UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
UNIOESTE – CAMPUS FOZ DO IGUAÇU

APLICAÇÃO DE TÉCNICAS DA INTELIGÊNCIA COMPUTACIONAL AOS PROBLEMAS DE LOCALIZAÇÃO E MAPEAMENTO DE AMBIENTES DESCONHECIDOS EM ROBÓTICA MÓVEL

GLAUBER ZÁRATE VALENÇA

FOZ DO IGUAÇU - 2004

APLICAÇÃO DE TÉCNICAS DA INTELIGÊNCIA COMPUTACIONAL AOS PROBLEMAS DE LOCALIZAÇÃO E MAPEAMENTO DE AMBIENTES DESCONHECIDOS EM ROBÓTICA MÓVEL

GLAUBER ZÁRATE VALENÇA

APLICAÇÃO DE TÉCNICAS DA INTELIGÊNCIA COMPUTACIONAL AOS PROBLEMAS DE LOCALIZAÇÃO E MAPEAMENTO DE AMBIENTES DESCONHECIDOS EM ROBÓTICA MÓVEL

Monografia apresentada como requisito parcial para a obtenção do grau de bacharel em Ciência da Computação, sobre Orientação de Ana Paula Martins de Araújo e Co-orientação de João Alberto Fabro.

FOZ DO IGUACU - 2004

O que realmente me interessa é se Deus tinha uma escolha na criação do mundo.

(Albert Einstein)

Dedico o presente trabalho a Deus por ter sempre me mostrado o caminho a ser trilhado.

Agradeço aos meus pais, especialmente minha mãe, Silvana Zárate Valença, pelo incentivo, dedicação e força me passado incondicionalmente nas horas boas e principalmente nas horas difícieis, ao meu orientador João Alberto Fabro e a todos que direta e indiretamente me ajudaram a conclusão deste trabalho.

RESUMO

O presente trabalho envolve conceitos de robótica móvel autônoma, simulação e os problemas envolvidos principalmente na área de navegação e mapeamento de ambientes desconhecidos, utilizando-se para isso técnicas da Inteligência Artificial Computacional e técnicas relacionada à navegação e mapeamento de ambientes.

Para a implementação da navegação do robô foi utilizada a técnica de Redes Neurais Artificiais, foram construídas 2 (duas) Redes Neurais para a simulação de desvio de obstáculos à direita e à esquerda, sendo estas redes intercaladas durante a navegação do robô.

Para a realização do mapeamento do ambiente onde o robô está inserido, a técnica utilizada é a técnica de Grades de Ocupação, onde o ambiente é representado por uma matriz bidimensional, sendo que, cada célula da matriz corresponde uma região do ambiente.

Durante a navegação do robô é realizada a busca por pontos espalhados pelo ambiente, onde, esta busca é realizada quando o robô não está próximo a nenhum obstáculo, ou seja, livre de colisão.

Ao final da busca pelos pontos espalhados pelo ambiente é construído um mapa *fuzzy* do mesmo, onde os obstáculos presentes no ambiente são representados por um vetor de obstáculos e suas determinadas coordenadas, sendo que o conjunto dos obstáculos presentes no vetor de obstáculos forma o mapa *fuzzy* do ambiente.

Diversos experimentos em diferentes ambientes foram realizados para verificar a aplicabilidade da proposta utilizada para a construção do mapa métrico do ambiente, onde ao final deste trabalho são relatados os experimentos realizados e a conclusão do trabalho.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

IA – INTELIGÊNCIA ARTIFICIAL

RN – REDE NEURAL

RNA’S – REDES NEURAIS ARTIFICIAIS

SUMÁRIO

1 INTRODUÇÃO

A partir do momento em que os seres humanos começaram a utilizar máquinas para automatizar algumas operações, um dos grandes desejos do homem tem sido a criação de máquinas que possam operar automaticamente, sem o controle humano. Uma máquina cuja independência seja desenvolvida de acordo com seu próprio aprendizado e que tenha a capacidade de interagir com ambientes incertos (desconhecidos por ela), uma máquina que possa ser chamada de autônoma e/ou inteligente. Sendo que um dos objetivos dos estudos relacionados a robótica é o desenvolvimento de tais máquinas capazes de lidar com uma variedade de eventos inesperados no ambiente em que opera. Estas máquinas continuariam a se adaptar e realizar tais tarefas gradativamente com maior eficiência, mesmo que em condições de ambiente imprevisíveis. Então, seriam muito úteis onde a interação humana é perigosa, tediosa ou impossível; como em reatores nucleares, combate ao fogo, operações militares, e exploração do espaço.

A robótica móvel é uma área de pesquisa que lida com o controle de veículos autônomos ou semiautônomos, e uma de suas ênfases está nos problemas relacionados com a operação (locomoção) em ambientes complexos de larga escala, que se modificam dinamicamente, compostos tanto de obstáculos estáticos como de obstáculos móveis. Para operar neste tipo de ambiente o robô deve ser capaz de adquirir e utilizar conhecimento sobre o ambiente, estimar sua posição, possuir a habilidade de reconhecer obstáculos, e responder em tempo real as situações que possam ocorrer. Além disso, todas estas funcionalidades devem operar em conjunto. As tarefas de como perceber o ambiente, localizar-se e se mover são problemas fundamentais na pesquisa dos robôs móveis autônomos.

Os seres humanos são uma fonte de motivação para o desenvolvimento destas máquinas, e proporcionam diversas heurísticas para o desenvolvimento de algoritmos de aprendizado e adaptação. Assim, espera-se que algumas das

características de aprendizado e adaptação dos organismos biológicos possam ser replicadas nestas máquinas.

A complexidade envolvida na navegação de um robô autônomo em um ambiente desconhecido requer a necessidade de algoritmos complexos para a navegação adequada do robô em tal ambiente. Estudos de técnicas envolvendo tais algoritmos recaem principalmente em:

i) **Redes Neurais:** utilizadas principalmente na execução da navegação/locomoção do robô, fazendo com que este possua certas habilidades, como desviar de obstáculos, e tomar ações necessárias em decorrência dos eventos ocorridos durante a sua navegação.

ii) **Lógica Nebulosa (*Fuzzy*):** é utilizada geralmente quando há problemas a serem resolvidos e estes problemas levam a fatores como ambigüidade ou a incerteza dos dados adquiridos, porém estes dados não necessariamente precisam ser exatos ou precisos, podendo assim ser tratado utilizando a técnica da lógica nebulosa, que por sua vez é uma técnica que utiliza conceitos de conjuntos nebulosos, ou seja, conjunto formado por dados estimados por um conjunto de regras nebulosas. Um dos trabalhos relacionados que abrange essas técnicas é descrito por Fabro [FABRO 2001], onde é realizada a construção de um mapa do ambiente utilizando características nebulosas.

A navegação autônoma do robô deve simular a navegação autônoma de uma pessoa em determinado ambiente, tendo características semelhantes, como por exemplo desvio de obstáculos (móveis e estáticos), localização (saber onde está e onde pretende chegar), mapeamento do ambiente durante sua interação com este, planejamento de melhores trajetos e execução de tarefas durante a navegação.

1.1 Motivação

O desenvolvimento de sistemas de controle para robôs móveis autônomos tem se mostrado um grande desafio para a IA até os dias atuais. Diferentes abordagens para o projeto de sistema de controle para robôs móveis autônomos vem sendo utilizadas em diversas áreas de pesquisa. No entanto, os robôs móveis autônomos ainda não causaram muito impacto em aplicações domésticas ou industriais, principalmente devido à falta de um sistema de controle robusto, confiável e flexível que permita a estes robôs operem em ambientes dinâmicos, pouco estruturados, e habitados por seres humanos. O desenvolvimento de um sistema de controle que possibilite a autonomia do robô dentro do ambiente é uma das principais motivações para a pesquisa na área de robótica móvel atualmente [FARLEI 2002].

1.2 O Problema da Localização e Mapeamento

O problema que se pretende abordar consiste em, inicialmente, explorar um ambiente desconhecido, e armazenar informações sobre o mesmo em uma estrutura de dados pré-definida. Após a realização da exploração, podem ser utilizados diversos algoritmos para planejar o melhor caminho, sair em busca de determinados elementos presentes no ambiente, ou chegar a certos locais reagindo adequadamente aos eventos inesperados, como a presença de obstáculos móveis ou estáticos desconhecidos, em sua trajetória, ou mesmo a construção de um mapa nebuloso do ambiente. Para realizar esta exploração, é necessário que o robô navegue pelo ambiente sem colidir com obstáculos, e reaja com ações repulsivas ou atrativas de acordo com cada evento ocorrido durante sua navegação. Para realizar a navegação, uma técnica bastante utilizada é o uso de redes neurais artificiais, indicando ao robô qual o próximo passo a ser tomado ao longo de sua iteração com o ambiente [ZUBEN 1999]

Outro problema típico na robótica móvel autônoma é de como definir uma estrutura para realizar o mapeamento do ambiente, ou seja, qual a melhor forma de representação do ambiente de modo que esta representação possa obter eficiência

de processamento, ou seja, onde os atributos do mapa são utilizados para a elaboração de algoritmos, como por exemplo, à determinação da posição (x,y) do robô no ambiente para a verificação se o robô já passou pela determinada posição ou mesmo a determinação da posição dos obstáculos no ambiente, e se a carga de processamento para a atualização do mapa não se torna inviável pelo fato das informações necessárias para o carregamento do mapa ser oriundas dos sensores do robô e, estas informações necessitarem de um pré-processamento para a atualização do mapa, como por exemplo, a determinação da posição dos obstáculos no ambiente.

Existe ainda um outro problema inerente à robótica móvel autônoma que é a incerteza dos dados obtidos pelos sensores no processo de mapeamento do ambiente, devido à existência de ruídos que são gerados pelos sensores do robô.

1.3 Organização do Trabalho

O restante deste trabalho é organizado como a seguir:

- **Capítulo 2:** neste capítulo é apresentada a revisão bibliográfica do trabalho em questão, onde são relatadas as técnicas de IA relacionadas à robótica móvel autônoma, bem como as principais técnicas de navegação e mapeamento de ambientes.
- **Capítulo 3:** neste capítulo são abordadas as arquiteturas para a implementação de sistemas de controle para robôs móveis autônomos.
- **Capítulo 4:** neste capítulo são relatadas as técnicas que foram implementadas no simulador robótico Khepera, bem como o porquê de sua utilização.
- **Capítulo 5:** neste capítulo são relatados os experimentos realizados durante a implementação das técnicas, principalmente no que diz respeito à construção dos padrões de treinamento para a RNA.
- **Capítulo 6:** neste capítulo é apresentada a conclusão do trabalho bem como propostas para trabalhos futuros.

