Pontifícia Universidade Católica
do Rio de Janeiro

Nilton Cesar Anchayhua Arestegui

Localização e mapeamento de robôs móveis utilizando inteligência e visão computacional

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Mecânica Aplicada do Departamento de Mecânica da PUC–Rio

Orientador: Prof. Marco Antônio Meggiolaro

Rio de Janeiro
Setembro de 2009

Nilton Cesar Anchayhua Arestegui

Localização e Mapeamento de Robôs Movéis Utilizando Inteligência e Visão Computacional

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Engenharia Mecânica da PUC - Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Marco Antonio Meggiolaro
Orientador
Departamento de Engenharia Mecânica PUC - Rio

Prof. Mauro Speranza Neto
Departamento de Engenharia Mecânica PUC - Rio

Prof.ª Karla Tereza Figueiredo Leite
Departamento de Engenharia Elétrica - PUC - Rio

Prof. José Eugenio Leal
Coordenador Setorial do Centro Técnico Científico - PUC - Rio

Rio de Janeiro, 14 de Setembro 2009.

Nilton Cesar Anchayhua Arestegui

Graduou-se em engenharia na Universidade Nacional de Engenharia (UNI)– (Lima, Peru), cursando Engenharia Mecatrônica. Especializou-se na Universidade Nacional de Engenharia na Área de Eletrônica (Lima, Peru) em Processamento Digital de Sinais (DSP), e participou no centro de desenvolvimento de inteligência computacional.

Ficha Catalográfica

Anchayhua Arestegui, Nilton Cesar

Localização e mapeamento de robôs móveis utilizando inteligência e visão computacional / Nilton Cesar Anchayhua Arestegui; orientador: Marco Antônio Meggiolaro. — Rio de Janeiro : PUC–Rio, Departamento de Mecânica, 2009.

v., 111 f: il. ; 29,7 cm

1. Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Mecânica.

Inclui referências bibliográficas.

1. Mecânica – Tese. 2. inteligência computacional. 3. Robô Móvel. 4. Scale Invariant Feature Transform (SIFT). 5. Algoritmos Genéticos. 6. Redes Neurais. 7. Lógica difusa. I. Meggiolaro, Marco Antônio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Mecânica. III. Título.

CDD: 621

Agradecimentos

Ao meu orientador Professor Marco Antônio Meggiolaro pelo apoio e a quem admiro muito e é exemplo para mim, obrigado pela simpatia de sempre, e incentivo para a realização deste trabalho.

Ao CAPES e PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho no poderia ter sido realizado.

A meus pais Beltran, Lucila, a meus irmãs Tania, Susan que sempre estiveram comigo e meu família.

Aos meus colegas da PUC-Rio, David, Rocem, johana, Gerardo, Cristian, Cesitar, Markini, jorge, juancarlos el loco, quem me fizeram adorar esse lugar.

Aos professores Marco Aurelio, Marley e Omar que me ofereceram a oportunidade desta cooperação.

Ao pessoal do departamento de Mecânica pela ajuda de todos os dias.

Resumo

Anchayhua Arestegui, Nilton Cesar; Meggiolaro, Marco Antonio. **Localização e Mapeamento de Robôs Móveis Utilizando Inteligência e Visão Computacional.** Rio de Janeiro, 2009. 111p. Dissertação de Mestrado - Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Esta dissertação introduz um estudo sobre os algoritmos de inteligência computacional para o controle autônomo dos robôs móveis, Nesta pesquisa, são desenvolvidos e implementados sistemas inteligentes de controle de um robô móvel construído no Laboratório de Robótica da PUC-Rio, baseado numa modificação do robô ER1. Os experimentos realizados consistem em duas etapas: a primeira etapa de simulação usando o software *Player-Stage* de simulação do robô em 2−D onde foram desenvolvidos os algoritmos de navegação usando as técnicas de inteligência computacional; e a segunda etapa a implementação dos algoritmos no robô real. As técnicas implementadas para a navegação do robô móvel estão baseadas em algoritmos de inteligência computacional como são redes neurais, lógica difusa e *support vector machine* (SVM) e para dar suporte visual ao robô móvel foi implementado uma técnica de visão computacional chamado *Scale Invariant Future Transform* (SIFT), estes algoritmos em conjunto fazem um sistema embebido para dotar de controle autônomo ao robô móvel. As simulações destes algoritmos conseguiram o objetivo, mas na implementação surgiram diferenças muito claras respeito à simulação pelo tempo que demora em processar o microprocessador.

Palavras–chave

Inteligência Computacional; Robô Móvel; Scale Invariant Feature Transform (SIFT); Algoritmos Genéticos; Redes Neurais; Lógica Difusa.

Abstract

Anchayhua Arestegui, Nilton Cesar; Meggiolaro, Marco Antonio (Advisor). **Computational Intelligence Techniques for Visual Self-Localization and Mapping of Mobile Robots.** Rio de Janeiro, 2009. 111p. M.Sc. Dissertation – Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

This theses introduces a study on the computational intelligence algorithms for autonomous control of mobile robots, In this research, intelligent systems are developed and implemented for a robot in the Robotics Laboratory of PUC-Rio, based on a modification of the robot ER1. The verification consist of two stages: the first stage includes simulation using Player-Stage software for simulation of the robot in 2-D with the developed of artificial intelligence; an the second stage, including the implementation of the algorithms in the real robot. The techniques implemented for the navigation of the mobile robot are based on algorithms of computational intelligence as neural networks, fuzzy logic and support vector machine (SVM); and to give visual support to the mobile robot was implemented the visual algorithm called Scale Invariant Future Transform (SIFT), these algorithms in set makes an absorbed system to endow with independent control the mobile robot. The simulations of these algorithms had obtained the objective but in the implementation clear differences had appeared respect to the simulation, it just for the time that delays in processing the microprocessor.

Keywords

Computational Intelligence; Mobile Robot; Scale Invariant Feature Transform (SIFT); Genetic Algorithms; Neural Network; Fuzzy Logic.

Sumário

Lista de figuras

Lista de tabelas

"Nenhum homem realmente produtivo pensa como se estivesse escrevendo uma dissertação".

Albert Einstein., .

1
Introdução

1.1
Contexto e motivações

Consideráveis avances foram feitos em relação à programação de máquinas para processos industriais específicos, começando com o tear mecânico de Jacquard em 1801 (Mck95). Desenvolvimentos em tecnologia, incluindo a criação de computadores, controle de laço fechado dos atuadores, transmissão de potência através de engrenagens e instrumentação avançada, foram necessários para a utilização de robôs suficientemente flexíveis para o uso nos mais diversos processos de produção. Tais avances propiciaram o desenvolvimento de robôs moveis que se deslocam no ambiente onde estão inseridos (Joh04). Esses robôs podem navegar em locais conhecidos ou não (Wit02), recebendo informações sensoriais e reagindo de acordo com as mesmas (Fre92). O trabalho aqui desenvolvido trata, mais especificamente, desta categoria de robôs, a traves do desenvolvimento e análise de sistemas de controle de navegação que recebem informações sensoriais e fornecem um comportamento inteligente. Muitos fatores vêm contribuindo para o desenvolvimento da pesquisa na área de robótica móvel, bem como servindo de motivação para este trabalho (Sel92),(Ben99),(Joh89). Inicialmente, pode-se citar o processo atual de automatização que as indústrias estão sofrendo em nível mundial, e o alto custo envolvido em processos de automatização flexível. Neste tipo de automatização, robôs deslocam-se através da planta fabril, conduzindo os materiais entre as máquinas, realizando suas tarefas. a posição das maquinas pode ser modificada, bem como novos equipamentos podem ser adicionados sem grandes problemas, permitindo configurações e *layouts* variáveis. Outro fator importante diz respeito a problemas de segurança em indústrias nucleares, que necessitam de robôs para transporte e manuseio de materiais radioativos. Também, o risco de colocação de pessoas em ambientes hostis e insalubres conduz ao desenvolvimento de veículos submersos e da robótica espacial (Wit02). Trabalhos repetitivos, tediosos e cansativos, como os efetuados em laboratórios e na agricultura, são fatores de incentivo ao uso de robôs moveis (Sie04). A utilização

de robôs traz três vantagens sobre as operações manuais: aumento da produtividade, melhora da qualidade e diminuído da exposição humana a produtos químicos e intempéries. Atualmente, existe na literatura uma série de estudos sobre sistemas de controle para robôs móveis.

1.2 Robôs móveis - uma visão geral

Os sistemas robóticos podem ser classificados basicamente como robôs manipuladores e robôs móveis. Ambas as classes de sistemas robóticos vão passando por avanços tecnológicos que elevam cada vez mais seus graus de autonomia e confiabilidade, refletindo em uma crescente aceitação pelo mercado (industrial, militar, residencial, dentre outros) destes sistemas. Nas últimas décadas tanto o campo de aplicação quanto o número de maquinas existentes tem crescido bastante. Robôs estão sendo utilizados nas mais diversas áreas para auxiliar o trabalho humano, bem como em locais de risco e inóspitos. No meio industrial, os robôs móveis e de base fixa são encontrados com freqüência, propiciando grandes vantagens na produção, tais como: velocidade de operação, aumento da qualidade dos produtos, redução dos custos relativos a encargos sociais e execução ininterrupta das atividades do trabalho.

Basicamente um robô é composto por quatro partes principais conforme se pode ver na Figura 1.1. Essas são: *mecanismos, sistema de acionamento, sistema de percepção e sistema de controle.*

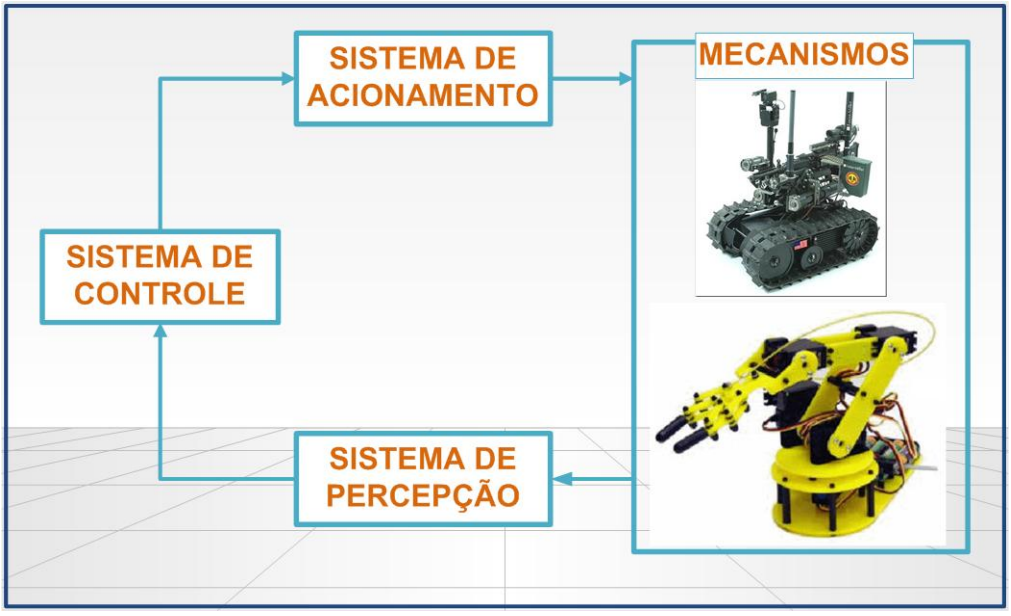

Figura 1.1: Componentes básicos de um robô: sistema de percepção, de controle, de acionamento e mecanismos de atuação.

O mecanismo, é um sistema mecânico que tem liberdade de movimentos. Esse pode ser dividido em quatro partes: dispositivos de deslocamento, braço, executor final. Os dispositivos de deslocamento permitem o movimento do robô através do ambiente. São formados por rodas, propulsões, esteiras, hélices entre outros. O braço determina o alcance do executor final no Espaço Euclidiano, enquanto o punho determina a sua Orientação. O executor é escolhido em função da tarefa a ser executada pelo robô, pode ser uma garra, pistola de solda, pistola de pintura entre outros dispositivos (Joh89).

O sistema de acionamento, transforma a energia elétrica, hidráulica e pneumática (ou uma combinação destas) em energia cinética, utilizada nos dispositivos de deslocamento, juntas do braço e do punho e no executor (Sel92).

O sistema de percepção, é responsável pela obtenção de informações sobre o estado do robô, sobre os objetos no seu espaço de trabalho e sobre as possíveis maquinas com as quais ele trabalhe. Os sensores podem ser de diversos tipos, por exemplo: sensores de infravermelho, sensores de ultra-som, câmeras filmadoras, sensores de voz entre outros (Sie04).

O sistema de controle, é o responsável pelo mapeamento, planejamento e controle dos movimentos do robô. Sua forma de ação pode consistir desde simples séries de paradas mecânicas ajustáveis até controles complexos, efetuados por processamento de dados, passando por avançados sistemas de mapeamento do ambiente. Em síntese, o sistema de controle é responsável pelo comportamento do robô. Comportamento este que pode ser do tipo fixo (repetitivo) ou inteligente, tomando decisões e reagindo conforme a situação encontrada, aprendendo com suas atitudes e incorporando-as ao seu conhecimento (Fre92).

A integração dos quatro sistemas resulta em máquinas com capacidade de realizar uma série de tarefas e movimentos. Quanto ao deslocamento no ambiente, podem-se classificar os robôs em duas grandes categorias: *fixos* e *móveis*. Inicialmente os robôs desenvolvidos eram fixos a uma determinada base e realizavam somente tarefas próximas a ela, ficando restrito o seu campo de atuação. Com o crescente interesse das diversas áreas pela utilização de robôs, tornou-se necessário o desenvolvimento de sistemas que se desloquem através do ambiente onde estão inseridos, interagindo e executando tarefas em função deste. Para viabilizar tais ações, várias questões precisam ser solucionadas. Assim, a pesquisa sobre robôs móveis, ao nível de *sistemas de controle, acionamento, percepção* e *mecanismo*, tem despertado bastante interesse. O trabalho desenvolvido aborda o estudo de sistemas de controle inteligente baseado na inteligência e a visão computacional para tais tipos de robôs.

1.3 Histórico

O desenvolvimento de pesquisas na área de robôs móveis, principalmente os autônomos, iniciou-se nos Estados Unidos com o uso de inteligência computacional (IA). A partir do início da década dos 80, houve um grande interesse sobre o assunto. Isto deveu-se às novas possibilidades oferecidas pelos microprocessadores, seja na área de instrumentação ou seja no tratamento dos dados. Os primeiros veículos desenvolvidos surgiram da necessidade da construção de robôs para serem utilizados na exploração espacial. Um dos primeiros robôs móveis foi o *Shakey*, desenvolvido em 1969 no *Stanford Research Institute*(ver figura 1.2).

Figura 1.2: Robô Shakey de Stanford

Em 1977, foi desenvolvido o veículo denominado *Stanford Cart* no *Stanford Artificial Intelligence Laboratory*, equipado com câmera de televisão, sendo substituído, em meados dos anos 80, por uma segunda versão. Nos seguintes anos instituições científicas continuaram com a tarefa de investigar robôs móveis, e assim nos anos 1980 e 1982 a marinha norte-americana criou o robô *Robart I*, que foi um dos primeiros modelos em obter sucesso, este robô tinha como função principal patrulhar ambientes fechados (*indoor*) e buscar situações indesejadas, tais como indícios de incêndio e vestígios de intrusão. Nos anos posteriores também surgiram novas versões de *Robart*, Fig 1.3.

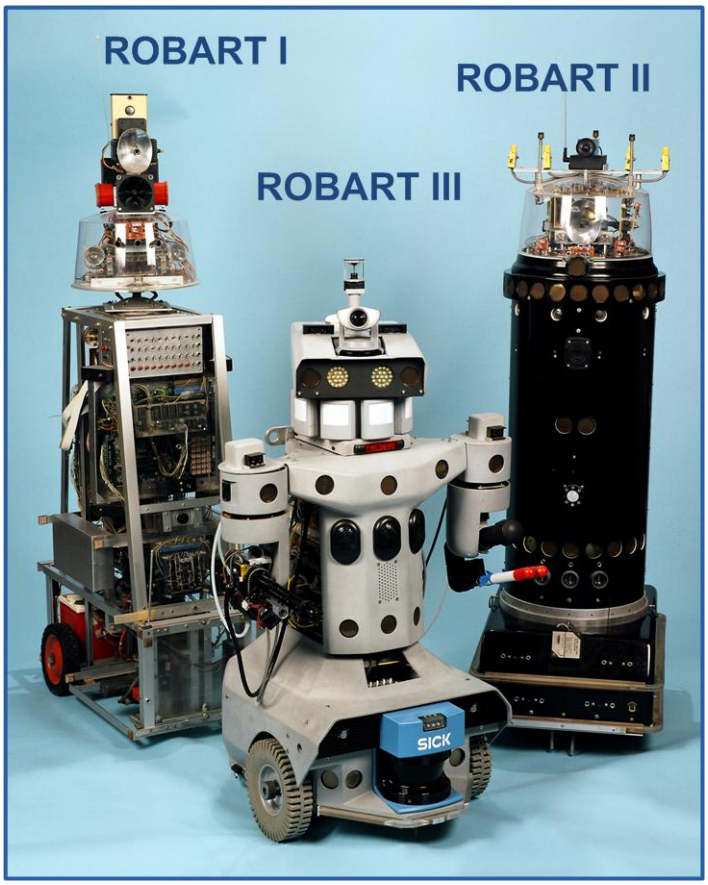

Figura 1.3: RobartI, RobartII e RobartIII projetados pela marinha norte-americana para patrulhamento em ambientes fechados

Aplicações como estas foram escolhidas para provar as utilidades destas funções, liberando pessoas destas tarefas e provando que não eram necessários dispositivos avançados, reduzindo a complexidade de todo sistema. O desenvolvimento continuou com a aparição de novos tipos de sensores, atuadores e programas de controle fazendo ao robô cada vez mais autônomo e preciso na realização das tarefas.

Existem hoje robôs extremamente avançados, a evolução nesta área é surpreendente. O caso mais famoso de projeto é o robô Sojourner, Fig 1.4. Construído pela *Jet Propulsion Laboratory(JPL)* do Instituto de Tecnologia da Califórnia, com a colaboração da *NASA*, foi enviado a Marte com o propósito de explorar as características do terreno desse planeta e realizar experimentos científicos (Sto96).

Existem muitos outros projetos desenvolvidos em diferentes centros de investigação a nível mundial, entre eles França, Espanha, Alemanha e EUA, todos eles construídos com a tecnologia atual usando elementos de percepção sensorial como câmeras digitais, sensores ultra-som, sensores infra-vermelhos, localização *Global Position System* (GPS), sensores a laser, etc.

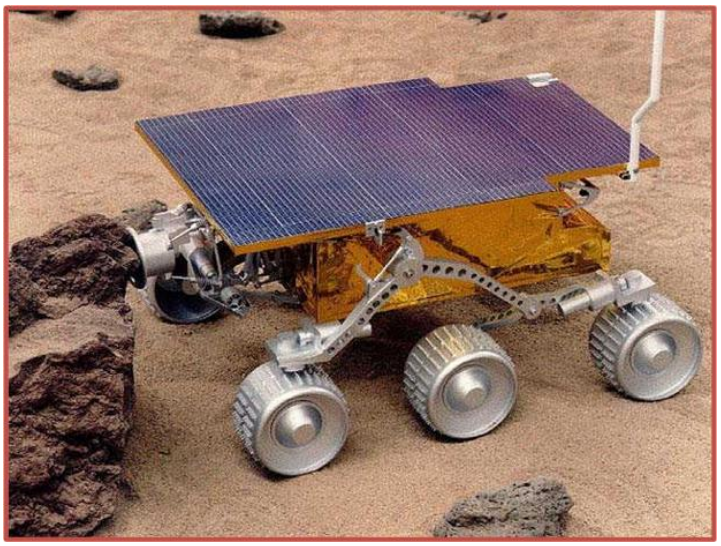

Figura 1.4: Robô Sojourner enviado para uma pesquisa a Marte

Adicionalmente todos estes robôs contam com capacidades inteligentes de interagir com o mundo real, vide Figura 1.5.

A PUC-Rio também tem um trabalho nesta área desenvolvida pelo Felipe Belo chamado "Desenvolvimento de Algoritmos de Exploração e Mapeamento para Robôs Móveis de Baixo Custo", neste trabalho ele fala sobre um algoritmo iterativo baseado em entropia para planejar uma estratégia de exploração visual, permitindo a construção eficaz de um modelo em grafo do ambiente. Utilizando a métrica de entropia o algoritmo determina nós potenciais para os quais deve se prosseguir a exploração através do procedimento de *Visual Tracking* em conjunto com a técnica SIFT (*Scale Invariant Feature Tansform*), o algoritmo auxilia a navegação do robô para cada nó novo.

A idéia dos nós potenciais também foram usados neste trabalho para o treinamento da rede neural e auxiliar o processo de guiar o robô para nós já conhecidos, mas de um jeito diferente e usando outros tipos de algoritmos como se detalha no estudo desta tese.

1.4 Objetivo

O objetivo desta dissertação é desenvolver um algoritmo de navegação inteligente para uma navegação autônoma e robusta do robô $ER1$ modificado num ambiente desconhecido estático. Isto envolve como primeira etapa a programação do algoritmo de controle inteligente no linguagem 'C' para simulação do sistema de navegação do robô no *software Player-Stage* na plataforma Linux; projeto mecânico do Robô $ER1$, projeto eletrônico do circuito de Potência e das interfaces dos sensores; posteriormente a pro-

Figura 1.5: a)Omni-França b)Minerva-Carnegie Mello c)Segbot-Stanford d)Familia Pioneer-Espanha e)Tourguide-Alemanha

gramação do algoritmo de controle no *Software Code Composer Studio* da *Texas Instrument* para sua implementação no processador digital de sinais $DigitalSignalProcessor(DSP)TMS320F2812$; programação do software de interface visual para o mapeamento da navegação do robô; e finalmente a etapa experimental.

1.5 Escopo

O robô $ER1$ modificado navegará num espaço fechado como ambientes de hospitais ou ambientes da Universidade, e sobre terrenos planos, fazendo a exploração do ambiente desconhecido e armazenando pontos característicos do ambiente para a aprendizagem do ambiente navegado.

1.6 Organização da Dissertação

A presente dissertação descreve a implementação do robô ER1 para que possa navegar num ambiente estático desconhecido mediante técnicas de inteligência computacional e percepção sensorial. Para este estudo, organizou-se o trabalho em seis capítulos, que são expostos da seguinte forma:

No **Capítulo 1**, se faz uma Introdução a esta dissertação, amostra-se uma perspectiva histórica da investigação com robôs móveis no mundo atual, e também apresenta-se o objetivo e algumas considerações e restrições.

No **Capítulo 2**, desenvolvesse os fundamentos teóricos desde o modelo do robô até as técnicas existentes para a navegação de robôs móveis.

No **Capítulo 3**, desenvolvesse o ambiente de trabalho, e se apresentam todos as ferramentas usadas neste trabalho, desde a implementação do hardware, o projeto da parte eletrônica e o desenvolvimento do software de controle.

No **Capítulo 4**, desenvolvesse o algoritmo de controle e as técnicas usadas para desenvolver um algoritmo inteligente para a navegação autônoma do robô, e se ilustram os diagramas de fluxo do algoritmo de navegação.

No **Capítulo 5**, se apresentam os resultados da exploração do robô, ilustrando primeiro a simulação e logo os gráficos do desempenho do treinamento da rede neural e do algoritmo de aprendizagem em cada nó.

No **Capítulo 6**, se detalham as conclusões do trabalho e contribuições para pesquisas posteriores.

Finalmente se apresentam os apêndices onde São mostrados os programas usados e desenvolvidos em linguagem *C*. O **apêndice A** contém a listagem dos programas em *C*. O **apêndice B** contém a biblioteca dos programas e as configurações dos periféricos do *DSP TMS320F2812*.

2
Fundamentação Teórica

2.1
Modelo cinemático do robô móvel

A cinemática é o estudo mais básico do comportamento do sistema mecânico. Nos robôs móveis precisamos compreender este comportamento para um apropriado projeto e compreensão de como criar o software de controle. O processo para compreender o movimento do robô começa com o processo de descrição na contribuição do movimento de cada roda do robô móvel.

2.1.1
Modelo cinemático e restrição

Deduzindo um modelo para um movimento completo de todo o processo, cada roda individualmente contribui ao movimento do robô, ao mesmo tempo, e estabelece restrições sobre o movimento do robô. As rodas são tratadas em conjunto com o chassi do robô, mas as forças e restrições de cada roda tem que ser expressadas em relação à referência do chassi; isto é particularmente importante na robótica móvel porque estas ações são precisamente a natureza do movimento (Sie04).

Representando a posição do robô

Através deste análise, modelamos o robô como um corpo rígido sobre rodas, operado em um plano horizontal. As dimensões do robô no plano são três, dois para a posição no plano e um para a orientação no eixo vertical que é ortogonal ao plano. Assume-se que existem graus de liberdade e flexibilidade adicionais devido ao eixo da roda, e as juntas que existem na roda. No entanto, neste análise considera-se o robô como corpo rígido ignorando as juntas e graus de liberdade internos do robô e de suas rodas.

Para especificar a posição do robô no plano, estabelecemos uma relação entre a referência global no plano e a referência local no robô, como se ilustra na Fig. 2.1. Os eixos X_I e Y_I definem uma base inercial arbitrária sobre o plano como uma referência global com origem $O : X_I, Y_I$. Para especificar a

posição do robô, escolhemos um ponto P sobre o chassi como sua posição de referência. A base X_R e Y_R define dois eixos relativos a P sobre o chassi do robô e deste modo o sistema de referência local do robô. A posição de P em relação ao sistema de referência global é especificada pelas coordenadas x e y, e a diferença angular entre as referencias global e local é dada por θ. Agora pode-se descrever a posição do robô com o vetor de três elementos, como se ilustra na equação 2-1.

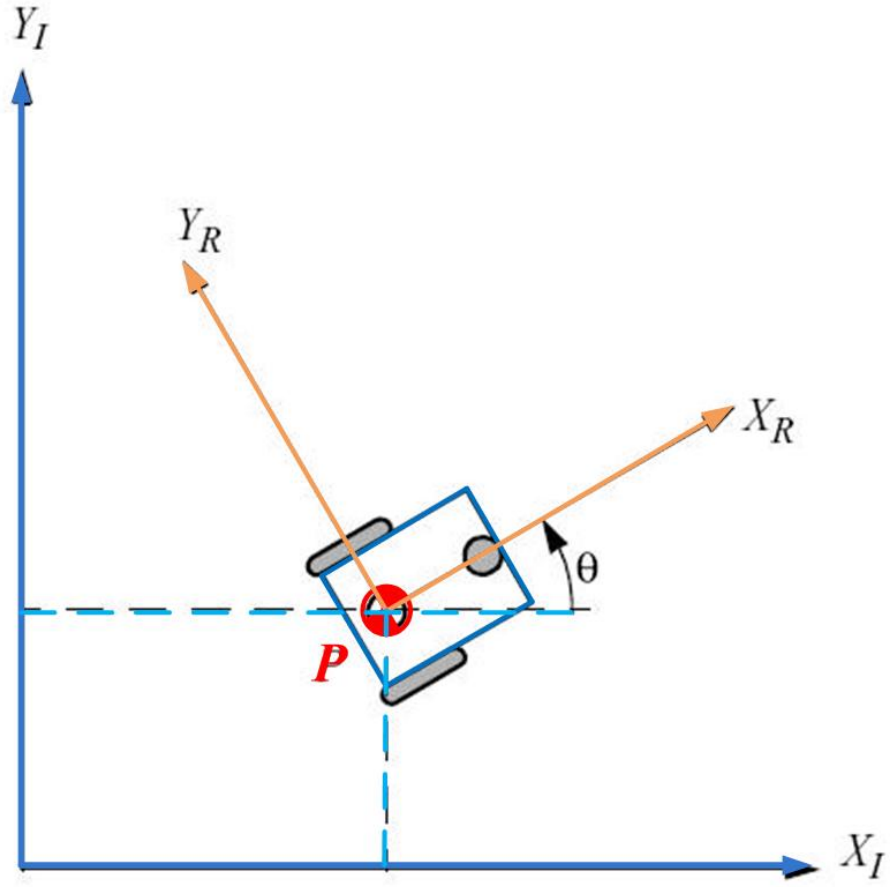

Figura 2.1: sistema de Referência global e sistema de referência local do robô

$$\xi_I = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \tag{2-1}$$

Para descrever o movimento do robô em termos de componentes de movimento, será necessário traçar o movimento ao longo do eixo de referência global para o movimento dos eixos locais sobre o robô. O traçado é acompanhado usando a matriz de rotação ortogonal.

$$R(\theta) = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2-2}$$

Esta matriz pode ser usada para traçar o movimento em relação ao sistema de referência global X_I, Y_I em termos do sistema de referência local X_R, Y_R. Esta operação é denotada por $R(\theta)\dot{\xi}_I$ porque o cálculo depende do valor de θ. Por exemplo, consideremos o robô da figura 2.2. Para este robô, $\theta = \frac{\pi}{2}$ podemos calcular a matriz de rotação instantânea do robô segundo as equações 2-3 e 2-4.

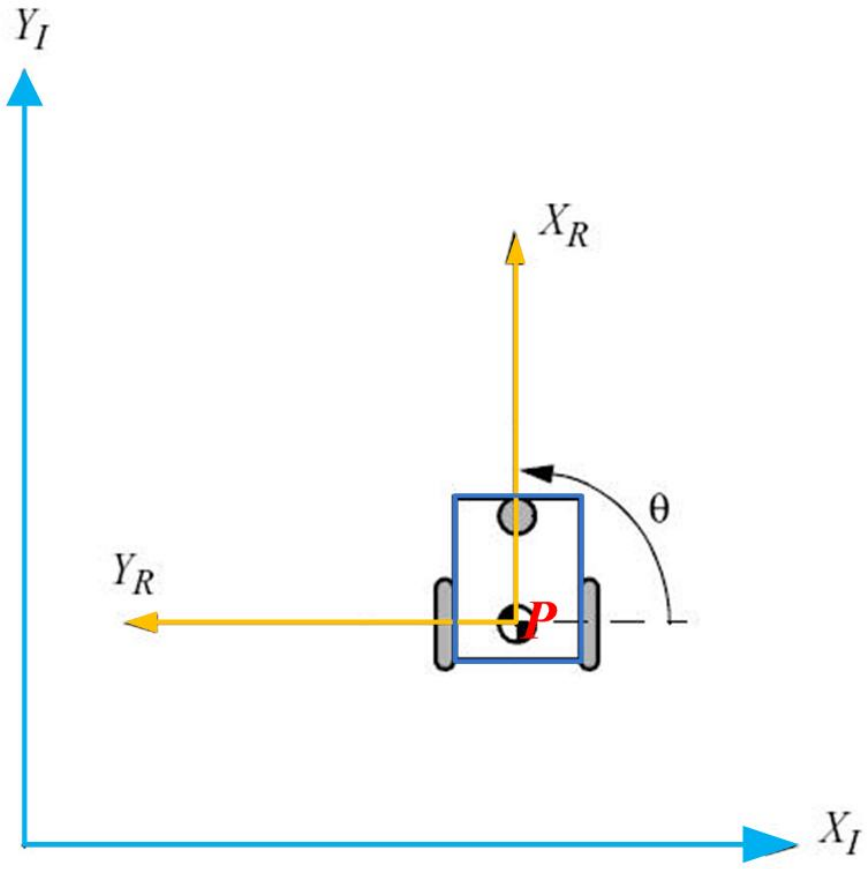

Figura 2.2: Robô móvel alinhado com os eixos do sistema de referência local

$$\dot{\xi_R} = R(\frac{\pi}{2})\dot{\xi}_I \tag{2-3}$$

$$R(\frac{\pi}{2}) = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2-4}$$

Dada uma velocidade $(\dot{x}, \dot{y}, \dot{\theta})$ no sistema de referência global, podemos calcular as componentes de movimento ao longo do eixo local.

$$\dot{\xi_R} = R(\frac{\pi}{2})\dot{\xi}_I = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{y} \\ -\dot{x} \\ \dot{\theta} \end{pmatrix} \tag{2-5}$$

Modelo cinemático direto

No caso mais simples, o mapeamento descrito pela equação 2-3 é suficiente para gerar a fórmula que captura a cinemática direta do robô móvel. Como se desloca o robô dado sua geometria e a velocidade da suas rodas?. Mais formalmente, considere-se o exemplo ilustrado na figura 2.3.

