta_r) \tag{5.18}$$

$$\omega_3 = -k_5 \alpha \theta_r \tag{5.19}$$

$$v_2 = k_6 \alpha (x_r \cos(\theta_r))^2 \tag{5.20}$$

$$v_3 = k_7 \alpha \tag{5.21}$$

5.3.1 Estabilidade

A dinâmica de um robô diferencial bidimensional pode ser modelada como

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \cos(\theta_r) & 0 \\ \sin(\theta_r) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{5.22}$$

onde v e ω são respectivamente as velocidades linear e angular de deslocamento do robô.

O sistema de controle de navegação proposto neste trabalho, bem como a planta controlada, são não lineares. Não foi realizado um estudo de estabilidade global, mas como é utilizada a variável α que satura as parcelas do controlador foi feito um estudo de estabilidade de forma separada para as regiões. Quando $\alpha \to 0$ o sistema se resume na simples lei de controle:

$$\begin{aligned} \omega &= k_2 \theta_s \\ v &= k_3 (y_r \cos(\theta_s/2))^2 \end{aligned} \tag{5.23}$$

Apesar de não garantir um bom desempenho, este subsistema será estável se $\{k2, k3\} > \{0, 0\}$, garantindo que o robô entre na região próxima à reta apontando para o ponto de segurança em um tempo finito. Uma vez próximo a reta $\alpha \to 1$ o controlador torna-se:

$$\begin{aligned} \omega &= -k_4 y_r \cos\theta_r - k_5 \theta_r \\ v &= k_7 + k_6 x_r^2 \cos{\theta_r}^2 \end{aligned} \tag{5.24}$$

Neste caso não existe estabilidade na variável x_r já que a proposta é que se percorra o eixo x_r, seguindo a reta central ininterruptamente até completada a travessia. Assim a variável x_r será desprezada desta análise e tratada como um distúrbio. Desta forma a Equação dinâmica (5.22) torna-se a Equação (5.25), onde v e ω oriundos da Equação (5.24).

$$\dot{\boldsymbol{p}} = \begin{bmatrix} \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} v \sin\theta_r \\ \omega \end{bmatrix} \tag{5.25}$$

Para avaliar as redondezas da reta central, pode-se utilizar o método indireto de Lyapunov para provar a estabilidade local do sistema em malha fechada. Para isso o sistema é linearizado por meio da matriz jacobiana em um ponto de equilíbrio ($\dot{\boldsymbol{p}} = 0$), no caso em $\{y_r, \theta_r\} = \{0, 0\}$.

$$\left.\frac{\partial \dot{\boldsymbol{p}}}{\partial \boldsymbol{p}}\right|_{\boldsymbol{p^0}} = \left[\begin{matrix} \dfrac{\partial}{\partial y_r}[v\sin(\theta_r)] & \dfrac{\partial}{\partial \theta_r}[v\sin(\theta_r)] \\ \dfrac{\partial w}{\partial y_r} & \dfrac{\partial w}{\partial \theta_r} \end{matrix}\right]\Bigg|_{\boldsymbol{p^0}} \tag{5.26}$$

Substituindo as leis de controle e aplicando no ponto de equilíbrio há:

$$\left.\frac{\partial \dot{\boldsymbol{p}}}{\partial \boldsymbol{p}}\right|_{y_r=\theta_r=0} = \begin{bmatrix} 0 & k_6 x_r^2 + k_7 \\ -k_4 & -k_5 \end{bmatrix} \tag{5.27}$$

O sistema resultante, exibido na Equação (5.27), é linear, de segunda ordem e invariante no tempo. Com ele é obtida uma aproximação do comportamento do sistema original nos arredores do ponto de equilíbrio, ou seja, quando o robô está próximo à reta e apontado em direção à saída. É possível analisar a estabilidade dessa região pelos autovalores de 5.27, garantindo que localmente, para os ganhos escolhidos, haverá uma dinâmica estável e previsível. Os autovalores são dados por:

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} -k_5 + \sqrt{k_5^2 - 4k_4(k_6 x_r^2 + k_7)} \\ -k_5 - \sqrt{k_5^2 - 4k_4(k_6 x_r^2 + k_7)} \end{bmatrix} \tag{5.28}$$

Todos os ganhos do controlador foram projetados para serem positivos. Sob essa

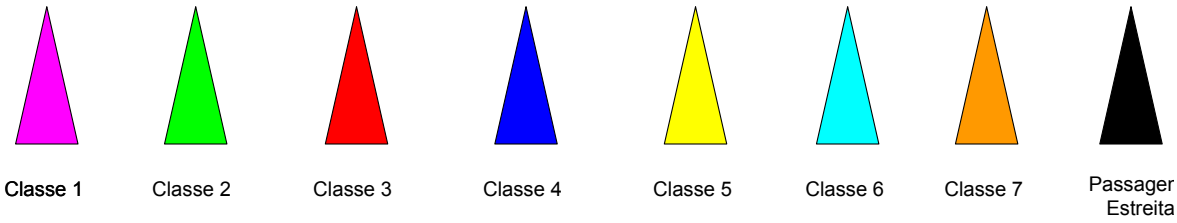
premissa, a análise dos polos garante estabilidade assintótica para qualquer valor de x_r. Nota-se que o aumento do valor absoluto de x_r, ou dos ganhos relacionados a ω_2, v_2 e v_3, influencia a parte complexa dos polos, introduzindo oscilações nas respostas temporais obtidas.

Não foi feita uma análise matemática da região de interseção ($0.5k1 < \alpha < 2k1$). Apesar de não haver uma prova, espera-se uma dinâmica estável obedecendo às restrições de ganho anteriores, já que é uma região transitória entre outras duas regiões, onde o controle tende a estabilizar o robô sobre a reta central.

6 Resultados

Este Capítulo apresenta as análises e os experimentos conduzidos para validar o sistema e as metodologias propostos, que foram expostos nos capítulos anteriores. Os registros das trajetórias seguidas pelo robô serão a principal ferramenta de análise. Para um maior detalhamento, além da posição e orientação, será exibido o estado na qual a CRI se encontrava em cada momento através de diferentes cores, ou seja, qual classe estava a cabeça do usuário ou se estava passando por uma passagem estreita. Estes estados ao longo das missões serão representados por setas de diferentes cores, como exposto na Figura 6.1, em conformidade com o esquema de classes da Figura 4.1.

Figura 6.1: Setas para Exposiçao dos Estados Durante Trajetórias



Fonte: Elaborada pelo autor

6.1 Estudo Preliminar

Inicialmente foi realizada uma análise preliminar em Matlab®. Utilizando as técnicas propostas no Capítulo 4 e 5, foi desenvolvida uma plataforma de simulação do sistema, incluindo uma CRI empregando a Equação (2.9). A partir do princípio de interseção entre retas, criou-se um sensor *rangefinder* virtual com 72 medidas em um campo de visão de aproximadamente 60 graus, para simular em diversos mapas compostos por retas.

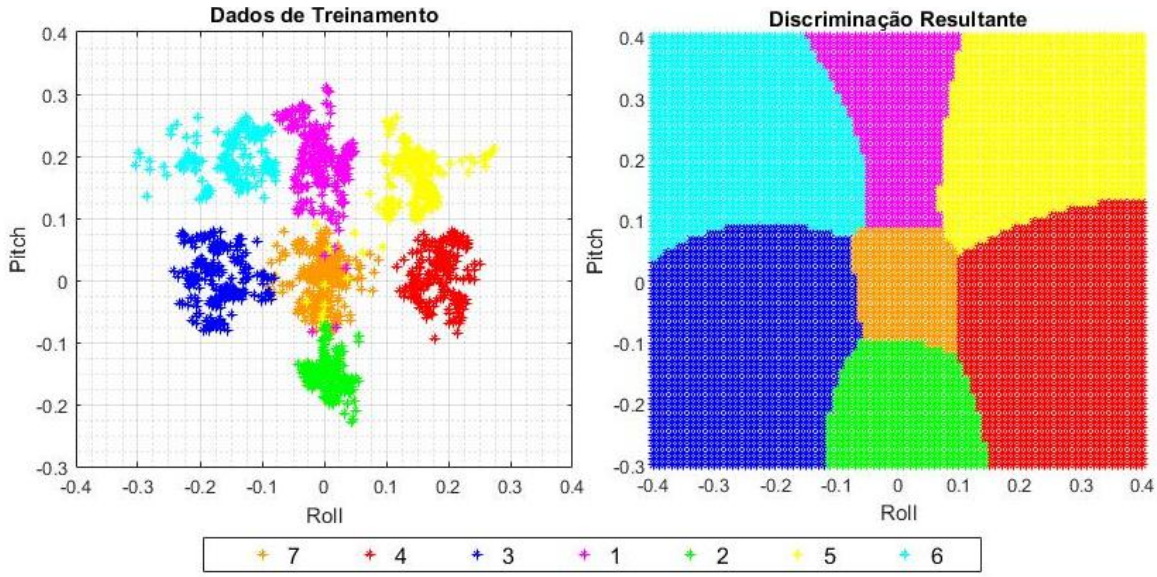
6.1.1 Análise da Navegação por Controle Compartilhado

Com o objetivo de avaliar o sistema de navegação assistida proposta, foram realizados testes em Matlab® utilizando a biblioteca de ROS intrínseca ao software para ler

o tópico *Rosserial*, referente a publicação dos Ângulos de Euler da cabeça.