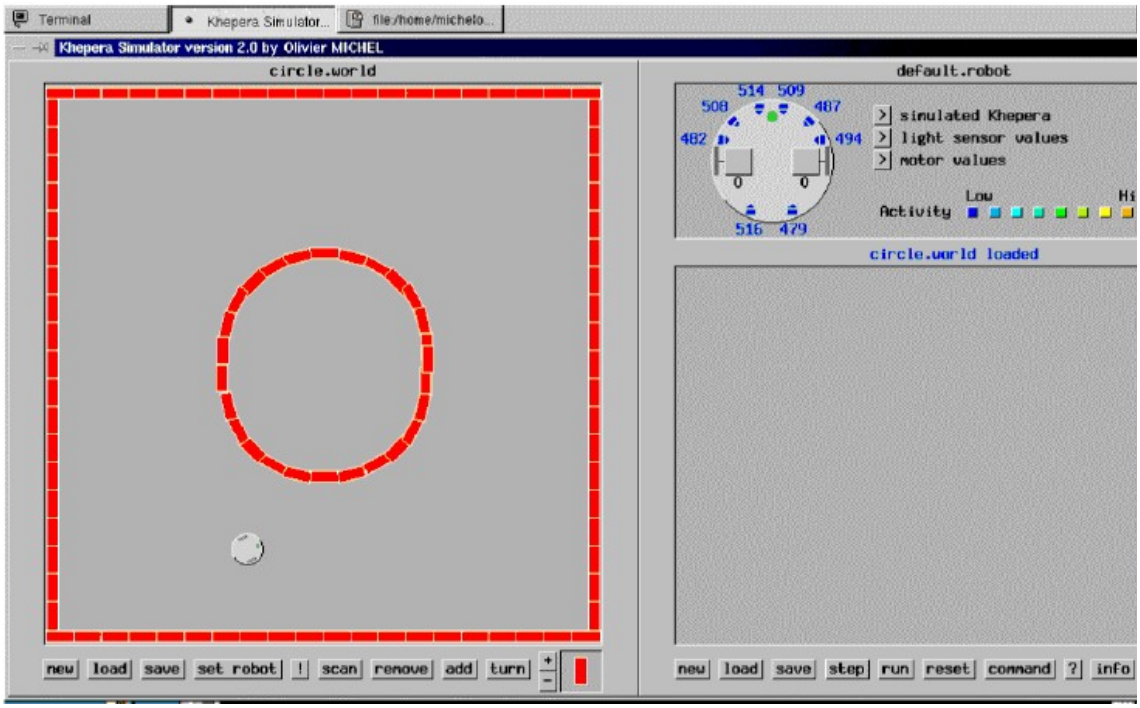
2 REVISÃO BIBLIOGRÁFICA

2.1 Introdução

Neste capítulo são apresentados os conceitos necessários à compreensão do restante do trabalho. O trabalho envolve conceitos de robótica móvel, simulação, os problemas envolvidos (principalmente localização e mapeamento), além de técnicas computacionais utilizadas para resolver os problemas. Entre estas técnicas, os conceitos de redes neurais, sistemas nebulosos e algoritmos genéticos, que juntos compõe a área de “inteligência computacional”, serão também abordados. Ao final do capítulo, trabalhos correlatos sobre a solução dos problemas de mapeamento e localização são apresentados, fornecendo uma base teórica sobre o assunto.

2.2 O Simulador

O simulador (figura 1) é responsável pela simulação de um robô dentro de um determinado ambiente, foi desenvolvido para realizar pesquisa em robótica móvel, utilizando para isso vários algoritmos como: redes neurais, lógica clássica, lógica nebulosa, ou qualquer outro tipo de algoritmo imaginável. O simulador funciona tanto em plataforma Windows quanto Unix, e pode ser implementado utilizando-se das linguagens C, C++ e Java.



Fonte: [MICHEL 1996]. **Figura 1** – O Simulador de Robôs Autônomos Khepera.

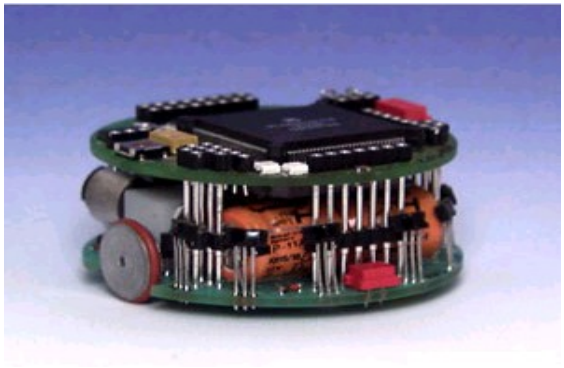
No simulador estão disponíveis quatro mundos pré-definidos, porém o simulador permite a criação de novos mundos. Ao iniciar a simulação, duas janelas são abertas, uma contendo as informações referentes aos sensores do robô, e outra contendo o mundo a ser explorado.

Um simulador robótico é o meio essencial para a elaboração de trabalhos científicos relacionados à área da robótica móvel autônoma, pois com ele é possível realizar experimentos e validações de técnicas computacionais relacionados à área da robótica móvel autônoma.

2.2.1 O Robô

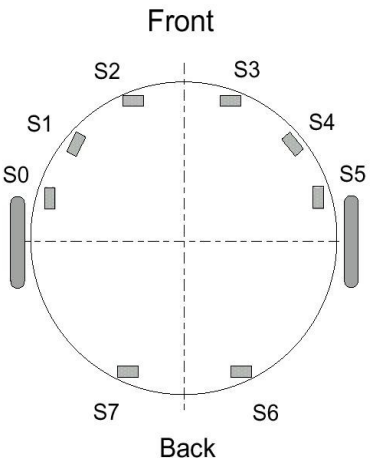
O robô modelado para utilização das simulações é do tipo Khepera (figura 2). É um robô cilíndrico, de 5 cm de diâmetro e 3 cm de altura, pesando 70 gramas, que se locomove por meio de motores de passo atuando sobre duas rodas, com codificador incremental onde 12 pulsos equivalem a um milímetro de

avanço. Cada um dos motores pode variar sua velocidade de 2 a 60 centímetros por segundo.



Fonte: [MICHELON 2000]. **Figura 2** – Robô Khepera

O robô possui um formato arredondado facilitando assim sua locomoção no ambiente. Possui também oito sensores infravermelhos dispostos ao seu redor, sendo seis destes na parte frontal e dois na parte de trás (figura 3). Este sensores fornecem informações tanto de distância à obstáculos, como luminosidade do ambiente. O robô pode detectar a proximidade dos objetos dispostos no ambiente, sendo que cada sensor retorna um valor entre 0 (zero – não há obstáculo sendo detectado na direção deste sensor) e 1023 (um mil e vinte e três – há um obstáculo muito próximo a este sensor – encostado). O robô em questão possui habilidades limitadas, conseqüentemente não executando tarefas mais complexas (como por exemplo o mapeamento do ambiente), havendo assim a necessidade da implantação de novos módulos, para a execução de tais tarefas.



Fonte: [MICHELON 2000]. **Figura 3** – Esquema de Distribuição dos Sensores.

2.2.2 Comportamento

Basicamente o comportamento de um robô inserido em um ambiente artificial simula o comportamento de uma situação real, neste caso é necessário que o robô possua habilidades suficiente para desempenhar seu papel dentro do ambiente, como por exemplo o desvio de um obstáculo, sua localização, determinação de trajetos, e execução de tarefas pré-determinadas. Portanto, para isso se tornar possível, o robô necessita de informações (conhecimento) sobre o ambiente em que se encontra, para realizar tais tarefas. Nas próximas seções são descritas algumas técnicas utilizadas para fornecer ao robô meios para que tais tarefas sejam executadas.

2.3 Redes Neurais

Redes neurais artificiais são sistemas de processamento de informação formados pela interconexão de unidades simples de processamento, denominados neurônios artificiais. As redes neurais recebem essa denominação devido ao fato de serem originadas de um modelo matemático simplificado de neurônios naturais do cérebro humano.

As redes neurais artificiais são compostas por nodos (neurônios artificiais) que calculam determinadas funções, geralmente não–lineares. Tais nodos formam camadas, que normalmente são denominadas camada de entrada (que não executa nenhuma funcionalidade, serve simplesmente para receber os sinais de entrada e passá-los para a próxima camada), camada intermediária (seus nodos recebem os valores provenientes da camada de entrada e os processam, passando seus resultados para a próxima camada) e camada de saída (produz o resultado da rede), sendo estas interligadas entre si. Podem existir uma ou mais camadas intermediárias.

2.3.1 Motivação

A motivação que está por trás deste paradigma alternativo de processamento de informação, está no fato da possibilidade de se elaborar soluções eficazes para problemas que seriam de difícil solução sem a utilização desta técnica.

A capacidade das redes neurais artificiais aprenderem com determinadas situações e agir a partir desta aprendizagem, é sem dúvida um paradigma que apresenta vantagens para soluções de problemas de diversas naturezas.

Nos últimos anos o estudo referente a esta técnica tem levado a soluções eficientes a problemas invariavelmente difíceis de serem resolvidos com relação aos paradigmas atualmente utilizados.

Uma motivação significante para a evolução da técnica de redes neurais artificiais deve sem dúvida ao fato da própria evolução da natureza dos problemas computacionais.

2.3.2 Aplicação

As redes neurais artificiais são normalmente utilizadas em problemas de análises multivaloradas de dados, e esta por sua vez pode se manifestar em três situações básicas:

i) Treinamento Supervisionado: os dados correspondem a relações de entrada e saída. Se fenômenos dinâmicos estão envolvidos, aspectos temporais estarão presentes. Caso contrário, apenas aspecto da distribuição espacial dos dados estarão em jogo [ZUBEN 1999]. A informação contida nas amostras pode advir da relação de causa e efeito, atributos e classes, sintomas e diagnósticos, ações de controle. Normalmente estas relações expressam associações não lineares entre as variáveis independentes (vetor de entrada), e as variáveis dependentes (vetor de saída), sendo que geralmente o propósito é sintetizar um mapeamento não-linear de entrada e saída a partir das amostras disponíveis. Em síntese, existe alguma lei que reage o comportamento de entrada e saída e as amostras retratam a aplicação desta lei para alguns casos específicos.

ii) Treinamento Não-Supervisionado: os dados correspondem a uma lista de atributos. Agora, as amostras não possuem a relação entre variáveis dependentes e independentes. Cada amostra descreve apenas uma lista de atributos que caracterizam um evento ou um objeto. A informação que se quer extrair a partir das amostras pode ser de diversas naturezas, em linhas gerais sabe-se que pontos próximos no espaço de atributos correspondem a amostras com forte semelhança nos atributos, assim como se sabe que pontos uniformemente distribuídos no espaço de atributos correspondem a amostras que apresentam o mesmo nível de correlação de atributos entre si.

iii) Memória Endereçável por Conteúdo: Os dados correspondem a amostras ruidosas ou com conteúdo parcial de memórias previamente armazenadas. O objetivo é recuperar a memória armazenada a partir de sua versão ruidosa ou

incompleta presente na amostra, ou seja, realizar o que se conhece por memória endereçável por conteúdo ou memória associativa.