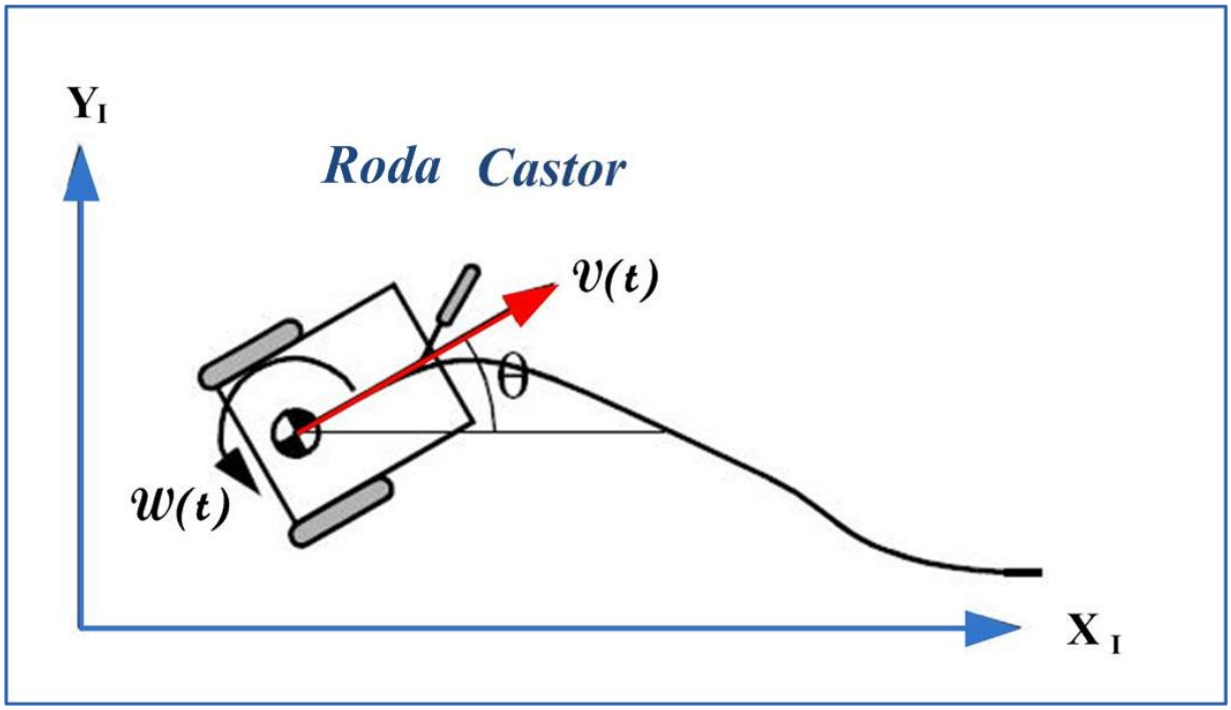

Figura 2.3: Referência global da unidade do robô

O robô possui duas rodas, cada uma com um motor próprio e independente, e cada roda tem um diâmetro r. Seja um ponto P centrado entre as duas rodas, cada uma a uma distancia l de P. Dados r, l, θ, e a velocidade própria de cada roda, $\dot{\varphi}_1$ e $\dot{\varphi}_2$, o modelo da cinemática direta poderia predizer a velocidade de todo o robô em relação ao sistema de referência global:

$$\dot{\xi}_I = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = f(l, r, \theta, \dot{\varphi}_1, \dot{\varphi}_2) \tag{2-6}$$

Da equação 2-3 sabemos que podemos calcular o movimento do robô no sistema de referência global do movimento em relação ao sistema de referência local: $\dot{\xi}_I = R(\theta)^{-1}\dot{\xi}_R$. Por conseguinte, a estratégia seria primeiro calcular a contribuição de cada roda no sistema de referência local $\dot{\xi}_R$.

Suponha que o sistema de referência local esteja alinhado tal que o robô se desloque ao longo do eixo $+X_R$, como se ilustra na figura 2.1. Primeiro consideremos a contribuição da velocidade de cada roda para a velocidade de

translação em P na direção de $+X_R$. Ambas rodas têm que girar à mesma velocidade se quisermos que o robô avance numa só direção, a contribuição de cada roda à velocidade do ponto P é a mesma, $\dot{x}_{r1} = r\dot{\varphi}_1$, $\dot{x}_{r2} = r\dot{\varphi}_2$ e $\dot{x}_{r1} = \dot{x}_{r2}$. Para poder assegurar uma velocidade correta no componente $\dot{x}_R$ de $\dot{\xi}_R$, fazemos com que a velocidade no ponto P seja uma velocidade ponderada das rodas. O valor de $\dot{y}_R$ é sempre zero, porque nenhuma roda contribui com o movimento nessa direção. Finalmente, calculamos o componente rotacional $\dot{\theta}_R$ de $\dot{\xi}_R$. Outra vez, a contribuição de cada roda pode ser calculada independentemente e adicionada; considere-se a roda direita (Roda1). O contínuo giro desta roda resulta num contador no sentido contrário de giro do relógio no ponto P. Lembrando que se a roda 1 gira sozinho, o robô gira com a Roda2(roda esquerda) como apoio. A velocidade de rotação ω_1 em P pode ser calculado porque a roda esta girando instantaneamente ao longe do arco de circulo de radio $2l$:

$$\omega_1 = \frac{r\dot{\varphi}_1}{2l} \tag{2-7}$$

A mesma equação 2-7 se aplica à roda esquerda, com a exceção que o giro resulta num giro no sentido de giro do relógio no ponto P, 2-8.

$$\omega_2 = -\frac{r\dot{\varphi}_2}{2l} \tag{2-8}$$

Combinando estas equações individuais 2-7 e 2-8, o modelado cinemático da unidade do robô é:

$$\dot{\xi}_I = R(\theta)^{-1} \begin{pmatrix} \frac{r\dot{\varphi}_1 + r\dot{\varphi}_2}{2} \\ 0 \\ \frac{r\dot{\varphi}_1}{2l} + \frac{r\dot{\varphi}_2}{2l} \end{pmatrix} \tag{2-9}$$

Esta aproximação do modelado cinemático 2-9 pode prover informação acerca do movimento do robô dado seus componentes de velocidade das rodas. De qualquer modo nós desejamos determinar o possível espaço do movimento para cada desenho de chassi do robô, para fazer isto se tem que estudar as restrições sobre o movimento do robô imposto pelas rodas.

Restrições na cinemática da roda

O primeiro passo para encontrar o modelo cinemático do robô é expressar as restrições sobre o movimento das rodas individualmente (Sie04). O movimento individual das rodas pode mais tarde ser combinado para calcular o movimento de todo o robô. Há muitos tipos de modelos básicos das rodas com propriedades cinemática variadas, sim embargo aqui só trataremos de dos tipos que são os que mais interessam neste trabalho.

De qualquer modo, muitas suposições importantes simplificaram esta apresentação, é assumido que o plano da roda sempre é vertical e que existe um único ponto de contato entre a roda e o chão; mais lá disto é assumido que não há deslizamento no ponto de contato. Isto é, a roda experimenta somente movimento baixo condições de rolamento e rotação pura no eixo vertical através do ponto de contato.

Baixo estas suposições, apresenta-se dois restrições para cada tipo de roda; a primeira restrição reforça o conceito de contato de rolamento - que a roda deve girar quando o movimento toma lugar na direção apropriada -; A segunda restrição reforça o conceito de não deslizamento lateral - que a roda não deve deslizar-se ortogonalmente ao plano da roda.

A roda que tratamos neste trabalho é a roda standard fixa que não tem eixo vertical de rotação para cabeceio. Seu ângulo ao chassi é fixo, e isto limita seu movimento só para atrás e adiante e no plano de rotação do ponto de contato com o chão, a figura 2.4, descreve uma roda fixa standard A e indica a posição de sua postura relativo ao sistema de referência local X_R, Y_R. A posição de A esta expressado em coordenadas polares pela distância l e o ângulo α. O ângulo do plano da roda relativo ao chassi esta denotado por β, o qual é fixa. A roda de radio r, pode girar no tempo, e então seu posição rotacional em torno de seu eixo horizontal é uma função do tempo $t : \varphi(t)$.

As restrições de rolamento para esta roda reforça que tudo movimento ao longe da direção do plano da roda deve ser acompanhado a ração de giro da roda tal que exista puro rolamento no ponto de contato:

$$\begin{pmatrix} \sin(\alpha+\beta) & -\cos(\alpha+\beta) & (-l)\cos\beta \end{pmatrix} R(\theta)\dot{\xi}_I - r\dot{\varphi} = 0 \qquad (2\text{-}10)$$

O primeiro termo da equação 2-10 denota o movimento total ao longe do plano da roda; os três elementos do vetor sobre a esquerda representam o mapeamento de cada variável $\dot{x}, \dot{y}, \dot{\theta}$ para as contribuições do movimento ao longe do plano da roda. Note que o termo $R(\theta)\dot{\xi}_I$ é usado para transformar os parâmetros do movimento $\dot{\xi}_I$ que esta no sistema de referência global X_I, Y_I dentro dos parâmetros do movimento no sistema de referência local X_R, Y_R que são mostradas na equação 2-3.

A restrição de deslizamento para esta roda reforça que o componente do movimento ortogonal da roda ao plano da roda deve ser zero, como se amostra na equação 2-11:

$$\begin{pmatrix} \cos(\alpha+\beta) & \sin(\alpha+\beta) & l\sin\beta \end{pmatrix} R(\theta)\dot{\xi}_I = 0 \qquad (2\text{-}11)$$

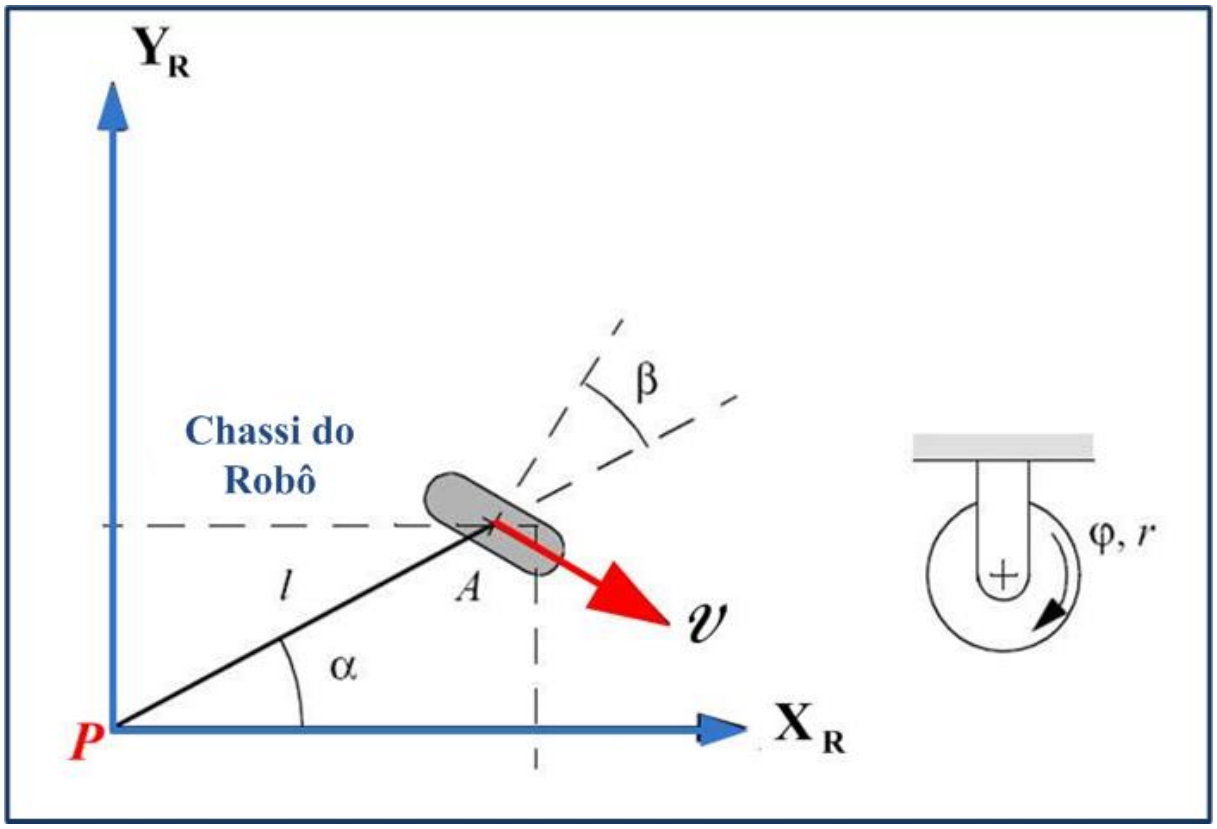

Figura 2.4: Roda standard fixa e seus parâmetros

Restrição na cinemática do robô

Dado o robô móvel com m rodas, pode-se calcular as restrições cinemática do chassi do robô (Sie04). Cada roda impõe restrições ao movimento do robô e então o processo é simplificado à combinação apropriada de todas as restrições cinemática aparecidas de todas as rodas.

A roda standard fixa que usamos neste trabalho tem impacto sobre a cinemática do chassi do robô e sim embargo requer considerações quando se calcula as restrições cinemática do robô. Suponha que o robô tem um total de N_f rodas standard e β_f seja a orientação dessas rodas fixas. Denotaremos $\dot{\varphi}_f(t)$ como a velocidade angular das rodas.

Então a restrição de rolamento de todas as rodas pode agora ser colecionada numa única equação 2-12:

$$J_1(\beta_f)R(\theta)\dot{\xi}_I - J_2\dot{\varphi} = 0 \tag{2-12}$$

Nesta expressão 2-12 se tem que J_2 é uma matriz constante diagonal $N \times N$ onde as entradas são os valores dos rádios r de cada roda; $J_1(\beta_)$ denota uma matriz com projeção para todas as rodas no movimento. Onde na equação 2-13 J_{1f} é uma matriz constante das projeções de todas as rodas fixas. Isto

tem o tamanho $(N_f \times 3)$, com cada fila consistindo dos três termos na matriz da equação 2-10 para cada roda fixa.

$$J_1(\beta_f) = \begin{pmatrix} J_{1f} \end{pmatrix} \tag{2-13}$$

Em resumo, a equação 2-12 representa a restrição que todas as rodas standard tem ao girar em torno a seu eixo horizontal. Agora usamos a mesma técnica para colecionar as restrições de deslizamento para todas as rodas numa única expressão:

$$C_1(\beta_f)R(\theta)\dot{\xi}_I = 0 \tag{2-14}$$

$$C_1(\beta_f) = \begin{pmatrix} C_{1f} \end{pmatrix} \tag{2-15}$$

Das equações 2-14 e 2-15, C_{1f} é uma matriz de $(N_f \times 3)$ onde seus termos são os mesmos da equação 2-11 para todas as rodas fixas. Assim a equação 2-14 é uma restrição sobre todas as rodas fixas e que seus termos de movimento ortogonal aos planos da roda devem ser zero. Esta restrição sobre todas as rodas fixas tem um grão significado sobre a manobrabilidade do chassi do robô. Pelo tanto a equação geral para a cinemática do robô queda como amostra-se na equação 2-16.

$$\begin{pmatrix} J_1(\beta_f \\ C_1(\beta_f \end{pmatrix} R(\theta)\dot{\xi}_I = \begin{pmatrix} J_2\varphi \\ 0 \end{pmatrix} \tag{2-16}$$

2.2 Percepção Sensorial

Uma das mais importantes tarefas de um sistema autônomo de alguma classe é adquirir conhecimento acerca do ambiente. Isto é realizado fazendo medições usando vários tipos de sensores e logo extraindo informação significativa dessas medidas.

Nesta seção se apresentam os mais comuns sensores usados em robôs móveis e logo se discute um pouco sobre a estratégia da extração da informação dos sensores (Eve95).

Sensores para robôs móveis

Há uma amplia variedade de sensores usados em robôs móveis, alguns destes sensores são usados para medidas de simples valores como a temperatura interna da eletrônica do robô ou a velocidade angular dos motores. Outros, mais sofisticados podem ser usados para adquirir informação acerca do ambiente do robô ou sobre a medida direita da posição global do robô. Nesta seção trata sobre sensores usados para a extração acerca do ambiente do robô.

H

Tabela 2.1: Classificação dos sensores usados nas aplicações de robôs móveis

Classificação General	Sensor y Sistema	PC ou EC	A ou P
Sensores Tacteies	ópticos	EC	A
Sensores do motor	encoders e potenciómetros	PC	P
Velocidade	doppler som	EC	A
Visão	CCD/CMOS câmeras	EC	P

Segundo (Sie04) classificam-se os sensores usando dos importantes eixos funcionais : *proprioceptive/exteroceptive* e *passive/active*.

Sensores *Proprioceptive* medem valores internos do sistema do robô; por exemplo, velocidade do motor, carga da roda, ângulo da junta do robô, voltagem da bateria.

Sensores *Exteroceptive* adquiram informação do ambiente do robô, por exemplo, medida da distancia, intensidade da luz, amplitude do som.

O sensor *Passive* mede as condições ambientais da energia de entrada ao sensor, por exemplo as sondas de temperatura, microfones e *CCD* das câmaras.

O sensor *Active* pode dirigir mais interações controladas com o ambiente, eles muitas vezes realizam um desempenho superior; de qualquer modo, estes sensores introduz muito risco como a interferência entre seus sinais.

A tabela 2.1 amostra a classificação dos sensores mais usados para aplicações de robôs móveis. Os sensores mais utilizados vão ser comentados brevemente. Onde A = ativos; P = passivos; P/A = passivos e ativos; PC = proprioceptive; EC, Exteroceptive.

Os tipos de sensores da tabela 2.1 estão ordenadas em ordem ascendente de complexidade. a continuação se descreve alguns sensores brevemente.

Encoder ópticos

Os encoder ópticos são os dispositivos mais populares para a medida angular da posição e velocidade para o controle do motor. Nos robôs móveis, os encoder são usados para o controle da posição e a velocidade das rodas. Eles estimam a posição do robô no sistema do robô e quando é aplicado ao problema da localização, algumas correções são requeridas para estes casos como será discutido mais adiante.

A Figura 2.5 ilustra um disco típico do sensor montado no eixo do motor para aplicações de leitura da posição e direção do motor. Quando o motor gira, o estator gera 2 quadratura de pulsos e um pulso index. A partir destes dados é possível inferir velocidades. Estes sinais são ilustradas na Figura 2.6:

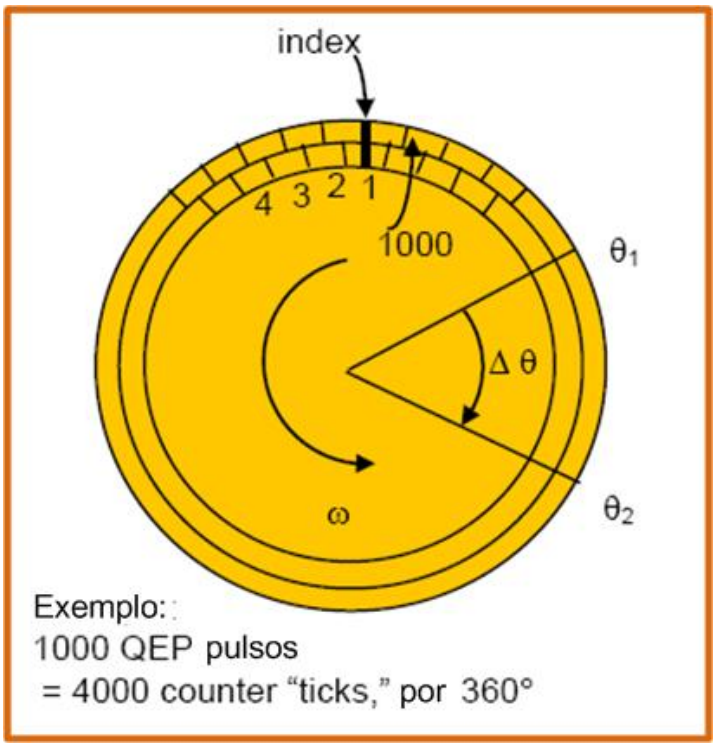

Figura 2.5: Disco do sensor de posição, usado para inferir velocidade

Sensores ultra-som

O principio básico de estes sensores é transmitir um paquete de pressão em forma de onda e medir o tempo que toma em refletir esta onda e retornar ao receptor. A distância d do objeto que causa a reflexão pode ser calculado baseado sobre a velocidade de propagação do som c e o tempo de vôo t.

$$d = \frac{c \times t}{2} \tag{2-17}$$

A velocidade do som no ar esta dado por:

$$c = \sqrt{\gamma RT} \tag{2-18}$$

Onde : γ = taça da temperatura específica; R = constante do gás; T = temperatura em K;

A forma de como se propaga esta onda se ilustra na Figura 2.7

2.3 Localização-Posicionamento

A navegação é um dos mais grandes desafios que enfrenta o robô móvel, o êxito da navegação requer o êxito na percepção, na localização, na cognição e no controle de movimento, o robô tem que modular a saída do motor para conseguir a trajetória desejada.

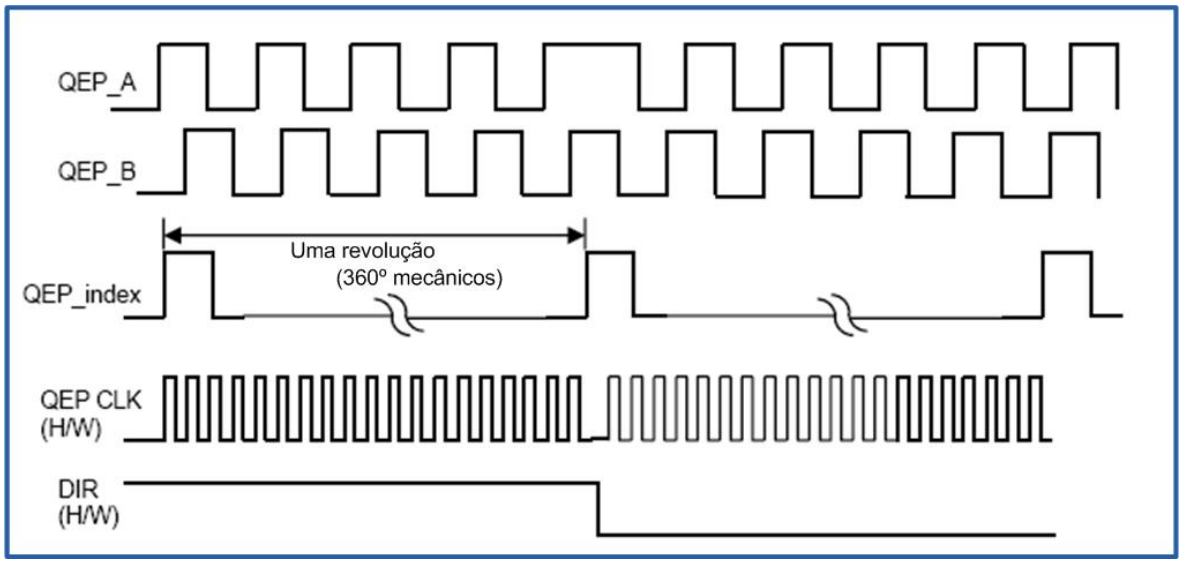

Figura 2.6: Pulsos de quadratura do encoder

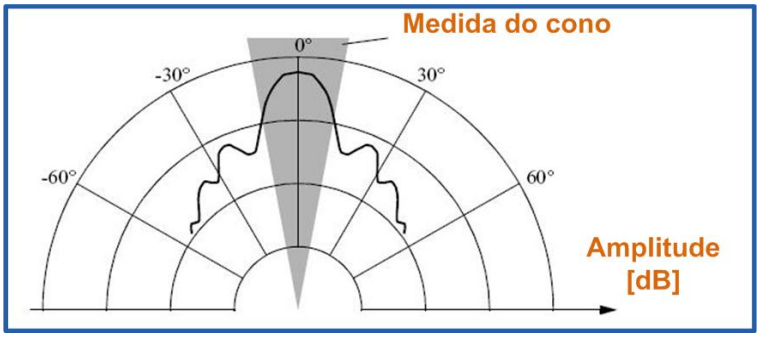

Figura 2.7: Intensidade típica da distribuição de um sensor ultra-sônico

Se poderia-se ajuntar um sensor *GPS* ao robô móvel, muitos dos problemas da localização poderiam ser obviados; este sensor poderia informar ao robô sua posição exata, dentro e fora do ambiente, logo a resposta à pergunta, Onde estou?, poderia ser disponível, infelizmente, tão sensor não é geralmente prático, primeiro por o alto custo e segundo porque a localização implica mais que o conhecimento da posição absoluta no sistema de referência terrestre. Considere-se um robô que inter-atua com humanos. Este robô poderia precisar identificar seu posição absoluta, mas seu posição relativa em relação ao humano também é igual de importante. Sua localização pode incluir a identificação de humanos usando um arranjo de sensores, logo calcular sua posição relativa ao humano.

Claramente, os sensores e atuadores do robô jogam um papel integral no processo de localização. Isto é porque da imprecisão e imperfeição dos sensores e atuadores que a localização dificulta-se.

Os sensores são fundamentalmente importantes para o processo de per-

cepção, e por conseguinte o grau ao qual estes sensores podem distinguir o estado do ambiente é crítico, o ruído nos sensores induz uma limitação sobre a consistência da leitura do sensor no mesmo ambiente. Muitas vezes a fonte do problema no ruído do sensor é que algumas características do ambiente não são capturadas pela representação do robô e são obviados.

Por exemplo o sistema sensorial humano, particularmente visual, tende a recepcionar entradas únicas em cada local único, é dizer que cada lugar se mira diferente. A capacidade deste único mapa é somente aparente quando um considera situações onde a falha se mantém; é dizer se um humano passa por um edifício não conhecido e completamente obscuro, quando seu sistema visual somente observa a obscuridade, o sistema de localização diminui rapidamente.

Nos robôs, a leitura do sensor não é único, isto é chamado *aliasing* do sensor, e é uma norma e não a exceção. Num robô com muitos sensores tem uma grande variedade de estados no ambiente por perceber e que muitos destes estados poderiam ativar o mesmo valor a estes sensores, assim o robô não pode distinguir de entre muitos estados, Outro problema de localização é o ruído do atuador, em particular uma só ação tomada por o robô pode ter muitos possíveis resultados diferentes, em resumo o atuador de um robô móvel introduz incerteza sobre o estado futuro, por conseguinte o simples fato do movimento tende a incrementar a incerteza do robô móvel.

É importante notar que desde o ponto de vista do robô, a incerteza do atuador é visto como um erro em odometria, o a incapacidade do robô de estimar seu própria posição no tempo usando o conhecimento da sua cinemática e dinâmica. A fonte do erro geralmente amostra um modelo incompleto do ambiente, porque o robô não modela o fato de que o chão possa ser escorregado e possa deslizar-se, todos estes fontes de erro não modeladas resultam numa imprecisão entre o movimento físico do robô, a intenção de movimento e a estimarão sensorial do movimento (Sie04).

2.3.1 Modelo do erro para a estimarão da posição por odometria

Geralmente a posição de um robô é representada por o vetor:

$$p = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \tag{2-19}$$

Para um robô diferencial a posição pode ser estimada começando do conhecimento da posição e integrando o movimento (somando o incremento da distancia percorrida), para um sistema discreto com um intervalo de

amostragem fixo $\triangle t$ o incremento das distâncias de percorrido $(\triangle x; \triangle y; \triangle \theta)$ são:

$$\triangle x = \triangle s \cos(\theta + \frac{\triangle \theta}{2}) \tag{2-20}$$

$$\triangle y = \triangle s \sin(\theta + \frac{\triangle \theta}{2}) \tag{2-21}$$

$$\triangle \theta = \frac{\triangle s_r - \triangle s_l}{b} \tag{2-22}$$

$$\triangle s = \frac{\triangle s_r + \triangle s_l}{2} \tag{2-23}$$

onde: $(\triangle x; \triangle y; \triangle \theta)$ = trajetória percorrida no ultimo intervalo de amostragem; $\triangle s_r; \triangle s_l$ = distancia percorrida pela roda direita e esquerda respectivamente; b = distância entre as dois rodas do robô diferencial.

Assim obtemos a posição atual p':

$$\begin{aligned} p' &= \begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = p + \begin{pmatrix} \triangle s \cos(\theta + \frac{\triangle \theta}{2}) \\ \triangle s \sin(\theta + \frac{\triangle \theta}{2}) \\ \triangle \theta \end{pmatrix} \\ &= \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} \triangle s \cos(\theta + \frac{\triangle \theta}{2}) \\ \triangle s \sin(\theta + \frac{\triangle \theta}{2}) \\ \triangle \theta \end{pmatrix} \end{aligned} \tag{2-24}$$

Usando a relação para $(\triangle s; \triangle \theta)$ das equações 2-23 e 2-22, mais adiante obtemos a equação básica para a atual posição de odometria (para robôs de manejo diferencial).

$$\begin{aligned} p' &= f(x, y, \theta, \triangle s_r, \triangle s_l) \\ &= \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} \frac{\triangle s_r + \triangle s_l}{2} \cos(\theta + \frac{\triangle s_r - \triangle s_l}{2b}) \\ \frac{\triangle s_r + \triangle s_l}{2} \sin(\theta + \frac{\triangle s_r - \triangle s_l}{2b}) \\ \frac{\triangle s_r - \triangle s_l}{b} \end{pmatrix} \end{aligned} \tag{2-25}$$

Tal como foi discutido antes, a atual posição de odometria pode dar somente uma estimarão da posição atual irregular. Devido à integração das incertezas do erro de p e o erro de movimento durante o movimento incremental $(\triangle s_r; \triangle s_l)$, a posição do erro baseado na integração da odometria cresce com

o tempo.

Agora se tem que estabelecer o modelo para a integração da posição p' para obter a matriz de co-variância $\sum_{p'}$ da posição estimada. Para fazer isto, assumimos que o ponto inicial da matriz de co-variância $\sum_p$ é conhecido; para o incremento do movimento $(\triangle s_r; \triangle s_l)$ assumimos a seguinte matriz de co-variância $\sum_{\triangle}$:

$$\sum_{\triangle} = covar(\triangle s_r, \triangle s_l) = \begin{pmatrix} k_r|\triangle s_r| & 0 \\ 0 & k_l|\triangle s_l| \end{pmatrix} \tag{2-26}$$

Onde $\triangle s_r$ y $\triangle s_l$ são as distâncias da trajetória de cada roda, e k_r, k_l são as constantes do erro representando os parâmetros não determinísticos do controle do motor e a inter-ação da roda com o chão, como se observa na equação 2-26 temos feito as seguintes suposições:

- Os dois erros do controle individual das rodas são independentes.
- As variâncias dos erros (ambas rodas), são proporcionais ao valor absoluto das distancias percorridas $(\triangle s_r; \triangle s_l)$.

Os erros do movimento são devido ao movimento impreciso por motivos de deformação das rodas, deslizamento, rugosidade do chão, erro nos encoder, etc. O valor das constantes de erro k_r y k_l dependem do robô e o ambiente e deveriam ser experimentalmente estabelecidos para melhorar o desempenho.

Se assumimos que p y $\triangle_{rl} = (\triangle s_r; \triangle s_l)$ não são correlacionados, e a derivada da equação 2-25 é favoravelmente aproximada pela expansão de Taylor de primeiro ordem (linearização), concluímos usando a lei de propagação do erro:

$$\sum_{p'} = \nabla_p f . \sum_{p} . \nabla_p f^T + \nabla_{\triangle_{rl}} f . \sum_{\triangle} . \nabla_{\triangle_{rl}} f^T \tag{2-27}$$

A matriz de co-variância $\sum_p$ é por suposto sempre permitido pela $\sum_{p'}$ do passo anterior, e pode assim ser calculado depois de especificar o valor inicial.