Primeiramente calibrou-se a interface *Head-Keypad* para o usuário. Utilizando o procedimento descrito na subseção 4.1.2 obteve-se o cenário observado na Figura 6.2, contendo os dados de treinamento e a discriminação de regiões resultante do classificador de Mahalanobis. É importante notar que propositalmente foram realizados movimentos bem suaves, com pouca amplitude, já que o conforto do usuário é um dos focos deste trabalho.

Figura 6.2: Discriminação de Classes



Fonte: Elaborada pelo autor

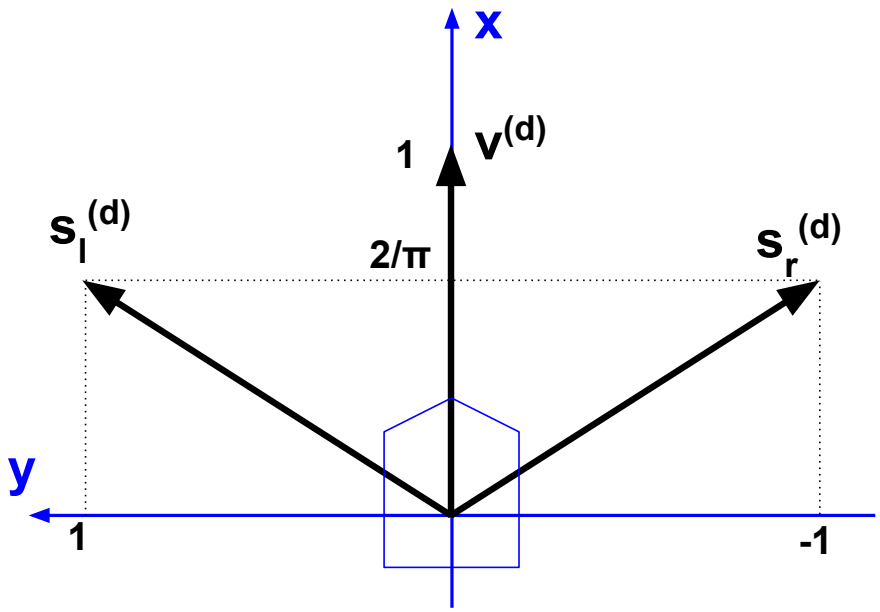
Fixado o raio da zona de proteção $R = 1.75m$, todos os obstáculos com medidas superiores serão desprezados. O próximo passo foi analisar e escolher os parâmetros a_1 e a_2, relacionados ao ganho β do campo repulsivo $\vec{C}$, Equação (4.2). Esta etapa foi resolvida empiricamente, analisando o comportamento dos campos sobre várias situações de obstáculos. Deseja-se que $C_x < -1$ quando próximo de obstáculos frontais e $|C_y| > 1$ quando muito próximos as laterais, sabendo que nessas situações as velocidades devem saturar em valores mínimos, como explicado no Capítulo 4.

Com base no comportamento de $\vec{\boldsymbol{C}}$ foram escolhidos os vetores padrão $\vec{v}^{(d)}$, $\vec{s}_r^{(d)}$ e $\vec{\boldsymbol{s}}_l^{(d)}$. Adotou-se $\vec{v}^{(d)} = [1 \;\; 0]^T$, pois, é desejado que $\vec{v}$ seja anulado quando $C_x < -1$. O ganho β é calculado conforme Equação 6.1, valorizando obstáculos laterais pois o campo de visão, (Figura 3.6), cobre pouco estas áreas.

$$\beta_i = 0.012\lambda_i^2 + 0.005 \tag{6.1}$$

Conforme Figura 6.3, os vetores de giro padrão foram adotados como: $\vec{s}_r^{(d)} = [2/\pi,\ -1]^T$ e $\vec{s}_l^{(d)} = [2/\pi,\ 1]^T$. A justificativa se encontra na Figura 6.4. Nesse cenário os ângulos de giro assumem valores unitários (± 1) quando não há campo repulsivo. Quando a componente $\vec{C}_y$ está no sentido oposto do movimento desejado, o valor absoluto do ângulo de giro decairá de forma aproximadamente linear, invertendo o sinal quando $|C_y| > 1$. No caso em que o campo esteja no mesmo sentido do movimento, a velocidade angular irá saturar em valor absoluto máximo 50% maior que o nominal.

Figura 6.3: Vetores Padrão



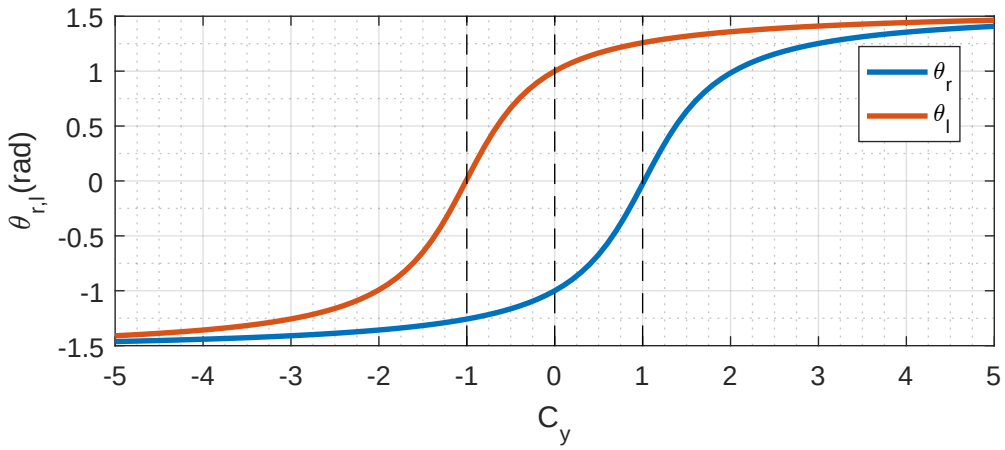
Fonte: Elaborada pelo autor

Em conformidade com a Tabela 4.4, deve-se decidir as velocidades mínimas v_{min}, w_{min} e as constantes $\alpha = [\alpha_1, ..., \alpha_8]$. Estes valores, com a classe de comando, o modo de operação e os variáveis v_x e $\theta_{r,l}$, irão definir o comportamento das velocidades da CRI.

$$\alpha = \begin{bmatrix} 0.5 & 0.8 & 0.4 & 0.6 & 0.45 & 0.3 & 0.7 & 0.4 \end{bmatrix} \tag{6.2}$$

Para avaliar, em conjunto, o reconhecimento de padrões e o controle compartilhado de velocidades, foi realizado um teste de percurso no modo *half speed* com curvas fechadas e estreitas. Foi utilizado HMI real para controlar o robô simulado, recebendo as

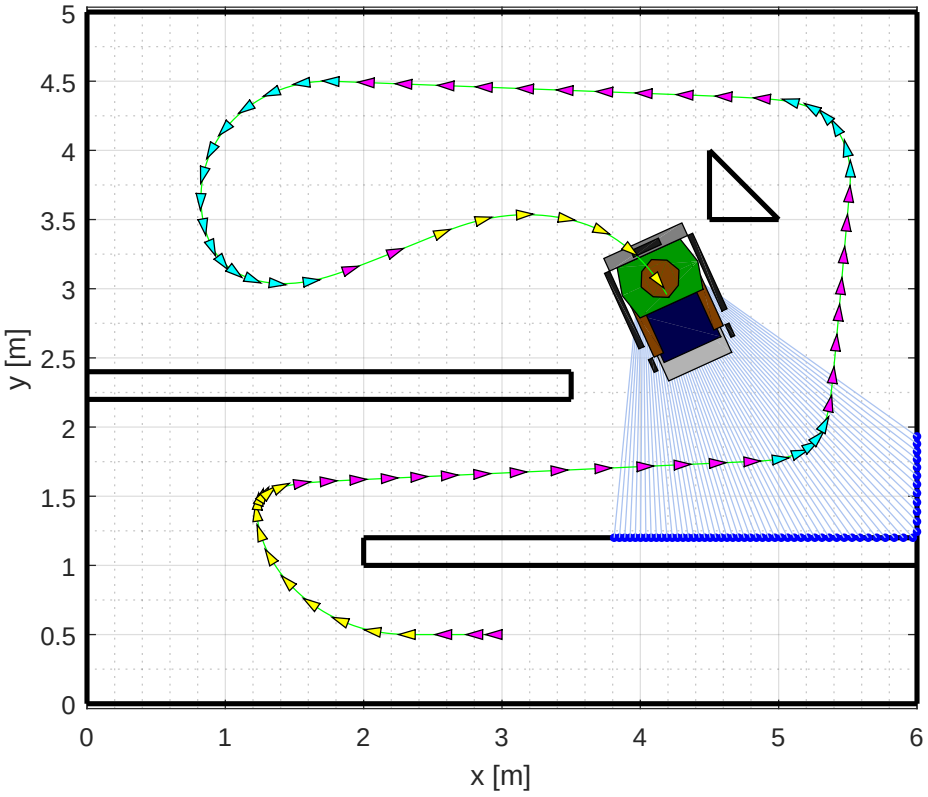
Figura 6.4: Interferência do Campo nos Ângulos de giro



Fonte: Elaborada pelo autor

informações de $roll_h$ *e* $pitch_h$ do protótipo. O Resultado está ilustrado na Figura 6.5.