Para tanto existem redes neurais recorrentes capazes de sintetizar sistemas dinâmicos não lineares que apresentam como pontos de equilíbrio exatamente as memórias disponíveis. Sendo assim, partindo de uma condição inicial equivalente a uma versão ruidosa ou com conteúdo parcial de uma das memórias, a simples sujeição da rede neural a uma dinâmica de relaxação é suficiente para conduzir a recuperação de uma das memórias armazenadas, pois vai ocorrer seguramente convergência para um ponto de equilíbrio.

Todos os problemas geralmente se encaixam e um dos três itens acima mencionado, no entanto a utilização de redes neurais não implica necessariamente na garantia de bom desempenho, muito menos na garantia de que ferramentas alternativas não produzirão desempenho superior, ou então o mesmo desempenho, mas com menor esforço. Porém, se redes neurais não resolvessem determinados problemas com grande eficiência e robustez, elas não estariam despertando tanto interesse ao redor do mundo. O essencial portanto é não superestimar o poder desta ferramenta, sem entretanto subestimá-la.

2.3.3 Arquitetura das RNA's

As conexões entre as camadas da rede neural podem gerar inúmeras estruturas diferentes, dependendo da necessidade do projetista, basicamente pode ser formada por uma ou mais camadas intermediarias. Tendo em vista estas características das RNA's, diferentes arquiteturas foram elaboradas, dentre elas uma arquitetura clássica utilizada é a que utiliza o modelo Perceptron.

2.3.3.1 Modelo Perceptron de Múltiplas Camadas (PMC)

Uma rede neural do tipo PMC é formada pelos seguintes elementos:

- **Camada de entrada**: cada elemento da camada de entrada faz a distribuição do sinal que ele recebe para todas as unidades de processamento.
- **Neurônios**: são constituídos de unidades sigma e de funções de transferência. As unidades sigma fazem a soma ponderada dos sinais vindos pelas conexões com os elementos de entrada. As funções de transferência determinam a saída de cada neurônio em função da soma ponderada.
- **Camada de saída**: onde os dados calculados pela camada intermediária serão atribuídos, para que com isso possa se prosseguir o processamento da codificação externa.

Quando a RNA possui uma única camada, os padrões de treinamento apresentados à camada de entrada são mapeados diretamente em um conjunto de padrões de saída da RNA, neste caso a codificação do mundo exterior deve ser suficiente para implementar esse mapeamento, essa restrição implica que padrões de entrada similares e saídas contraditórias não podem ser mapeados com RNA que possua essa representação. Contudo, o desenvolvimento do algoritmo de treinamento *backpropagation*, por Rumelhart, Hinton e Williams em 1986 [RUMELHART 1986], precedido por propostas semelhantes ocorridas nos anos 70 e 80, mostrou que é possível treinar eficientemente redes com camadas intermediárias, resultando no modelo de rede neural artificial mais utilizado atualmente, a rede PMC treinada com o algoritmo *backpropagation*.

2.4 LÓGICA NEBULOSA

2.4.1 Introdução

Quando temos problemas a serem resolvidos e estes problemas levam a fatores como ambigüidade e incerteza, para a resolução de tais problemas não necessitamos necessariamente de um conhecimento específico para obter a solução, ou seja, informações vagas ou não tão precisas adquiridas por

experiências passadas podem ajudar a resolver tais problemas. Nos problemas de difícil solução, onde se faz necessário o auxílio matemático/computacional, tais problemas se tornam ainda mais difíceis de serem resolvidos, pois a matemática convencional não trabalha com valores ambíguos ou imprecisos. Para lidar com isso de forma matemática, Lotfi A. Zadeh [ZADEH 1988] elaborou na Universidade da Califórnia em Berkeley um artigo, e dentre suas idéias ele introduziu o conceito de "variáveis lingüísticas" (admitem como valores apenas expressões lingüísticas, como "muito grande", "pouco frio", "mais ou menos jovem") Este artigo propôs a definição de tais variáveis através de conjuntos *fuzzy*, e esta teoria permite que sejam tratados níveis de incerteza e ambigüidade.

2.4.2 Conceito

Uma coleção de objetos, onde nada em especial é assumido sobre a natureza dos objetos individualmente, é chamado de conjunto. Na teoria clássica dos conjuntos, um determinado elemento pertence ou não a um conjunto específico, ou seja, são determinados os limites para determinar a pertinência de um dado elemento a um conjunto. Mais formalmente: seja U uma coleção de objetos, chamado universo de discurso, u um elemento particular deste universo, e A um dado conjunto contido em U, então, ou "u pertence a A" ou "u não pertence a A". Para cada elemento do universo de discurso pode-se, portanto, determinar sem ambigüidades quando pertence ou não a um conjunto. Podemos descrever um conjunto pela enumeração de todos os seus componentes, desde que seja contável, através de uma propriedade que deve ser satisfeita por seus membros. Porém muitos conjuntos e proposições não podem sempre ser determinadas de maneira tão exata. Por exemplo, a determinação de pessoas “altas” é um conjunto onde um limite exato não pode ser precisamente definido. Na lógica nebulosa a pertinência de elemento a um conjunto ocorre gradativamente, sendo expressa através de uma função de pertinência.

2.4.3 Funções de Pertinência

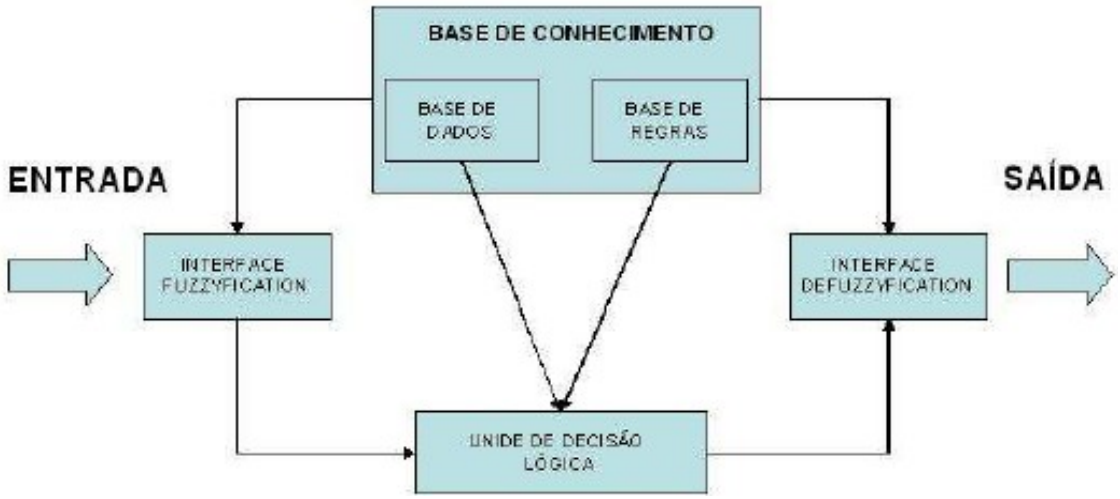
Cada conjunto fuzzy (nebuloso), A, é definido em termos de relevância a um conjunto universal, X, por uma função denominada *função de pertinência,* associando cada elemento x um número A(x), no intervalo fechado [0,1], que caracteriza o grau de pertinência de x em A.

2.4.4 Variáveis Lingüísticas

Uma das grandes vantagens da utilização da lógica *fuzzy* é a utilização das variáveis lingüísticas, pois com elas é possível transformar a linguagem natural em um conjunto de números, permitindo assim a sua manipulação computacional. As variáveis lingüísticas assumem valores lingüísticos, como por exemplo, FRIO, MORNA, QUENTE, quando relativos a variável TEMPERATURA DA ÁGUA.

2.4.5 Sistemas Nebulosos (Fuzzy)

Os sistemas *fuzzy* estimam funções a partir de uma descrição parcial do comportamento do sistema, onde um especialista deve prover o conhecimento heurístico, ou esse conhecimento pode ser inferido a partir de dados de entrada e saída do sistema. Um sistema de inferência pode ser composto por cinco blocos principais [OLIVEIRA 1999].



Fonte: [ZADEH 1988]. **Figura 4** – Sistema de Inferência *Fuzzy*

Base de Regras → contém um conjunto de regras nebulosas, onde variáveis antecedentes/conseqüentes são variáveis lingüísticas, e os possíveis valores de uma variável lingüística são representados por conjuntos nebulosos.

Base de Dados → define as funções de pertinência do conjunto nebuloso.

Unidade de Decisão Lógica → realiza as operações de inferência, para obter a partir da avaliação dos dados de entrada com as condições impostas pela base de regras, uma ação a ser realizada pelo sistema.

Interface de Fuzificação → utilizando as funções de pertinência pré-estabelecidas, ela mapeia cada variável de entrada do sistema em graus de pertinência.

Interface de Defuzificação → transforma os valores nebulosos, provenientes da unidade de decisão lógica, em valores de saída.

2.5 MÉTODOS DE NAVEGAÇÃO:

Existem dois tipos básicos de problemas de navegação. O primeiro está relacionado ao planejamento de caminhos em ambientes totalmente conhecidos, e o segundo está relacionado a ambientes desconhecidos, onde se utiliza apenas informação local obtida através dos sensores [MCK 1995].

Quando se possui um conhecimento total da estrutura do ambiente onde o robô está inserido, e sua localização dentro dele, utilizam-se técnicas de planejamento global de trajetórias. Quando não se possui este conhecimento total, ou quando o robô não possui informação nenhuma sobre o ambiente, e estas informações são oriundas somente dos sensores, utilizam-se técnicas de planejamento local. Sendo assim, é realizada a construção de um mapa sobre o ambiente a ser explorado. Em geral os métodos locais são chamados de reativos e são utilizados para tratar eventos inesperados. Um dos métodos locais amplamente conhecido é o método de seguir paredes. Quando existem informações parciais sobre a estrutura do ambiente, e o robô complementa ou atualiza estas informações através de seus sensores, combinando informações globais e locais, temos os métodos híbridos de planejamento.

2.5.1 Planejamento Exploratório:

É uma técnica de planejamento local. A construção de um mapa é feita através de uma longa interação do robô com o ambiente a ser desbravado. O robô utiliza sensores para mapear o ambiente e propor estratégias de busca a objetos ou caminhos a serem percorridos.

Um mapa pode ser obtido através de uma navegação direcionada ou através do planejamento exploratório [PRESTES 2003].

As estratégias comumente usadas [ROM 2000] são divididas em duas classes: abordagem reativa, e abordagem baseada modelo.