Usando a equação 2-25, podemos desenvolver os Jacobianos, $F_p = \nabla_p f$ e $F_{\nabla_{rl}} = \nabla_{\nabla_{rl}} f$:

$$\begin{aligned} F_p &= \nabla_p f = \nabla_p (f^T) = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial \theta} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & -\triangle s \sin(\theta + \frac{\triangle \theta}{2}) \\ 0 & 1 & \triangle s \cos(\theta + \frac{\triangle \theta}{2} \\ 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{2-28}$$

$$F_{\triangle_{rl}} = \begin{pmatrix} \frac{1}{2}\cos(\theta + \frac{\triangle\theta}{2}) - \frac{\triangle s}{2b}\sin(\theta + \frac{\triangle\theta}{2}) & \frac{1}{2}\cos(\theta + \frac{\triangle\theta}{2}) + \frac{\triangle s}{2b}\sin(\theta + \frac{\triangle\theta}{2}) \\ \frac{1}{2}\sin(\theta + \frac{\triangle\theta}{2}) + \frac{\triangle s}{2b}\cos(\theta + \frac{\triangle\theta}{2}) & \frac{1}{2}\sin(\theta + \frac{\triangle\theta}{2}) - \frac{\triangle s}{2b}\cos(\theta + \frac{\triangle\theta}{2}) \\ \frac{1}{b} & -\frac{1}{b} \end{pmatrix} \tag{2-29}$$

2.4 Navegação de robôs móveis

A navegação baseada em sensores propicia ao robô um comportamento dependente da situação interna do veiculo e do meio onde está inserido (Fre92). O sistema de percepção é o responsável pelo tratamento e envio de dados coletados por sensores, determinando o tipo de movimento a ser executado pelo veículo.

Periodicamente podem ser verificados vários parâmetros internos próprios do robô (nível de energia, presença de falhas entre outros). Esses fatores podem ser responsáveis por avarias e problemas que impeçam o perfeito desempenho do robô. Uma vez detectamos, poderão ser solucionadas pelo sistema de controle.

Este ambiente também pode ser sensoriado periodicamente, possibilitando diretamente a execução de uma determinada ação com base nos dados obtidos por sensoriamento, ou a execução da sub-tarefa de mapeamento do ambiente.

Na navegação baseada em sensores, as três sub-tarefas que compõem as técnicas de controle são realizadas em tempo real de operação. Ao contrário da navegação em locais conhecidos, em que o mapa do ambiente previamente conhecido, sendo o planejamento determinado numa fase anterior, a navegação por sensores, além de exigir um sistema de percepção mais complexo, necessita de técnicas de controle de alto desempenho de execução nas tomadas de decisão, e ao nível operacional, a cada instante. A capacidade de tratar ruídos outra característica que devem apresentar os sistemas baseados em sensores, pois esses são bastante comuns quando se trata com tais tipos de dispositivos . O sistema de controle deve ser tolerante a sinais ruidosos, conseguindo diminuir a sua influencia maléfica no comportamento do veículo.

Uma das vantagens propiciadas pela utilização de sensores é a possibilidade do robô móvel operar autonomamente. Tal fato possibilita a existência de veículos que possam realizar suas tarefas completamente desconectados de qualquer tipo de *hardware* de apoio, como também sem qualquer tipo de interferência externa (Ben96). Assim, um comportamento inteligente pode ser esperado dos robôs que realizam suas tarefas, baseados nas informações obti-

das do ambiente e de si próprios. O trabalho desenvolvido aborda o estudo de sistemas de controle que se enquadram nessa categoria de controle de navegação.

2.4.1 Controle de agentes

Ao se tratar com navegação baseada em sensores, consideram-se os robôs móveis envolvidos como agentes. Um agente é um processo capaz de possuir percepção, computação e ação dentro de seu mundo, podendo ser físico ou no.

Arquiteturas para controle de agentes

Baseado em planos (*planner-based approach*), que é uma estratégia deliberativa, onde a arquitectura de controle é do tipo *top-down*(Mat94). É usado um modelo de mundo centralizado para verificar a informação sensorial e gerar ações. Essa abordagem permite a formulação explícita das tarefas e metas do sistema, e estimarão da qualidade da desempenho do agente.

Muitos dos métodos tradicionais utilizados para planejamento de caminho em locais conhecidos estão sendo modificados para serem utilizados no caso de navegação por sensores *planner-based*. Estudos estão sendo realizados para utilizarem-se heurísticas do tipo campo potencial e diagrama de Voronoi na navegação sensoriada (Cho95). Por outro lado, incertezas que possam vir a ocorrer no sensoriamento e mudanças do ambiente podem requerer freqüente remanejamento, sendo o seu custo muitas vezes alto para sistemas complexos. A abordagem baseada em planejamento é criticado devido a esta dificuldade de convivência com a complexidade do problema conseqüentemente no permitindo reações em tempo real e tratamento de ruídos.

Várias pesquisas estão sendo realizadas com o objetivo de desenvolverem-se sistemas de controle de agentes em tempo real . Uma das abordagens mais proeminentes são do tipo *bottom-up* e implementam estratégias de controle para agentes como uma coleção de pares de ação-reação pré-programados com mínimos estados. Esses sistemas não mantém modelos internos de representação do ambiente, mas simplesmente comandam a ação apropriada para um determinado conjunto de valores sensoriados. Conta-se com uma relação direta entre os valores sensoriados e a ação , associada a uma rápida realimentação do ambiente. Estratégias puramente reativas tem-se mostrado eficazes para uma variedade de problemas que podem ser definidos temporiamente, mas são inflexíveis para a execução em tempo real dividido a sua incapacidade de armazenar informação dinamicamente.

Arquiteturas híbridas enfocam um compromisso entre abordagens puramente reativas e abordagens deliberativas pelo emprego de um sistema reativo para controle de baixo nível e um planejamento para decisões de alto nível. Sistemas híbridos formam um campo de elevado potencial para pesquisa. Como exemplo, pose-se citar o planejamento reativo ou execução reativa que utiliza primitivas de alto nível para planejamento,as quais cuidam de todos os detalhes da execução *(Reactive Actions Packages)*, bem como o PRS *(Procedure Reasoning System)*que é uma arquitectura para inovação de regras flexíveis (Mat94) entre outros. Esses sistemas tendem separar os módulos de controle em duas o mais partes independentes, mas intercomunicação. Em muitos casos, processos reativos de baixo nível cuidam das ações imediatas de sobrevivência do robô, enquanto os níveis mais altos selecionam as seqüencias de ações.

A abordagem baseada em comportamento é uma extensão de sistemas reativos, mas que apresenta alguns aspectos da abordagem baseada em planejamento (Jor03).Embora algumas vezes seja encontrada numa forma relativamente confusa na literatura, a estratégia baseada no comportamento consideravelmente mais potente do que as abordagens puramente reativas, uma vez que no possui limitações em seus estados internos, formado por pares restritos de ação-reação. Mesmo que os sistemas baseados no comportamento possuam algumas propriedades dos sistemas reativos, casualmente contenham componentes reativos, sua computação no limitada a consultas. Esses sistemas podem usar diferentes formas de representação interna do ambiente bem como executar computação distribuída, a fim de decidir qual ação-efeito aplicar (Mat94).

2.4.2 Técnicas de controle inteligentes

O comportamento inteligente pretendido pode ser alcançado com o uso de módulos controladores baseados em sistemas inteligentes artificiais, utilizando as arquiteturas aplicáveis a agentes (Lak98). Técnicas de controle advindas da inteligência computacional, tanto simbólica como conexionista, estão sendo utilizadas nas três sub-tarefas de controle (mapeamento, planejamento e execução). Estes módulos recebem os dados provenientes dos sensores, podendo, a partir dos dados sensoriados: montar mapas internos de representação, fornecer ações e comportamentos a serem executados, modificar-se conforme o ambiente entre outros. Controladores que utilizam redes neurais (Lak98), sistemas *fuzzy* (Les04), sistemas especialistas (Ben96) e algoritmos genéticos (Koz94), estão sendo utilizados como módulos de controle de comportamento. A seguir serão vistos mais detalhadamente cada um destes módulos.

Módulos com sistemas especialistas baseados em regras

Neste tipo de módulo de controle, as sinais provenientes dos sensores são analisados por sistemas especialistas (Vee95). Estes sistemas são formados por tabelas ou regras que tentam mapear as possíveis situações que possam vir a ser apresentadas ao robô. Leszek (Les04) propõe um sistemas de interpretação que, ao receber os dados dos sensores, compara-os com uma série de regras e, baseado nestas, toma decisões. Esses sistemas são previamente definidos por um especialista que procura cercar as situações possíveis de acontecer. Desta forma, a robustez do sistema está diretamente relacionada com o número de regras. a medida que o número de regras aumenta, tornam-se mais difíceis o gerenciamento e os testes. Os sistemas baseados em regras possuem um bom desempenho para manipulações simbólicas, o que no o caso da classificação dos dados sensoriados, onde existe a necessidade de transformação de dados numéricos em símbolos. Um outro problema é a dificuldade de, uma vez definido o sistema, expandi-lho para um número maior de regras, a fim de cobrir novas situações, a dependência de dados fidedignos também é grande neste tipo de tratamento.

Módulo por sistemas fuzzy

Módulos de controle por sistemas *fuzzy* utilizam a lógica *fuzzy* para a interpretação dos sinais recebidos dos sensores. A lógica *fuzzy* baseada na teoria de conjuntos e na teoria das possibilidades. Na teoria de conjuntos clássicos, um elemento pertence ou no a determinado conjunto (Zad84). J na teoria dos conjuntos *fuzzy*, um elemento pode pertencer parcialmente a um determinado conjunto, ou seja, existe um determinado grau de possibilidade de o elemento pertencer ao conjunto. Isso pode ser ilustrado pelo gráfico na figura 2.8

Se fosse desejado categorizar a distancia de um obstáculo ao robô da figura 2.8 em noções difusas de distância, seria possível dizer que: um obstáculo à distancia de 38 metros do robô encontrasse-se a uma distancia "um pouco"perto e "um pouco"média. J um obstáculo a 10 metros, estaria totalmente perto. Tal sentença traduz a noção que o ser humano tem das grandezas físicas (Zad84). A mente humana no sabe exatamente o quanto a água esta fria ou quanto está morna, porém consegue tomar decisões baseadas nessa noção difusa das grandezas. A abordagem *fuzzy* bastante simbólica, por tratar com noções qualitativas do universo de tratamento.

O controle *fuzzy* está baseado na idéia difusa das grandezas. O controlador recebe os valores advindo dos sensores e, então, os fuzzyfica, ou seja, ele determina qual o grau de possibilidade da medida pertencer a cada um dos conjuntos *fuzzy* de entrada, neste caso representando os possíveis estados do

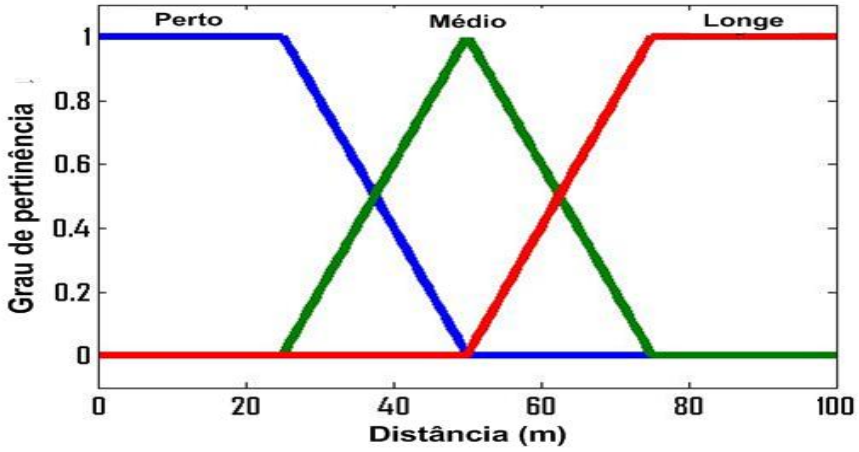

Figura 2.8: Conjuntos *fuzzy* - A variável distância apresenta três conjuntos *fuzzy* que representam os três conceitos lingüísticos: perto, médio, longe

robô em relação ao meio (por exemplo, perto de obstáculos, longe do alvo entre outros). Feito isso, consulta-se a base de regras e toma-se uma decisão. A base de regras um conjunto de proposições do tipo SE..ENTÃO... Por exemplo: SE perto ENTÃO desvie, onde a saída um comportamento do robô. Uma vez avaliadas todas as regras, realiza-se uma espécie de ponderação das mesmas, de modo a se obter um comportamento final da saída (Zad84).

Da mesma forma que o módulo de controle por sistemas especialistas, o controle por *fuzzy* composto por regras do tipo SE/ENTO, as quais são facilmente entendidas se comparadas com equações matemáticas. A vantagem que, devido às suas funções, esses sistemas permitem uma interface entre o nível simbólico (das regras) e o nível de sinal (dos sensores), realizando perfeitamente o casamento entre os dois. Ressalta-se o fato de que, nos sistemas *fuzzy*, o elemento a ser tratado pertence parcialmente a um conjunto de regras, havendo trânsito gradual entre os diversos tipos, o que no ocorre nas regras heurísticas (Les04).

Infelizmente, esse tipo de estrutura em forma de regras dificulta modificação do sistema e conseqüente adaptação a novas situações, que seria uma das exigências básicas para o funcionamento dos sistemas ditos autônomos. Outra dificuldade é a extração de regras dos conjuntos de dados a ser realizada por um operador humano. Em decorrência desses fatos, estuda-se sua utilização em combinação com outros sistemas mais flexíveis.

Módulo por redes neurais artificiais

Redes neurais artificiais (RNAs) foram criadas na tentativa de obter-se um modelo que descrevesse o funcionamento do cérebro. RNAs são formadas,

em sua maioria, por elementos no lineares, altamente conectados, denominados de neurônios artificiais (Ben96). As redes neurais apresentam a habilidade de aprender com a experiência. O aprendizado se d através da modificação do valor das conexões entre os neurônios (pesos).

Módulos de controle por RNAs utilizam-nas para a interpretação dos dados provenientes dos sensores e geração de alguma ação nos atuadores. Esta dissertação utiliza várias características que as redes neurais possuem, e que são requeridas pelos princípios de autonomia em ambientes reais. Algumas dessas propriedades são as seguintes: flexibilidade e generalização, tolerância a falhas em componentes físicos, facilidade de convivência com a micro-estrutura do robô (podendo modelar sua própria estrutura de maneira a explorar as melhores características senso-motoras do robô)(Fre92), tolerância a ruído, natureza paralela (veloz para aplicações em tempo real)(Vee95), e se no forem impostos limites arquitectura da rede (conexões e funções de transferência), ter-se- um dispositivo com grande potencial para o tratamento de estruturas temporais e mapeamentos complexos.

A utilização de redes neurais para controle de RNAs pode ser executada através do desenvolvimento de redes cujo aprendizado (e conseqüente modificação dos valores dos pesos) acontece de forma que a rede aprenda a tomar atitudes corretas na presença de determinadas situações sensoriais verificadas.

A rede recebe como entrada os valores providos dos sensores e fornece como saída um determinado comportamento a ser executado pelo sistema de acionamento do robô. Essas redes podem ser treinaras anteriormente (supervisionadas) ou podem dinamicamente sofrer modificações (no-supervisionado)(Fre92).

Como desvantagem do uso de redes neurais para controle de robôs está o fato de que o mapeamento das características e a classificação dos sinais, internamente na rede, não são visíveis e são de difícil entendimento. Assim, no se sabe de que forma a rede está armazenado determinado conhecimento, nem em que local isso é feito (Fre92).

Outro problema existente quanto convergência de aprendizado, o que algumas vezes pode ser muito lento, prejudicando assim o desempenho do robô. Além disso, em alguns casos, o próprio aprendizado não é garantido, o que inicializa a sua posterior utilização.

Support vetor machine

"*support vetor machine*"(SVM)é um procedimento construtivo universal de aprendizagem baseado em "*statistical learning theory*". O termo universal significa que o SVM pode ser utilizado para o aprendizado de várias repre-

sentações como as redes neurais, as funções de base radial, "*splines*"e funções polinomiais. Atualmente é uma técnica usada com muitas vantagens sobre outros métodos de aprendizado de maquinas e com pesquisas em SVM baseados no algoritmo *Simultaneous Localization and Mapping* ($SLAM$)que permite a um robô autônomo móbil navegar num ambiente dinâmico ou estático (Jia07). Fazendo uma comparação com redes neurônais, o SVM tem funções kernel que mapeiam um espaço dimensional muito grande, o espaço de busca só tem um único mínimo local, o treinamento e a classificação é extremosamente eficiente fornecendo a precisão e a robustez.

Módulo de controle por algoritmos genéticos

Algoritmos genéticos (AGs) são algoritmos de procura baseados em mecanismos de seleção natural e genética natural (Ash06). Ao contrário das redes neurais, nas quais a busca de soluções pode ser obtida através do uso de funções matemáticas de minimização de erros (Les04), os algoritmos genéticos combinam a capacidade de sobrevivência de estruturas entre suas cadeias, onde as melhores estruturas são trocadas randomicamente, gerando novas estruturas e gerando um algoritmo de busca inovador. Os AGs eficientemente exploram informações históricas para especular e gerar novos ponto de busca com a expectativa de aumento do desempenho dos sistemas.

Módulos de controle por algoritmos genéticos podem ser utilizados individualmente onde o controlador recebe os dados provenientes dos sensores, gerando uma população de possíveis ações que serão avaliadas e reproduzidas, resultando numa ação final de execução. O algoritmo-base deste tipo de módulo pode ser considerado como um tipo de tentativa-e-erro [Fuk94], o que pode levar o sistema a encontrar um mínimo global e no local, como em outros métodos. Por esta razão, algoritmos genéticos são uma potente ferramenta de otimização que pode ser utilizados por outros módulos de controle.

Módulo de controle híbrido

Fukada (Fuk94) faz uma comparação entre os diversos tipos de módulos de controle inteligente ver tabela 2.2 . Como se pode notar, todos os métodos apresentam diferentes características. Com o objetivo de aliarem-se as vantagens presentes em cada abordagem, está-se voltando para o uso de módulos de controle híbridos. Como exemplo, podem-se usar redes neurônais e sistemas *fuzzy* como pré-processadores de regras heurísticas, transformando dados numéricos em conjuntos de dados simbólicos a serem manipulados.

Tabela 2.2: Comparação entre controladores: FO-Forte, ME-Médio, RA-Razoável e FR-Fraco

1	MM	AP	TR	RC	NL	OT
Locais Conhecidos	FO	FR	RA	FR	FR	FR
Redes Neurais	FR	FO	FO	FR	FO	ME
Fuzzy	ME	FR	FO	RA	FO	FR
Sistemas especialistas	RA	FR	FR	FO	RA	FR

Onde: MM - Modelamento Matemático, AP - Aprendizado, TR - Tempo Real, RC - Representação do conhecimento, NL - No Linearidade, OT - Otimização.

Outra aplicação decorre de sistemas em possuem um elevado número de parâmetros de entrada, o que torna difícil para um operador humano determinar o completo conjunto de regras de descrição do sistema. Pode-se aplicar redes neurônais, pois essas no exigem tais procedimentos, uma vez que apenas criam relações entre pares entra/sada (Fuk94). Combinações de sistemas *fuzzy* com redes neurônais estão sendo desenvolvidas. Redes *fuzzy* neurônais possuem o seu conhecimento estruturado e fornecido por especialistas através de funções de pertinência, ao mesmo tempo em que essas funções são modificadas por processos de aprendizado (Koz94). Algoritmos genéticos estão sendo utilizados para otimizar topologías de redes neurais, propiciando a estas possibilidades de evoluo (conformar discutido anteriormente). Também a capacidade de manipulação de símbolos apresentada pelos algoritmos genéticos pode produzir novas regras ou conhecimento para sistemas especialistas.

2.4.3 Técnicas por Visão Computacional

Correspondência de imagens é fundamental em diversos problemas de visão computacional como reconhecimento de objetos, reconhecimento de cenas, montagem automática de mosaicos, obtenção da estrutura 3D de múltiplas imagens, correspondência estéreo e perseguição de movimentos. Uma abordagem para se trabalhar com correspondência de imagens é se usar descritores locais para se representar uma imagem. Descritores são vetores de características de uma imagem ou de determinadas regiões de uma imagem e podem ser usados para se comparar regiões em imagens diferentes. Este vetor de características é normalmente formado em imagens diferentes. Este vetor de características é normalmente formado por descritores locais ou globais. Descritores locais computados em pontos de interesse provaram ser bem sucedido em aplicações como correspondência e reconhecimento de imagens (Mik03).

Descritores são distintos, robustos à oclusão e não requerem segmentação. Existem diversas técnicas para se descrever regiões locais em uma imagem (Mik03). O mais simples descritor é um vetor com as intensidades dos pixels da imagem. A medida de correlação cruzada pode ser então usada para computar a similaridade entre duas regiões. Porém, a alta dimensionalidade de tal descritor aumenta a complexidade computacional da comparação. Então, esta técnica é principalmente usada para se encontrar correspondências ponto a ponto entre duas imagens. A vizinhança de um ponto também pode ser escalada de modo a reduzir sua dimensão. Outro descritor simples é a distribuição de intensidades de uma região representada por seu histograma.

Trabalhos recentes têm se concentrado em fazer descritores invariáveis a transformações nas imagens. Mikolajczyk e Schimid (Mik04) propuseram um detector de pontos de interesse invariável a transformações afins através da combinação de um detector invariável à escala e da técnica *"second moment of Harris corners"* (Har88). Ling e Jacobs (lin05) propuseram um sistema para a construção de descritores de intensidade locais invariáveis a deformações em geral. Lowe (Low99) propôs uma maneira rápida e eficiente de computar características invariáveis a transformações em escala, que medem a distribuição do gradiente em regiões detectados invariáveis à escala.

Transformação *SIFT*

SIFT (*Scale Invariant Feature Transform*) é uma técnica de processamento de imagens que permite a detecção e extração de descritores locais, favoravelmente invariáveis a mudanças de iluminação, ruído de imagem, rotação, escala e pequenas mudanças de perspectiva. Estes descritores podem ser utilizados para se fazer a correspondência de diferentes visões de um objeto ou cena. Descritores obtidos com a técnica *SIFT* são altamente distintos, ou seja, um determinado ponto pode ser corretamente encontrado com alta probabilidade em um banco de dados extenso com descritores para diversas imagens. Um aspecto importante da técnica *SIFT* é a geração de um número grande de descritores que conseguem cobrir densamente uma imagem quanto a escalas e localizações. A quantidade de descritores é particularmente importante para o reconhecimento de objeto, onde a capacidade de se encontrar pequenos objetos em ambientes desordenados requer ao menos 3 pontos encontrados em comum para uma identificação confiável. A obtenção de descritores *SIFT* é feita através das seguintes etapas:

– Detecção de extremos: Nesta primeira etapa é feita procura para todas escalas e localizações de uma imagem. Isto é feito utilizando-se a difer-

ença de filtros gaussianos de modo a se identificar pontos de interesse invariáveis à escala e rotação.

– Localização de pontos chave: Para cada localização em que foi detectado um extremo, um modelo detalhado é ajustado de modo a se determinar localização e escala. Pontos chaves, ou pontos de interesse, são então selecionados baseando-se em medidas de estabilidade.

– Definição de orientação: É definida a orientação de cada ponto chave através dos gradientes locais da imagem. Toda operação a partir de então será feita com relação a dados da imagem transformados em relação à orientação, escala e localização de cada ponto chave. Desta maneira se obtém invariância a estas transformações

– Descritor dos pontos chaves: Nesta etapa é feita a construção dos descritores ao se medir Gradientes locais em uma região vizinha a cada ponto de interesse. Estas medidas são então transformadas para uma representação que permite níveis significativos de distorção e mudança na iluminação.

Em tarefas de comparação de imagens e reconhecimento, descritores *SIFT* são extraídos das imagens para então poderem ser comparados.

Transformada Hough

A transformada *hough* (**?**) é uma técnica de extração de características chaves numa imagem, sim embargo pode ser extensiva para a identificação de posições de formas diversas. A transformada *hough* pode ser entendida de modo genérico como uma tabela de parâmetros que descrevem um modelo, a tabela seria preenchida para cada dado de um conjunto de dados apresentados, encontrando todos os modelos possíveis que coincidissem com cada ponto e atualizando a tabela, aumentando às células referentes aos possíveis parâmetros dos modelos encontrados.

A idéia básica da transformada *hough* se ilustra na Figura 2.9, onde uma linha é um conjunto de pontos (u, v) tal que 2-30:

$$u \times \cos\theta + v \times \sin\theta = d \tag{2-30}$$

Para algum ponto (u, v), existe uma família de parâmetros de linhas através deste ponto, dado pela equação 2-30. Cada ponto obtém um voto por cada linha na família; se há uma linha que tem muitos votos, então isso deveria ser a linha que passa através dos pontos. Com esta técnica é possível encontrar nas imagens objetos como linhas, círculos o elipses.

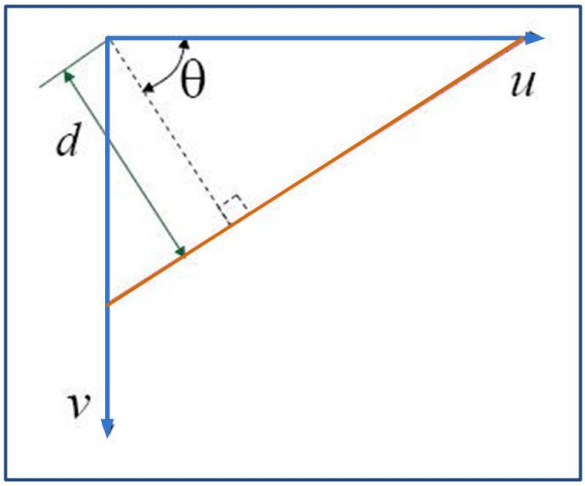

Figura 2.9: Idéia básica da Transformada Hough

Transformada *RANSAC*

O algoritmo *RANSAC* (*Random Sample Consensus Algorithm*), é apresentado em (Fis81)(Lac00), como um método para estimar os parâmetros de um modelo para um conjunto de dados conhecidos mas com presença de diversos dados errôneos.

O algoritmo não é aplicado independentemente porque seu eficiência é limitada quando há um grande número de dados, não conseguindo uma boa estimação do modelo. Por isso é sugerido que se aplique anteriormente Hough, o outra técnica conseguindo gerar um conjunto mais robusto que o inicial.

RANSAC é um algoritmo bem simples definido como segue. Dado um modelo com parâmetros $\vec{x}$ se deseja estimar-os. Para tal, é assumido:

- Os parâmetros podem ser estimados a partir de um número N de ítens num conjunto de dados conhecidos.
- A probabilidade de um dado selecionado aleatoriamente para ser parte de um bom modelo é dada por p_g.
- A probabilidade de que o algoritmo termine sim que se encontre um bom modelo é dada por p_{falla}.

O algoritmo é então executado através das seguintes etapas:e

1. N ítens são escolhidos de modo aleatório.
2. A partir dos ítens escolhidos, $\vec{x}$ é estimado.
3. Logo encontra-se o número de ítens que encaixam ao modelo para determinada tolerância especificada. Este número é chamado de K.

4. Caso K seja grande ou suficiente, para um limite escolhido, o algoritmo termina com sucesso.

5. O algoritmo é repetido de 1 a 4 um número L de vezes.

6. Caso o Algoritmo não tenha terminado depois de L tentativas, o algoritmo falho.

2.5 Planejamento da Trajetória

O planejamento da trajetória *path planning*, já seja global ou local, consiste em encontrar uma rota segura capaz de levar ao veículo desde a posição atual até a especificada do destino. O conceito de rota segura implica o cálculo de um caminho ao menos continuo na posição, que seja livre de obstáculos. Em virtude desta rota, o gerador constituirá as referencias que entregam-se ao controle de movimento.

Existem vários métodos de planificação, todos eles se fundamentam em uma primeira fase de construção de algum tipo de grafo sobre o espaço livre, segundo a informação adquirida do entorno, para posteriormente usar um algoritmo de busca em grafos que encontre o caminho ótimo segundo à função de custo, entre eles temos os seguintes:

- Planificação baseada em grafos de visibilidade.
- Planificação baseada em diagramas de Voronoi.
- Planificação baseada em modelado do espaço livre.
- Planificação baseada na descomposição de células.
- Planificação baseada em campos potenciais.

A fundamentação teoria apresentada neste capitulo será utilizada na proposta de uma metodologia para auto localização e mapeamento de um robô móvel. no próximo capitulo, o ambiente de desenvolvimento é descrito, incluindo o robô móvel utilizado e seus sensores e atuadores.

3
O ambiente de desenvolvimento

3.1
Introdução

A idéia destes algoritmos surgiram de tratar de assemelhar ao robô com a pessoa humana; o robô conta com uma câmera que assemelha ao olho humano, os descritores SIFT assemelha à percepção do olho, as redes neurais ao aprendizagem do cérebro, a lógica fuzzy a toma de decisão da pessoa e algoritmos genéticos a idéia de otimizar a menor trajetória entre dois pontos, como se ilustra na figura Figura 3.1.

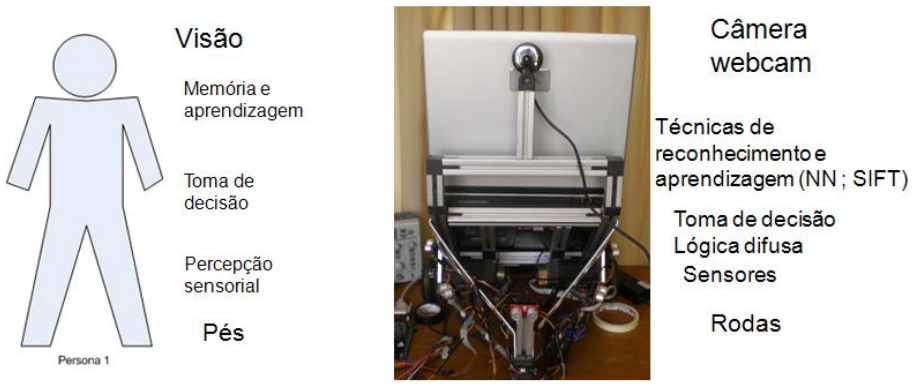

Figura 3.1: Semelhança robô e a pessoa humana

O robô ER1 é usado pela facilidade da estrutura mecânica y a facilidade de modificar nele outro mecanismo de mobilidade para facilitar nosso trabalho.

O uso do DSP e a notebook é para o seguinte: o DSP vai gerar a sinais de controle do motor usando o processamento embebido da lógica fuzzy além disso também faz a leitura dos sensores de proximidade para treina os na rede neural. Para a aquisição das imagens o sistema usa uma camera e para o processamento a notebook, devido à dificuldade do processamento o DSP não pode processar imagens, então a notebook ajuda ao DSP no processamento de imagens para ter informação dele e poder comunicar ao DSP para uma correta navegação.

Os sistemas de controle desenvolvidos neste trabalho visam ao tratamento dos dados provindos dos sensores, efetuando-se a navegação no ambiente desconhecido, O sistema de controle dever evitar que o veículo colida com os obstáculos, ao mesmo tempo que procura nós para poder mapear o ambiente desconhecido. Logo disso o robô tem que ser capaz de reconhecer cada nó do ambiente já explorado e poder navegar para qualquer ponto dele.

Para que o robô tenha sucesso com a tarefa de exploração é preciso que o robô tenha sucesso com a etapa da percepção sensorial e com a etapa de controle dos atuadores(motores) para isso se fiz um projeto mecânico, um projeto eletrônico, e o desenho do software.

3.2 Projeto mecânico do robô

3.2.1 Robô ER1

O robô móvel utilizado neste projeto foi construído pela empresa *Evolution Robotics*, e é por isso que se denomina $ER1$. O ER1 se locomove por direção diferencial tipo torque através de duas rodas ativas acionadas por motores de passo. Este robô foi extensivamente modificado neste trabalho para poder ter maior capacidade de mobilidade, trocando os motores de passo pelos motores *brushless* que são melhores no tempo de resposta e pode fazer movimentos muito mais precisos; além disso também o sistema de controle foi trocado por um processador digital de sinais (DSP) que com a ajuda de um *notebook* LG vão fazer as tarefas de controle usando algoritmos de navegação;o robô foi implementado com 11 sensores infravermelhos , 6 sensores ultra-sônicos e uma câmera como se mostra na Figura 3.2

O robô $ER1$ é composto de:

- Chassi: Vigas de perfil x de alumínio, conectores de plástico, porcas e placas laterais em alumínio.
- Mecanismo pré-montado sobre placas de alumínio que serve de suporte dos motores, tem dois rodas de 4 polegadas de diâmetro, correias e polias e também contém um rodízio de 360º.
- Bateria recarregável de $12V$ e $5,4A-h$ com fusível interno para proteção.