Figura 6.5: Simulação Controle Compartilhado

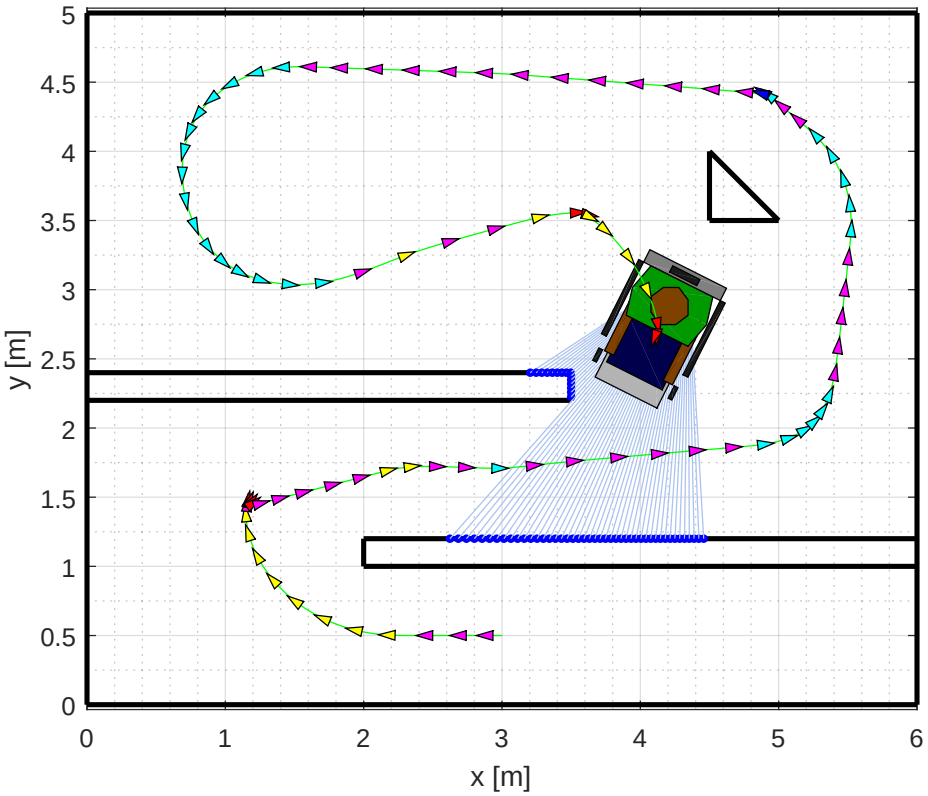


Fonte: Elaborada pelo autor .

O trajeto pode ser feito com facilidade, com apenas 9 movimentações de pescoço. Foram utilizadas apenas as classe 1,5 e 6, ou seja, o trajeto foi fluente e em nenhum momento houve a necessidade de parar a cadeira. O controle compartilhado facilitou a navegação reduzindo a velocidade linear quando há obstáculos próximos e controlando a

velocidade angular de forma inteligente. Para efeito de comparação, realizou-se o mesmo trajeto sem atuação dos campos vetoriais, onde cada classe opera com um padrão de velocidade fixo (Figura 6.6).

Figura 6.6: Simulação Sem Controle Compartilhado

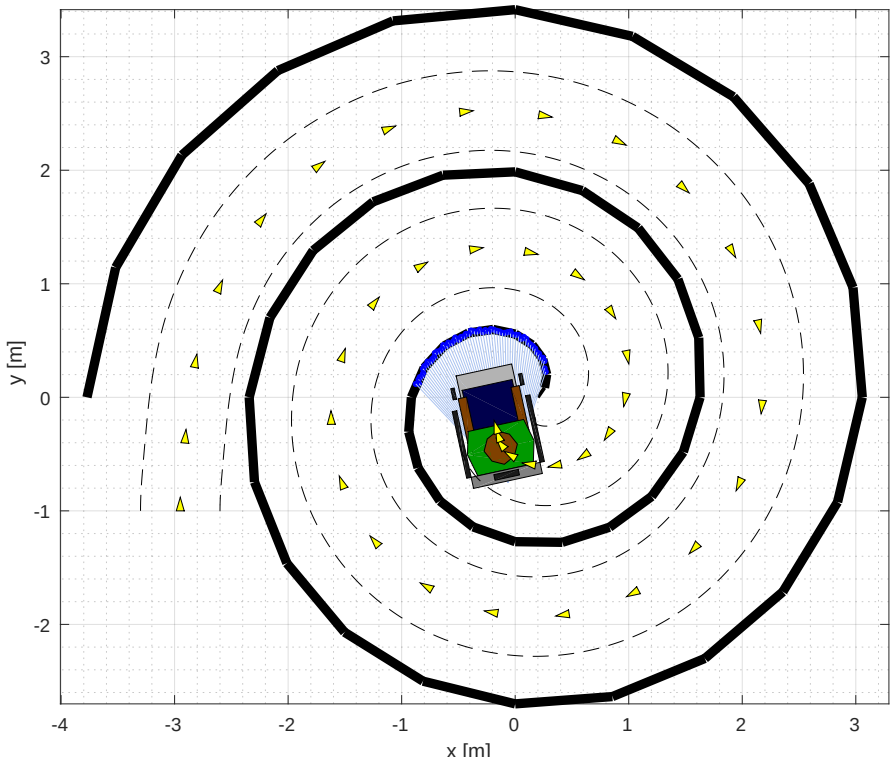


Fonte: Elaborada pelo autor

Apesar de ter concluído o trajeto com sucesso, sem o controle compartilhado não houve segurança e foram necessárias 24 movimentações do pescoço. Outra questão foi a necessidade de utilizar as classes 3 e 4, momentos em que foi realizado apenas o movimento de angular seu próprio eixo. Ficou evidente o desconforto devido às variações bruscas de inclinação da cabeça e as acelerações que atuariam sobre o corpo do usuário.

A metodologia mostrou-se promissora em relação a conforto e segurança, não foram notados erros de classificação durante os testes preliminares. Um último teste desta subseção foi realizado em uma situação de seguidor de parede (*wall-fallowing*), Figura 6.7, onde foi mantida a cabeça em todo o percurso na classe 5. A CRI ajustou as velocidades de forma a realizar todo o percurso em espiral sem colisões ou troca de classes, mantendo-se em paralelo com a parede.

Figura 6.7: Simulação Wall-Fallowing



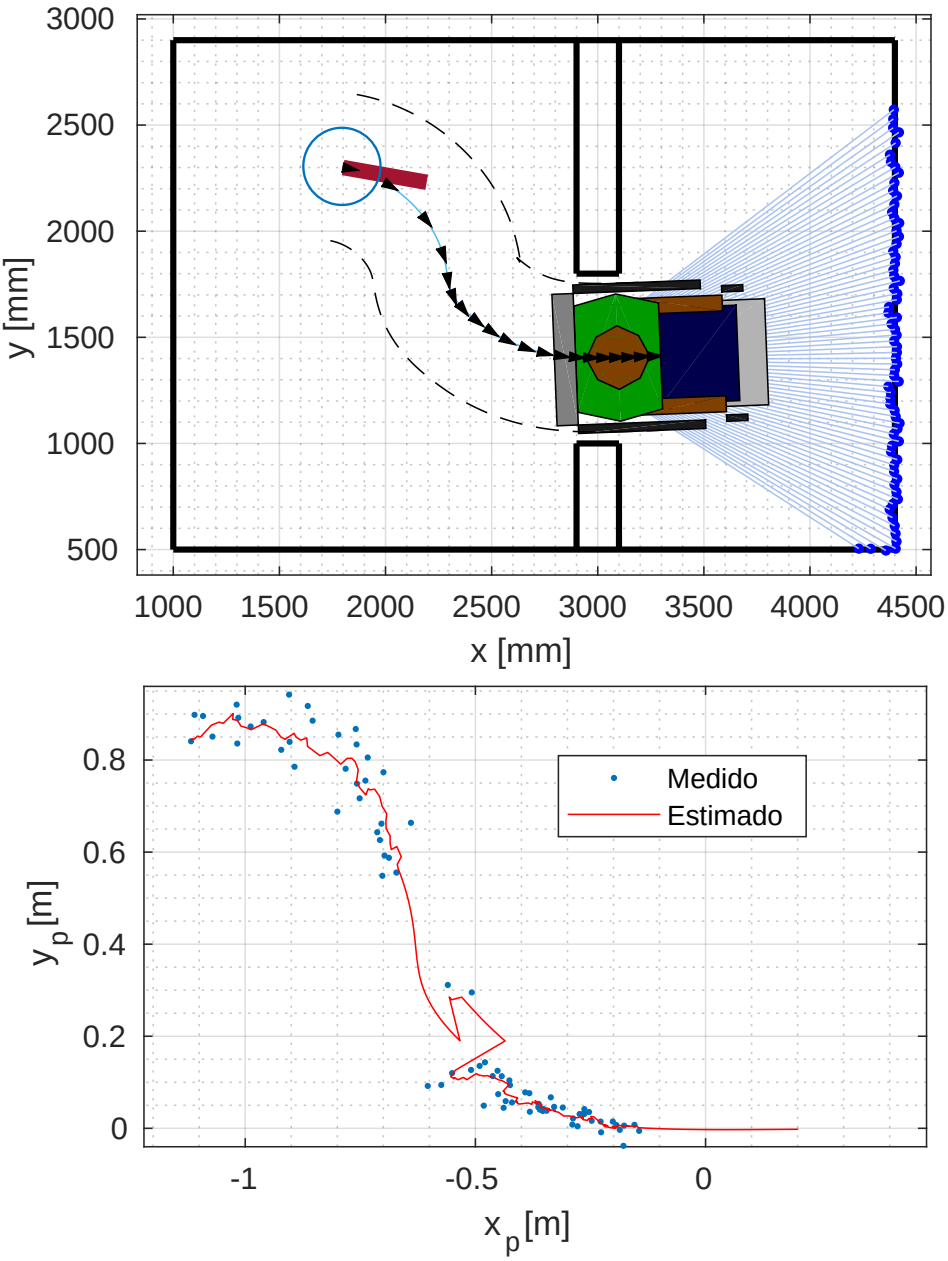
Fonte: Elaborada pelo autor

6.1.2 Análise do Algoritmo de Passagens Estreitas

Objetivo desta análise é, através de simulações em Matlab®, estudar e avaliar o método proposto de travessia de passagens estreitas em ambientes desconhecidos. Foram calibrados empiricamente todos os parâmetros envolvidos, como os do Filtro de Kalman, reconhecimento de passagem e ganhos do controlador. Foram realizadas duas simulações, em situações com alto grau de dificuldade para a travessia: a passagem é apenas 10cm maior que a largura do robô e composta por duas paredes grossas, além das missões partirem de posições distantes da reta central.