A abor dagem reativa geralmente usada é a abordagem de seguir paredes [MCK 1997, MAT 1990, CRO 1985] APUD [PRESTES 2003] , ela consiste em guiar o robô paralelamente às paredes, enquanto extrai o mapa topológico [MAT 1990] APUD [PRESTES 2003], ou geométrico [CRO 1985] APUD [PRESTES 2003] do ambiente. Por outro lado existem métodos capazes de determinar qual passo a ser tomado baseado no estado corrente da navegação do

robô. Esta função é chamada de função de custo, função de recompensa ou função heurística [PRESTES 2003]. Um exemplo é a exploração baseada em fronteiras de Yamauchi [YAM 1997], onde os limites entre as regiões conhecidas e desconhecidas do ambiente, chamadas fronteiras, direcionam o robô através de um comportamento baseado no clássico algoritmo busca em profundidade. Similarmente, Thrun [THR 1999] desenvolveu uma estratégia onde uma função heurística é calculada iterativamente através de um algoritmo da programação dinâmica chamado iteração valorada. Após a convergência a função indica o caminho a ser seguido a fim de minimizar a quantidade de regiões desconhecidas pelo robô.

O planejamento exploratório pode ser classificado em [PRESTES 2003]:

i) Guiado por Objetivo:

Quando o robô não possui nenhum conhecimento prévio sobre o ambiente e deve obter informações sobre objetos dispostos pelo ambiente a partir unicamente pela leitura de seus sensores. Na maioria das vezes a solução para este problema é a modelagem em grafo utilizando o algoritmo de busca conveniente. Em alguns casos também se realiza a construção parcial de um mapa utilizado para alcançar os objetivos pré-determinados.

ii) Não Guiado:

O objetivo é a aquisição e o armazenamento da estrutura do ambiente. Um exemplo e a estratégia de Sigthseer [RAO 1993] APUD [PRESTES 2003], que necessita que exista pelo menos um obstáculo visível ao robô. A estratégia basicamente consiste em uma busca em profundidade, onde o robô sempre deve visitar o obstáculo mais próximo e marcá-lo como visitado; se todos os obstáculos ao redor tiverem sido visitados o robô retorna ao obstáculo mais recentemente visitado e repete o processo. Ao final é gerado um grafo não direcionado mostrando a ordem da visita do grafo.

No processo exploratório os obstáculos são os protagonistas, eles definem as regiões do ambiente onde são determinadas as ações atrativas ou repulsivas

do robô. Além disso, eles impõem a ordem da exploração, que na maioria das vezes e baseada na distância deles ao robô. É possível também observar que a exploração está intimamente ligada à busca em grafos [PRESTES 2003].

2.5.2 Componentes Básicos:

Para que o sistema seja aplicado em um robô real, é necessário que os seguintes problemas sejam resolvidos [PRESTES 2003]:

i) percepção/representação: qualquer decisão executada pelo robô, precisa ser feita através da leitura de seus sensores e de seu conhecimento global.

ii) auto–localização e mapeamento concorrente: para realizar a construção de um modelo confiável do ambiente, é necessário levar em conta os erros executados pelo robô durante sua navegação no ambiente. Esses erros correspondem aos erros de odometria, que são observados através de comparações entre a percepção local e do conhecimento global sobre o ambiente, e devem ser reduzidos através de um processo de auto-localização. além disso o processo de mapeamento do ambiente deve ser processado em tempo factível durante sua navegação atualizando seus dados através dos sensores embarcados ro robô.

iii) planejamento de movimentos: o robô deve saber qual o melhor caminho a seguir baseado no conhecimento armazenado oriundo de seus sensores.

2.5.3 Representação do Ambiente:

Uma das questões fundamentais em robótica móvel é como o ambiente será representado internamente no robô. As representações comumente utilizadas na literatura são as grades de ocupação, objetos, lugares, rotas [ROM 2000]. Em

geral essas representações podem ser enquadradas em dois grandes grupos: mapas métricos e mapas topológicos.

i) Mapas métricos: os mapas métricos podem ser subdivididos naqueles que realizam a decomposição espacial do ambiente (como as grades de ocupação), e aqueles que utilizam uma representação geométrica, como os mapas baseados em características e os mapas probabilísticos.

As grades de ocupação [ELF 1987, ELF 1989] APUD [PRESTES 2003] representam o ambiente como uma matriz de células, onde cada célula esta relacionada com uma região quadrada do ambiente e armazena um valor que indica a probabilidade desta área estar ocupada. Este tipo de representação possui problemas de granularidade, escalabilidade e extensibilidade [BAI 2001] APUD [PRESTES 2003].

A tabela abaixo representa algumas vantagens e desvantagens dos mapas métricos e topológicos.

Tabela 1 – Vantagens e desvantagens relacionada aos mapas métricos e topológicos.

MÉTRICOS	***TOPOLÓGICOS***
Requer determinação da posição precisa do robô	**Não necessita da posição especifica.**
Fácil de construir mas não de manter	**Difícil de construir e manter.**
Dependendo do tamanho do ambiente e da configuração das células, o planejamento de trajetórias pode se tornar um problema grave.	**Permite eficiente planejamento.**
Alto custo computacional.	**Baixo custo computacional.**

Os mapas baseados em características representam o ambiente através da localização de todas as características relevantes do ambiente e suas propriedades geométricas (mapas de elevação, mapas poligonais, bolhas tridimensionais) [CRO 1985, DEV 1995, CHA 1996] APUD [PRESTES 2003]. A principal vantagem na utilização de primitivas geométricas é a eficiência na

representação do ambiente. Entretanto neste tipo de situação existem 3 principais problemas [ROM 2000].

- Perda da estabilidade: pois a representação pode ser prejudicada por pequenas variações das leituras dos sensores.
- Dificuldade na localização: pois devido à representação ser baseada em primitivas geométricas é possível que a representação não seja única.
- Perda de poder expressivo: decorrente da dificuldade de representar completamente todas as características do ambiente através de primitivas geométricas.

Em relação às grades de ocupação, [DAM 1999] APUD [PRESTES 2003] faz as seguintes comparações:

- se o ambiente é desconhecido, um mapa baseado em características é muito mais complicado de ser construído.
- um mapa baseado em características é muito mais difícil de ser construído em ambientes não estruturados, devido à dificuldade de modelar alguns obstáculos usando primitivas geométricas.
- a posição do robô pode ser calculada de forma mais eficiente utilizando mapa baseado em características do que utilizando grades de ocupação.
- a principal vantagem da utilização da representação baseada em grade é a possibilidade de representar tanto o espaço livre quanto o espaço ocupado do ambiente, os quais podem ser utilizados em algoritmos de navegação.

Os mapas probabilísticos são ferramentas para tratar as incertezas espaciais originadas dos sensores e dos movimentos do robô na representação espacial, ao invés de considerá-las como um problema geométrico a parte a ser tratado [SMI 1987] APUD [PRESTES 2003]. Neste tipo de representação, tanto os obstáculos encontrados quanto o robô são representados em um único conjunto de objetos descritos pela sua localização juntamente como uma matriz de covariância que descreve o relacionamento espacial entre eles [HEB 1985] APUD [PRESTES 2003]. Este tipo de representação é utilizado junto com o filtro de Kalman [PRESTES 2003].

2.5.4 Filtro de Kalman

O filtro de Kalman foi proposto em 1960 por R. E. Kalman, como uma ferramenta recursiva para a o problema de filtragem linear de dados discretos. Sua principal característica é estimar dados passados com dados correntes e futuros mesmo quando a dinâmica é desconhecida [PRESTES 2003]. Esta técnica de filtragem tem por característica combinar todas as medidas de dados disponíveis, mais o conhecimento prévio sobre o sistema e dispositivos de medida, para produzir uma estimativa das variáveis desejadas de tal maneira que o erro é minimizado estatisticamente.

Esta propriedade é utilizada para atualizar e corrigir o estado do sistema e sua incerteza associada. Essa estratégia tem sido usada em ambientes externos, subaquático [SMI 1997] APUD [PRESTES 2003], e internos [FED 1999, LEO 1992] APUD [PRESTES 2003].

O principal problema é a própria matriz de covariância (que é determinada por uma matriz identidade para dar o inicio do processo da filtragem). Esta matriz cresce quadraticamente com a quantidade de objetos observados, resultando em alto custo computacional quando o ambiente a ser mapeado é grande, o que inviabiliza sua aplicação em aplicações de tempo real [DIS 2001] APUD [PRESTES 2003].

Resumindo, o filtro de Kalman pode ser utilizado sem restrições em ambientes onde o número de *landmarks* é pequeno, pois neste caso, os recursos computacionais exigido são limitados e o tempo de resposta é menor. Caso contrário, o processo de filtragem torna-se geralmente inviável.

ii) Os mapas topológicos correspondem a um grafo composto de vértices e arestas, onde os vértices correspondem aos locais distintos, e as arestas correspondem às ligações entre diferentes locais contendo informações que permitirão ao robô navegar entre os vértices [KUI 1991] APUD [PRESTES 2003]. Os vértices correspondem aos locais distinguíveis do ambiente baseados no

padrão observados nos dados sensoriais [BAI 1999] APUD [PRESTES 2003]. Estes padrões são representações comuns de portas, intersecções de corredores. Em ambientes internos as arestas correspondem a corredores e os caminhos médios correspondem a vértices [PRESTES 2003].

De acordo com Dudek [DUD 1993] APUD [PRESTES 2003], a principal vantagem desta abordagem é a sua natureza qualitativa e sua relação com as teorias da cognição humana, além disso os mapas topológicos são geralmente menos complexos que os mapas geométricos, e muito mais eficiente no planejamento de caminhos.

2.5.5 Auto-localização

O problema da auto-localização corresponde a uma área vasta da robótica móvel que tem recebido muita atenção durante as ultimas décadas, pois se um robô não sabe sua posição, ele não pode eficientemente planejar movimentos, localizar ou alcançar objetivos [OLS 1999] APUD [PRESTES 2003].

Bailey [BAI 2001] APUD [PRESTES 2003], destaca que um robô autônomo genuíno é aquele que consegue se auto-localizar em um ambiente usando apenas seus sensores embarcados sem a necessidade que o ambiente seja artificialmente modificado.

Planejadores de movimento clássico contam apenas com odômetros para determinar sua posição. Este mecanismo, chamado de *dead-reckoning*, corresponde a integrar os dados fornecidos pelos odômetros das rodas do robô, os quais contam a quantidade de rotações realizadas por cada roda. Este processo provou ser suscetível a erros [THR 1999] por diversos motivos: insuficientes mecanismos de alinhamento das rodas; derrapagem; erros do sinal dos sensores; variações nas trajetórias devido as diferenças de superfície; entre outros. Em particular os odômetros geram dois tipos de erros: sistemáticos e não sistemáticos. Os erros sistemáticos são gerados a partir da plataforma móvel, e os erros não sistemáticos são gerados a partir da interação de tal plataforma móvel, ou seja, a interação do robô com o ambiente.

As estratégias para solução da auto-localização podem ser divididas em dois grupos: técnicas de posicionamento relativo e absoluto. O posicionamento relativo leva em conta os odômetros do robô, e o posicionamento absoluto é baseado nos sensores absolutos como por exemplo o GPS - *Global Positioning System* – Sistema de Posicionamento Global [BET 1994] APUD [PRESTES 2003], cuja localização é conhecida no ambiente.