Figura 3.2: Robô ER1 modificado no momento de testes

3.2.2 Motores brushless

O robô foi implementado com dois motores *brushless DC (BLDC)* tipo $5143DO12$ da séries 5100 da família *Pittman-Elcom* como se ilustra na Figura 3.3; os motores BLDC são um dos tipos de motores que esta ganhando popularidade nas aplicações industriais pelas vantagens sobre os motores DC, como o nome deles diz, os motores BLDC não usam escovinha para sua comutação, no seu lugar usa comutação eletrônica; este motor tem as seguintes vantagens respeito aos motores DC com escovinha e os motores de indução:

- Melhor característica de velocidade versus torque.
- Alta resposta dinâmica.
- Alta eficiência.
- Largo tempo de vida.
- Operação sim ruídos.
- Altos ranges de velocidade.

A operação do motor *BLDC* é uma seqüencia de comutação numa das bobinas com voltagem positivo a segunda bobina é negativa e a terceira bobina não tem voltagem. O torque é produzido porque há interação entre o campo magnético gerado pelo estator e o ímã permanente. Idealmente o

Figura 3.3: Motor Brushless DC

torque mais alto ocorre quando estes dois campos estão a 90º um do outro e cai quando se movem juntos. Em disposição de manter o motor funcionando, o campo magnético produzido pelas bobinas deveriam trocar de posição. Como o movimento do rotor é capturado com o campo do estator, isto leva a definir uma seqüencia de comutação de 6 passos para ter voltagem nas bobinas, isto também leva a determinar a posição exata das bobinas no momento de giro, para isso o motor *brushless* esta implementado com sensores de efeito hall como se ilustra na Figura 3.4, para determinar as fases corretas e a forma de energiza-o seqüencialmente.

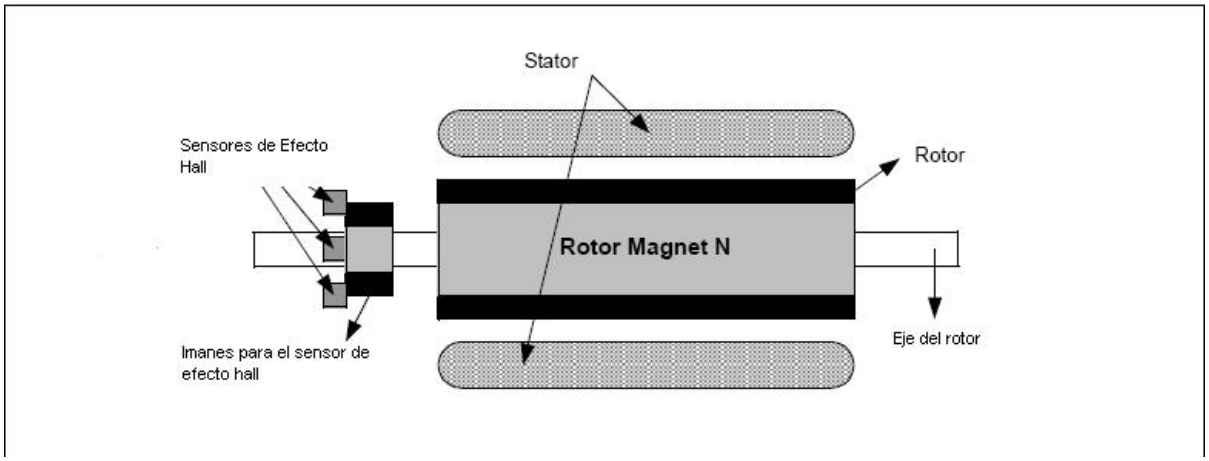

Figura 3.4: Vista transversal do motor brushless

3.2.3 Sensores

O robô $ER1$ foi implementado por 6 sensores ultra-som e 11 sensores infravermelhos os quais foram colocados simetricamente na estrutura do robô.

Sensor ultra-som

O sensor ultra-som que foi implementado no robô é o $SRF08$ (Figura 3.5), que tem um alcance de $3cm$ até $6m$ e a interface de comunicação é mediante o protocolo de comunicação IIC (*Inter-Integrated Circuit*); o $SRF08$ registra 17 subsequentes pulsos de ecos que correspondem ao objeto mais perto isto permite fazer uma medida mais acertada da distância ao objeto.

Figura 3.5: Sensor de Ultra-Som SRF08

A leitura destes sensores tem dois tipos de modalidades, a primeira é que o sensor pode armazenar o resultado da medida em cm, s ou $polegadas$ e a segunda modalidade é que pode armazenar o resultado da medida num registro de 32 bytes para o fácil uso se fosse o caso de precisar utilizar uma rede neuronal e ter como entrada a leitura do sensor; onde cada byte representa o espaço de $352mm$, é dizer se um objeto se encontra a $3m$ então o registro 8 seria ativado a 1 lógico e assim as entradas à rede neuronal estaria pronto para o treinamento.

Sensor infravermelho

O sensor que foi implementado é o $GP2D15$ da família $sharp$(Figura 3.6), este sensor tem um alcance de $10cm$ a $80cm$ e produz um voltagem proporcional à distância do objeto mas neste projeto ele é usado como um detector digital a uma distância de $25cm$, é dizer se existe um objeto a uma distância menor a $25cm$ o detector dará informação de 1 lógico.

Figura 3.6: Sensor Infravermelho GP2D15 da família Sharp

Encoder

O encoder utilizado para a estimar a velocidade dos motores é o Encoder $HEDL5540$ (Figura 3.7) de $500ppv$ com três canais e com um driver $RS422$ como se ilustra na Figura 3.8.

Figura 3.7: O Encoder HEDL5540

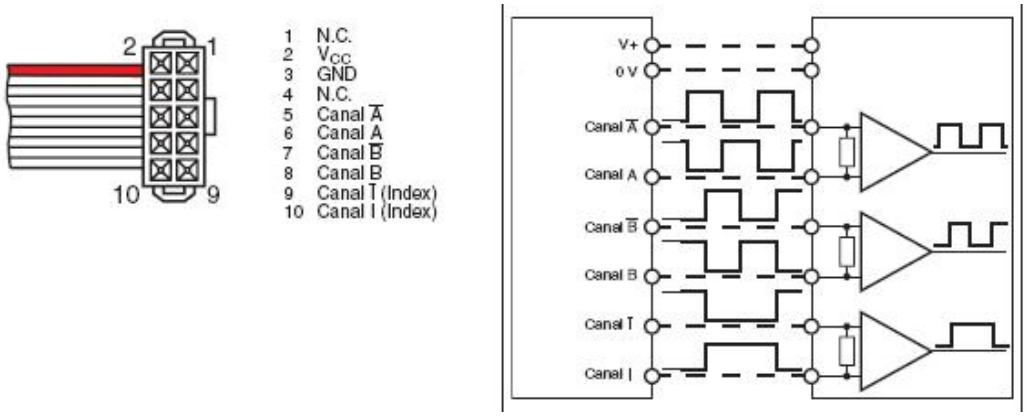

Figura 3.8: conexão e pulsos do encoder

Câmera

A câmera *webcam* utilizada é uma tipo *USB* da família *Clone* tão como se ilustra na Figura 3.9, ele tem uma resolução de 80×60 até 640×800 e é de 4 Megapixels.

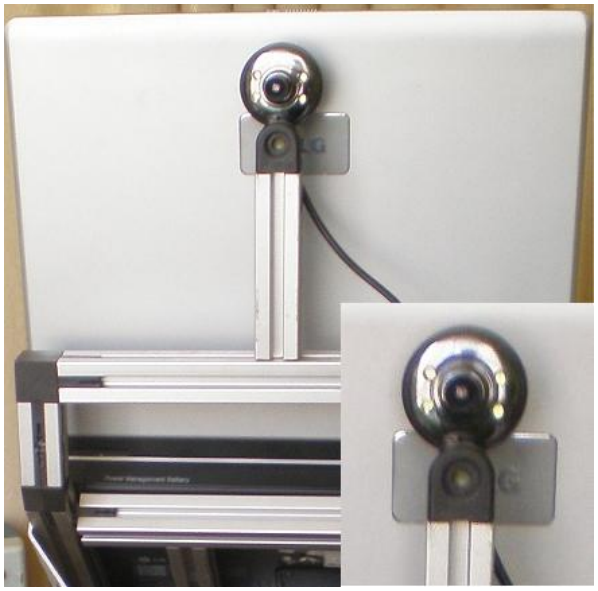

Figura 3.9: Câmera digital USB Clone

3.3 Projeto eletrônico do robô

No desenho eletrônico tivesse presente que o DSP é um processador de baixa potência, é dizer que ele trabalha a um voltagem de $3,3 voltios$, é por isso que a interface eletrônica com os outros dispositivos que trabalham a $5 voltios$ foi preciso.

Microprocessador TMS320F2812

O DSP (*Digital Signal Procesor*) é o dispositivo de controle e junto com o *notebook* é o cérebro do robô, é um processador de *Texas Instruments*, uma das maiores empresas fabricantes de ultimas tecnologias.

Um *DSP* é um processador de propósito particular (dedicado), com características especiais a nível estrutural, que permitem maximizar seu rendimento em términos de capacidades de memória e velocidade do processo; na Figura 3.10 se ilustra a arquitectura do *DSP*; atualmente este dispositivo é utilizado na maioria das tecnologias existentes no mundo, como são em sensores inteligentes, automóveis, aeronáutica, equipamentos médicos, etc.

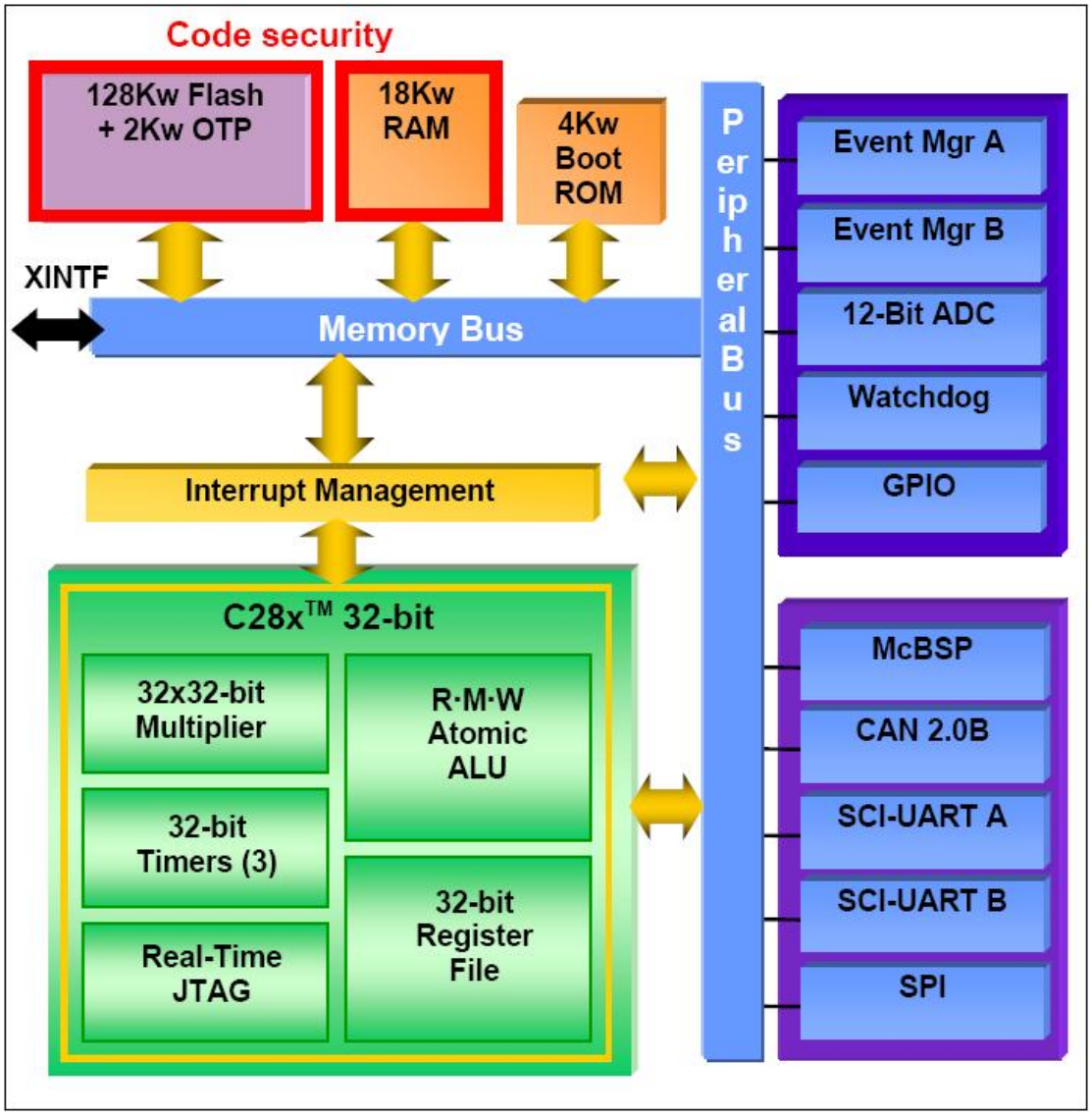

Figura 3.10: Arquitectura do TMS320F2812

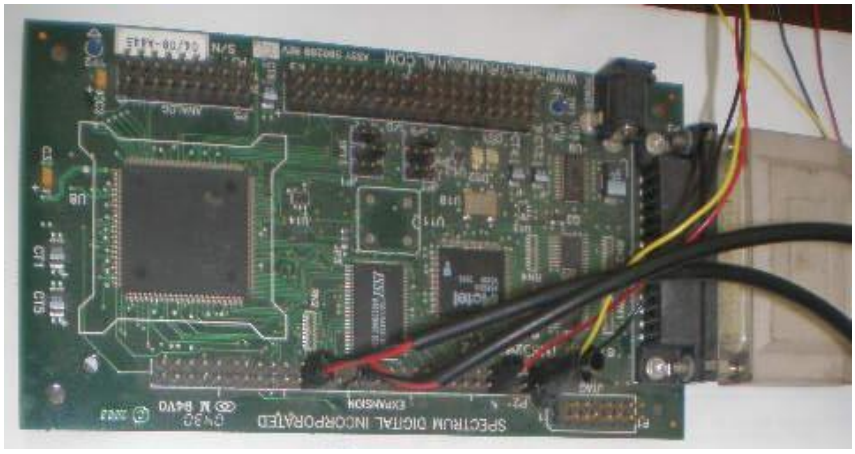

Figura 3.11: Processador de Sinais DSP TMS320F2812

El *DSP* $TMS320F2812$(Figura 3.11), do que pode se ver uma completa descrição em (sprs174), é um dispositivo mais potente da família $C2000$ de *Texas Instruments* dos todos os *DSPs* orientados ao controle de processos. Incorpora um núcleo de arquitetura Harvard de 32 bits de coma fixa a 150 MHz, capaz de executar funções MAC (multiplicar e acumular) num só ciclo de trabalho. Além disso, incorpora todos os periféricos necessários para o controle de motores: geradores de sinais PWM(*Pulse Width Modulation*) com tempo morto, conversores A/D (*Analogic to Digital Converter*), capturadores para o encoder, além disso tem algumas funções adicionais que ajudam no controle de processos, como se pode ver na Figura 3.10.

O *DSP* também incorpora memória *Flash* para o código e memória *RAM* para as variáveis. Também incorpora uma pequena memória ROM (*Read Only Memory*), onde se albergam os modos de arranque e umas tabelas para o cálculo matemático de algumas funções mediante interpolação numérica.

Como periféricos de comunicação incorpora controladores para distintos standards de comunicação, como pode ser o *UART Universal Asynchronous Receive and Transmit*), o protocolo *CAN* (*Controller Area Network*) e *McBSP* (*Multichannel Buffered Serial Port*). Neste dispositivo concreto (TMS320F2812), o bus de endereço e de dados, é acessível, com o que se pode ampliar seus recursos.

Os periféricos que interessam mais para este trabalho são três: os geradores de sinais *PWM*, os conversores A/D e os capturadores para o encoder, todos eles estão englobados no que se chama *Event Manager*(EV). O $TMS320F2812$ dispõe de dois EV, o A e o B. Cada um deles funciona como uma unidade independente capaz de gerar seus interrupções e administrar todos seus recursos. Tanto o A como o B funcionam da mesma forma, com o que com um só *DSP* é possível controlar dois motores *brushless* ao mesmo tempo. O EV dispõe de dois *Timers* de 16 bits com o que se gera as bases do tempo tanto para os registros de comparação como para o capturador da sinal do encoder. Três registros de entrada (CMPR1, CMPR2, CMPR3) geram as seis sinais complementarias dois a dois e se agrega um tempo morto totalmente programável desde $0us$ até $12us$.

Além, o *DSP* dispõe de uma sinal de entrada de falha do conversor que automaticamente e sim uso da *CPU* (*Central Process Unit*), pôr em alta impedância as seis saídas *PWM* e gera uma interrupção de software, que permite administrar a falha de forma rápida e segura para o entorno.

O capturador para o encoder funciona de forma muito fácil; ele se limita a medir o tempo entre dois pulsos consecutivos provenientes do encoder. Dispõe de uma lógica interna para decodificar o sentido de giro.

O conversor A/D é de 12 bits e dispõe de dois S/H (*Sample and Hold*; a entrada analógica tem um margem dinâmico de 0.0 V a 3.0 V, e é capaz de operar a $12,5$ *MSPS* (*Mega Samples Per Second*), velocidade suficiente para aplicações de controle de processos industriais. Tem vários modos de funcionamento e de sincronização com outros periféricos, em especial os *EV*, com o qual pode se fazer o amostragem das sinais no instante adequado sem necessidade de usar a *CPU*.

Circuito de potência dos motores

O projeto se fez tendo em conta as características de resposta em freqüência do robô e do controle, também se tomou em conta que era preciso ter uma interface entre os terminais do *PWM* que trabalha a $3,3V$ com os drivers dos *mosfet*, os *mosfet* são a parte importante do circuito de potência, eles são os que trabalham diretamente com o motor, e o tempo de resposta deles tem que ser muito maior que a velocidade de trabalho dos PWMs para ter um ótimo controle, na Figura 3.12 se ilustra o projeto feito no software *Orcad Capture*.

Interface dos sensores

Com o fim de reduzir o consumo da potência dos dispositivos na atualidade os dispositivos tem um menor consumo de energia, é por isso que os atuais dispositivos *DSP* são alimentados por uma fonte de voltagem de $3,3$ vóltios, isto é compatível com componentes e circuitos integrados (ICs) no mercado. Neste caso do *DSP* as entradas e saídas lógicas do controlador do motor trabalha a $3,3V$ então temos que ter em conta para o projeto da interface dos sensores com o *DSP* os seguintes casos (spra550):

A interface do sensor infravermelho é como se ilustra na Figura 3.13, o sensor quando percebe um objeto desativa o *gate* do *mosfet* então o *mosfet* deixa de conduzir corrente e a entrada do DSP é $3,3V$.

A interface com o sensor ultra-som é como se ilustra na Figura 3.14, o projeto é desse jeito porque o $TMS320F2812$ não tem o módulo de comunicação $I2C$ mas tem outro módulo com o qual pode se fazer o protocolo de comunicação $I2C$.

A interface com o encoder é como se ilustra na Figura 3.15, neste caso se fez uma interface para que o pulso de $5V$ seja de $3,3V$ à entrada do *DSP*.

3.4

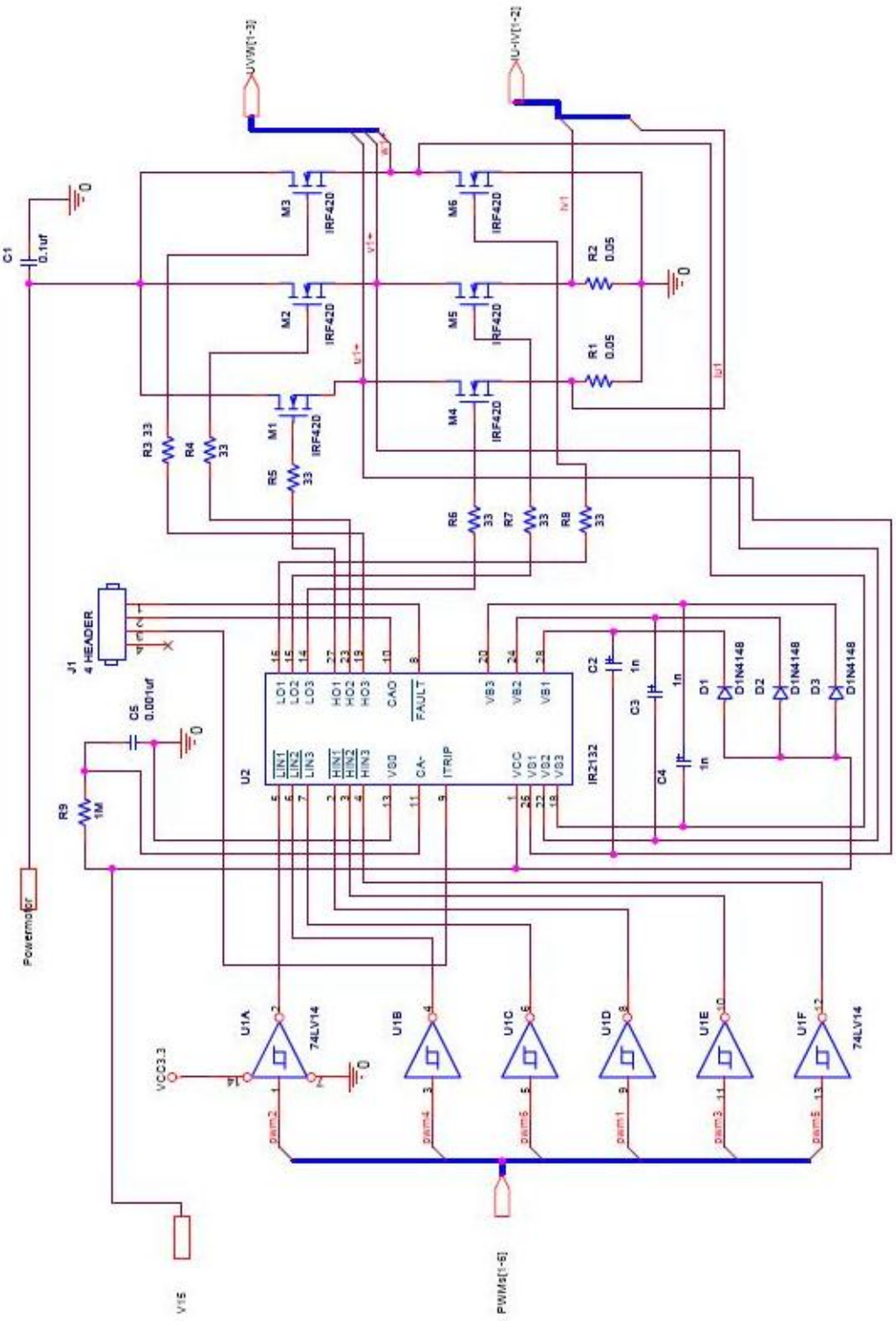

Figura 3.12: Circuito de potência do motor Brushless

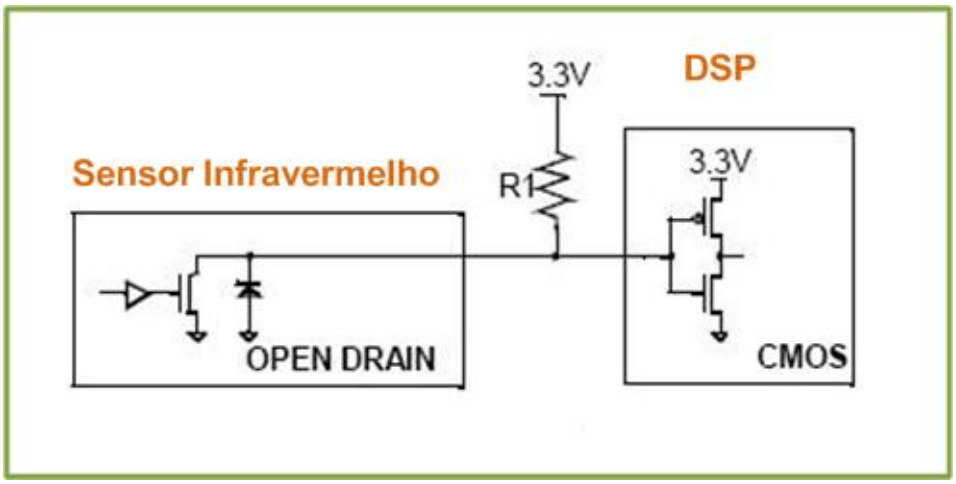

Figura 3.13: Interface do sensor infravermelho com o DSP

Projeto do software

Neste projeto se tem o desenvolvimento de duas etapas, o primeiro a etapa de simulação e o segundo a etapa de controle do robô que é o trabalho de dois software que estão em comunicação constante, a descrição de cada um deles se da a continuação:

3.4.1 Software de simulação

O software que se utilizou para a simulação foi o *Player-Stage* que é um simulador *open source* da plataforma *Linux*, que possibilitam simulações em $2D$ para robôs móveis com seus respectivos sensores e drivers comerciais já implementados no software tais como laser, sonares, gps, etc.; Possuem módulos para construção e localização de mapas *cross compiling*, etc.

O linguagem que se utiliza neste programa o C++ um linguagem muito fácil de usar, na Figura 3.16 se mostra um exemplo do ambiente de simulação com o *player-stage*.

3.4.2 Software de controle

O software de controle foi implementado com o uso de dois programas, eles se encarregam de todo o processo de exploração do robô, como descrito a seguir.

Code Composer Studio

O software que proporciona o fabricante para interatuar com o *DSP* é o *Code Composer Studio*(CCS), o mesmo programa serve para todas as famílias

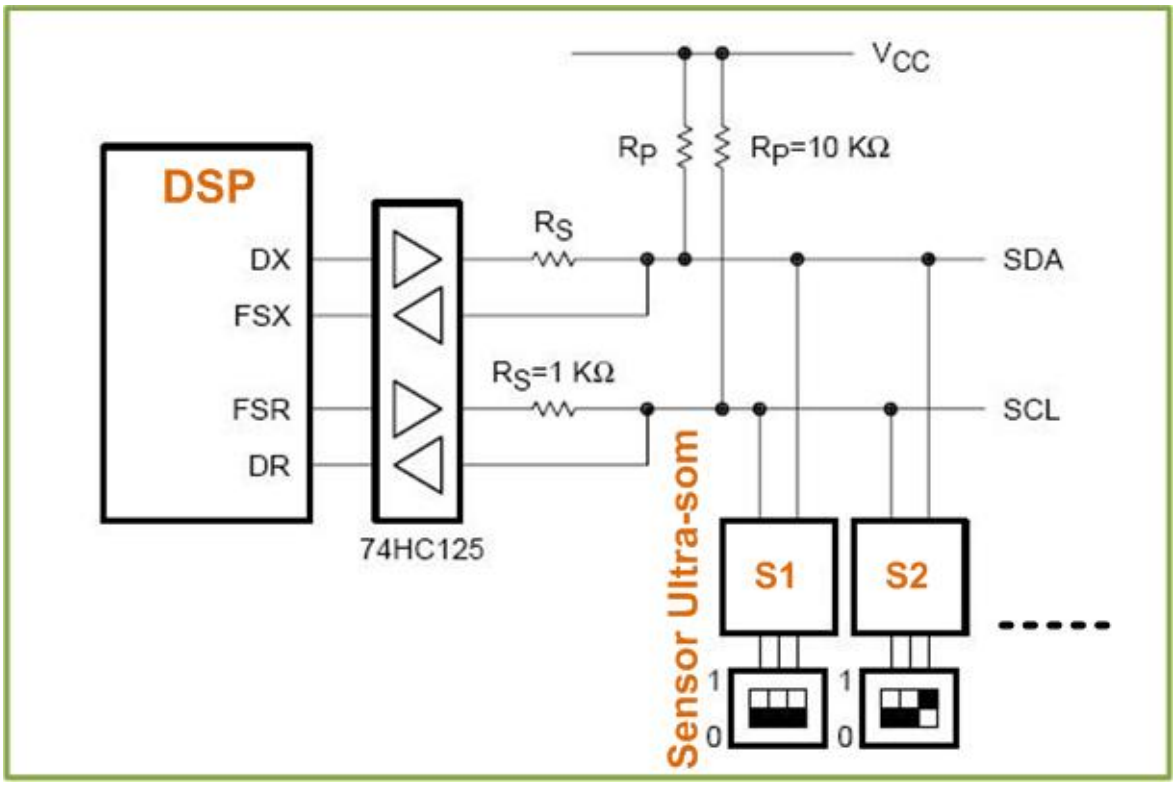

Figura 3.14: Interface do Sensor Ultra-som com o DSP

de *Texas Instruments*. O CCS forma o que se chama um sistema integrado de desarrolho, já que desde ele pode se escrever, compilar e carregar o código ao *DSP* em linguagem C e em Assembler; além disso é uma potente ferramenta para o análises e o depurado do código em tempo real mediante pontos de detecção (*breakpoints* ou interrupções). Permite também visualizar e modificar variáveis internas em tempo real sem deter a *CPU*.

As funções permitidas pelo entorno tem como objetivo facilitar a sintonização dos valores influentes no controle e o ajuste dos valores dos *PID* (controlador proporcional, integral e derivativo) em tempo real da forma mais eficaz. Na Figura 3.17 se ilustra uma imagem da janela principal do programa *CCS*, a qual à vez está dividida em diferentes janelas que amostram a informação necessária para um perfeito seguimento do controle. Como pode se notar apresenta varias janelas, algumas onde se desarrolha o programa e outras onde se amostra o código assembler e também o valor das variáveis e seu respectiva gráfica em tempo de execução.