Os resultados estão apresentados nas Figuras 6.8 e 6.9, exibindo as simulações de trajetória em suas partes superiores. Na parte inferior é apresentada a estimação de estados do EFK com as posições recebidas pela observação. Devido ao ruído das medidas de distância *rangefinder*, sua discretização angular e a espessura da parede, o algoritmo de localização em relação à passagem sofre grandes variações, que tendem a se reduzir no final da missão ao aproximar-se da passagem. A aplicação do Filtro de Kalman reduziu expressivamente este ruído.

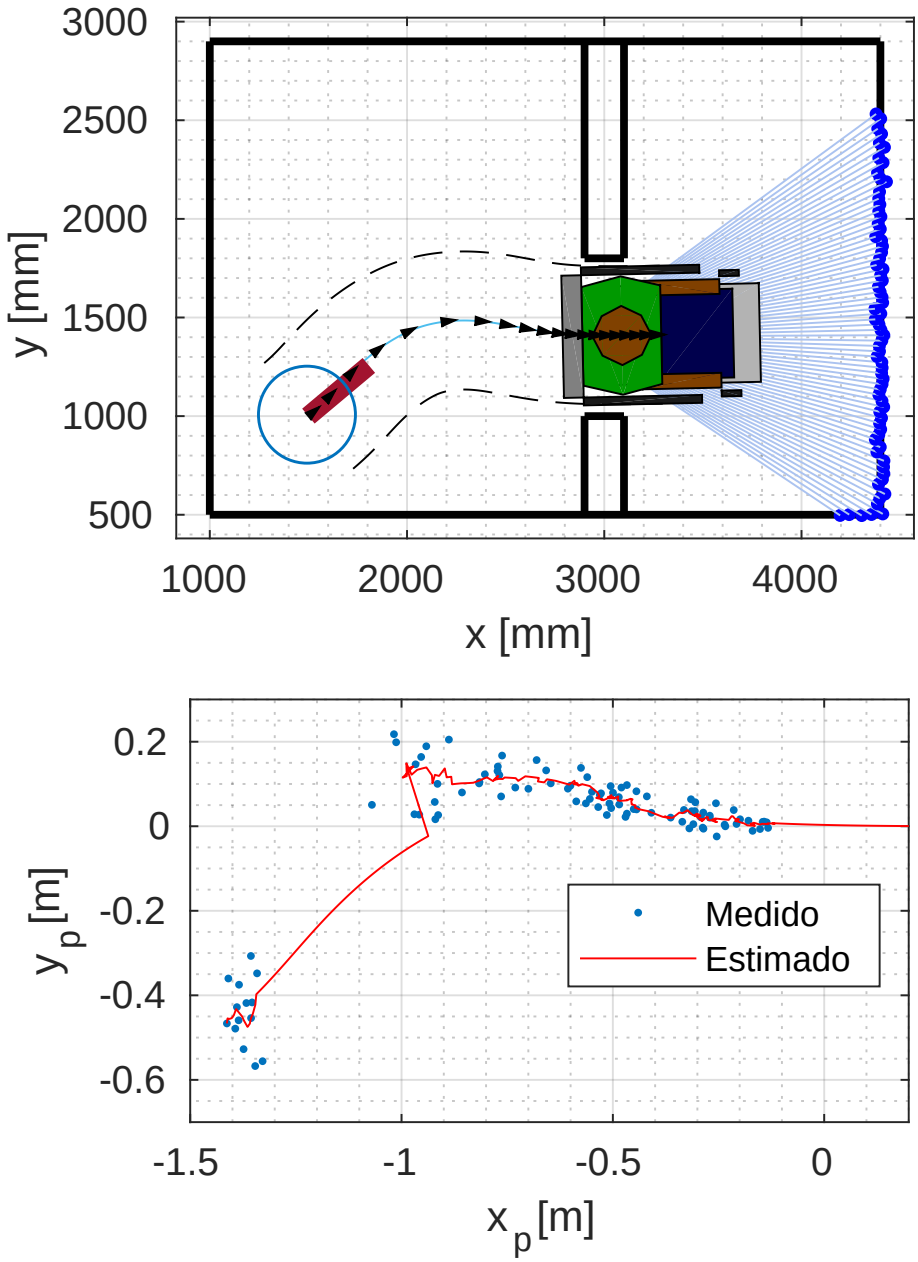
Figura 6.8: Simulação Narrow Passages



Fonte: Elaborada pelo autor

Durante o trajeto a cadeira primeiramente tende a se aproximar do ponto de segurança, escolhido a -1 m. Em sequência, ao entrar na região próxima a da reta central ela tende a seguí-la. O sistema deixa de enxergar a passagem em alguns momentos, mas continua a missão somente com os sensores odométricos (predição do EKF). Ao retornar a observá-la, o erro integrativo é corrigido. As missões foram completadas com um tempo em torno de 15 segundos. Aumentando os ganhos relacionados com as velocidades lineares, podem-se reduzir estes tempos à custa de uma maior oscilação e risco de choque com as laterais da passagem.

Figura 6.9: Simulação Narrow Passages 2



Fonte: Elaborada pelo autor

6.2 Avaliação Experimental

Nesta seção será avaliado o sistema como um todo, onde as propostas serão testadas utilizando os *hardwares* e *softwares* propostos no Capítulo 3.

6.2.1 Testes na Plataforma Gazebo

Gazebo é um software de simulação de robôs que permite simular, de forma precisa e eficiente, a dinâmica de robôs em ambientes internos e externos, com processos

físicos fiéis a realidade. Esta plataforma apresenta um *pluggin* para o ROS, o que permite interagir com o programa a partir de tópicos e mensagens.

Neste trabalho criou-se um mundo virtual, contendo uma casa com obstáculos e portas para testar as metodologias propostas. Usou-se uma CRI simulada (Figura 6.10), desenvolvida em Abhishek Patil (2017). Este modelo conta com diversos atuadores e sensores simulados, que interagem com a plataforma ROS. O modelo conta com duas rodas tracionadas à frente, e duas *caster* atrás. A Figura 6.11 apresenta este modelo inserido no ambiente criado para os testes.

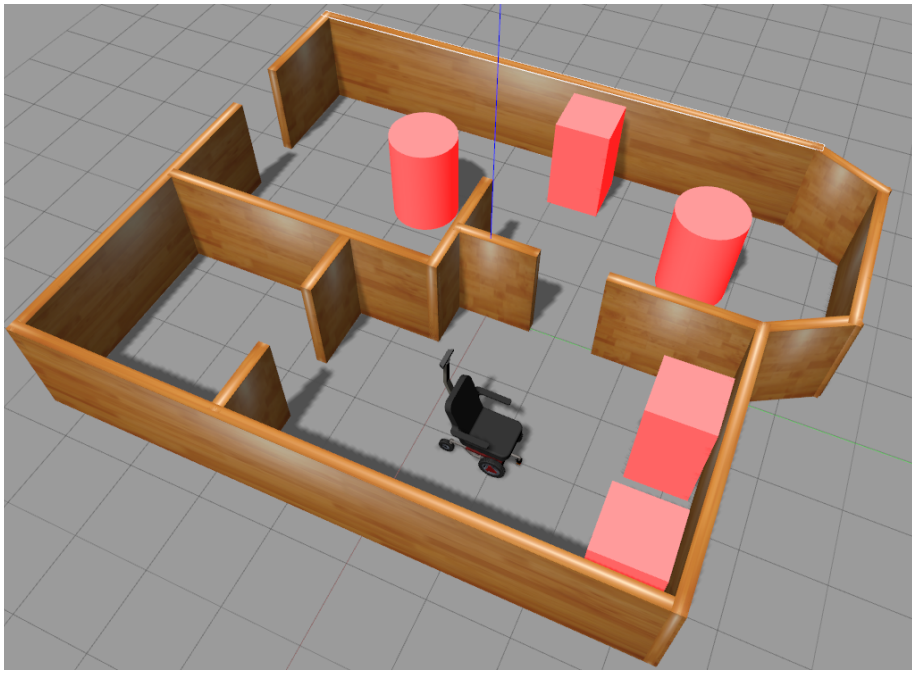
Figura 6.10: CRI simulada no Gazebo

Fonte: Elaborada pelo autor

A CRI utilizada possui um conjunto de atuadores e sensores com *pluggins* para a plataforma ROS, criando uma estrutura de tópicos para ser acessada pelo usuário, em destaque:

- **Odometria:** Um tópico simula a odometria do robô, fornecendo atual posição e orientação, além das velocidades linear e angular.
- **Controle de Velocidade:** Possui um tópico que recebe mensagens de alto nível de velocidade e simula um controlador interno de velocidade das rodas para– atender a requisição.