A maioria dos métodos de localização falha ao realizar os cálculos de posição do robô, devido à limitação da capacidade dos sensores de proximidade utilizados. Roy [ROY 1999] propõe uma solução chamada navegação costeira, onde o robô navega próximo às regiões que contém alto conteúdo informativo, baseando-se na navegação em barcos onde estes não possuem o GPS, esta técnica utiliza localização markoviana (caso um estado do ambiente contenha toda a informação relevante então ele é chamado de markoviano ou que tem a propriedade de Markov). Pode-se observar esta propriedade tomando-se, por exemplo, o movimento de uma bola atirada por um canhão. Para determinar seu vôo futuro, não importa com que velocidade saiu e de que posição veio, basta conhecer a velocidade e a posição atual. Quando utilizada sobre uma representação baseada em grades, esta técnica guia o robô por aquelas regiões que possuem o maior grau de informação, ou seja, regiões com o maior grau de ocupação.

A localização markoviana é também utilizada no problema de localização global. Também chamado de o problema do robô raptado, este problema foi proposto por Engelson [ENG 1994] APUD [PRESTES 2003], e consiste em determinar a posição do robô em um *frame* de referência global com a incerteza sobre sua posição.

2.5.6 Mapeamento e Localização Concorrentes:

A maioria das aplicações em robótica móvel necessita que o robô represente o ambiente em uma estrutura interna, na forma de um mapa, para que com este o robô possa desempenhar atividades e objetivos dentro do determinado ambiente, é necessário que este mapa seja confiável e estável e que seja construído durante o desbravamento do ambiente, para que com isso possa realizar de forma eficiente sua localização e planejamento de rotas.

O mapeamento do ambiente é necessário principalmente em casos em que o ambiente é desconhecido. Ele consiste em determinar entidades de interesse, tais como: *landmarks*, obstáculos, objetos em um *frame* de referência global [FOX 1999] APUD [PRESTES 2003].

O problema do mapeamento de um ambiente está estritamente ligado a auto-localização do robô no ambiente, pois para se construir o mapa, o robô necessita saber sua posição atual para, e precisa do mapa para se auto-localizar. Ambos os problemas levam a outro problema: Mapeamento e Localização Simultâneos ou Concorrentes [BET 1994] APUD [PRESTES 2003].

O mapeamento e localização concorrentes eliminam a necessidade de uma infra-estrutura artificial para a localização do robô e de conhecimento a priori topológico ou geométrico sobre o ambiente [PRESTES 2003].

Com a solução deste problema é possível que o robô execute a navegação em ambientes como: exploração planetária, ambientes sub-aquáticos, mineração e aplicações militares.

De acordo com Dissanayke [DIS 1999] APUD [PRESTES 2003], os problemas relacionados ao mapeamento e localização concorrentes seguem 3 principais filosofias:

i) a mais popular é a utilização do filtro de Kalmam. Sua popularidade deriva tanto da solução recursiva ao problema da navegação como o mecanismo para calcular a incerteza sobre a localização dos *landmarks* do veículo.

ii) a segunda filosofia é a utilização do conhecimento qualitativo dos locais relativos dos *landmarks* e do veículo [KUI 1991] APUD [PRESTES 2003].

iii) a terceira usa grades de ocupação probabilística juntamente com o método de relaxação para a navegação, e o mapeamento e localização concorrentes [THR 1999, YAM 1997].

2.6 Trabalhos Relacionados:

Inicialmente, uma proposta para solução do problema de mapeamento e localização concorrente foi descrita por Smith e Durrant–White [LEO 1992] APUD [PRESTES 2003], o qual estabelece uma base estatística para descrever o relacionamento entre *landmarks (*marcas terrestres) e as incertezas geométricas manipuladas. O elemento chave deste trabalho foi mostrar que existe um alto grau de correlação entre as estimativas da localização dos *landmarks* e que essas correlações crescem a medida que novas observações são obtidas. Isto ocorre devido ao erro comum existente na estimativa da posição do veículo [SMI 1987] APUD [PRESTES 2003].

Em missões de longa interação do robô com o ambiente, o número de *landmarks* aumenta e os recursos computacionais não são suficientes para manter a atualização do mapa em tempo real. Para resolver isto, Guivant [GUI 2001] APUD [PRESTES 2003] propõe uma implementação de um algoritmo de Mapeamento e Localização Concorrentes que utiliza uma forma especial de matrizes em um novo filtro compressor que reduz significativamente os requisitos computacionais sem inserir qualquer penalidade na precisão dos resultados. Este algoritmo é utilizado no armazenamento e na manutenção de todas as informações obtidas em uma área local. A transferência da informação local para o mapa global é feita quando esta informação se tornar estável e confiável e com custo igual ao do Mapeamento e Localização Concorrentes completo com a diferença de ser feito em somente uma iteração. A proposta de Guivant prevê que o usuário determine o número de landmarks baseado nos recursos

computacionais disponíveis, e o sistema irá selecionar os que tiverem maior poder informativo.

Fox [FOX 1999] APUD [PRESTES 2003], propõe uma abordagem estatística para estimativas em espaços de alta dimensionalidade baseada no algoritmo EM (Expectation Maximization), o qual no contexto das cadeias escondidas de Markov é chamado de algoritmo alfa-beta. O algoritmo basicamente possui dois passos:

1 – Localização: é responsável por calcular a posição e orientação real do robô baseado no mapa previamente calculado e no conjunto de observações e ações realizadas.

2 – Mapeamento: O passo de mapeamento é responsável por produzir o mapa mais provável baseado na estimativa da posição calculada anteriormente. Esta estratégia é utilizada na geração de mapas topológicos [SHA 1997] APUD [PRESTES 2003] e baseados em grades [FOX 1999] APUD [PRESTES 2003] utilizando sensores do tipo sonar.

Thrun [THR 1999] propõe uma estratégia de programação dinâmica, chamada iteração valorada aplicada sobre uma representação baseada em grades. As células do mapa armazenam um valor que representam o custa navegacional do robô e são atualizadas através de uma função baseada em probabilidade condicional obtida com o uso dos sensores sonar. A iteração valorada atua sobre este custo direcionando o robô para áreas de custo mínimo, as quais correspondem as células ainda não exploradas.

Alguns problemas relacionados ao método de Thrun são discutidos por Romero [ROM 2000]:

i) devido ao movimento do robô ser, em algumas ocasiões, próximo dos obstáculos, o risco de colisões tende a aumentar.

ii) O robô pode ficar perdido quando inserido em ambientes esparsos.

iii) O robô muda de direção freqüentemente. Isto tende a aumentar os erros odométricos.

Mataric [MAT 1990], propõe uma nova estratégia, baseada na estratégia de seguir paredes, para a extração de mapas topológicos de ambientes internos, onde os

nós do mapa representam *landmarks* detectados durante o processo de seguir paredes. Após a identificação de um nó, ele é inserido no mapa juntamente com algumas informações, tais como: parede esquerda, parede direita e corredor, e informações sobre a distância relativa aos outros nós que são utilizados no processo de localização do robô. Em conjunto a isto, um protocolo é utilizado para garantir que *landmarks* iguais não se tornem nós distintos no mapa.

2.7 Planejamento de Trajeto por Algoritmos Genéticos baseados em um Mapa Fuzzy

As duas classes principais para a abordagem do planejamento de trajetos são o planejamento de trajeto baseado em um modelo do ambiente ou a navegação utilizando informações de sensores [FABRO 2001].

As técnicas baseadas em modelo utilizam o modelo pré-determinado para planejar a trajetória a ser percorrida, evitando a colisões em obstáculos e seguindo em direção ao alvo. A eficácia deste método depende principalmente da precisão do modelo do ambiente, este por sua vez é um dos maiores problemas encontrados neste método. Estas técnicas são mais apropriadas para planejamento de trajetos em ambientes estáticos e controlados [FABRO 2001].

As abordagens baseadas em sensores utilizam estes para obter informações do ambiente a ser desbravado para determinar a próxima ação do robô, evitar a colisão com obstáculos e seguir em direção ao alvo. A principal vantagem na utilização de sensores é que o robô pode ser inserido em um ambiente dinâmico e desconhecido, pois as únicas informações necessárias para a movimentação são obtidas dos próprios sensores. Entretanto as dificuldades encontradas em tal abordagem são inúmeras, principalmente devido ao fato das leituras realizadas pelos sensores não serem totalmente confiáveis, a impresivibilidade do ambiente, e as inúmeras situações reais que o robô pode encontrar.

As técnicas das abordagens de planejamento de trajeto baseado em modelo e baseada em sensores podem ser combinadas em abordagens híbridas,

nas quais o planejamento global do trajeto é realizado com o uso de um modelo incompleto do ambiente, enquanto que as técnicas baseadas em sensores são utilizadas para desviar de obstáculos e traçar o caminho planejado.

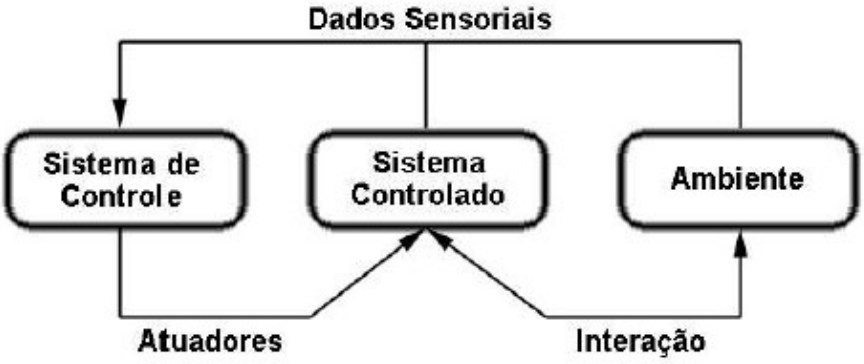
Uma proposta utilizando a abordagem híbrida é relatada por Fabro [FABRO 2001], que propõe o seguinte contexto: O planejamento do trajeto é feito através de um algoritmo genético que, utilizando informações provenientes de um mapa nebuloso do ambiente, procura o caminho a ser percorrido pelo robô de um ponto inicial ao ponto final, sem colidir com obstáculos mapeados. Através de um controle feito por uma rede neural, é possível mover o robô pelo ambiente, fazendo com que o robô desvie obstáculos não detectados no mapa, bem como obstáculos móveis, e siga a sua determinada rota.

Nesta abordagem é utilizado um método chamado de fusão nebulosa de obstáculos, onde obstáculos próximos entre si (eles devem estar tão próximos entre si ao ponto que o robô não passe entre eles), são fundidos em um único obstáculo. Com esta capacidade de fusão, é possível fundir vários obstáculos entre si, transformando paredes inteiras de obstáculos em um único obstáculo nebuloso. Esta informação, quando utilizada pelo algoritmo genético para a geração dos caminhos, traz uma grande carga de conhecimento heurístico para a busca, acelerando a localização de caminhos viáveis por onde não existam obstáculos. Com isto é possível criar um mapa nebuloso do ambiente, que pode ser utilizado pelo algoritmo genético para encontrar o trajeto a ser percorrido pelo robô entre dois pontos .