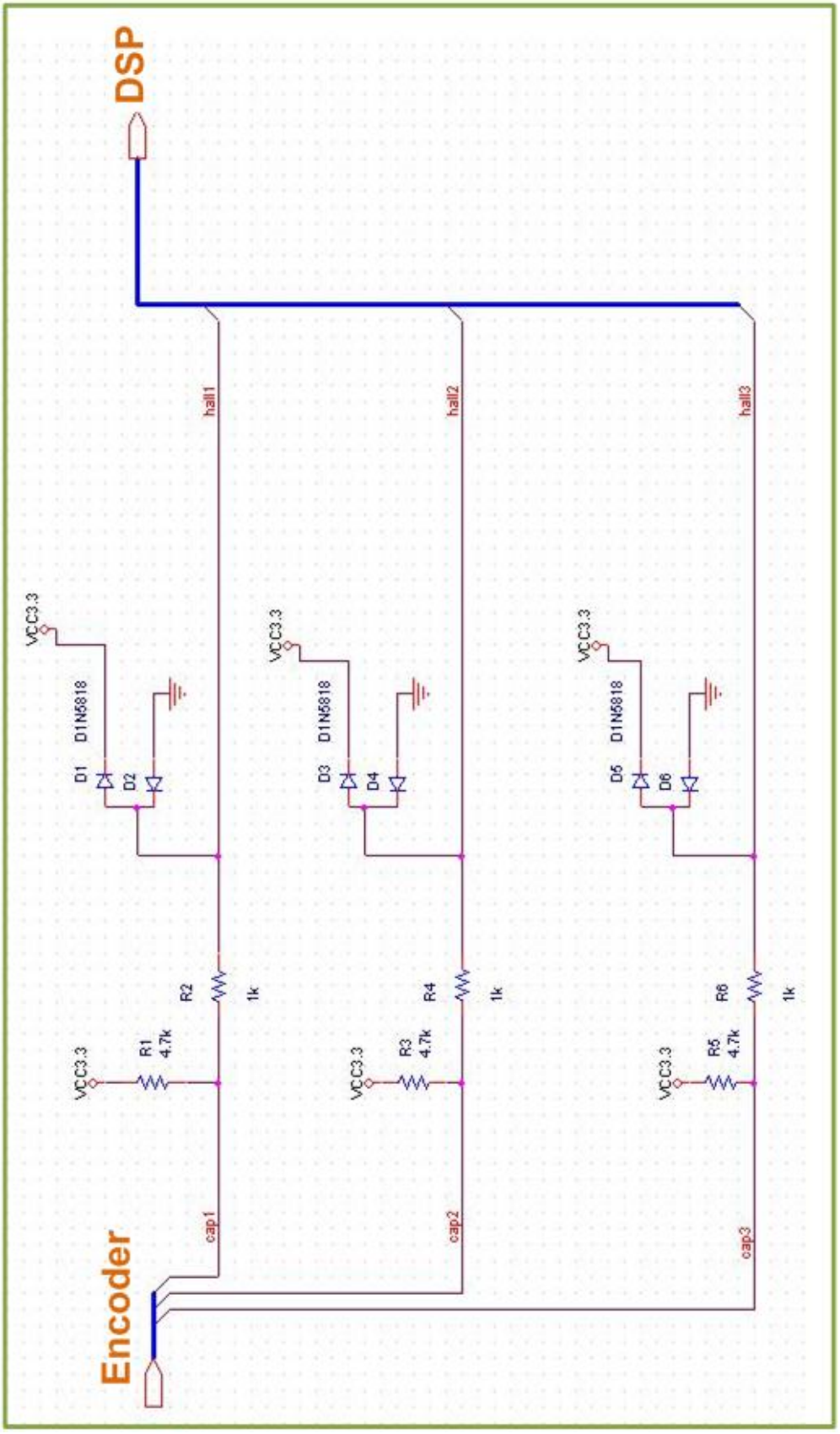

Figura 3.15: Interface do Encoder com o DSP

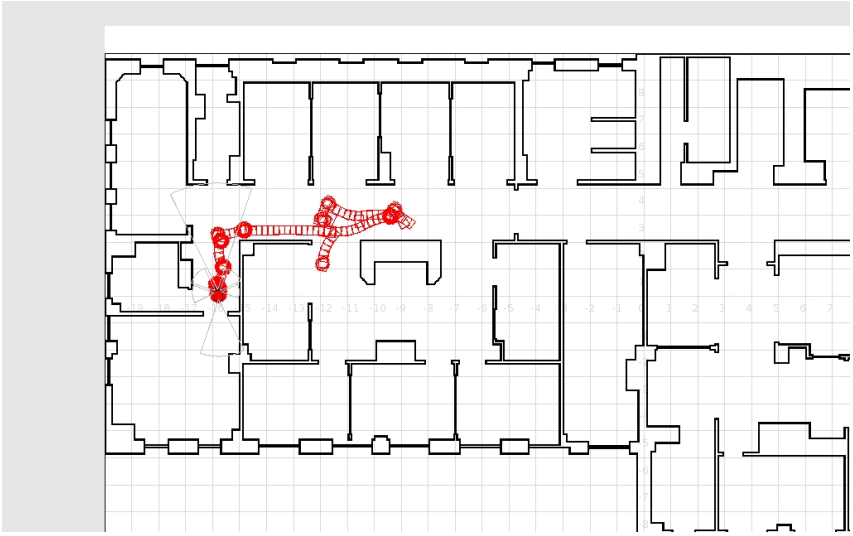

Figura 3.16: Ambiente de simulação do player-stage em 2-D

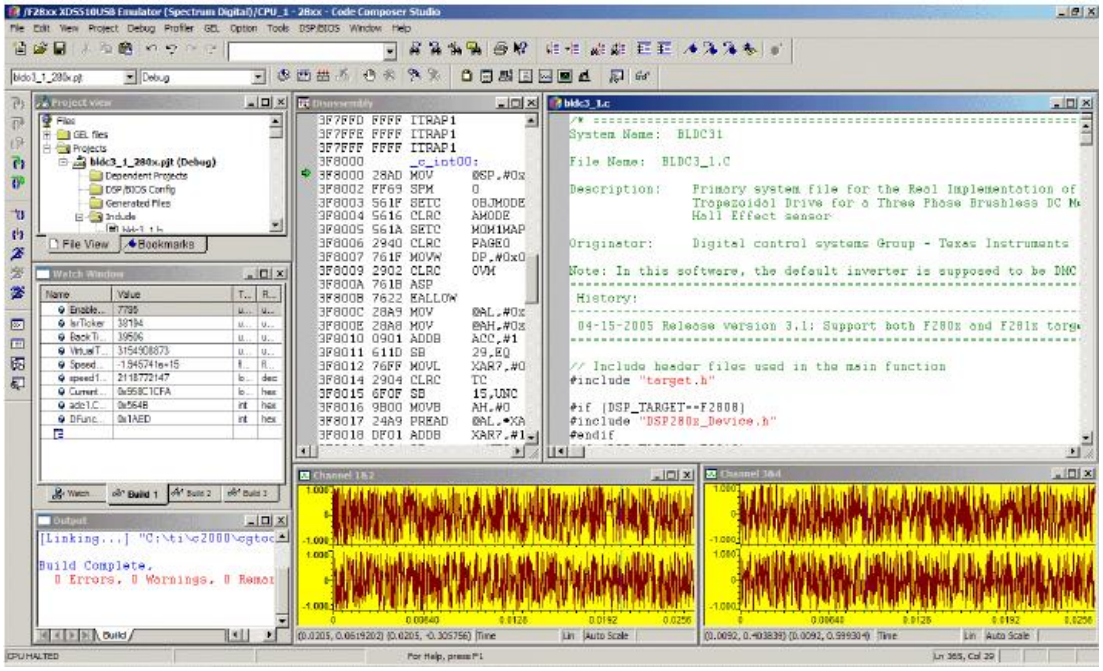

Figura 3.17: Aspecto da janela do Code Composer Studio

Matlab

Matlab é um software muito conhecido no mundo científico, e tem em suas livrarias um módulo que faz a interface entre o *DSP* em tempo real e o Matlab, isto é de grande ajuda porque neste trabalho a parte da comunicação entre o matlab e o *CCS* é importante para o desarrolho ótimo desde trabalho, o tutorial para aprender as funções pode ser encontrado escrevendo no promp do matlab o comando ccstutorial.

Controle do motor Brushless

Nesta seção vai se explicar qual é o tipo de técnica de controle utilizado e desarrolhado para o controle do motor *brushless*; existem várias técnicas de controle para este tipo de motores um dos mais simples é o controle baseado em comutação trapezoidal (Ham04), e que foi implementado para este trabalho mas posteriormente devido as deficiências de exatidão do controle foi trocado pelo controle vetorial o qual é o controle mais complexo e que requer maior potência de cálculo mas é a que melhor controle proporciona.

O controle vetorial ou *Field Oriented Control (FOC)* controla o vetor de correntes diretamente no espaço de referência ortogonal e rotacional, chamado espaço D-Q *(Direct-Quadrature)*. Dito espaço de referência está normalmente alinhado com o rotor de forma que permite que o controle do fluxo e o par do motor se realize de forma independente. A componente direta permite controlar o fluxo e a componente em quadratura o par.

Para poder realizar este controle é necessário transformar matematicamente as medidas das três correntes referidas ao espaço estático das bobinas do motor ao espaço rotacional D-Q. Embora esta transformação pode implementar se em um só passo, mas para o melhor entendimento se faz em dois transformações; a transformada de Clarke e a transformada de Park (Ham04).

O diagrama de controle se ilustra na figura 3.18, onde cada bloco representa programas feitos no linguagem C.

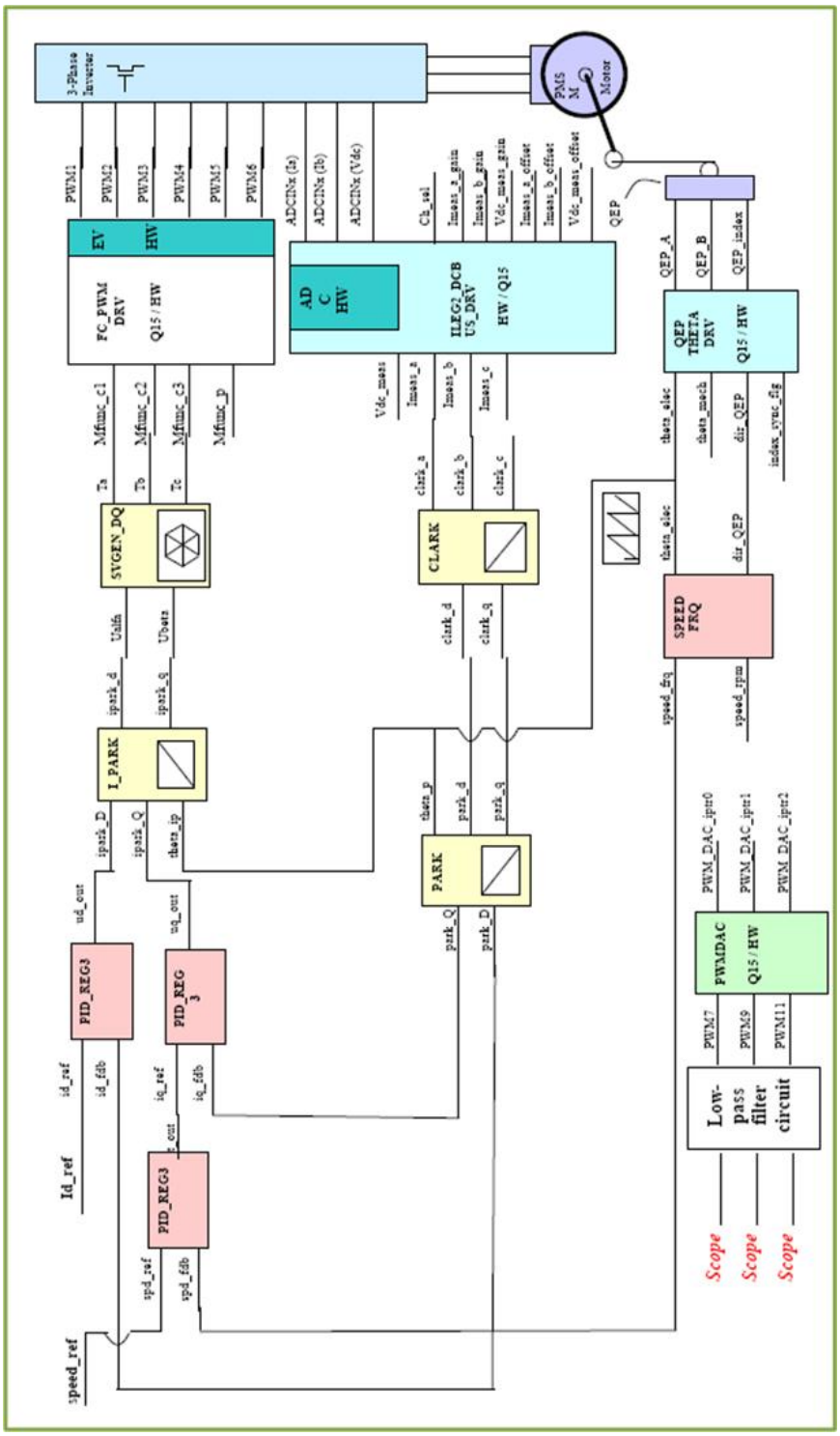

Figura 3.18: Diagrama de blocos do controle vetorial de velocidade para o robô

No próximo capitulo, a metodologia para auto localização e mapeamento do ambiente pelo robô é descrita.

4
Desenvolvimento de Algoritmos de controle

4.1
Introdução

Os estudos das características sensoriais e equações de movimento realizados com o robô ER1, junto com as ferramentas de simulação, permitem o estudo e elaboração de algoritmos de controle de robôs móveis. Este capítulo inicialmente define o problema a ser tratado. São definidas as tarefas que o robô deve realizar e os motivos de tal escolha. O ambiente no qual está inserido também é determinado. Depois disso, são abordados os algoritmos e as técnicas são divididas em três etapas: etapa de exploração do ambiente desconhecido, etapa de navegação num ambiente já conhecido e a etapa de otimização da trajetória.

4.2
Definição do problema

O objetivo dos sistemas de controle desenvolvidos é fazer que o robô navegue num ambiente desconhecido, aprenda certos pontos do ambiente que ele considere importante (*landmarks*, que saiba como navegar através desses pontos já conhecidos e que possa depois navegar de um ponto a outro ponto otimizando a trajetória percorrida desviando obstáculos; para isso foram desenvolvidos algumas técnicas que a continuação se explicam.

4.3
Percepção Geral

As seguintes considerações são baseadas no robô móvel com três graus de liberdade num movimento planar como se amostra na Figura 4.2. O robô é equipado com um anilho de 6 sensores ultra-sônicos, e outro anilho de 11 sensores infravermelhos (Figura 4.1), o anilho de sensores infravermelhos são usados como detectores digitais a uma distância de $25cm$ para cobrir as zonas onde os sensores ultra-sônicos não podem perceber nenhuma informação perto do robô; O anilho de sensores ultra-sônicos vai se considerar como se ilustra na figura 4.2.

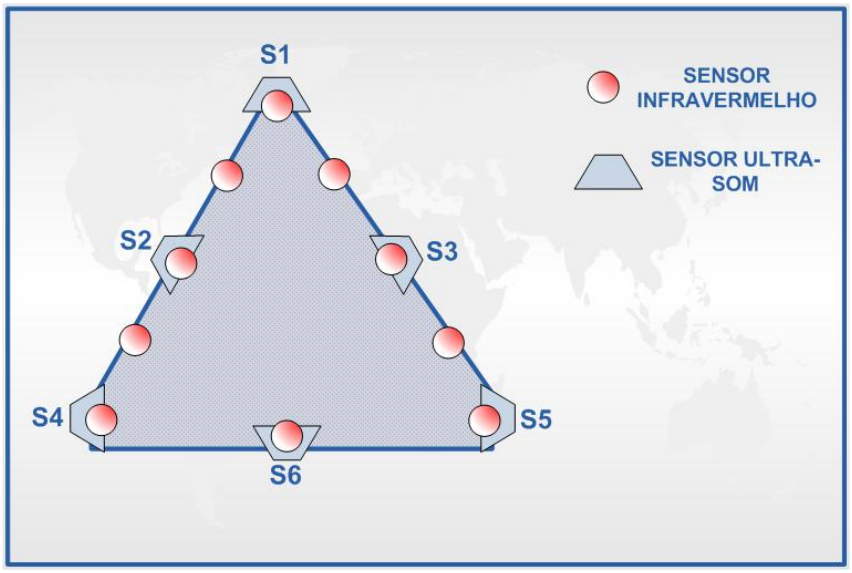

Figura 4.1: Posição dos sensores no robô ER1

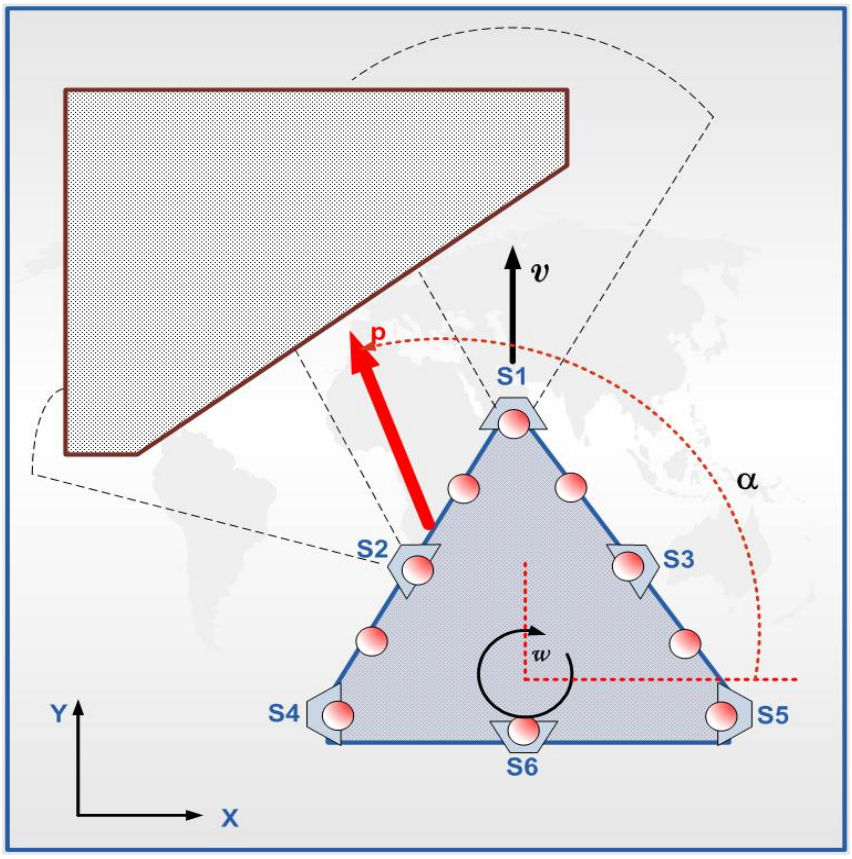

Figura 4.2: Rob móvel com o vetor de percepção

A idéia desta técnica que cada sensor i do robô móvel seja referido a um vetor p_i, esta direção vai ser a orientação dos eixos do sensor e a função de longitude vai ser em função da distancia d_i medido pelo sensor.

$$p_i = \frac{d_{max} - d_i}{d_{max} - d_{min}} \tag{4-1}$$

onde d_{min} e d_{max} refere-se distancia mínima e máxima respectivamente onde pode estar posicionado um objeto detectado. p_i limitado no range $[0 - 1]$

A percepção geral do vetor p composto da suma individual das percepções p_i, seu direção a suma das percepções de todos os sensores e seu módulo o maior módulo de todas as percepções individuais.

$$\mathbf{p} = p_{i,max} \times \frac{\sum(pi)}{|\sum(pi)|} \tag{4-2}$$

A razão de cambio da percepção geral no tempo representado por $\dot{p}$ e expressado por um escalar.

$$\dot{p}_i = \frac{dp_i}{dt} = \frac{\triangle d_i}{\triangle t \times (d_{max} - d_{min})} \tag{4-3}$$

Com estas equações do conceito da percepção geral podemos fazer uso das técnicas da lógica *fuzzy* para fazer um controle da navegação inicial do robô móvel.

4.4 Algoritmo de Exploração

Para a primeira etapa da exploração do ambiente desconhecido se fez algumas suposições:

- As paredes do ambiente desconhecido não é do mesmo cor e não é homogêneo.
- A posição inicial do robô vai ser sempre escolhido de modo que o robô tenha espaço para navegar, isto para segurar que o robô vai ter uma direção para começar a navegar.
- A exploração tem que ser num ambiente que tenha iluminação adequada.

Feitas estas suposições na Figura 4.3 se ilustra o diagrama de fluxo do algoritmo de navegação e a continuação se descreve passo a passo o processo.

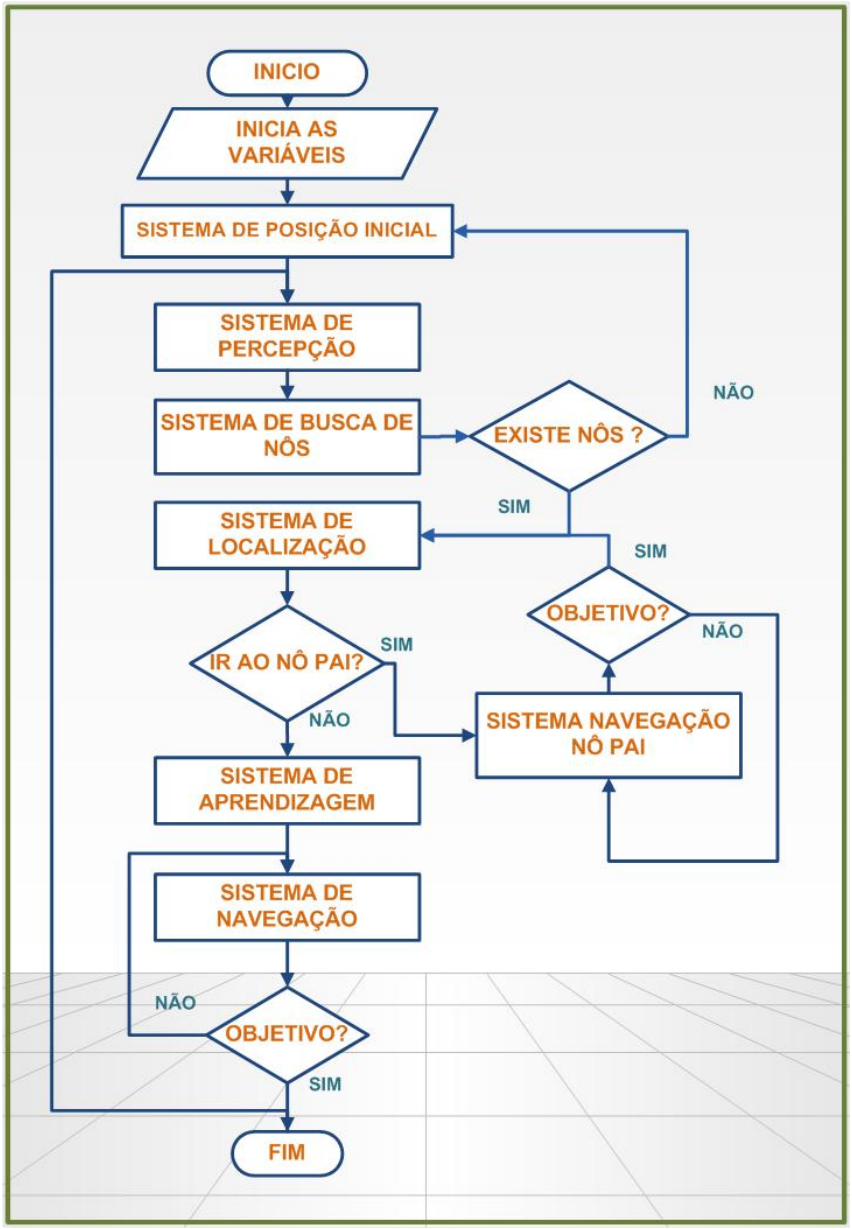

Figura 4.3: Diagrama de fluxo da etapa de exploração

1. O robô procura uma posição inicial para começar a exploração.
2. O robô ativa seu sistema de percepção para guardar a informação do nó.
3. Se procura os possíveis nós a explorar.
4. Se existe nós para explorar se executa o sistema de localização se não se procura outra posição inicial.
5. Se verifica se existe nós filhos a explorar se não existe se executa o sistema de retorno ao nó pai de outro modo se executa o sistema de navegação ao nó filho.

6. Se executa o sistema de aprendizagem.

7. Se executa o sistema de navegação ao nó filho.

8. Se o robô ainda não chega ao objetivo então continua com a navegação fazendo as leituras dos sensores.

9. Se chegou ao possível nó filho então faz novamente a etapa de percepção e continua o processo novamente desde 2.

Este algoritmo se assemelha ao proposto por (Fel06), pelo arvore de nós, mas a principal diferença está no uso de redes neurais com informações obtidas pelos sensores de ultra-som e infra-vermelho, além do uso da câmera. O trabalho de (Fel06), se limitou ao uso da câmera e de algoritmos determinísticos para a identificação dos nós, sem o uso de técnicas de inteligência computacional. Os sensores infra-vermelhos foram usados em (Fel06), apenas para cumprir o papel de sensores de contato(*bump sensores*).

4.4.1 Sistema de posição inicial

Nesta etapa o robô começa com uma leitura dos sensores e procura uma posição inicial segundo a percepção geral num ângulo definido como 0º, isto significa que a posição inicial é na direção do menor vetor de percepção geral. Se encontra esta direção o robô já tem um nó inicial mas se não encontra o robô procura a direção do objeto mais próximo ao robô, este algoritmo se ilustra na Figura 4.4.

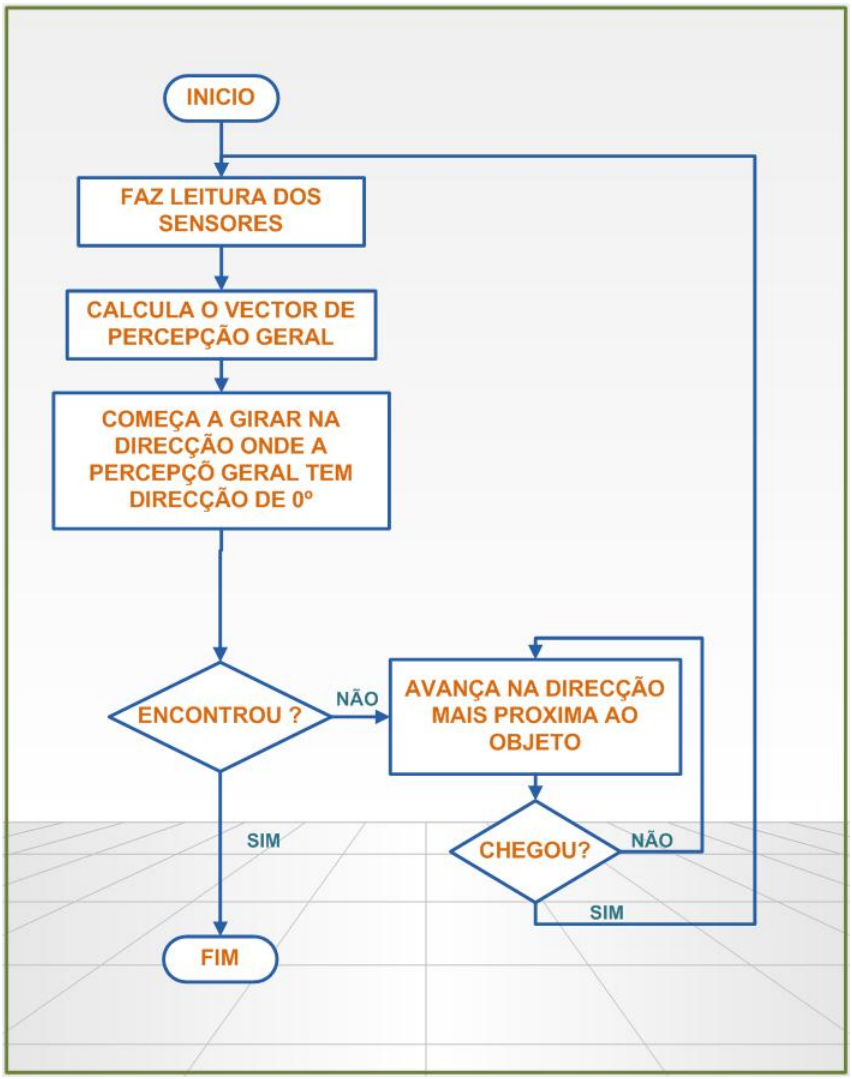

Figura 4.4: Diagrama de fluxo do sistema de posição inicial

4.4.2 Sistema de percepção

Nesta etapa o robô faz a leitura dos sensores ultra-sônicos e infravermelhos na direção inicial (ângulo = 0º), e também tira uma foto e calcula os descritores *SIFT* e os guarda num vetor; depois disso o robô gira um ângulo de 15º e repete a leitura dos sensores e o calculo dos descritores e assim por diante até completar o giro de 360º, logo todos os dados são guardados. A leitura dos sensores é feito pelo *DSP* e o cálculo dos descritores é feito pelo *notebook*, o diagrama se ilustra na Figura 4.5.

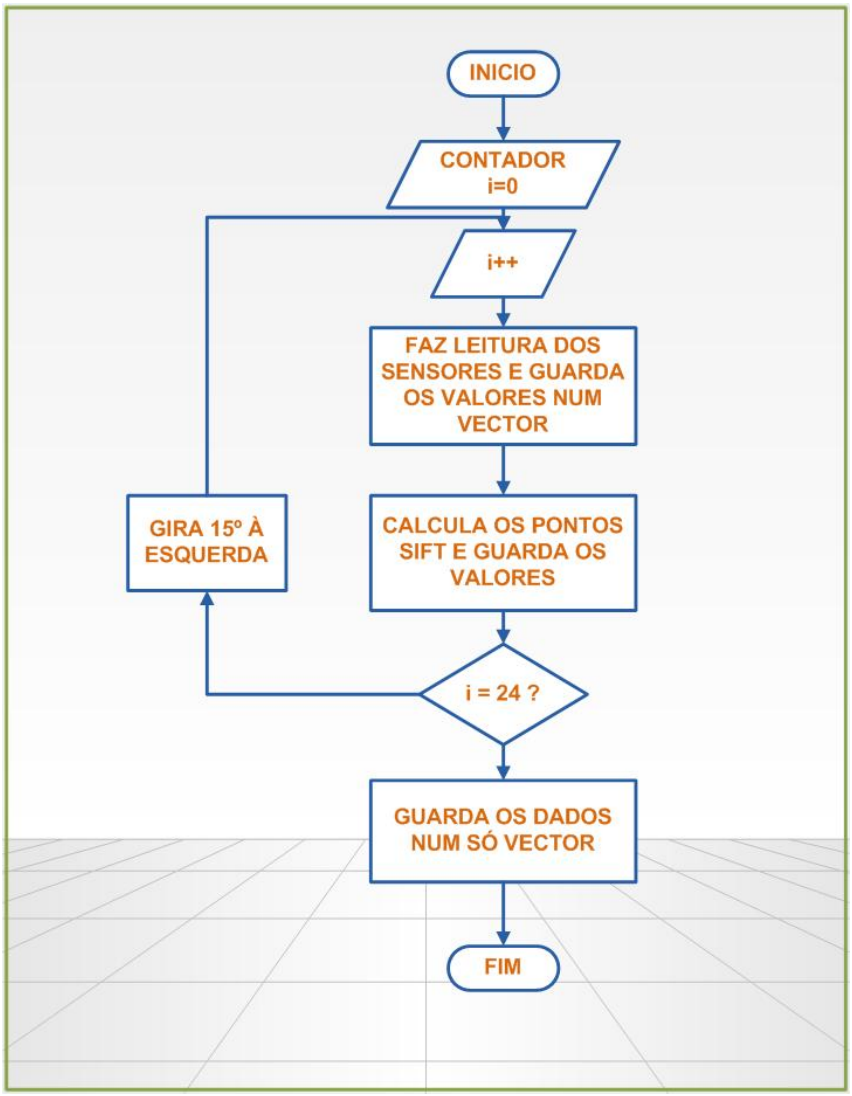

Figura 4.5: Diagrama de fluxo do sistema de Percepção

4.4.3 Sistema de busca de nós

Nesta parte do algoritmo procura se a informação do sensor ultra-sônico $S1$ para cada direção cada 45º e ordenados em forma crescente até 360º, se existe distâncias ao objeto maiores a $720mm$ (escolhido em relação ao diâmetro do robô), então essas direções se consideram possíveis nós filhos para explorar depois é verificado se existe dois possíveis nós filhos adjacentes, se isso é certo então a direção que tem menor distância ao objeto se considera que vai ser um nó folha, se faz isto para evitar que dois direções adjacentes tenham os mesmos nós filhos; se não existisse direções para explorar então se ativa um *flag* que indique isso; o diagrama de fluxo se ilustra na Figura 4.6.