Figura 6.11: Ambiente de Simulação



Fonte: Elaborada pelo autor

- **Laser Scan:** Apesar de não estar sendo utilizado no trabalho, a CRI conta com um modelo de *laser scan* Hokuio à frente com 360 leituras entre -90 e 90 graus.
- **Sensor Kinect:** A CRI possui um modelo de câmera RGB-D em cima da cabeça do usuário, e publica exatamente os mesmo tópicos de um sensor real sobre o pacote *Oppeni2*, incluindo as imagens de profundidade, RGB e nuvem de pontos.

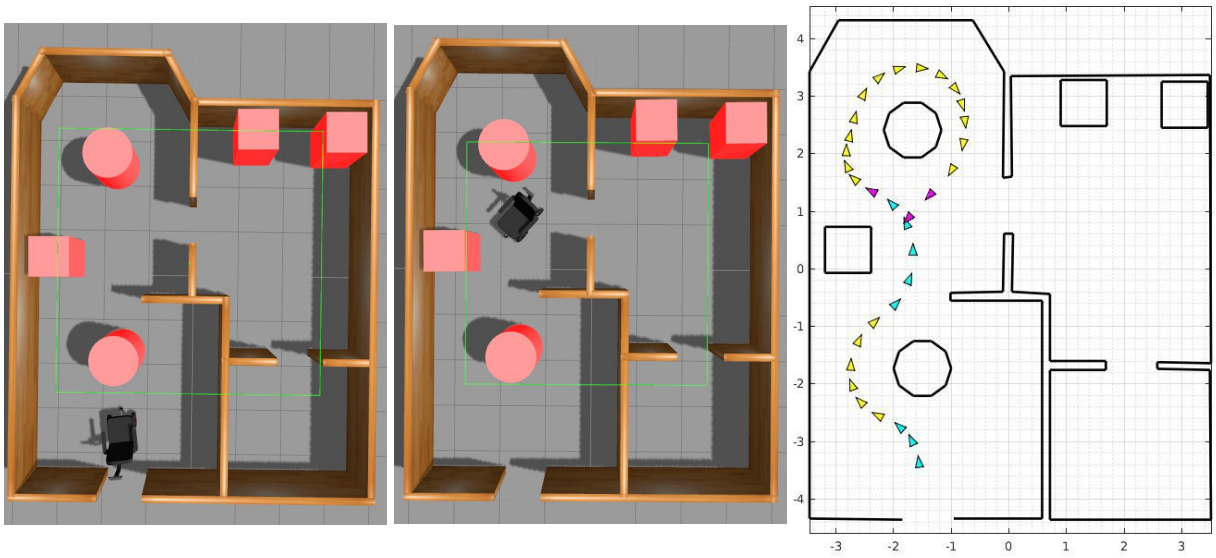
O objetivo é testar a arquitetura do sistema real embarcada na *Raspberry Pi 3* atuando sobre um modelo simulado de CRI. Na *Raspberry*, foram conectados ao Arduino e Headset, nela se executou quase todos os programas incluindo: nó mestre (roscore), reconhecimento de voz (pocketsphinx), áudio (soundplay), leitura serial do Arduino (ros-serial) e nós em *Python* específicos da dissertação (Control Center e Shared Control). Já em um notebook Intel i5 - 6GB de memória RAM e Nvidia GT 740m, cujo ROS foi configurado para reportar ao IP mestre da *Raspberry*, é executado o simulador Gazebo e o pacote depthimage_to_laserscan.

Teste em Gazebo 1

O foco do primeiro teste é avaliar o desempenho da navegação assistida em um ambiente realístico. O teste começou em um ponto inicial no estado "*parked*" e foi realizada

a sequência de comandos de voz: "*manual mode*"e "*half speed*". Em sequência realizou-se um trajeto contornando obstáculos e, para finalizar a missão, foi dado o comando "*stop*". A Figura 6.12 apresenta o cenário inicial e final, e o trajeto envolvido. O vídeo desta missão está disponível em: ⟨https://youtu.be/vygcShtkTXg⟩.

Figura 6.12: Primeiro Teste em Gazebo



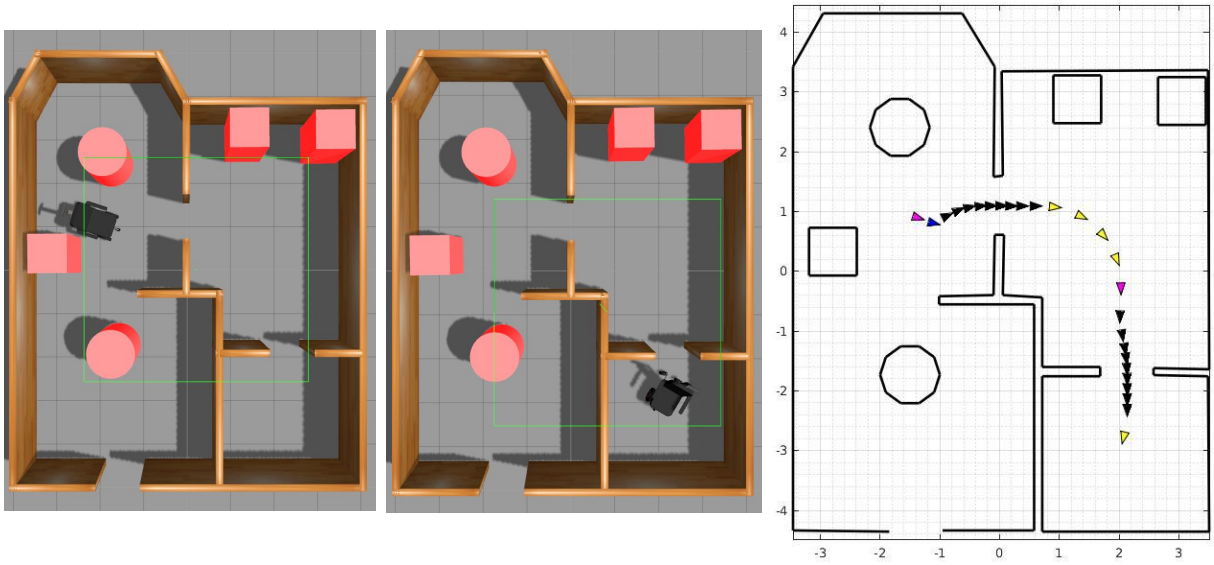
Fonte: Elaborada pelo autor

Teste em Gazebo 2

O enfoque do segundo teste foi o algoritmo de passagens estreitas. Primeiramente foi testado seu funcionamento partindo de diversas posições iniciais em relação à passagem, o vídeo deste experimento está no link: ⟨https://youtu.be/XVu1e8XM038⟩.

Depois realizou-se uma missão partindo de um ponto inicial no estado "*parked*" dando a sequência de comandos de voz: "*manual mode*"e "*half speed*". Em sequência, por dois momentos consecutivos, aproximou-se de uma porta e utilizou-se o comando "*narrow passage*". A Figura 6.13 apresenta o cenário inicial e final, e a trajetória desenvolvida. Vale a pena salientar que no momento da travessia o controle é 100% autônomo, podendo ser interrompido apenas por comando de voz. O vídeo de registro está no link: ⟨https://youtu.be/JF2Qi0CgP6U⟩.

Figura 6.13: Segundo Teste em Gazebo

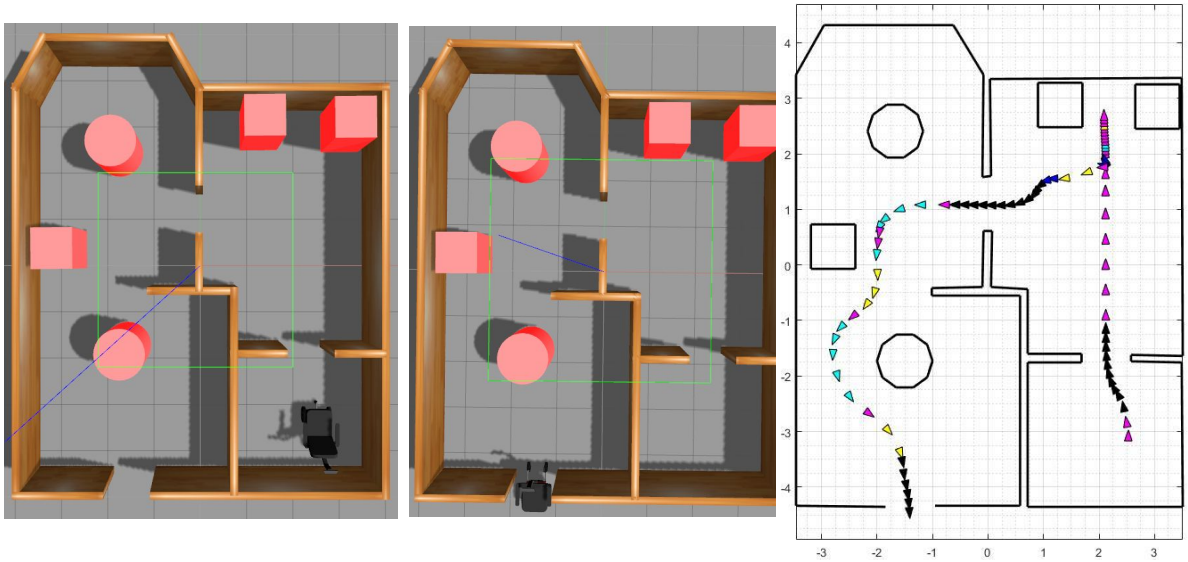


Fonte: Elaborada pelo autor

Teste em Gazebo 3

No último teste em Gazebo foi realizada uma navegação completa, passando por vários estados e situações debatidas neste trabalho. Simulando uma situação em que o usuário navega por todos os cômodos da residência, foi realizado: (i) desvios de obstáculos, (ii) travessias de passagens, (iii) manobra de estacionar e (iv) mudanças de velocidade. Os pontos inicial e final, além do trajeto, podem ser observados na Figura 6.14. No link ⟨https://youtu.be/31mFXlEV4gA⟩ está o vídeo da missão.