3 SISTEMAS DE CONTROLE DE NAVEGAÇÃO PARA ROBÔS MÓVEIS AUTÔNOMOS

3.1 Introdução

Para definir como funciona um sistema de controle para robôs móveis deve-se inicialmente definir quais são os elementos participantes do sistema, ou seja, os elementos que interagem com o sistema de controle.



Fonte: [FARLEI 2002]. **Figura 5** – Sistema de Controle.

A tarefa do Sistema de Controle é fazer com que o sistema como um todo alcance um determinado estado. Alcançar este estado pode envolver ou depender de mudanças que ocorram no sistema controlado ou mesmo devido à interação entre os dois.

Mas como é possível relacionar esta terminologia com a de controle de robôs móveis autônomos? Considere-se o seguinte exemplo: um robô está se locomovendo dentro de uma área tentando evitar a colisão com os muros. Neste caso o sistema controlado é o robô, e os muros são parte do ambiente. A interação entre eles ocorre através da fricção das rodas do robô com o chão desta área, e se o robô acidentalmente colidir com um muro, o sistema de controle obtém informações sobre o próprio robô e sobre o ambiente através dos sensores. O sistema de controle então utiliza estas informações para controlar os atuadores (por atuadores definimos todos os mecanismos capazes de modificar o estado do

sistema controlado em relação ao ambiente. Ex: motores, pistões hidráulicos, etc.) e manter o robô distante dos muros, que seria o estado desejado do sistema.

Um sistema de controle pode utilizar sensores inseridos no sistema controlado ou no ambiente, mas isso é opcional. É possível que o sistema de controle utilize sensores somente no sistema controlado, somente no ambiente, ou em lugar nenhum. Existem diferentes maneiras de se obter informações sobre o estado do ambiente e do sistema controlado. A seguir três técnicas são brevemente descritas, citadas por [HAL 1990].

- Sistemas de controle **"*Open Loop*"**: não utilizam nenhum sensor. Por exemplo, considere-se um sistema de controle que deve fazer um robô se mover a uma velocidade de 6 quilômetros por hora. Cálculos baseados no modelo físico do sistema podem ser utilizados para construir um modelo que pode prever quanto energia deve ser fornecida aos motores do robô para que ele atinja a velocidade desejada.
- Sistemas de controle **"*Feedforward*"**: utilizam sensores somente para perceber o ambiente. Neste tipo de sistema de controle, medições do ambiente são utilizadas para atualizar variáveis no modelo do sistema. Utilizando como base o exemplo anterior, podemos adicionar um sensor que informa a inclinação do terreno. Esta informação pode ser utilizada para recalcular a força necessária para que o robô atinja a velocidade desejada.

As técnicas anteriores podem ser utilizadas somente quando o ambiente é praticamente estático e previsível. Isso não ocorre na maioria dos casos de controle robótico. Em robótica o sistema de controle mais utilizado é o "*Feedback*".

- Sistema de controle **"*Feedback*"**: monitora continuamente a situação dos sensores e ajusta seus atuadores de acordo. Retornando ao exemplo anterior, um velocímetro pode ser utilizado ao invés de um sensor de inclinação. Utilizando o velocímetro a diferença entre a velocidade atual e a velocidade desejada é utilizada para ajustar continuamente a força enviada para os motores do robô.

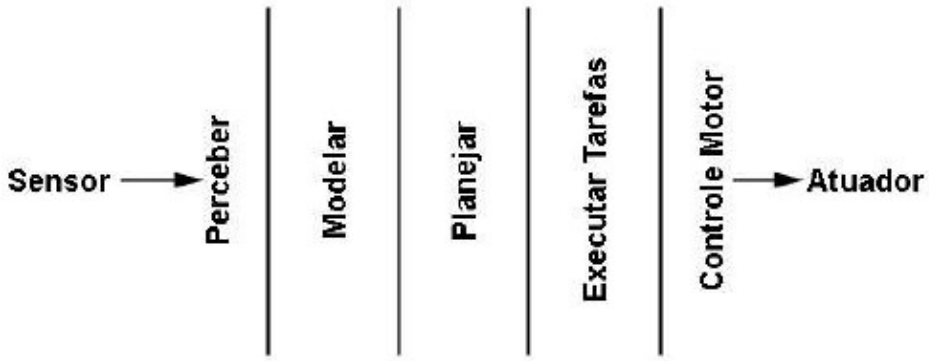
3.2 Arquiteturas de Controle

Uma arquitetura de software pode ser definida como a componentização e as interações entre estes componentes [PRESSMAN 1995], mas de acordo com o trabalho em si, a arquitetura de um robô pode ser definida de como é organizar a tarefa de gerar ações através de sua percepção, ou seja, através da leitura de seus sensores [RUSSEL 1995].

Existem duas principais abordagens para a arquitetura de controle, denominadas arquitetura Horizontal e Vertical [BRO 1991] APUD [FARLEI 2002].

3.2.1 Arquitetura Horizontal

Uma arquitetura tradicional em sistemas robóticos é a arquitetura Horizontal [BRO 1991], uma abordagem é descrita na figura 6.



Fonte: [FARLEI 2002]. **Figura 6** – Arquitetura Horizontal.

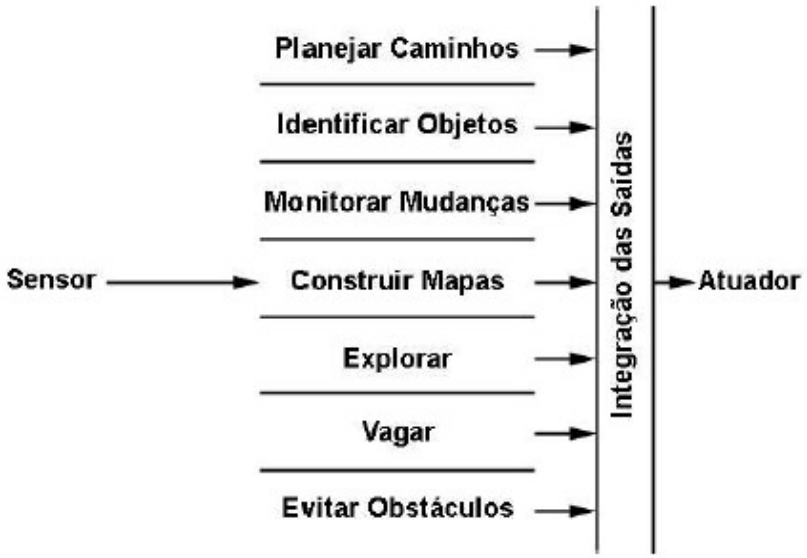
Sistemas de controle baseados nesta arquitetura executam as tarefas em etapas, sendo que a primeira etapa (Perceber) corresponde às entradas dos dados sensoriais. Baseada nesta entrada, a segunda etapa (Modelar) é basicamente responsável pela construção de um modelo do ambiente onde o sistema de controle está inserido; logo após, na terceira etapa (Planejar), define-se o que deve ser realizado pelo sistema controlado, passando as decisões

tomadas para a quarta etapa (Executar Tarefas) onde as tarefas são executadas através da comunicação com a quinta etapa (Controle do Motor) onde são então passados os valores que agem diretamente sobre os atuadores do robô.

A idéia geral dessa arquitetura é receber os dados dos sensores do robô, pegar os valores relevantes do ambiente e construir um modelo mais completo possível sobre o mesmo, ou seja, construir um mapa do ambiente; com este modelo é possível utilizar algoritmos de planejamento, auto-localização e ações a serem executadas pelo sistema controlado afim de alcançar objetivos.

3.2.2 Arquitetura Vertical

Esta arquitetura foi introduzida por Brooks em 1986 [BROOKS 1986]. Nesta arquitetura, ao invés das tarefas serem divididas em partes dependendo da funcionalidade a ser aplicada, a divisão agora deve ser baseada em comportamentos que executam as tarefas, como mostra a figura 7.



Fonte: [FARLEI 2002]. **Figura 7** – Arquitetura Vertical.

Um sistema de controle de robôs móveis baseado em comportamentos é constituído de diversos comportamentos executados em paralelo. Cada comportamento calcula suas saídas diretamente para os atuadores utilizando as

entradas sensoriais. As saídas sugeridas pelo comportamento com a mais alta prioridade são então utilizadas para controlar os atuadores do robô. Em muitas aplicações de sucesso que utilizaram uma arquitetura vertical para construir o sistema de controle, a integração das saídas é feita utilizando-se um esquema de prioridades fixas.

A grande diferença nesta arquitetura está no fato que uma entrada sensorial não precisa mais passar por uma série de camadas de processamento antes de se transformar em uma saída para os atuadores de robô. Nesta arquitetura somente os dados sensoriais relevantes para cada comportamento são utilizados, pelo mesmo, gerando assim as saídas para os atuadores do robô.

4 PROPOSTA DE SISTEMA DE MAPEAMENTO E CONTROLE DE NAVEGAÇÃO PARA ROBÔS MÓVEIS AUTÔNOMOS

Este capítulo tem por finalidade a apresentação da proposta em si, descrevendo os métodos utilizados para a implementação da navegação e a construção do mapa do ambiente.

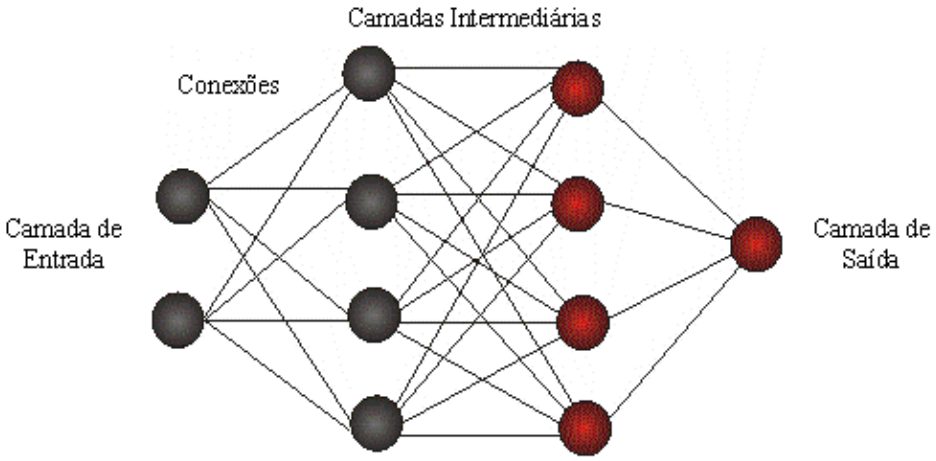
A arquitetura utilizada para a implementação é a arquitetura horizontal, pois o modelo de implementação do sistema faz com que os módulos sejam dependentes entre si, fazendo com que o resultado final da inter-relação entre os módulos ajam sobre os atuadores do robô.