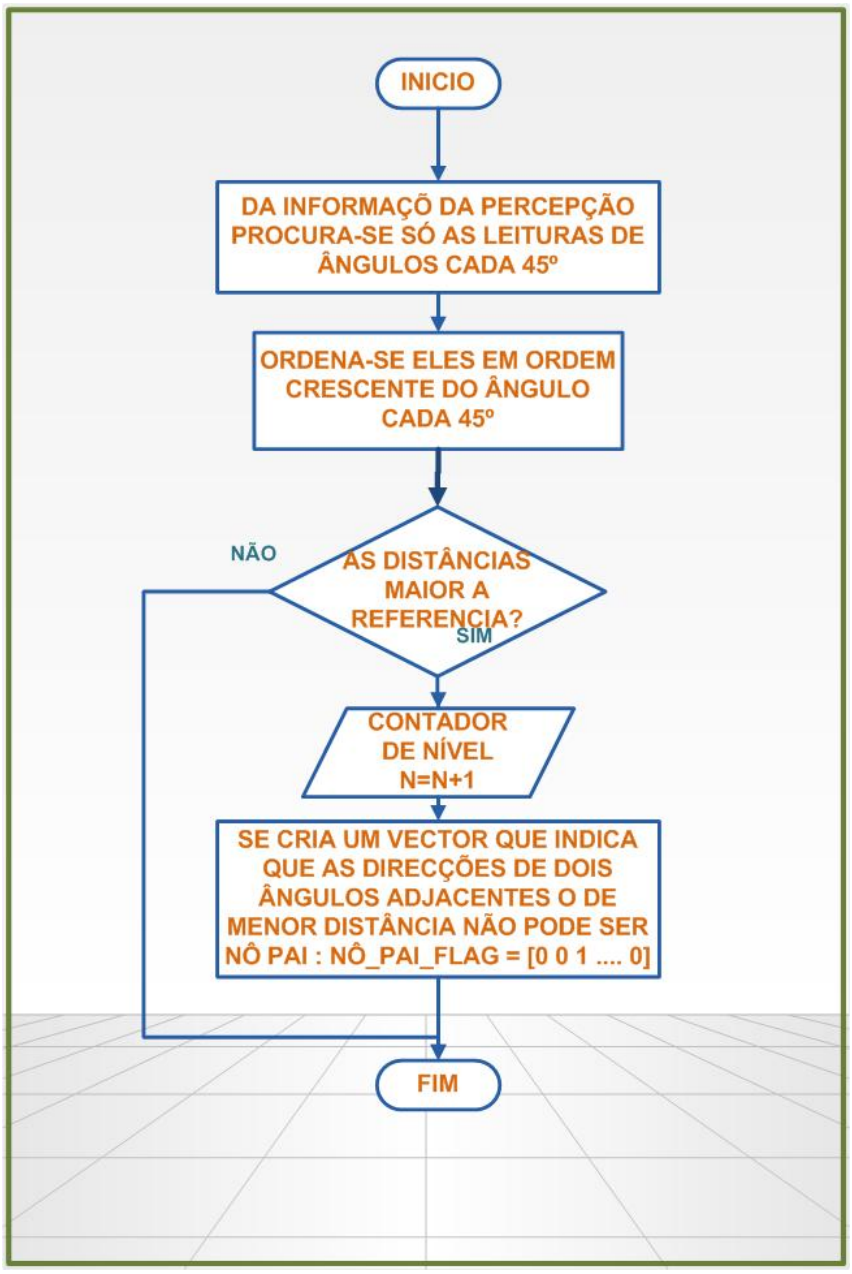

Figura 4.6: Diagrama de fluxo do sistema de Busca dos nós

4.4.4 Sistema de localização

A idéia da localização é que o robô tenha presente em que nó ele se encontra, com isto o robô sabe se vai explorar a um nó filho ou voltar a um determinado nó pai, isto se ilustra num grafo de árvore (Figura 4.7).

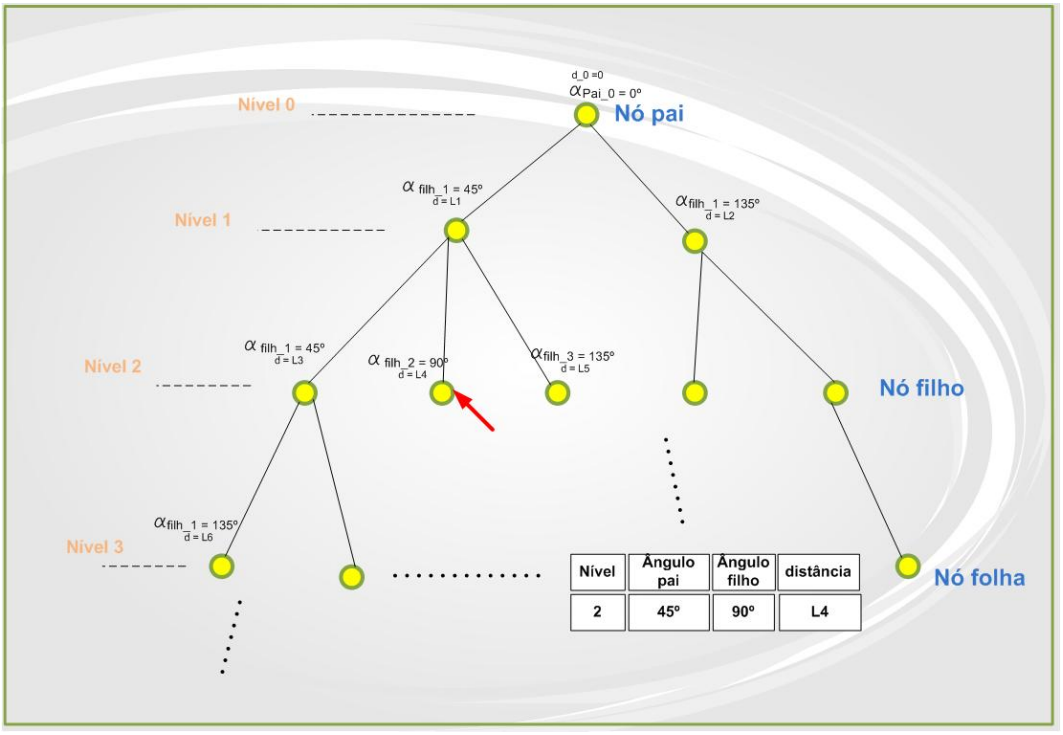

Figura 4.7: Grafo de árvore de nós da exploração

Para poder identificar o nó temos um vetor de identificação que se amostra na Figura 4.7; Por exemplo se queremos saber qual é a identificação do nó que está apontado com a flecha, teremos 4 parâmetros para isso, o primeiro indica o nível de profundidade da exploração, o segundo indica o ângulo do nó pai que saiu, o terceiro indica o nó filho atual que se encontra o robô, e o quarto indica a distância percorrida do nó pai até o nó filho.

O algoritmo desta etapa de localização se ilustra na Figura 4.8, onde pode-se observar as atualizações de todas as variáveis, contadores e *flags* para saber onde se encontra o robô; depois disso se aumenta em 45º a direção do nó filho a explorar mas se ele tem uma distância menor ao de referência então essa direção não é explorada pelo tanto aumenta novamente em 45º até chegar a 360º; se já chegamos então tem que voltar ao nó pai.

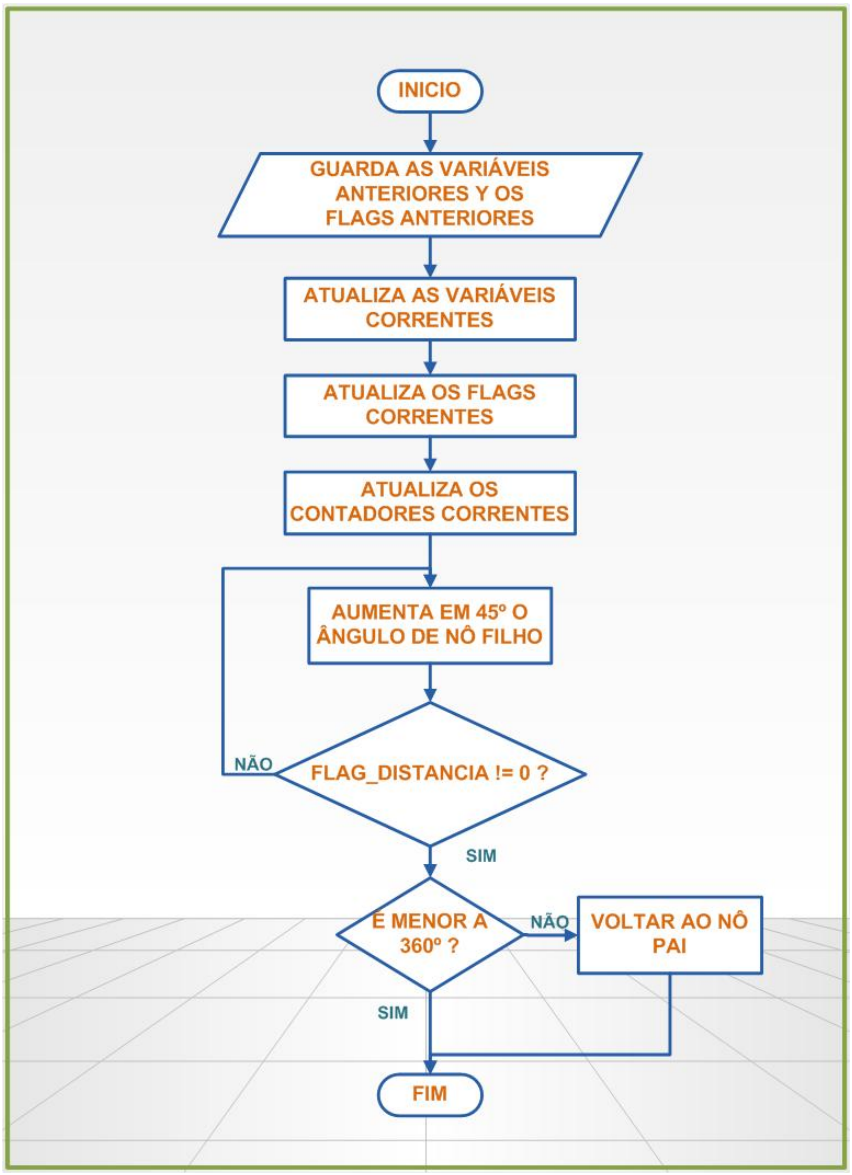

Figura 4.8: Diagrama de fluxo do sistema de Localização

4.4.5 Sistema de navegação para o nó pai

O algoritmo para esta etapa se ilustra na Figura 4.9, o primeiro que se faz é saber o ângulo do nó pai, o ângulo do nó filho atual, a distância percorrida e o nível a retornar; o robô gira o ângulo para retornar ao nó pai, logo executa o sistema *fuzzy* de velocidade até chegar ao nó pai, quando o robô está perto ao nó pai, se faz a leitura dos sensores infravermelhos e a aquisição dos descritores *SIFT* com uma velocidade do robô é mínima para comprovar a chegada do robô ao nó pai.

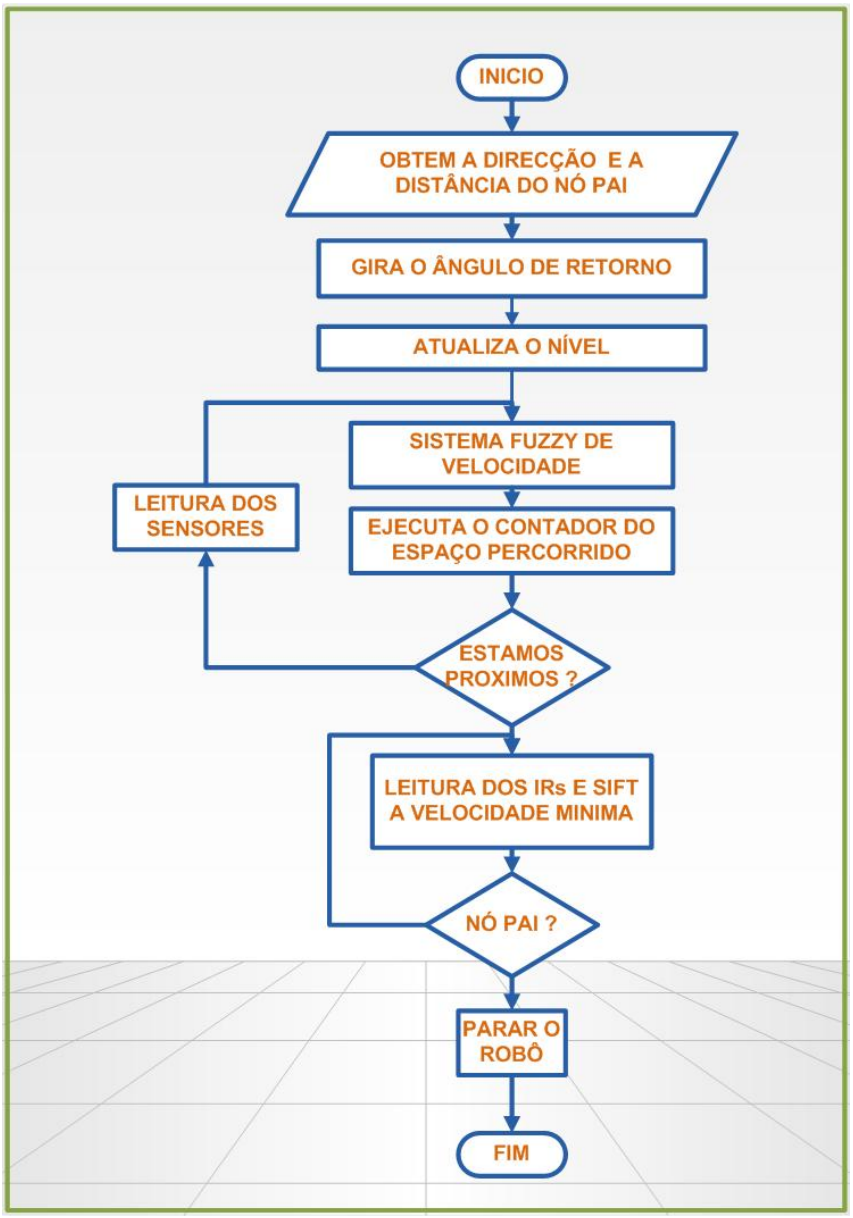

Figura 4.9: Diagrama de fluxo do sistema de Navegação ao nó pai

4.4.6 Sistema de Aprendizagem

Na parte do aprendizagem dos nós é feito pela rede neuronal que se ilustra na Figura 4.10, onde as entradas são as leituras dos sensores codificados em sistema binário e os descritores *SIFT* normalizados, o número de entradas à rede neuronal é 42; as variáveis de saída são toda a informação que pode obter-se do nó (o nível onde se encontra, o ângulo pai de onde se deriva o nó atual, o ângulo atual, a distância desde o nó pai, e o número de nós filhos que tem o nó atual); estas variáveis de saída estão codificadas em sistema binário e o número das variáveis de saída são 23.

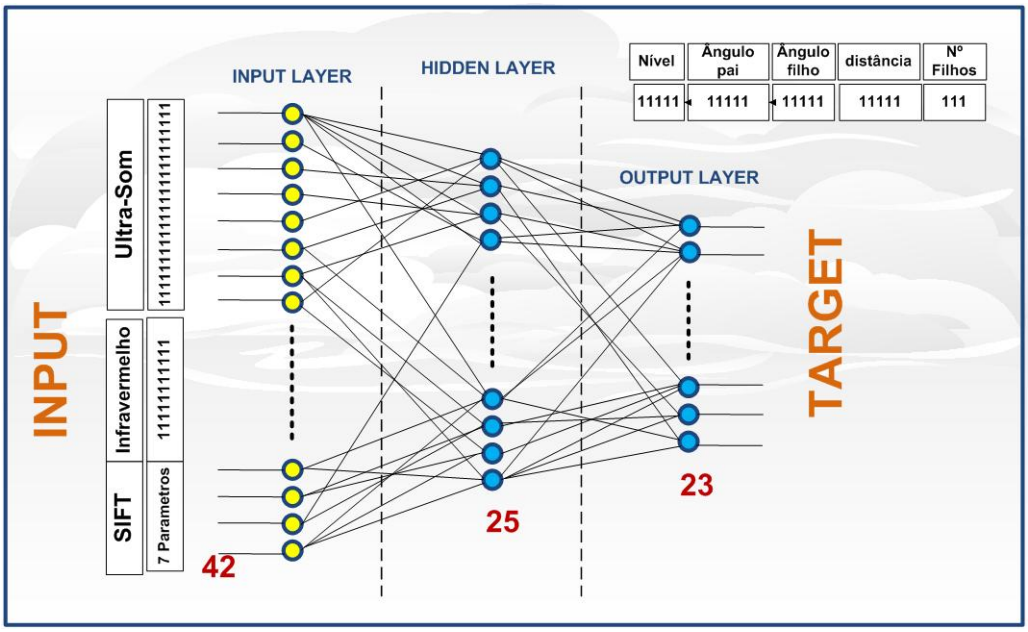

Figura 4.10: Rede Neuronal proposto

O diagrama de fluxo do sistema de aprendizagem se ilustra na Figura 4.11, o sistema começa com o tratamento inicial dos dados e depois se faz o treinamento da rede se o treinamento é ruim se faz novamente o treinamento, se o treinamento é ótimo então se guarda os pesos (W_i) da rede treinado. Esta parte do treinamento é feito pela *notebook* enquanto o *DSP* faz o controle dos motores para a navegação.

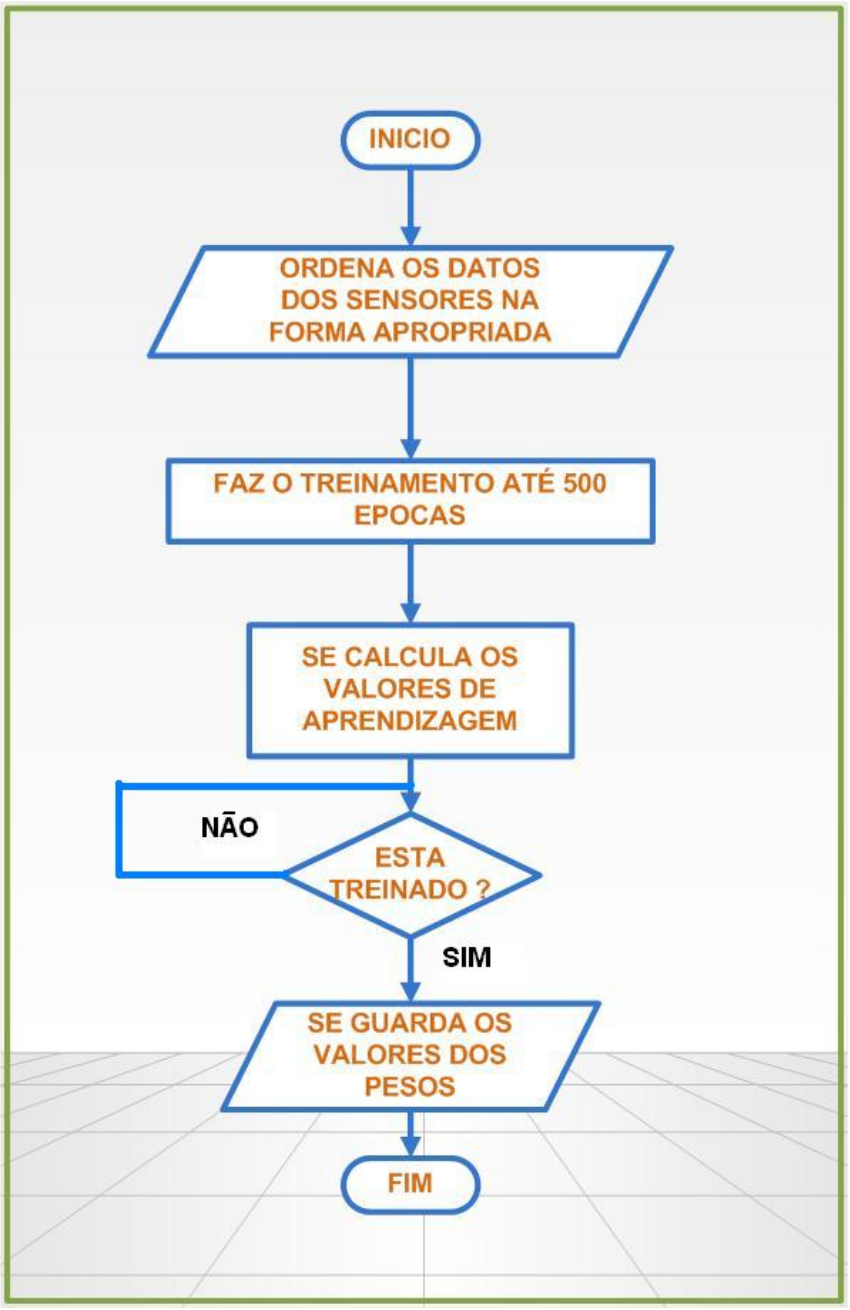

Figura 4.11: Diagrama de fluxo do sistema de Aprendizagem

4.5 Controlador Fuzzy

O controlador fuzzy foi desenhado baixo a suposição de um ambiente estático, é dizer não existe obstáculos que estão no movimento; existem dois controladores independentes fuzzy no robô $ER1$, um controlador para o movimento angular do robô e outro controlador para a rapidez do robô, cada um de estes controladores executam seu control diretamente sobre os motores *brushless*, permitindo assim controlar o movimento e a navegação do robô $ER1$.

4.5.1 Entradas fuzzy

O controlador fuzzy consiste de uma regra total de 27 regras, os quais representam instruções para estimar o comportamento do robô em diferentes situações. Os valores de as entradas são o ângulo α, quem esta entre o vector de percepção geral e a velocidade do robô, a intensidade do vector percepção geral p, e seu mudança no tempo $\dot{p}$.

O robô classifica as situações usando a percepção descrita no item anterior. Para este propósito α, p e $\dot{p}$ são chamados "*anguloperceposo*", "*percepo* e "*mudapercepo*" respectivamente cada um deles com seus respectivos conjuntos fuzzy, tal como se ilustra nas Figuras (4.12),(4.13) y (4.14).

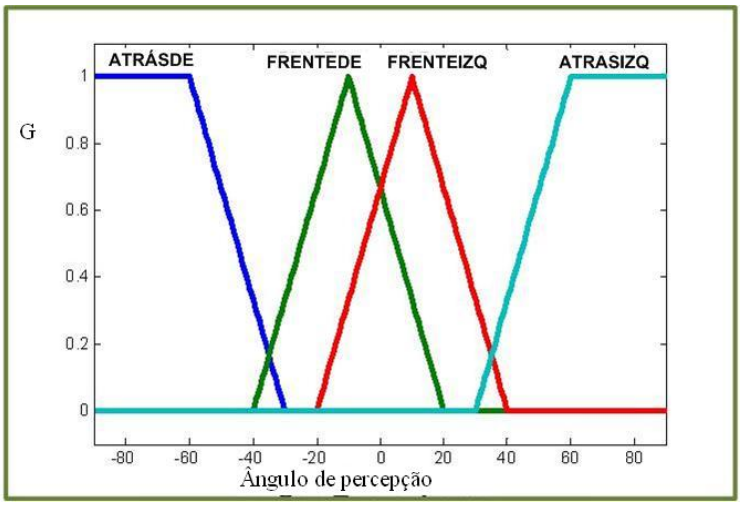

Figura 4.12: Conjuntos fuzzy da variável ângulo de percepção

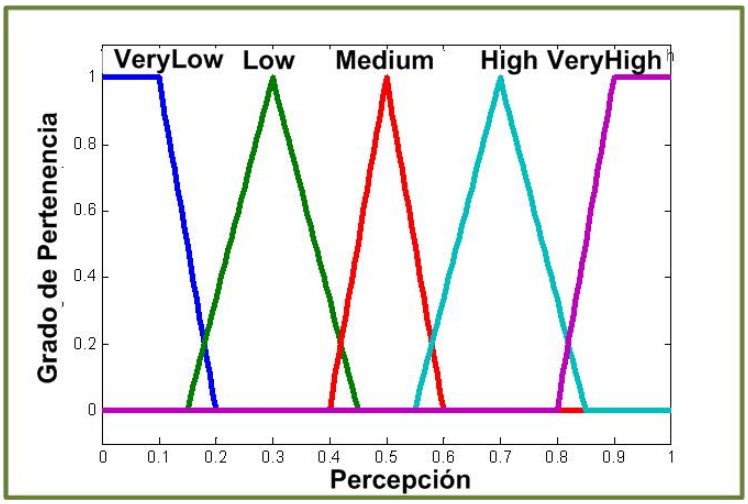

Figura 4.13: Conjuntos fuzzy da variável percepção

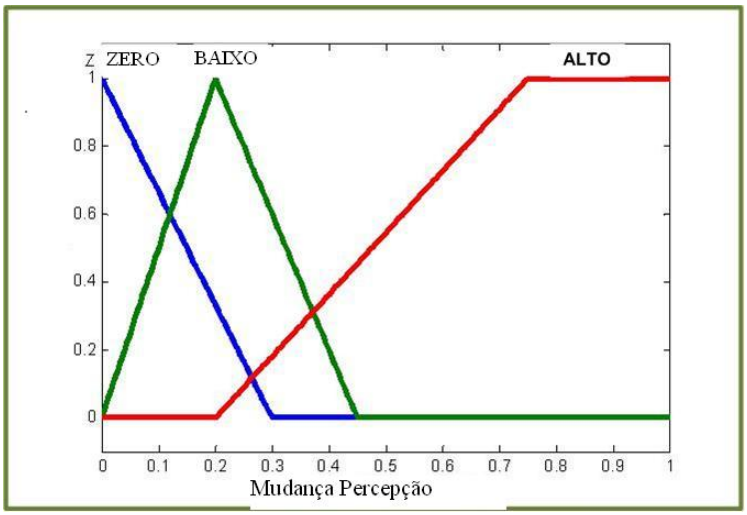

Figura 4.14: Conjuntos fuzzy da variável mudança de percepção

4.5.2 Saídas fuzzy

Las variables de saída fuzzy são a *direction* e a aceleração; a variável de saída fuzzy "direction"executa o controle de velocidade angular do robô desde -60^o a 60^o respeito ao eixo de velocidade linear do robô. A variável de saída fuzzy "aceleração"executa o controle da aceleração linear do robô, os conjuntos fuzzy das variáveis de saída fuzzy se ilustram nas Figuras (4.15) y (4.16).

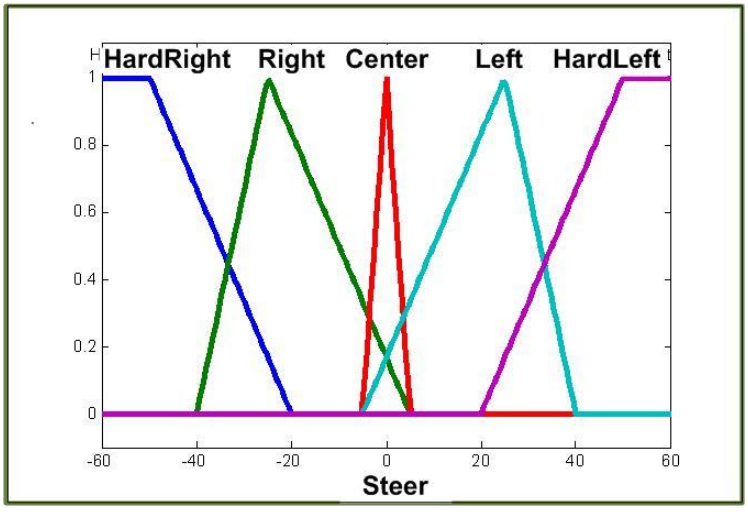

Figura 4.15: Variável de saída fuzzy *Direction*

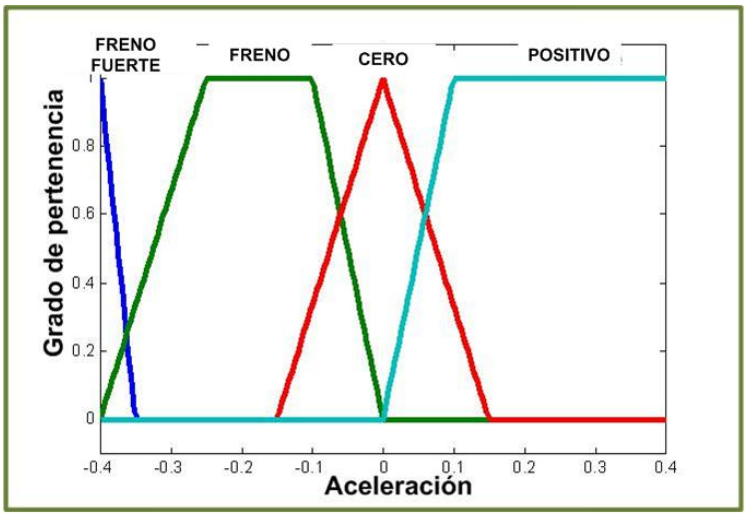

Figura 4.16: Variável de saída fuzzy aceleração

4.5.3 Controle de direção

Neste controle fuzzy se usam como variáveis de entrada o "ângulo da percepção"e a "percepção"e como variável de saída a *"direction"* tal como se ilustra na Figura 4.17.

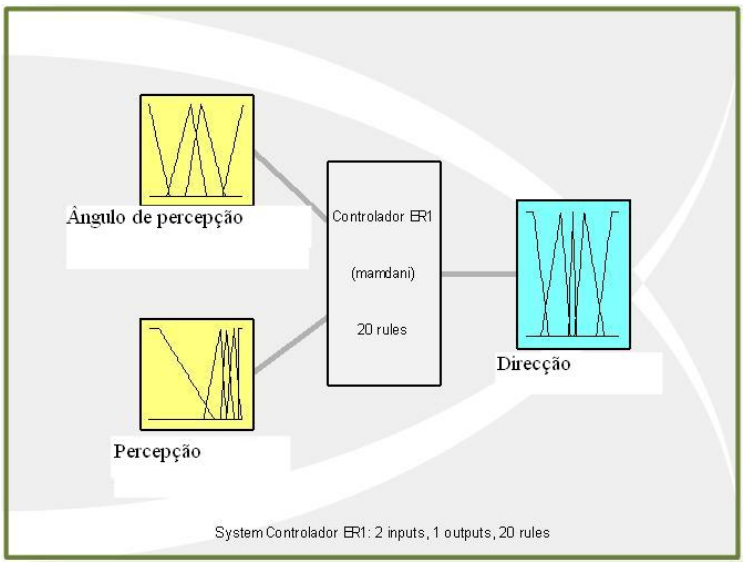

Figura 4.17: Controle de direção do robô ER1

As regras de controle fuzzy para o controle de direção tratam de manter o robô separado dos obstáculos e navegue até espaços livres do ambiente desconhecido, a tabela 4.1 amostra todas as regras usadas neste controle.

Tabela 4.1: Regras do controle de direção do robô ER1

α/p	VL	L	M	H	VH
RB	HR	R	C	R	C
RF	R	R	L	L	HL
LF	HL	L	R	R	HR
LB	HL	L	C	L	C

Tabela 4.2: Regras do controle de velocidade do robô ER1

$\dot{p}/p$	VL o VH	Lo H	ME	—
ZE	ZE	P	P	—
L	EB	B	Z	—
H	—	—	—	EB

4.5.4 Controle de velocidade

Este controle tem 7 regras, que se ilustram na tabela 4.2 , este control tem como entradas às variáveis fuzzy "percepção"e "mudança de percepção"as quais controlam a variável de saída "aceleração". Este controle faz acelerar o robô o diminuir a aceleração do robô; a gráfica do controle de velocidade se ilustra na Figura 4.18

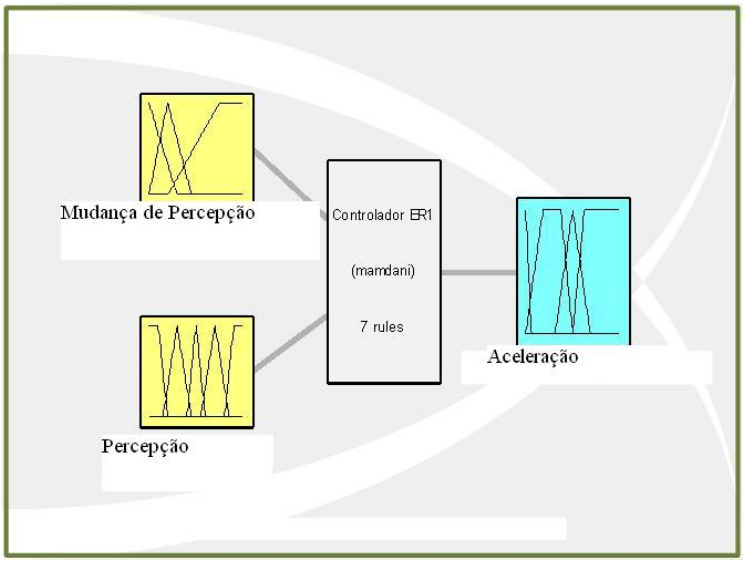

Figura 4.18: Controle de velocidade do robô ER1

5
Testes e Resultados

Neste capítulo serão apresentados experimentos que ilustram as implementações desenvolvidas e propostas no capítulo anterior. São mostrados experimentos que investigam o algoritmo de exploração proposto mas feito inicialmente para um ambiente fechado pequeno em relação ao robô.