Figura 6.14: Terceiro Teste em Gazebo



Fonte: Elaborada pelo autor

6.2.2 Testes com robô real - Pioneer P3DX

Devido a não disponibilidade de uma CRI real, foram realizados testes com o robô Pionner-P3DX, com o propósito de verificar o comportamento do sistema em ambiente real, simbolizado por uma versão em miniatura. Como o usuário não pode estar sobre o robô, a operação foi feita à distância, sendo necessário processar dois pacotes fora da *Raspberry*, em outro computador: o de reconhecimento de voz e o de leitura da orientação da cabeça através da IMU.

A Figura 6.15 apresenta o modelo utilizado, e a Figura 6.16, a sala montada para testes de robótica móvel que será utilizada nas missões. Além de uma estrutura fixa feita em madeira, há diversos objetos para construção de obstáculos dinâmicos.

Figura 6.15: Robô Utilizado

Fonte: Elaborada pelo autor

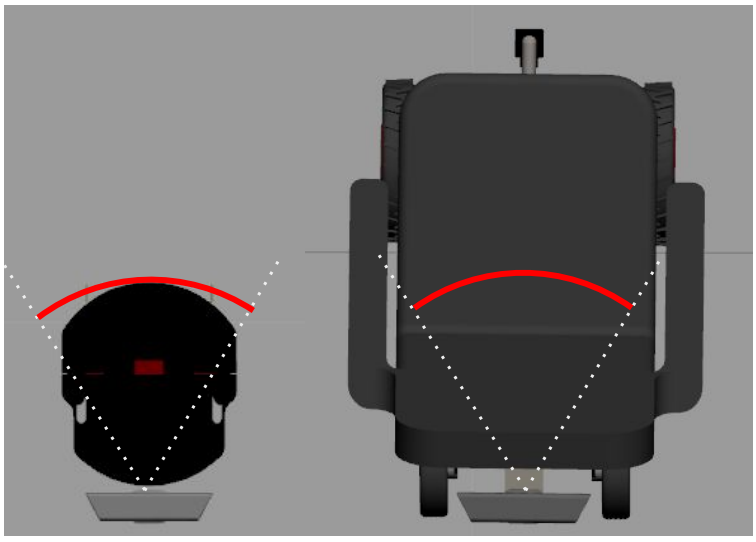
Como apresentado na Figura 6.17, em decorrência do tamanho do robô P3DX e do alcance mínimo de $0.45m$ da câmera de profundidade, é obtido um campo de visão de sensoriamento pouco efetivo comparado com uma CRI. As técnicas propostas serão menos eficientes, já que haverá pouca influência de campos vetoriais laterais e o robô deixará de enxergar a passagem previamente durante uma travessia. Mesmo com o empecilho, os

Figura 6.16: Sala de Testes

Fonte: Elaborada pelo autor

processos apresentaram a robustez necessária para serem comprovados com êxito neste cenário. Todos os parâmetros do sistema tiveram que ser atualizados para estas novas dimensões.

Figura 6.17: Limite Inferior do Campo de Visão



Fonte: Elaborada pelo autor

Teste da Navegação Assistida

O primeiro teste desta seção é relativo à análise em ambiente real da atuação da interface proposta em conjunto com os campos vetoriais , conforme Capítulo 4. Foi elaborado um pequeno cenário composto por corredores estreitos e curvas fechadas. O objetivo foi provar que através da proposta do trabalho é possível uma navegação sem colisões em ambientes apertados e com obstáculos, sem demanda excessiva de esforço

físico e mental por parte do usuário.

Figura 6.18: Teste da Navegação Assistida

Fonte: Elaborada pelo autor

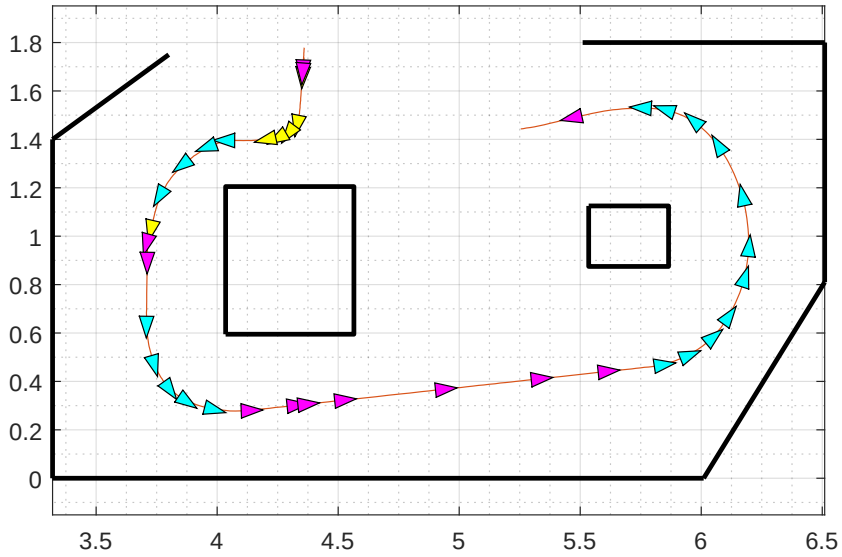
A Figura 6.18 exibe o cenário inicial do teste, que pode ser assistido no link: ⟨https://youtu.be/fydTb9-qug8⟩. A Figura 6.19 contém a estimativa de trajeto decorrente do registro odométrico, nota-se que o trajeto foi realizado com sucesso sem a necessidade de muitas transições entre classes de comando, ou seja, sem número excessivo de movimentações da cabeça. O robô seguiu a direção requerida mantendo-se distanciado das paredes.

Teste do Método de Passagens Estreitas

Para a avaliação de desempenho do algoritmo apresentado no Capítulo 5, foram realizados dois testes de travessia partindo de posições iniciais distintas. A Figura 6.20 apresenta as posições iniciais, as Figuras 6.21 e 6.22 exibem o *log* das missões, expondo o comportamento da estimação do Filtro de Kalman em comparação com as medições de posição em relação a passagem, e os registros odométricos. O vídeo está disponível em: ⟨https://youtu.be/Y51gtIKyb5U⟩.

Em ambos os testes o robô partiu de pontos distantes da reta central. Durante a manobra de aproximação ao ponto de segurança, a passagem deixa de ser observável. Ao se aproximar da reta e se orientar, o sistema volta a enxergar o acesso, realizando agora

Figura 6.19: *Log* do Teste de Navegação Assistida



Fonte: Elaborada pelo autor

Figura 6.20: Teste de Passagems Estreitas

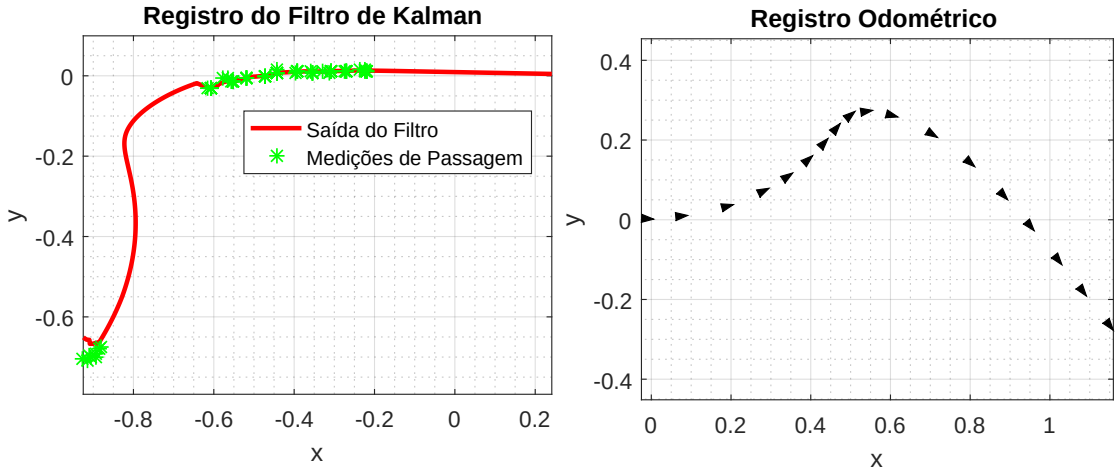
Fonte: Elaborada pelo autor

um ajuste fino de localização. Devido à limitação estrutural exposta na Figura 6.17, a partir de $x \cong -0.2$ o robô completa a missão apenas com os sensores odométricos, o que não acarretou nenhuma falha.