Para cada módulo são descritas as técnicas para a implementação nas seções que seguem.

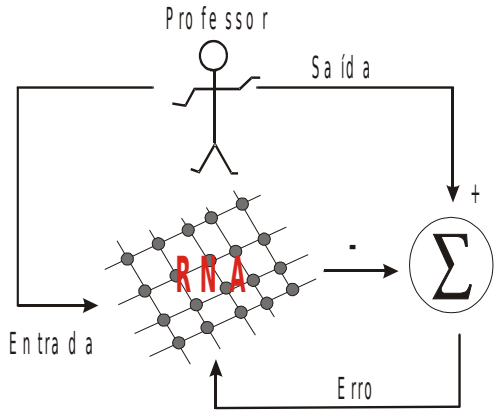
4.1 Navegação

Quando se fazem referências à navegação de robôs móveis autônomos, o principal objetivo é descrever as técnicas que fornecem os meios para que um robô autônomo se mova de forma segura de um local a outro dentro do ambiente.

Para fazer com que o robô navegue pelo ambiente evitando colisões, e por sua vez possua uma navegação estável desviando de obstáculos, é utilizada a técnica de redes neurais artificiais (descrita no capítulo 2). A rede utilizada é uma do tipo perceptron de múltiplas camadas, sendo uma rede acíclica (*feedforward*), com aprendizado supervisionado pelo algoritmo *backpropagation* (figura 8). No aprendizado supervisionado as entradas e as respectivas saídas são fornecidas e controladas por um agente externo (figura 9). A principal forma de ajuste da rede é feita através da correção de erros a partir da saída obtida na rede para uma dada entrada em comparação com a saída ideal fornecida.



Fonte: [BRAGA 2000]. **Figura 8 -** Rede neural de múltiplas camadas acíclica (*feedforward*)



Fonte: [BRAGA 2000]. **Figura 9 -** Aprendizado Supervisionado

Durante o treinamento com o algoritmo *backpropagation*, a rede opera em uma seqüência de duas fases. Na primeira fase, denominada de fase de propagação (ou *forward*), um padrão é apresentado à camada de entrada da rede. Este padrão é propagado através da rede, camada por camada, até que a resposta seja produzida pela camada de saída. Na segunda fase, denominada de fase de retropropagação (ou *backward*), a saída obtida é comparada à saída desejada para esse padrão particular. Se não estiver correta, o erro é calculado. O erro é então propagado a partir da camada de saída até a camada de entrada, e os pesos das

conexões das unidades das camadas internas vão sendo modificados proporcionalmente ao erro.

A fase *forward* envolve os seguintes passos:

1. A entrada é apresentada á primeira camada da rede, a camada C^0.
2. Para cada camada C^i a partir da camada de entrada:
   2.1. Após os nodos da camada C^i ($i > 0$) calcularem seus sinais de saída, estes servem como entrada para a definição das saídas produzidas pelos nodos da camada C^{i+1}.
3. As saídas produzidas pelos nodos da última camada são comparadas às saídas desejadas.

A fase *backward* envolve as etapas listadas a seguir :

1. A partir da última camada, até chegar na camada de entrada:
   1.1. Os nodos da camada atual ajustam seus pesos de forma a reduzir seus erros.
   1.2. O erro de um nodo das camadas intermediárias é calculado utilizando os erros dos nodos da camada seguinte conectados a ele, ponderados pelos pesos das conexões entre eles.

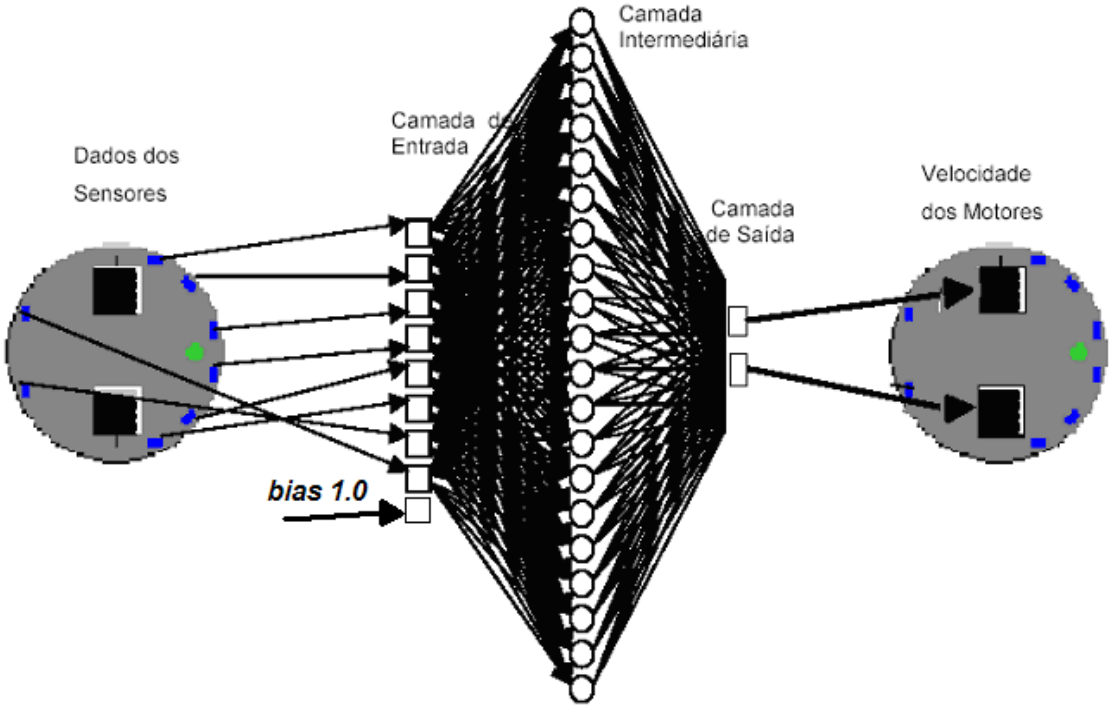
O algoritmo *backpropagation*, que faz uso destas duas fases, é apresentado a seguir:

1. Inicializar pesos e parâmetros
2. Repita até obter um erro mínimo desejado, ou realizar um dado número de ciclos:
   2.1. Para cada padrão de treinamento X:
      2.1.1. Definir saída da rede através da fase *forward.*
      2.1.2. Comparar saídas produzidas com as saídas desejadas.
      2.1.3. Atualizar pesos do nodos através da fase *backward*.

O algoritmo *back-propagation* é baseado na regra delta proposta por Widrow e Hoff [WIDROW 1960] APUD [RUMELHART 1986], sendo por isto também chamada de regra delta generalizada. Este algoritmo propõe uma forma de definir o erro dos nodos das camadas intermediárias, possibilitando o ajuste de seus pesos [ZUBEN 1999].

4.2 O Treinamento

Para realizar o treinamento da rede neural é necessário definir os padrões de entrada e as respectivas saídas. A entrada da rede é fornecida pelos sensores do robô de forma que os respectivos pares de saídas são determinados pelo agente treinador da rede, que por sua vez acionaram os atuadores do robô (figura 10).



Fonte: [MICHELON 2000]. **Figura 10** – Entradas e Saídas da RNA.

Para realização do treinamento foram utilizadas duas RNA's, sendo que uma direciona o robô para a esquerda (tabela 3), e outra direciona o robô para a direita (tabela 4). Com a utilização de duas RNA's é possível determinar com maior precisão os comportamentos de desvio de obstáculos à esquerda e à direita.

Com a utilização de uma única RNA para o controle da navegação do robô, em determinadas situações, como por exemplo quando os sensores de um dos lados do robô estavam ativos, o robô possuía uma ação correta, ou seja, a RNA utilizada fazia com que a saída atribuída aos motores do robô fossem corretas, porém em situações contrárias, a execução do desvio do determinado obstáculo não era satisfatória, pois identificou-se que existiam padrões que quando realizado o treinamento, geravam conflitos, ou seja, padrões de entrada com muita semelhança e com respectivas saídas contraditórias, faziam com que o robô não possuísse a ação correta de desvio do obstáculo. Devido a estes comportamentos conflitantes, verificou-se a necessidade da construção de duas RNA's para a execução da navegação do robô. A tabela 2 determina um padrão de treinamento utilizado para a construção da RNA.

Tabela 2 – Padrão de Treinamento.

Entradas
0.0
0.0
0.30
0.30
0.0
0.0
0.0
0.0
1.0
Saídas
-0.5
1.0

Este padrão de treinamento indica que o robô está “vendo” um obstáculo a sua frente, e a ação tomada por ele será desviar do obstáculo virando para a esquerda, fazendo assim que o robô não colida com o obstáculo.

Tabela 3 – Direcionamento a Esquerda.

	E1	E2	E3	E4	E5	E6	E7	E8	Bias	Saída	S1	S2
P1	0.70	0.10	0.25	0.25	0.0	0.0	0.0	0.0	1.0	→	1.0	-0.5
P2	0.0	0.0	0.25	0.25	0.10	0.70	0.0	0.0	1.0	→	-0.5	1.0
P3	0.0	0.0	0.30	0.30	0.0	0.0	0.0	0.0	1.0	→	-0.5	1.0
P4	0.0	0.0	0.0	0.0	0.0	0.40	0.70	0.70	1.0	→	1.0	1.0
P5	0.40	0.0	0.0	0.0	0.0	0.0	0.70	0.70	1.0	→	1.0	1.0
P6	0.80	0.40	0.0	0.0	0.40	0.80	0.0	0.0	1.0	→	1.0	1.0
P7	0.80	0.50	0.40	0.40	0.50	0.80	0.0	0.0	1.0	→	0.5	-0.5
P8	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	1.0	→	0.5	-0.5
P9	0.70	0.40	0.60	0.60	0.40	0.0	0.70	0.70	1.0	→	0.5	-0.5
P10	0.80	0.10	0.0	0.0	0.0	0.0	0.0	0.0	1.0	→	1.0	1.0
P11	0.0	0.0	0.0	0.0	0.10	0.80	0.0	0.0	1.0	→	1.0	1.0
P12	0.30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	→	-0.5	0.5
P13	0.0	0.0	0.0	0.0	0.0	0.30	0.0	0.0	1.0	→	-1.0	0.5

Tabela 4 – Direcionamento a Direita.