5.1
Testes e Resultados no Player-Stage

No Player-Stage se fez até agora três tipos de simulações, o primeiro é um algoritmo de seguimento da parede utilizando o conceito da percepção geral e um sistema de controle com lógica fuzzy, esta simulação foi feito para avaliar a capacidade da percepção geral usando só 6 sensores ultra-sônicos o resultado se ilustra nas Figuras5.1-5.10:

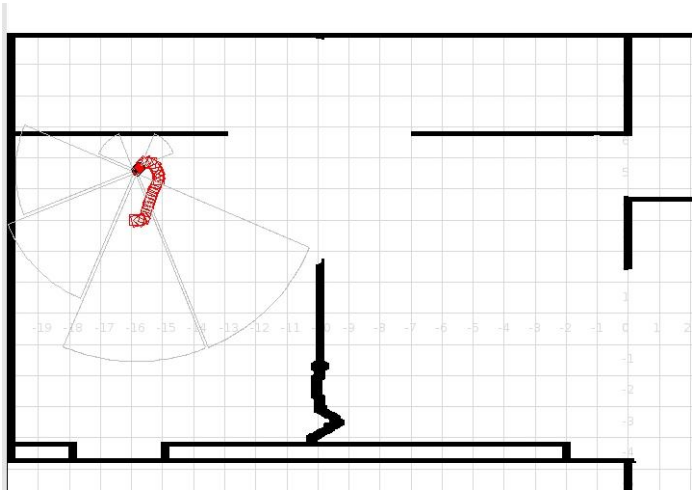

Figura 5.1: Exploração WallFollow (etapa 1/10) - robô se move na direção de vetor percepção geral até chegar próximo da parede

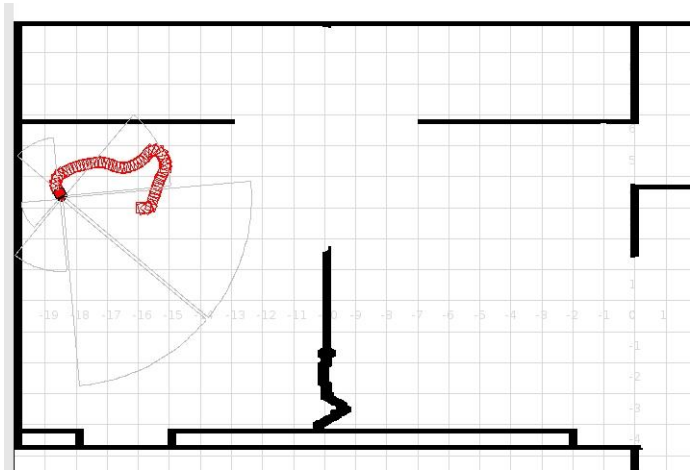

Figura 5.2: Exploração WallFollow (etapa 2/10) - robô se move seguindo a parede

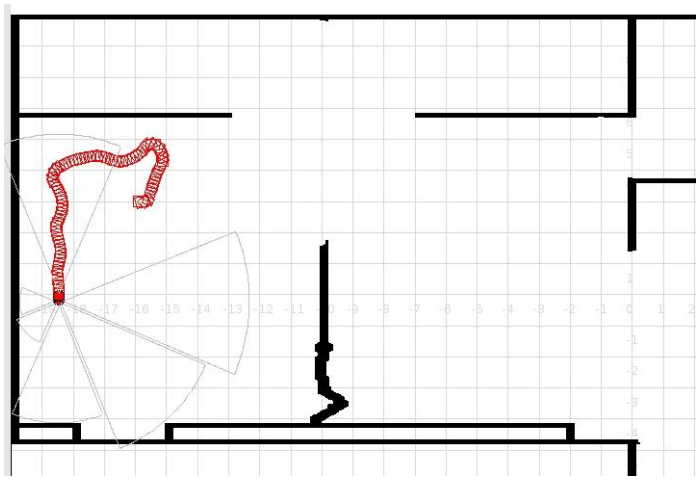

Figura 5.3: Exploração WallFollow (etapa 3/10) - robô trata de seguir a parede com certa oscilação

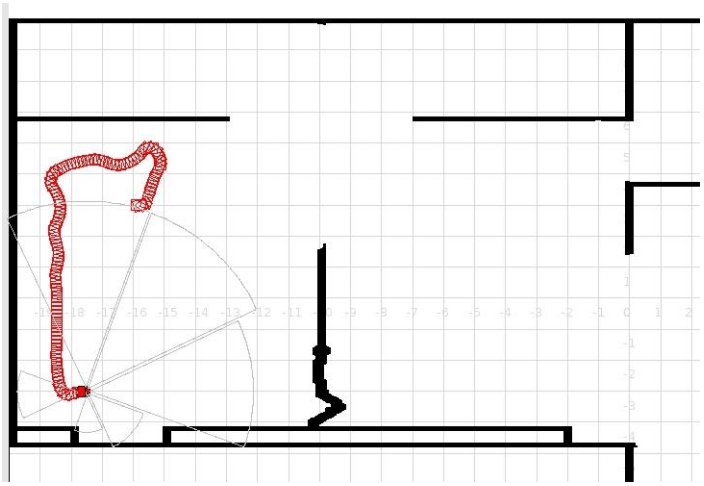

Figura 5.4: Exploração WallFollow (etapa 4/10) - robô se move paralelo à parede e vira à esquerda

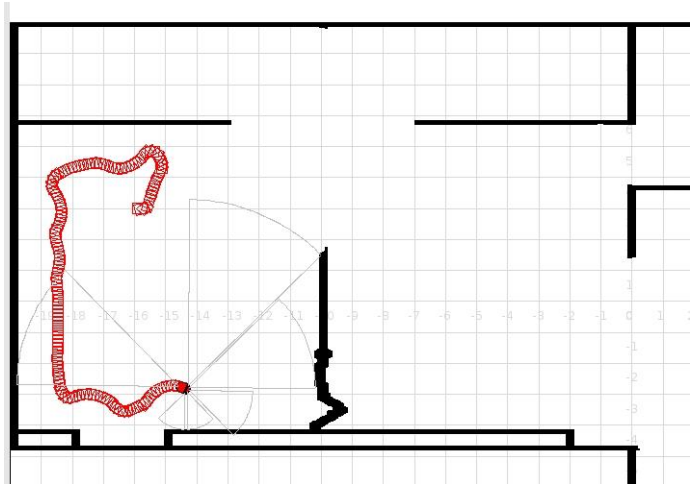

Figura 5.5: Exploração WallFollow (etapa 5/10) - robô se move perto da parede com oscilações devido à parede irregular

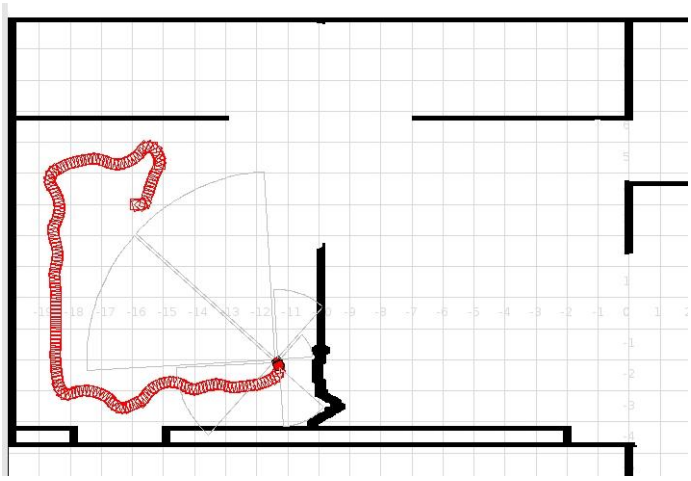

Figura 5.6: Exploração WallFollow (etapa 6/10) - robô se move paralelo à parede

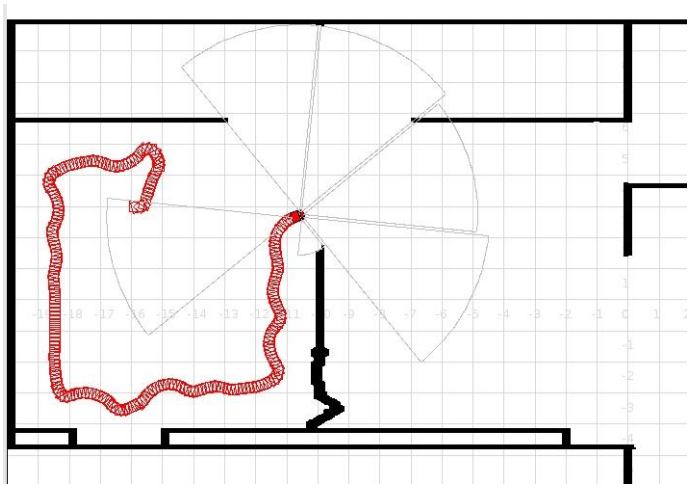

Figura 5.7: Exploração WallFollow (etapa 7/10) - robô vira à esquerda e segue à parede

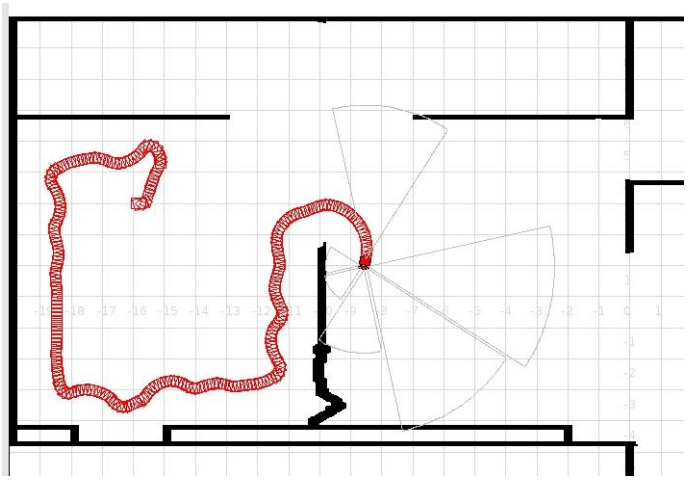

Figura 5.8: Exploração WallFollow (etapa 8/10) - robô vira com aceleração para conseguir virar 180º

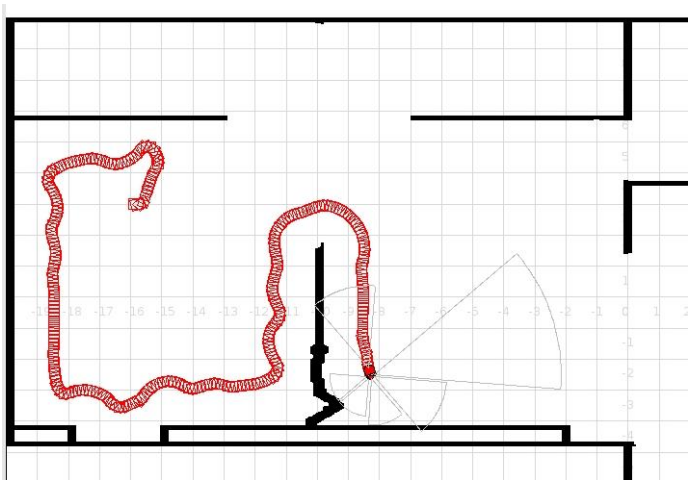

Figura 5.9: Exploração WallFollow (etapa 9/10) - robô se move paralelo à parede

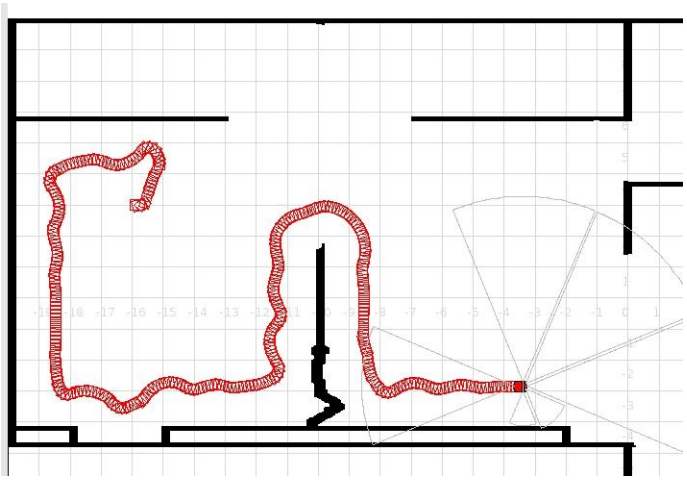

Figura 5.10: Exploração WallFollow (etapa 10/10) - robô segue paralelo à parede da parte inferior do mapa

O segundo tipo de simulação é a navegação do robô seguindo os espaços livres no ambiente. Isto também foi feito usando o conceito da percepção geral, neste caso a idéia é que o robô navegue pelo espaço onde não tem obstáculos, é dizer ao contrario do caso anterior o vetor de percepção geral tem que ser o menor possível (espaço livre), isto se ilustra nas Figuras 5.11-**??**:

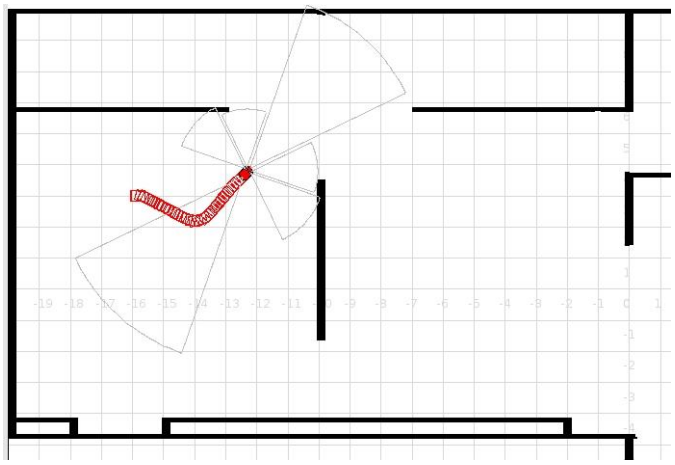

Figura 5.11: Exploração do Espaço Livre (etapa 1/6) - robô procura o espaço aberto

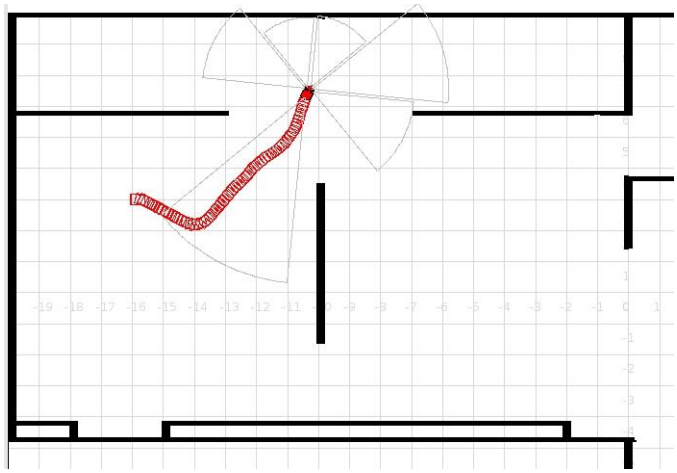

Figura 5.12: Exploração do Espaço Livre (etapa 2/6) - robô procura achar o caminho livre de obstáculos

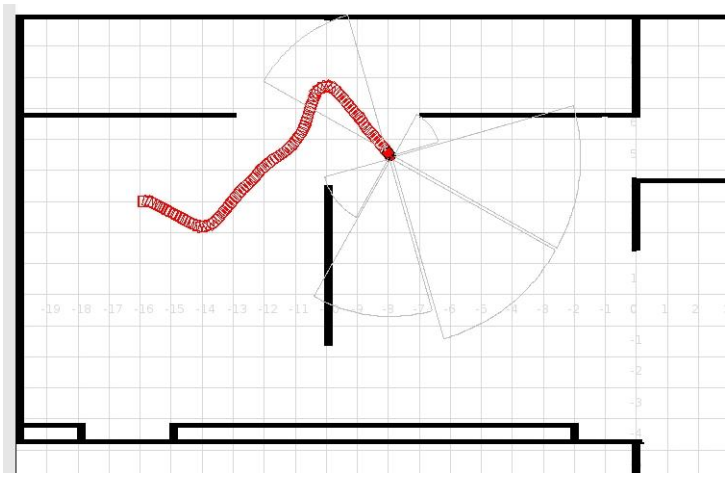

Figura 5.13: Exploração do Espaço Livre (etapa 3/6) - robô segue na direção do menor valor do vetor percepção geral

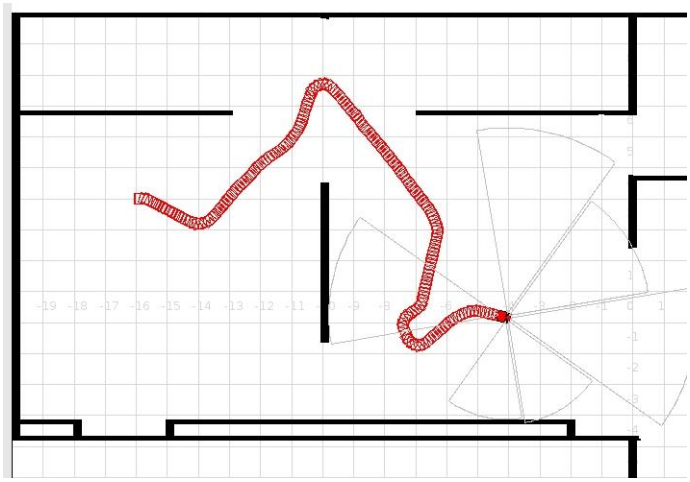

Figura 5.14: Exploração do Espaço Livre (etapa 4/6) - robô segue procurando o espaço livre

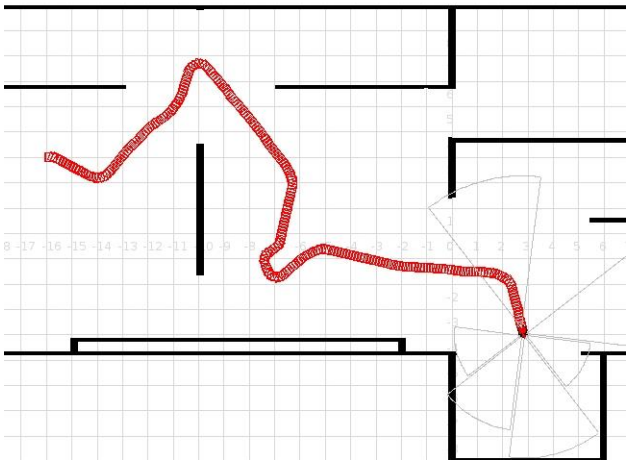

Figura 5.15: Exploração do Espaço Livre (etapa 5/6) - robô encontra um obstáculo e procura virar para sair em outra direção

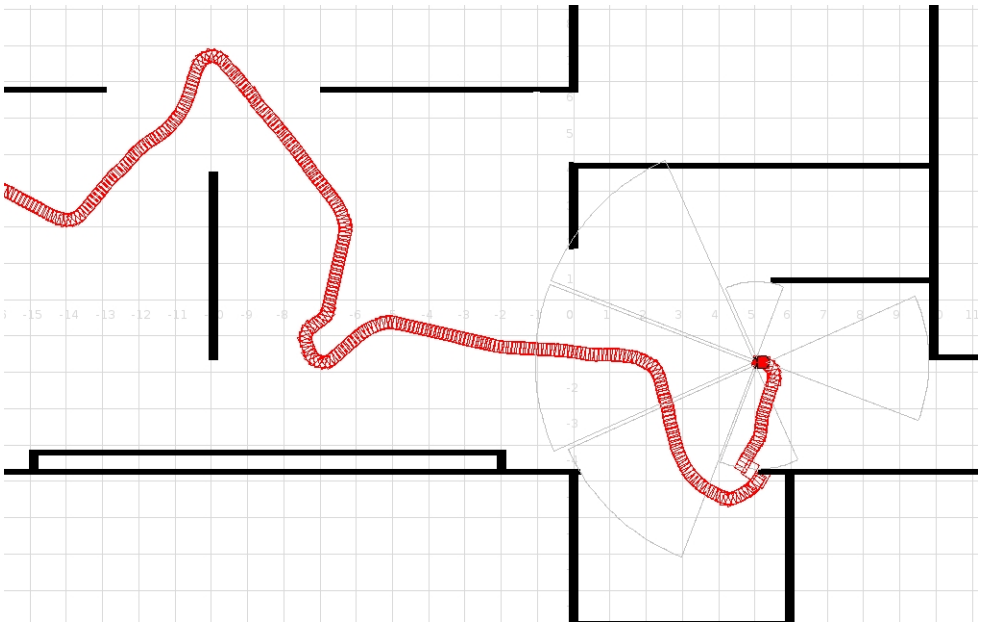

Figura 5.16: Exploração do Espaço Livre (etapa 6/6) - robô consegue sair do obstáculo e novamente procura o espaço livre

A terceira Simulação foi feita usando o algoritmo proposto no capítulo anterior mas sem o algoritmo do *SIFT* porque não é possível simulá-lo no Player-Stage. Embora foi feito só para um lugar pequeno, a simulação tem bom resultado ilustrando que o algoritmo pode cumprir a exploração sem problemas num ambiente pequeno, a navegação se ilustra nas Figuras 5.17-5.32:

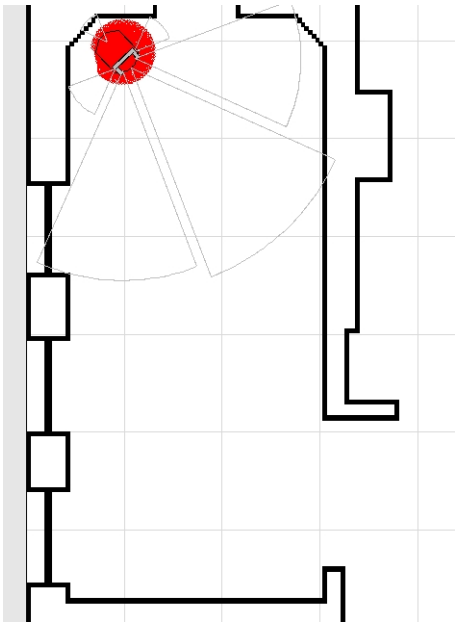

Figura 5.17: Navegação num ambiente desconhecido(etapa 1/16) - robô na posição inicial procura as possíveis direções de navegação

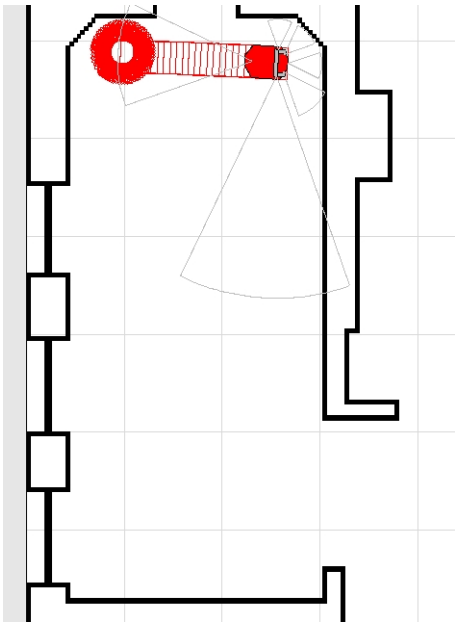

Figura 5.18: Navegação num ambiente desconhecido (etapa 2/16) - robô começa navegar na primeira direção

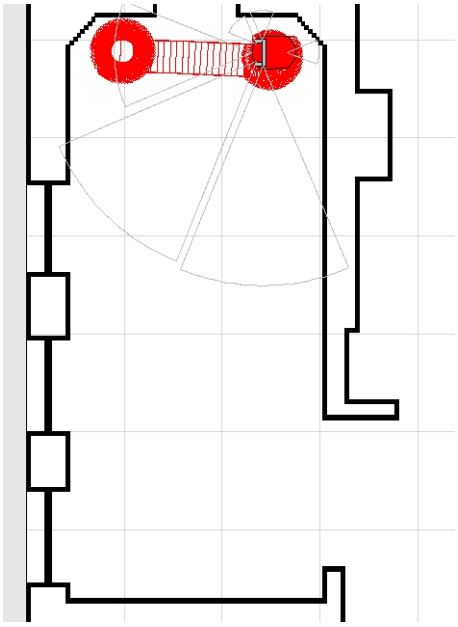

Figura 5.19: Navegação num ambiente desconhecido (etapa 3/16) - robô procura voltar novamente ao nó pai

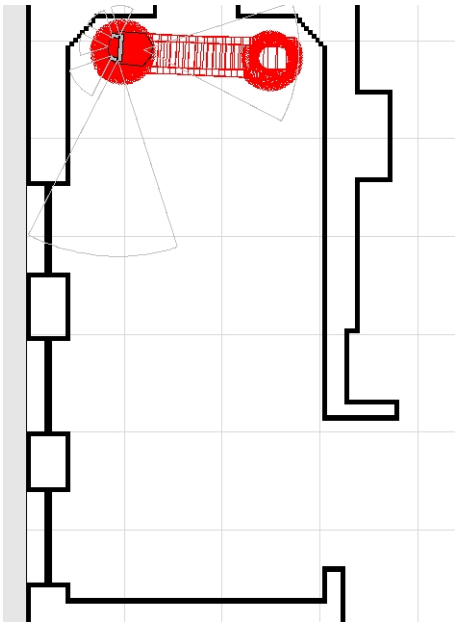

Figura 5.20: Navegação num ambiente desconhecido (etapa 4/16) - robô chega ao nó pai

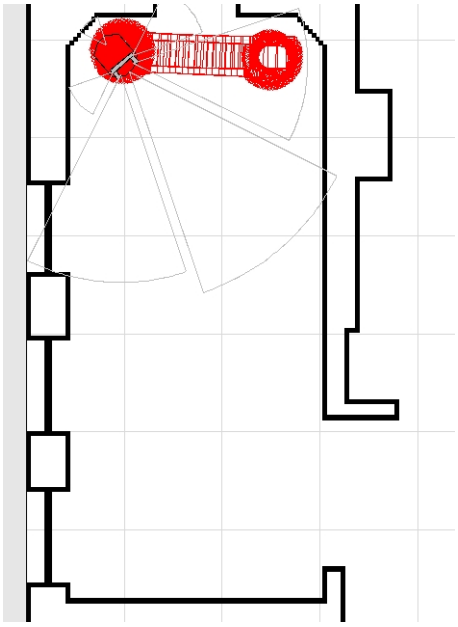

Figura 5.21: Navegação num ambiente desconhecido (etapa 5/16) - robô procura outra direção

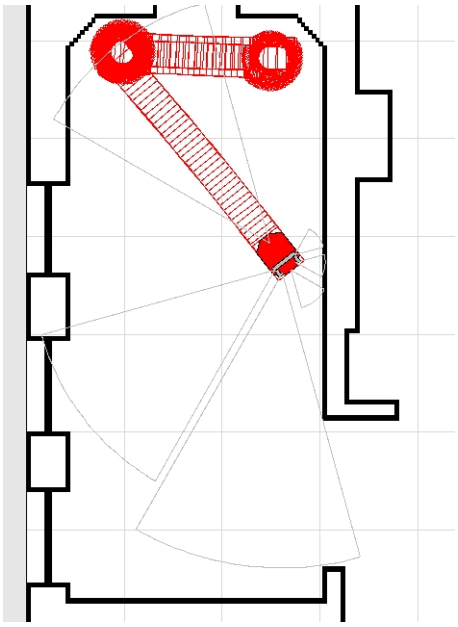

Figura 5.22: Navegação num ambiente desconhecido (etapa 6/16) - robô chega até estar próximo à parede

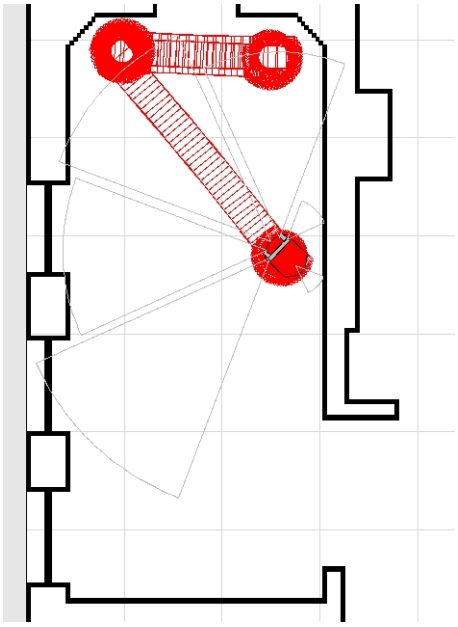

Figura 5.23: Navegação num ambiente desconhecido (etapa 7/16) - robô procura a direção do nó pai

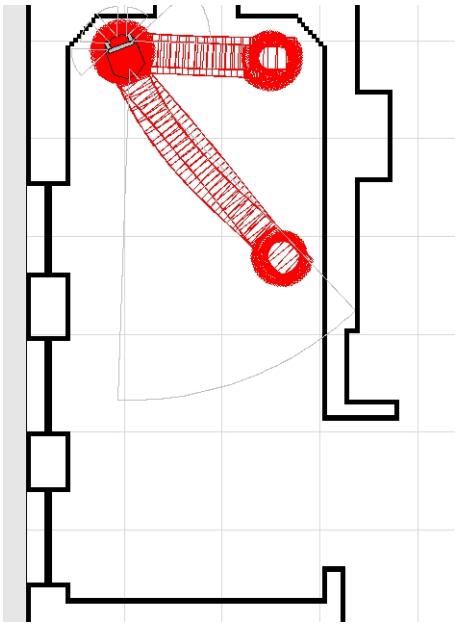

Figura 5.24: Navegação num ambiente desconhecido (etapa 8/16) - robô segue uma trajetória diferente até alcançar o nó pai

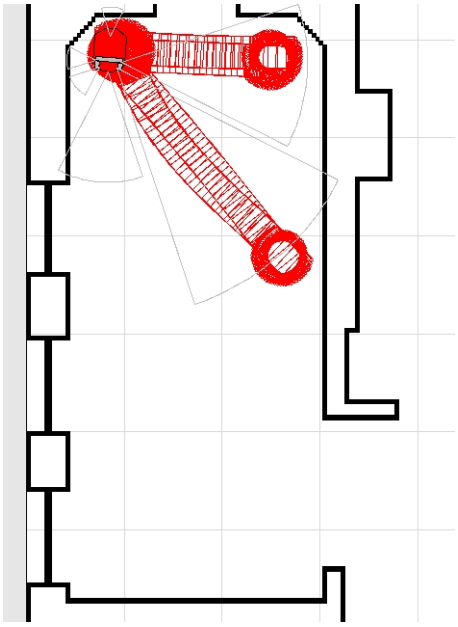

Figura 5.25: Navegação num ambiente desconhecido (etapa 9/16) - robô procura uma nova direção

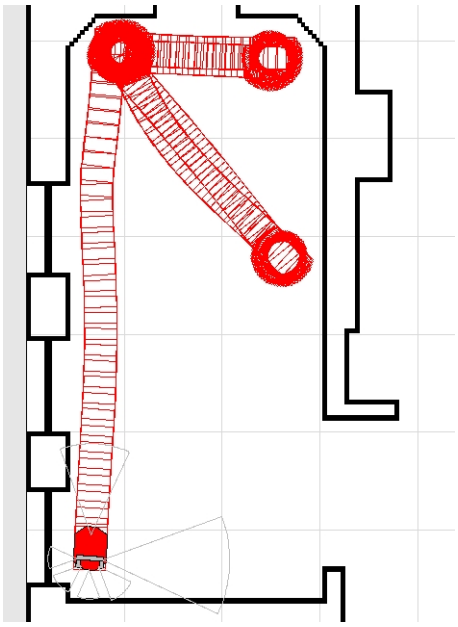

Figura 5.26: Navegação num ambiente desconhecido (etapa 10/16) - robô faz um wallfollow nesta trajetória