Teste Final

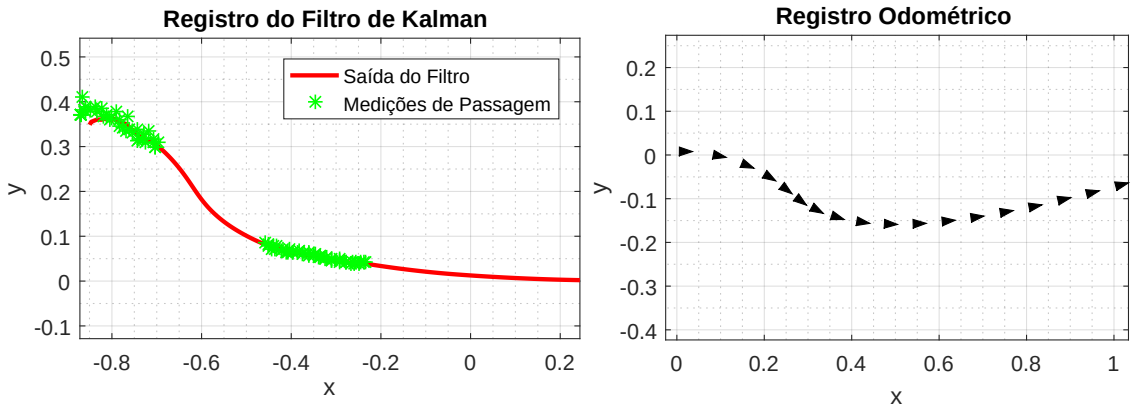
O último teste consistiu em uma navegação completa, abrangendo uma gama grande de tópicos deste trabalho. O vídeo da missão foi filmado em três perspectivas: do

Figura 6.21: *Log* do Teste de Passagens Estreitas 1



Fonte: Elaborada pelo autor

Figura 6.22: *Log* do Teste de Passagens Estreitas 2



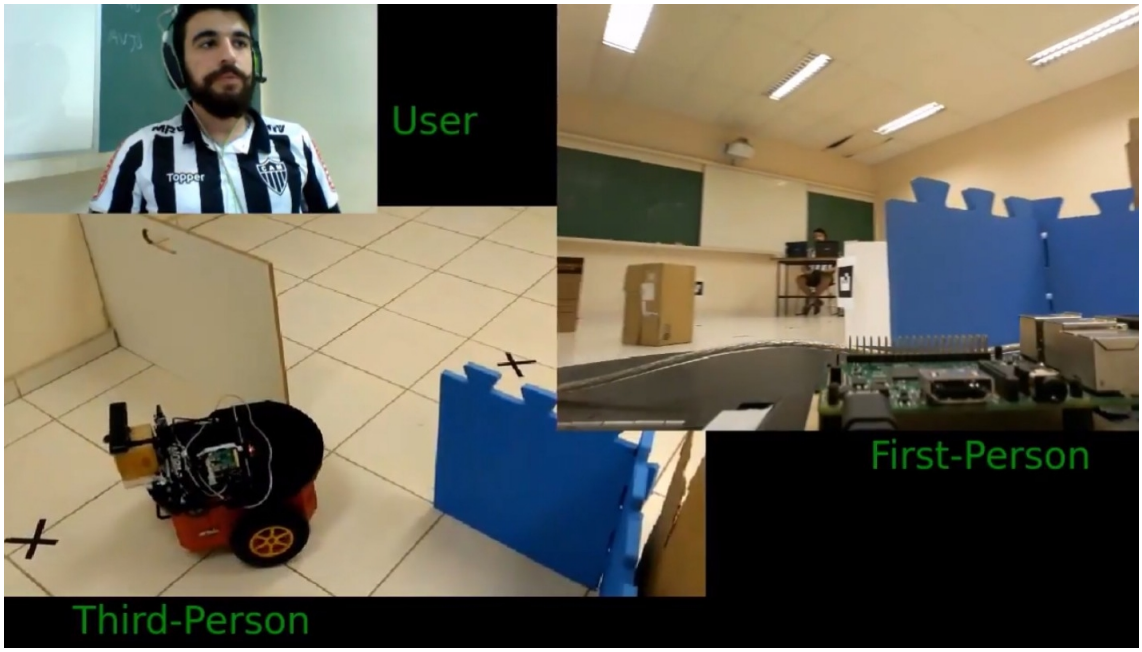
Fonte: Elaborada pelo autor

usuário, da visão em primeira e em terceira pessoa, conforme exibido na Figura 6.23. As filmagens podem ser assistidas no *link*: ⟨https://youtu.be/Vj3PGxQYczE⟩.

Além do vídeo, a missão foi registrada pela odometria do robô, Figura 6.24, apresentando uma trajetória muito fidedigna com a realidade. Nota-se novamente que, mesmo com as limitações, o robô teve êxito na travessia da passagem estreita, além do controle compartilhado ter atuado com eficácia, permitindo uma navegação sem colisões e com leves movimentações de cabeça.

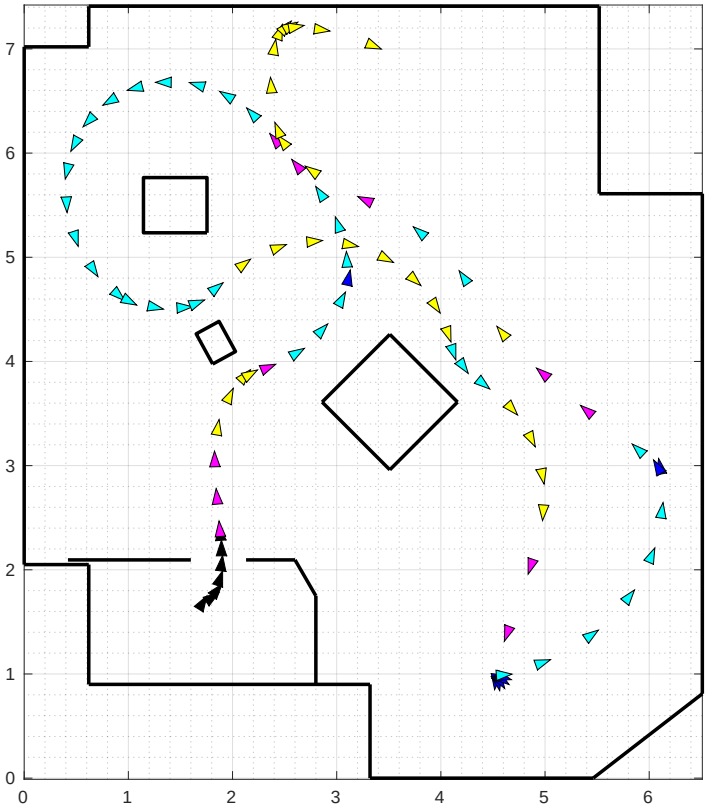
.

Figura 6.23: Teste Final



Fonte: Elaborada pelo autor

Figura 6.24: Trajetória Aproximada pelo Registro Odométrico



Fonte: Elaborada pelo autor

7 Conclusões e Trabalhos Futuros

Esta dissertação apresentou soluções de baixo custo para aplicação em Cadeiras de Rodas Inteligentes. O sistema e o usuário se comunicam por frases sonoras, de forma a configurar os modos de operação, que estão modelados em uma estrutura de eventos discretos, baseados em uma Rede de Petri. O reconhecimento de voz apresentou boa acurácia e em poucas ocasiões os comandos não promoveram as transições desejadas.

Foi desenvolvida uma interface para tetraplégicos, denominada *Head-Keypad*, baseada em leves movimentos de pescoço. É utilizada uma IMU para classificar a atual inclinação da cabeça em 7 diferentes classes de atuação. Através de um protocolo de treinamento, o classificador por distância de Mahalanobis indicou boa precisão para o contexto, sendo necessárias baixas amplitudes e tornando-se uma solução confortável e confiável.

Por motivo de custo, optou-se em utilizar uma câmera de profundidade ao invés de um *laser scanner*. Mesmo com o campo de visão reduzido, foi possível a comprovação das metodologias propostas neste trabalho. Foi desenvolvido um sistema de controle compartilhado, onde o usuário decide a direção do movimento e a inteligência computacional, baseada nos campos vetoriais repulsivos, decide a magnitude das velocidades da CRI. Mesmos com as limitações, os resultados foram promissores, possibilitando uma navegação segura e confortável.

Mesmo com o controle compartilhado, o processo de travessia por passagens estreitas se mostrou necessário, evitando inúmeros movimentos de pescoço. Foi proposta uma metodologia de identificação, localização, planejamento e controle para a tarefa. Após a sintonia do controlador, foi possível realizar a travessia em todos os cenários testados sem colisões, em carácter real e simulado.

Em trabalhos futuros, é almejada a implementação do sistema em uma CRI real. Outro ponto desejado é a otimização de *software* visando melhor desempenho, apesar da *Raspberry* ter triunfado em sua função, notou-se um pequeno atraso comparado com um *Desktop*. Ações como a troca da Linguagem *Python* pela *C++* e a edição dos pacotes

ROS utilizados podem provocar um aumento da performance. Em relação ao algoritmo de passagem estreita, é pretendido obter um sistema de reconhecimento que abranja mais tipos de geometria de passagens estreitas, como por exemplo, corredores longos. É aspirado buscar a prova de estabilidade global por uma função candidata de Lyapunov, mesmo com a alta complexidade do conjunto planta e controlador.

Antes da tentativa de obter um produto final, é pretendido um estudo do sistema com usuários tetraplégicos em parceria com pesquisadores da área da saúde. Pode-se analisar o grau de mobilidade e o desempenho destes pacientes para os mesmos testes até então realizados.

Bibliografia

Abhishek Patil. *nuric wheelchair model*. 2017. Disponível em: ⟨https://github.com/patilnabhi/nuric_wheelchair_model_02⟩. Acesso em: 04/05/2017.

AMIRYAN, J.; JAMZAD, M. Adaptive motion planning with artificial potential fields using a prior path. In: IEEE. *Robotics and Mechatronics (ICROM), 2015 3rd RSI International Conference on*. [S.l.], 2015. p. 731–736.

BOUCHER, P. et al. Design and validation of an intelligent wheelchair towards a clinically-functional outcome. *Journal of neuroengineering and rehabilitation*, BioMed Central, v. 10, n. 1, p. 58, 2013.

BUREAU, M. et al. Non-invasive, wireless and universal interface for the control of peripheral devices by means of head movements. In: IEEE. *Rehabilitation Robotics, 2007. ICORR 2007. IEEE 10th International Conference on*. [S.l.], 2007. p. 124–131.

BURHANPURKAR, M. et al. Cheap or robust? the practical realization of self-driving wheelchair technology. In: IEEE. *Rehabilitation Robotics (ICORR), 2017 International Conference on*. [S.l.], 2017. p. 1079–1086.

CAMPANELI, H.; MESTRIA, M. Otimização de trajetórias através de caminhos mínimos para a locomoção de cadeira de rodas robótica. *Simpósio Brasileiro de Automção Inteligente*, 2013.

CARDOSO, J.; VALETTE, R. *Redes de petri*. [S.l.]: Editora da UFSC, 1997.

CHAUHAN, R. et al. Study of implementation of voice controlled wheelchair. In: IEEE. *Advanced Computing and Communication Systems (ICACCS), 2016 3rd International Conference on*. [S.l.], 2016. v. 1, p. 1–4.

COELHO, F. et al. Filtro de kalman estendido baseado em visão computacional e odometria aplicado à localização de robôs móveis. In: . [S.l.]: XIII Simpósio Brasileiro de Automação Inteligente (SBAI), 2017.

CORKE, P. *Robotics, vision and control: fundamental algorithms in MATLAB*. [S.l.]: Springer, 2011. v. 73.

COSTA, E. B. Metodologia de otimização em dois níveispara a geração de sinal sub-Ótimo deexcitação e estimação de parâmetros desistemas não lineares restritos. Universidade Federal de Juiz de Fora, 2017.

DITUNNO, J. et al. The international standards booklet for neurological and functional classification of spinal cord injury. *Spinal Cord*, Nature Publishing Group, v. 32, n. 2, p. 70–80, 1994.

ESCOLANO, C.; ANTELIS, J. M.; MINGUEZ, J. A telepresence mobile robot controlled with a noninvasive brain–computer interface. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, IEEE, v. 42, n. 3, p. 793–804, 2012.

ESPINOSA, F. et al. Odometry and laser scanner fusion based on a discrete extended kalman filter for robotic platooning guidance. *Sensors*, Molecular Diversity Preservation International, v. 11, n. 9, p. 8339–8357, 2011.

FAHIMI, F. *Autonomous robots: modeling, path planning, and control*. [S.l.]: Springer Science & Business Media, 2008. v. 107.

IBGE, B. Censo demográfico, 2010. *Acesso em*, v. 13, 2015.

KALMAN, R. E. et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, v. 82, n. 1, p. 35–45, 1960.

KIM, J. et al. The tongue enables computer and wheelchair control for people with spinal cord injury. *Science translational medicine*, American Association for the Advancement of Science, v. 5, n. 213, p. 213ra166–213ra166, 2013.

KIM, P. *Kalman filter for beginners: with MATLAB examples*. [S.l.]: CreateSpace, 2011.

LEISHMAN, F.; HORN, O.; BOURHIS, G. Smart wheelchair control through a deictic approach. *Robotics and Autonomous Systems*, Elsevier, v. 58, n. 10, p. 1149–1158, 2010.

LOPES, A. C.; PIRES, G.; NUNES, U. Assisted navigation for a brain-actuated intelligent wheelchair. *Robotics and Autonomous Systems*, Elsevier, v. 61, n. 3, p. 245–258, 2013.

MACIEL, G. M. et al. Development of a control system for electric wheelchairs based on head movements. In: IEEE. *Proceedings of SAI Intelligent Systems Conference*. [S.l.], 2017.

MACIEL, G. M. et al. Planejamento e controle não linear para passagens estreitas em robótica móvel assistiva. XIII Simpósio Brasileiro de Automação Inteligente (SBAI), 2017.

MOHANTY, P. K.; PARHI, D. R. Controlling the motion of an autonomous mobile robot using various techniques: a review. *Journal of Advance Mechanical Engineering*, v. 1, n. 1, p. 24–39, 2013.

NAEEM, A.; QADAR, A.; SAFDAR, W. Voice controlled intelligent wheelchair using raspberry pi. *International Journal of Technology and Research*, Technology and Research Publications, v. 2, n. 2, p. 65, 2014.

NEGENBORN, R. Robot localization and kalman filters. *Utrecht Univ., Utrecht, Netherlands, Master's thesis INF/SCR-0309*, 2003.

NETO, W. A. et al. Construção e sintetização de modelos de estado para o controle de processos. Universidade Federal de Juiz de Fora (UFJF), 2014.

OLIVI, L. et al. Shared control for assistive mobile robots based on vector fields. In: IEEE. *Ubiquitous Robots and Ambient Intelligence (URAI), 2013 10th International Conference on*. [S.l.], 2013. p. 96–101.

OLIVI, L. R. et al. Navegação de robôs móveis assistivos por controle compartilhado baseado em campos vetoriais. [sn], 2014.

PACNIK, G.; BENKIC, K.; BRECKO, B. Voice operated intelligent wheelchair-voic. In: IEEE. *Industrial Electronics, 2005. ISIE 2005. Proceedings of the IEEE International Symposium on*. [S.l.], 2005. v. 3, p. 1221–1226.

PEREIRA, G. Comparison of human machine interfaces to control a robotized wheelchair. *XIII Simpósio Brasileiro de Automação Inteligente Porto Alegre, RS*, p. 2301 – 2306, 2017.

QUIGLEY, M.; GERKEY, B.; SMART, W. D. *Programming Robots with ROS: a practical introduction to the Robot Operating System*. [S.l.]: "O'Reilly Media, Inc.", 2015.

REISIG, W. *Understanding petri nets: modeling techniques, analysis methods, case studies*. [S.l.]: Springer, 2013.

ROFER, T.; MANDEL, C.; LAUE, T. Controlling an automated wheelchair via joystick/head-joystick supported by smart driving assistance. In: IEEE. *Rehabilitation Robotics, 2009. ICORR 2009. IEEE International Conference on*. [S.l.], 2009. p. 743–748.

ROHMER, E. et al. Laser based driving assistance for smart robotic wheelchairs. In: IEEE. *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*. [S.l.], 2015. p. 1–4.

RUIZ, M. F. R. et al. Desenvolvimento de um sistema de localização híbrido para navegação autônoma de veículos terrestres em ambiente simulado. sn, 2014.

RUZAIJ, M. F.; POONGUZHALI, S. Design and implementation of low cost intelligent wheelchair. In: IEEE. *Recent Trends In Information Technology (ICRTIT), 2012 International Conference on*. [S.l.], 2012. p. 468–471.

SEKI, H. et al. Practical obstacle avoidance using potential field for a nonholonmic mobile robot with rectangular body. In: IEEE. *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*. [S.l.], 2008. p. 326–332.

SHI, K.; DENNY, J.; AMATO, N. M. Spark prm: Using rrts within prms to efficiently explore narrow passages. In: IEEE. *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. [S.l.], 2014. p. 4659–4666.

SIMPSON, R. C. Smart wheelchairs: A literature review. *Journal of rehabilitation research and development*, Superintendent of Documents, v. 42, n. 4, p. 423, 2005.

TAHER, F. B. et al. A collaborative and voice configured electric powered wheelchair control system based on eeg and head movement. In: *3rd International Conference on Automation, Control Engineering and Computer Science*. [S.l.: s.n.], 2016.

THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic robotics*. [S.l.: s.n.], 2005.

TOMARI, M. R. M.; KOBAYASHI, Y.; KUNO, Y. Development of smart wheelchair system for a user with severe motor impairment. *Procedia Engineering*, Elsevier, v. 41, p. 538–546, 2012.

TORKIA, C. et al. Power wheelchair driving challenges in the community: a users' perspective. *Disability and Rehabilitation: Assistive Technology*, Taylor & Francis, v. 10, n. 3, p. 211–215, 2015.

WANG, J.; CHEN, W.; LIAO, W. An improved localization and navigation method for intelligent wheelchair in narrow and crowded environments. *IFAC Proceedings Volumes*, Elsevier, v. 46, n. 13, p. 389–394, 2013.

Wiki ROS. *ROS Documentation*. 2017. Disponível em: ⟨http://wiki.ros.org/⟩. Acesso em: 04/12/2017.

Wikimedia Commons. *Wikimedia Commons, the free media repository*. 2017. Disponível em: ⟨https://commons.wikimedia.org/wiki/Main_Page/⟩. Acesso em: 04/12/2017.