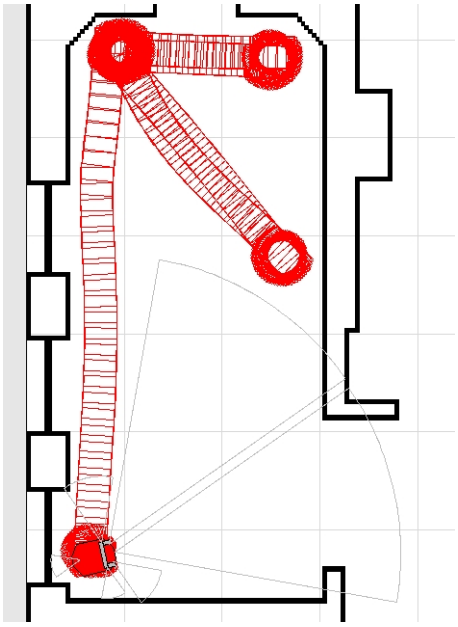

Figura 5.27: Navegação num ambiente desconhecido (etapa 11/16) - robô encontra uma possível trajetória

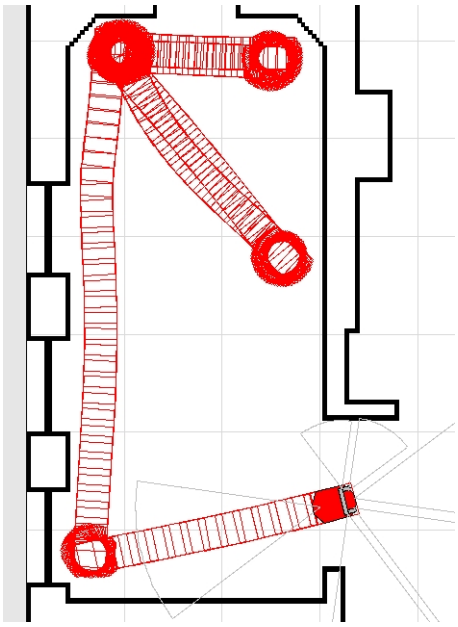

Figura 5.28: Navegação num ambiente desconhecido (etapa 12/16) - robô navega para outro nó

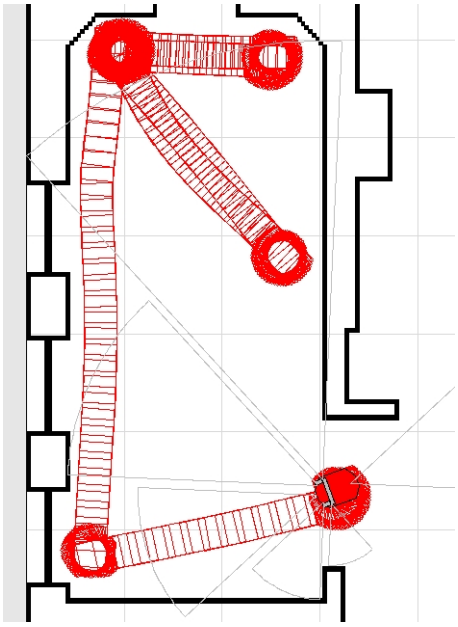

Figura 5.29: Navegação num ambiente desconhecido (etapa 13/16) - robô tem uma condição de retorno

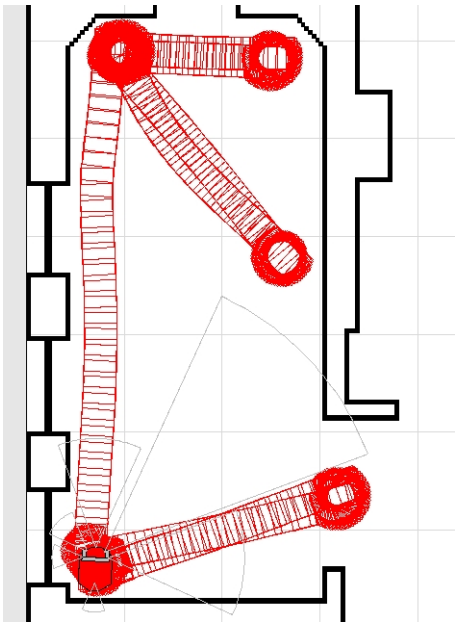

Figura 5.30: Navegação num ambiente desconhecido (etapa 14/16) - robô volta para o nó pai e gira até encontrar a direção do nó inicial

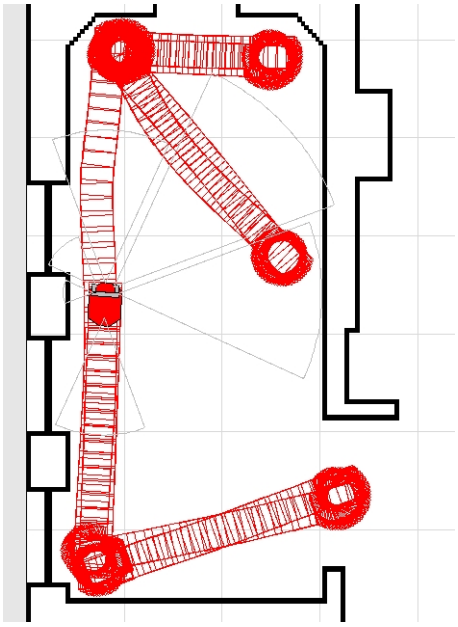

Figura 5.31: Navegação num ambiente desconhecido (etapa 15/16) - robô novamente faz um wallfollow para chegar ao nó pai

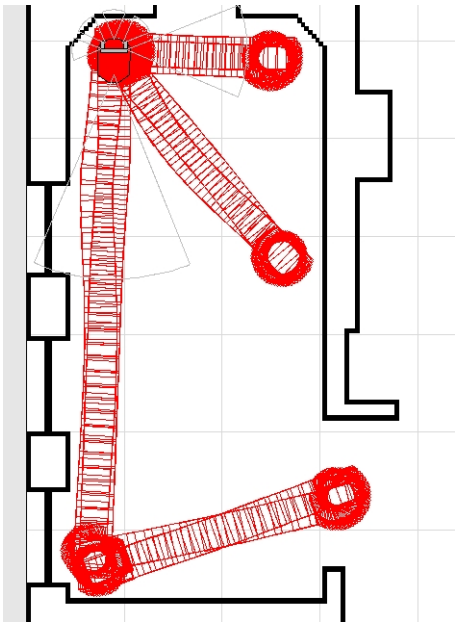

Figura 5.32: Navegação num ambiente desconhecido (etapa 16/16) - robô consegue chegar ao nó pai e consegue a condição de parada

5.2 Experimentos de identificação dos nós

Se realizou testes com o robô, o primeiro foi o reconhecimento de patrões no nó pai (Figura 5.33), se tirou fotos do nó 24 vezes cada 15º até completar uma volta completa; se achou os descritores *SIFT* para cada direção e se fez a leitura dos sensores, para um posterior treinamento da rede neuronal proposto no capitulo anterior. É dizer com a leitura dos sensores em cada um das 24 direções e com os descritores SIFT como entradas à rede neural pode-se generalizar e aprender o nó atual. Os resultados dos descritores se ilustram nas Figuras :

Figura 5.33: Nó pai no inicio da navegação

Figura 5.34: Descritores SIFT do nó depois do processamento no matlab num range total de visão

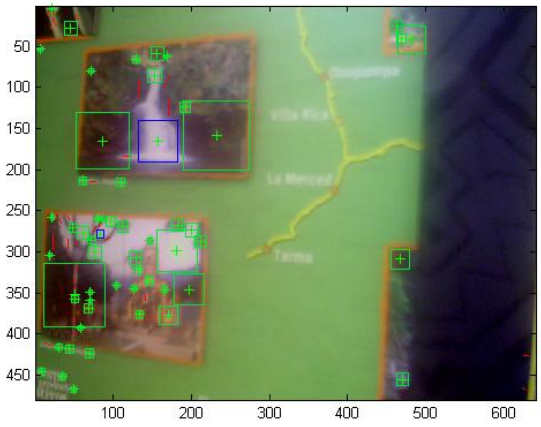

Figura 5.35: Descritores SIFT do nó pai num ângulo de 15º

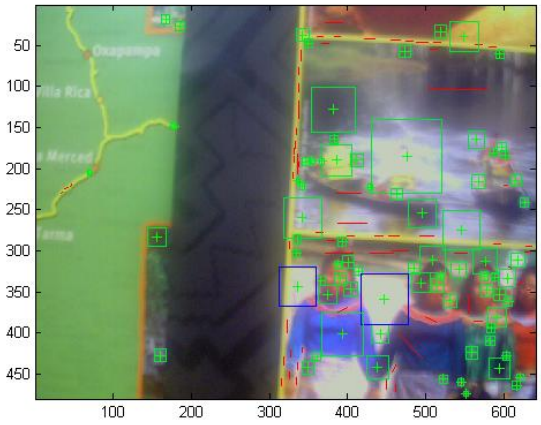

Figura 5.36: Descritores SIFT do nó pai num ângulo de 30º

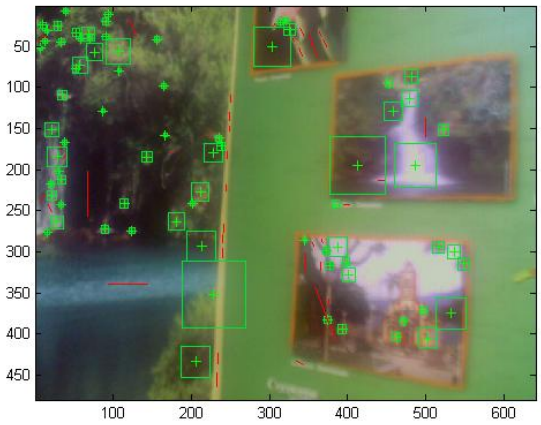

Figura 5.37: Descritores SIFT do nó pai num ângulo de 45º

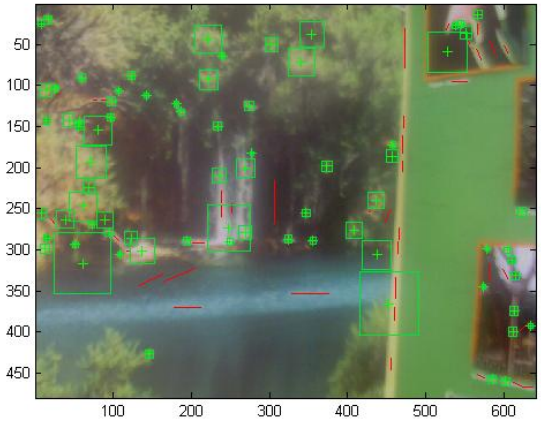

Figura 5.38: Descritores SIFT do nó pai num ângulo de 60º

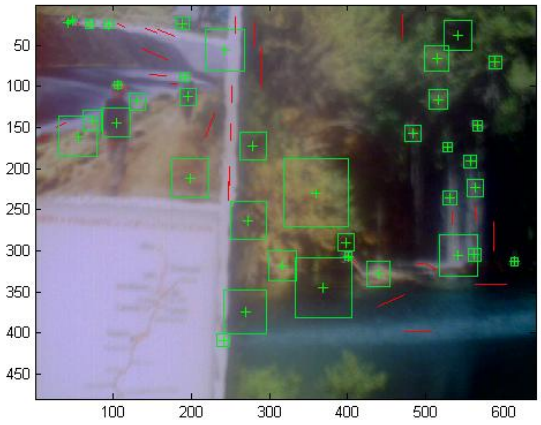

Figura 5.39: Descritores SIFT do nó pai num ângulo de 75º

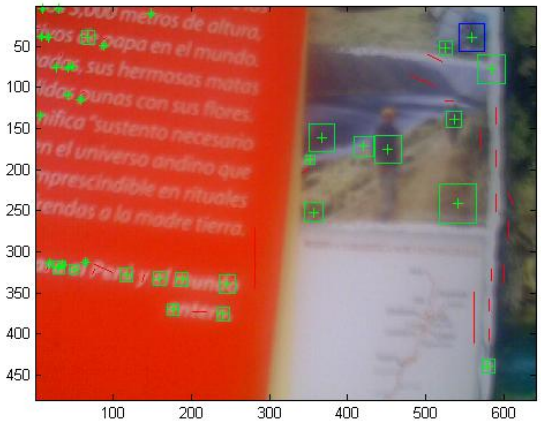

Figura 5.40: Descritores SIFT do nó pai num ângulo de 90º

Os eixos X e Y são as dimensões de cada foto e que é uma variável importante para o tempo de processamento para achar os descritores SIFT, se a foto é muito grande então o custo do processamento é maior.

Depois de fazer a leitura dos sensores em cada direção e tendo os descritores *SIFT* se procedeu ao tratamento dos dados e logo ao treinamento da rede neuronal proposto tendo os seguintes resultados para diferentes números de neurônios da camada escondida como se ilustra nas Figuras 5.41 ao 5.45.

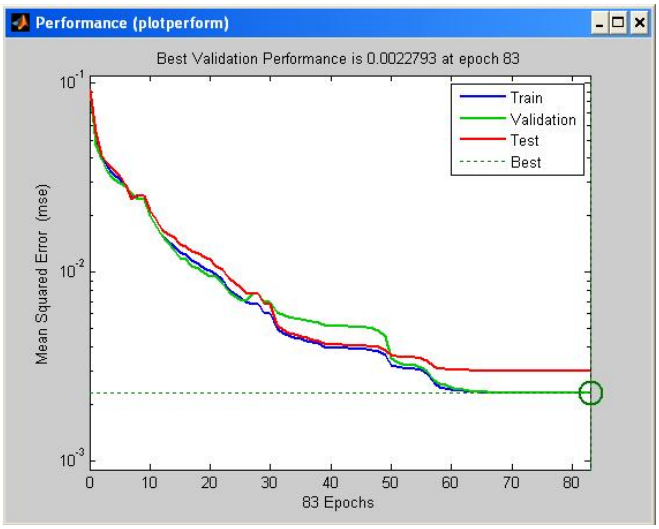

Figura 5.41: Desempenho da rede neuronal para uma camada escondida de 20 neurônios

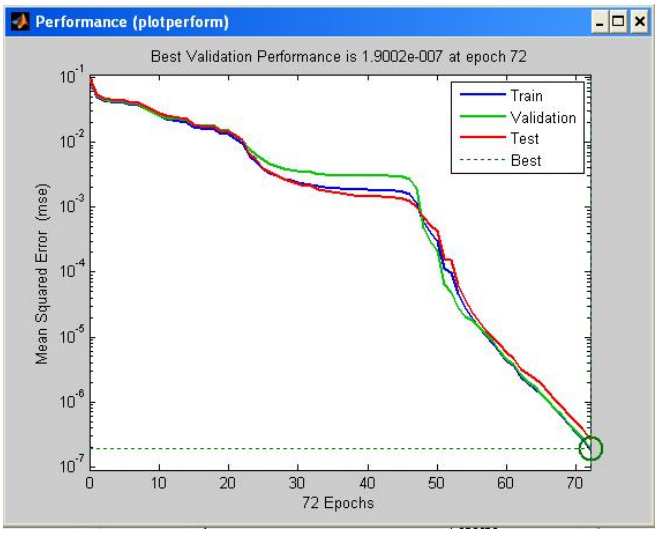

Figura 5.42: Desempenho da rede neuronal para uma camada escondida de 25 neurônios

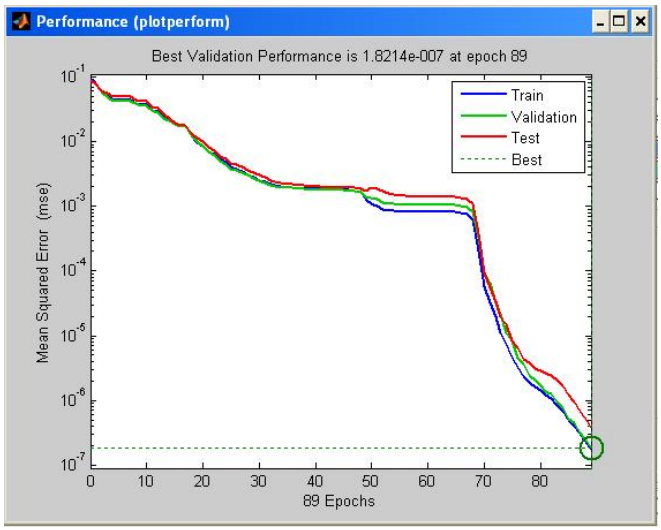

Figura 5.43: Desempenho da rede neuronal para uma camada escondida de 26 neurônios

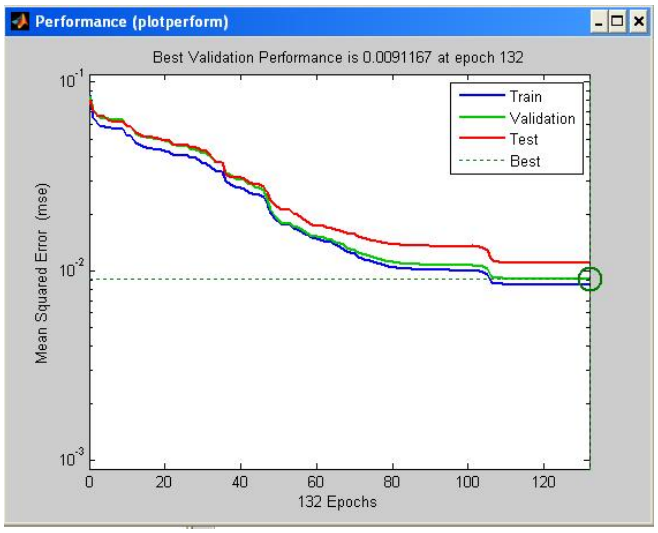

Figura 5.44: Desempenho da rede neuronal para uma camada escondida de 27 neurônios

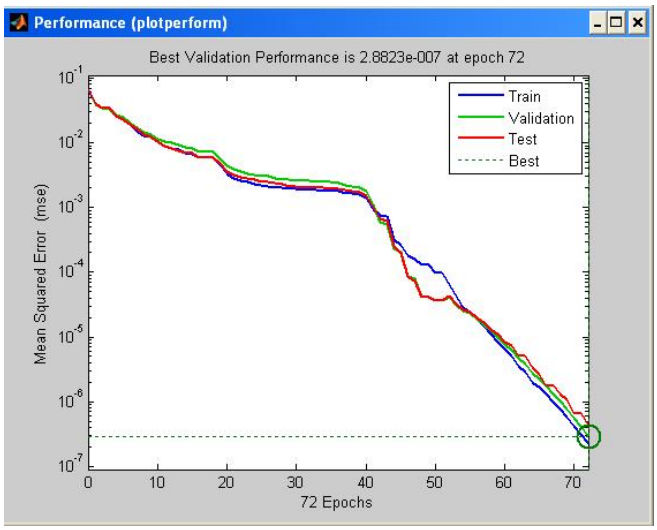

Figura 5.45: Desempenho da rede neuronal para uma camada escondida de 30 neurônios

Das gráficas podem se observar que o melhor desempenho que tem a rede neuronal é com uma camada escondida de 26 neurônios, isto quer dizer que a rede com as entradas que tem pode aprender o nó atual da exploração. Se realizou testes só com os valores dos sensores e se teve também bons resultados só que os descritores SIFT são dados importantes para fazer a diferença com outros nós.

A dificuldade de usar redes neurônais é o custo computacional que é demasiado, é por isso que foi usado o método de SVM, melhorando o tempo de treinamento e a precisão da classificação.

O SVM é ao igual que a rede neural uma caixa preta que consegue fazer o aprendizagem mas com poucas épocas ou iterações.

5.3 Experimentos de exploração completa

Neste experimento de prova de este sistema de controle de navegação, o robô tem que ser capaz de sortear um obstáculo, tomando em conta tudo o problema de navegação, o sorteio de obstáculos poderia reduzir a dois aspectos importantes:

1. Quando começa o robô a sortear o obstáculo.

2. Como sorteia o obstáculo.

5.3.1
Começo de navegação

Nesta parte da navegação, o robô para evitar um obstáculo inesperado começa a realizar cálculos das leituras dos sensores e obtém a mínima distância entre o robô e o obstáculo em cada movimento realizado no ambiente desconhecido, esta mínima distância é programada pelo controlador fuzzy, o qual ante uma presença de um obstáculo muito próximo o robô vai tratar de evadir o obstáculo para não colidir com o obstáculo, na Figura 5.46 se apresenta o momento quando o robô começa a interatuar ante a presença do obstáculo.

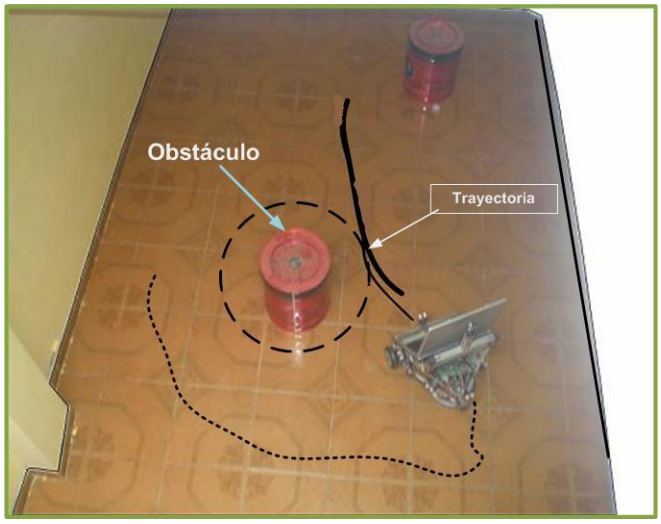

Figura 5.46: O robô ER1 evita um obstáculo

5.3.2
Sorteio dos Obstáculos

O processo de sortear obstáculos que realiza o robô consiste em seguir uma trajetória que siga o contorno do obstáculo no caso que exista só um obstáculo, mas se existisse muitos obstáculos percebidos no mesmo instante, o robô segue uma trajetória evitando colidir com qualquer obstáculo pelo tanto o robô também pode seguir uma trajetória paralela à parede como se fosse um controle *wall-following*[1] (ver Figura 5.47).

[1]controle de seguimento de parede

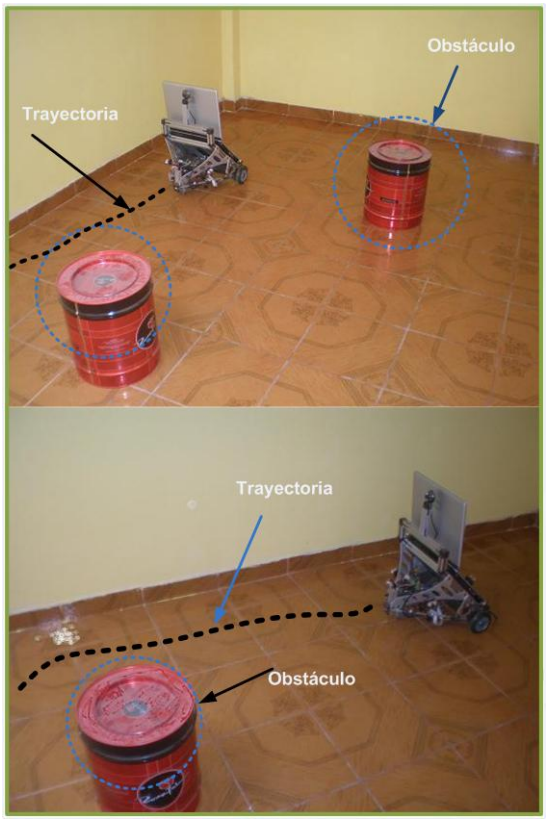

Figura 5.47: O robô faz um controle de seguimento de parede

6
Conclusão e trabalhos futuros

Nesta dissertação investigamos métodos de aprendizado de máquina onde o algoritmo support vector machine resultou um melhor algoritmo respeito à rede neuronal reduzindo o tempo de treinamento e melhorando a classificação para encontrar os nós característicos do ambiente explorado, se encontrou também que o SVM é extremosamente robusto mas nesta tese só se utilizou o SVM para substituir a rede neuronal mas poderia se usar este poderoso algoritmo para a navegação autônoma do robô.

Para a fusão dos sensores, a lógica difusa é utilizada. A aplicação desta técnica reside no fato de facilitar a tomada de decisão em regiões de ponderação, permitindo, assim, o tratamento de incerteza no processo de fusão e tem a capacidade de tomar a decisão correta frente a situações inusitadas, baseado na combinação do conhecimento embutido nas regras.

Os descritores SIFT das imagens capturadas melhorou o aprendizado dos nós aumentando a data de entrada da rede neuronal, mas o tempo de processado da imagem é longo quando a imagem tem dimensões maiores, porém isto provoca usar uma resolução menor, o que interfere a quantidade dos descritores da imagem e porém uma menor data de treinamento de aprendizagem, também é importante dizer que se a imagem não é colorida os descritores sift será muito pequeno e em conseqüência também o aprendizagem terá um umbral pequeno.

O objetivo do sistema proposto foi a navegação num ambiente pequeno desconhecido estático aplicando técnicas de inteligência computacional e visão computacional. O sistema foi validado através de simulação. Apesar do objetivo ser aplicado a ambiente fechados, seus princípios e arquitetura pode ser utilizados em outras áreas, por exemplo: na indústria, para inspeção e localização de objetos coloridos.

Respeito ao trabalho feito pelo Felipe Belo, este trabalho fez a fusão de todos os algoritmos num só algoritmo tal como foi proposto neste trabalho, verificando assim uma melhora na navegação do robô ER1.

Como trabalhos futuros pode melhorar o algoritmo aplicando *support vector machine* para a generalização e classificação do árvore de nós dentro

do espaço de exploração e também usar algoritmos genéticos e *support vector machine* para otimizar o planejamento da trajetória desde um ponto inicial até o ponto objetivo. Também na etapa de simulação poderia se utilizar o programa gazebo de 3 dimensões para ter um melhor programa de simulação. O robô poderia ser melhorado para que possa se deslocar em terrenos irregulares incluindo novos sensores como por exemplo inclinômetros y testar com uma arquitetura de controle híbrido.

Também poderia-se utilizar outro *DSP* especialmente usado para o processamento de imagens para não utilizar a *notebook* e reduzir assim o custo do projeto, também existe um microprocessador da texas instrument chamado *beagleboard* que pode ser usando para melhorar o processamento e controle em tempo real, *www.beagleboard.org*.

Referências Bibliográficas

[Ash06] ASHLOCK, D.. **Evolutionary Computation for Modeling and Optimization**. Springer Science - Business Media Inc., Unnited State of America, 2006. 2.4.2

[Ben96] PATRICK VAN DER SMAGT, B. .. **Introduction to Neural Networks**. The University of Amsterdam, Kruislaan 403, NL-1098 SJ Amsterdam, 1996. 2.4, 2.4.2, 2.4.2

[Ben99] SANDLER, B. Z.. **Robotics - Designing the Mechanisms for Automated Machinery**. A Solomon Press Book, United State of America, 1999. 1.1

[Cho95] CHOSET, H; BURDICK, J.. **International conference on robotics and automation**. In: INCREMENTAL CONSTRUCTION OF THE GENERALIZED VORONOI GRAPH., Nagoya, Japan, July 1995. IEEE, IEEE. 2.4.1

[Eve95] EVERETT, H.. **Sensors for Mobile Robots, Theory and Applications**. Natick, MA, A.K. Peters, Ltd, New York, 1995. 2.2

[Fel06] BELO, F. A. W.. **Desenvolvimento de algoritmos de exploração e mapeamento visual para robôs móveis de baixo custo**. Master's thesis, Pontificia Universidade Católica de Rio de Janeiro, Rua Marquês de São Vicente,225 - Gávea, Janeiro 2006. 4.4

[Fis81] FISCHLER, M. A.; BOLLES, R. C.. **Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography**, volumen 24. Comm. of the ACM,, 1981. 2.4.3

[Fre92] FRED MARTIN, P. O.. **The 6.270 Robots Builder's Guide**. The Massachusetts Institute of Technology, United State of America, 1992. 1.1, 1.2, 2.4, 2.4.2

[Fuk94] FUKUDA, T.. **Fuzzy-neuro-ga based intelligence robotics**. Technical report, IEEE Computation Intelligence Imitating Life, 1994. 2.4.2, 2.4.2

[Ham04] CAMPBELL, H. A. T. S. G.. **DSP-Based Electromechanical Motion Control**. CRC Press, Texas AM University, 2004. 3.4.2

[Har88] HARRIS, C.;STEPHENS, M.. **A combined corner and edge detector**. Technical report, Proc. Alvey Vision Conf., university of Manchester, p. 147-151, 1988. 2.4.3

[Jia07] SHEN, J.; HU, H.. **Svm based slam algorithm for autonomous mobile**. Technical report, Department of Computer Science, University of Essex Wivenhoe Park, colchester, 2007. 2.4.2

[Joh89] CRAIG, J. J.. **Introduction to Robotics - Mechanics and Control**. Addison Wesley Longman, United State of America, 1989. 1.1, 1.2

[Joh04] HOLLAND, J. M.. **Designing Autonomous Mobile Robots**. Linacre House, Jordan Hill, Oxford OX2 8DP, UK, United State, 2004. 1.1

[Jor03] ANGELES, J.. **Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms**. Springeer-Verlag New York, Inc, United State of America, 2003. 2.4.1

[Koz94] KOZA, J.. **Evolution of Subsumption Using Genetic programming.** Ilmenau - Germany, In: IWK'94 Ilemanu, 1994. 2.4.2, 2.4.2

[Lac00] "LACEY, A. J.; PINITKARN, N. T. N. A.. **An Evaluation of the Performance of RANSAC Algorithms for Stereo Camera Calibration**. British Machine Vision Conference, BMVC, 2000. 2.4.3

[Lak98] JAIN, L. C.. **Fusion of Neural Networks, Fuzzy Systems and Genetic Algorithms: Industrial Applications**. CRC Press, CRC Press LLC, United State of America, 1998. 2.4.2

[Les04] RUTKOWSKI, L.. **Flexible Neuro-Fuzzy Systems**. Kluwer Academic Publishers, United State of America, 2004. 2.4.2, 2.4.2, 2.4.2, 2.4.2

[Low99] LOWE, D.. **Object recognition from local scaleinvariant features**. Technical report, ICCV, Kerkyra, 1999. 2.4.3

[Mat94] MATARIC, M.. **Interaction and Intelligence Behavior**. PhD thesis, Massachusetts Institute of Technology. 2.4.1

[Mck95] JOHN, M. P.. **Introduction to Robotic**. Addison-Wesley Pub (Sd), United State, 1991. 1.1

[Mik03] MIKOLAJCZYK, K.; SCHMID, C.. **A performance evaluation of local descriptors**. Technical report, IEEE Conf. Vision Pattern recognition. V.2, 2003. 2.4.3

[Mik04] MIKOLAJCZYK, K.; SCHMID, C.. **Scale and affine invariant interest point detectors**. Technical report, Computer Vision V. 60, N°. 1, p.62–86, 2004. 2.4.3

[Sel92] M., S. J.. **Introductory Robotics**. Prentice Hall International, United State of America, 1992. 1.1, 1.2

[Sie04] NOURBAKHSH ILLAH, S. .. **Autonomous Mobile Robots**. A bradford Book - The MIT Press, United State, 2004. 1.1, 1.2, 2.1.1, 2.1.1, 2.1.1, 2.2, 2.3

[Sto96] W., S. H.. **Mars pathfinder microrover: A small, low-cost, low-power spacecraf**. Proceedings of the 1996 AIAA Forum on Advanced Developments in Space Robotics., 1996. 1.3

[Vee95] J., V. L. P.. **Analysi and Applications of Artificial Neural Networks**. Prentice Hall International, Campus 400, Maylands Avenue Heml Hempstead, 1995. 2.4.2, 2.4.2

[Wit02] JACAK, W.. **Intelligence Robotic Systems, Design, Planning and Control**. Kluwer Academic Publishers, New York, Boston, Dordrecht, London, Moscow, 2002. 1.1

[Zad84] ZADEH, L.. **Making Computers think like peaple**. IEEE Espectrum, Unnited State of America, 1984. 2.4.2, 2.4.2

[lin05] LING, H.; JACOBS, D.. **Deformation invariant image matching**. Technical report, ICCV, p. 1466-1473, 2005. 2.4.3

[spra550] YU, Z.. **3,3V DSP for Digital Motor Control**. Texas Instrument TM, Texas - Unites State. 3.3

[sprs174] INSTRUMENTS, T.. **TMS320F2810, TMS320F2812 Digital signal Processor Data Manual**. Texas Instruments Incorporated, Dallas, Junio 2002. 3.3