	E1	E2	E3	E4	E5	E6	E7	E8	Bias	Saída	S1	S2
P1	0.70	0.10	0.25	0.25	0.0	0.0	0.0	0.0	1.0	→	1.0	-0.5
P2	0.0	0.0	0.25	0.25	0.10	0.70	0.0	0.0	1.0	→	-0.5	1.0
P3	0.0	0.0	0.30	0.30	0.0	0.0	0.0	0.0	1.0	→	1.0	-0.5
P4	0.0	0.0	0.0	0.0	0.0	0.40	0.70	0.70	1.0	→	1.0	1.0
P5	0.40	0.0	0.0	0.0	0.0	0.0	0.70	0.70	1.0	→	1.0	1.0
P6	0.80	0.40	0.0	0.0	0.40	0.80	0.0	0.0	1.0	→	1.0	1.0
P7	0.80	0.50	0.40	0.40	0.50	0.80	0.0	0.0	1.0	→	0.5	-0.5
P8	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80	1.0	→	0.5	-0.5
P9	0.70	0.40	0.60	0.60	0.40	0.0	0.70	0.70	1.0	→	0.5	-0.5
P10	0.80	0.10	0.0	0.0	0.0	0.0	0.0	0.0	1.0	→	1.0	1.0
P11	0.0	0.0	0.0	0.0	0.10	0.80	0.0	0.0	1.0	→	1.0	1.0
P12	0.30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	→	-0.5	0.5
P13	0.0	0.0	0.0	0.0	0.0	0.30	0.0	0.0	1.0	→	1.0	-0.5
P14	0.90	0.90	0.0	0.0	0.0	0.0	0.0	0.0	1.0	→	1.0	-0.5

Os valores de E1 à E8 são os valores normalizados dos sensores do robô, e ainda existe a nona entrada formada consistindo no *bias* que assume o valor 1.0, utilizado para o aceleramento do treinamento da RNA. Com relação aos valores S1 e S2, estes são dados que são utilizados para fornecer os valores que

são atribuídos aos atuadores do robô, ou seja, os valores dos motores do robô, determinando assim a ação tomada pelo robô, seus valores variam de (-1) à (1) devido à função de transferência ser formada pela tangente hiperbólica, onde a saída desta função é multiplicada por uma constante para logo após ser atribuída aos motores do robô.

4.3 Intercalação entre as RNA'S

Inicialmente a RNA utilizada é a RNA com padrões de direcionamento à esquerda. Para que ocorra a intercalação das RNA's primeiro verifica-se se há obstáculos próximos ao robô, caso haja é identificada a rede atual em execução, (RNA de desvio à esquerda ou RNA de desvio à direita), e dependendo da rede que esteja em execução verificam-se os sensores ativos do robô, do lado esquerdo (sensores de 0 à 2), ou lado direito (sensores de 3 à 5), após determinado qual o lado ativo, ou a rede atual é mantida, ou o comando é passado para a outra rede. Por exemplo, quando as leituras dos sensores indicam presença de obstáculo à esquerda, a RNA treinada para desviar obstáculos à esquerda é ativada, caso contrário a outra RNA é ativada.

Durante a navegação do robô existe uma função que identifica se o robô já passou pela posição (x,y) do ambiente. Caso o robô já tenha visitado aquela posição, verificam-se os sensores 2 e 3 (frente do robô), caso estes não estejam ativos, o robô continua normalmente sua navegação, caso contrário, é identificada a rede em execução para que o robô tome a decisão de girar para esquerda ou para a direita, até que os sensores 2 e 3 não estejam mais ativos (se entende por ativo quando o retorno do valor do sensor for maior ou igual a 90), fazendo assim a trocas entre as RNA's.

Verificou-se que com a utilização de duas RNA's ao invés de uma, a navegação do robô se tornou mais estável e confiável pois, utilizando o algoritmo de intercalação entre as redes, pode-se obter um maior controle sobre a ação que deve ser tomada pelo robô para que com isso o mesmo não venha a colidir com algum obstáculo presente no ambiente.

As figuras 11 e 12 mostram a atuação das duas RNA’s, uma desviando um obstáculo à esquerda, e outra à direita, respectivamente:

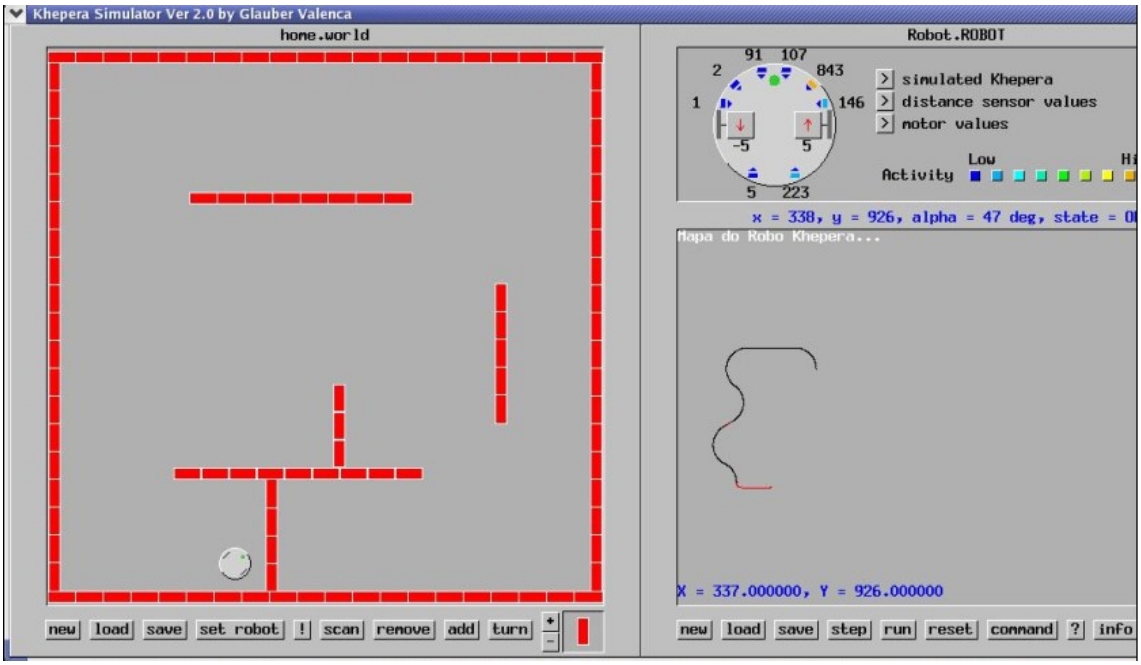


Figura 11 – RNA de desvio a esquerda.

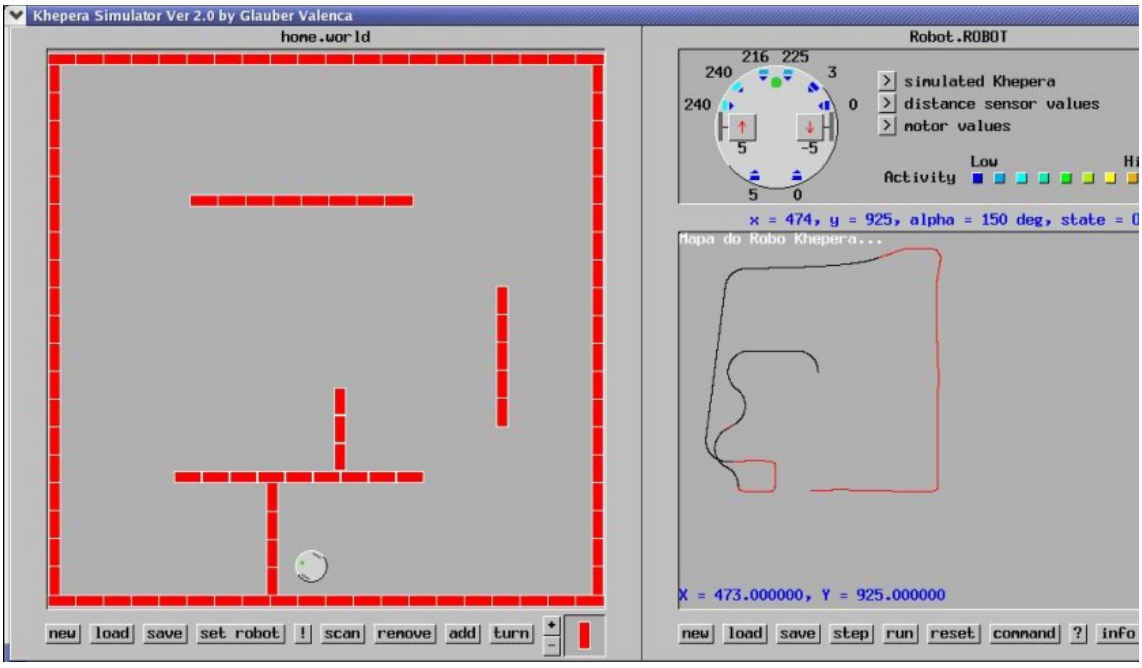


Figura 12 – RNA de desvio a direita.

4.4 Técnicas Navegacionais

O método de navegação utilizado é o método Local (guiado por objetivo), também chamado de método Reativo, ou seja, o robô não possui nenhum conhecimento “a priori” da estrutura do ambiente a ser explorado e estas informações são oriundas somente dos sensores do robô. Para que as informações sejam capturadas e inseridas no mapa do robô, utiliza-se também a abordagem de “seguir paredes”.

4.4.1 Abordagem de “Seguir Paredes”

Esta técnica consiste em guiar o robô paralelamente às paredes, enquanto se extrai o mapa métrico do ambiente. Os padrões de treinamento da rede neural que realizam a tarefa de guiar o robô paralelamente as paredes são descrito na tabela 5, onde estes padrões fazem parte dos padrões de treinamento especificados nas tabelas 3 e 4:

Tabela 5 – padrão que define o robô a andar paralelamente as paredes.

	E1	E2	E3	E4	E5	E6	E7	E8	Bias	Saída	S1	S2
P1	0.80	0.10	0.0	0.0	0.0	0.0	0.0	0.0	1.0	→	1.0	1.0
P2	0.0	0.0	0.0	0.0	0.10	0.80	0.0	0.0	1.0	→	1.0	1.0
P3	0.30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	→	-0.5	0.5
P4	0.0	0.0	0.0	0.0	0.0	0.30	0.0	0.0	1.0	→	0.5	-0.5

- P1: indica que os sensores do robô estão notando a presença de um obstáculo (parede) à sua esquerda, pois os sensores E1 e E2 estão ativos e a respectiva saída é S1=1.0 e S2=1.0, fazendo com que o robô se mova para frente.
- P2: é o recíproco do padrão P11, neste caso os sensores do robô estão percebendo a presença de um obstáculo (parede) à sua direita, indicado pelos sensores E5 e E6, e as respectivas saídas S1=1.0 e S2=1.0, fazendo também com que o robô vá para frente.
- P3: caso o robô se desloque um pouco para a sua direita, os valores atribuídos aos atuadores fazem com que ele volte a se aproximar da parede, fazendo com que o robô se desloque para a esquerda.
- P4: caso o robô se desloque um pouco a sua esquerda os valores atribuídos aos atuadores do robô fazem com que ele volte a se aproximar da parede, fazendo com que o robô se desloque para a direita.

A figura 11 mostra o caminho percorrido pelo robô, guiando paralelamente às paredes do ambiente enquanto extrai o mapa métrico do mesmo, sendo que quando os sensores estão ativos a RNA toma controle sobre o robô e seus atuadores, fazendo com que o robô se locomova pelo ambiente.

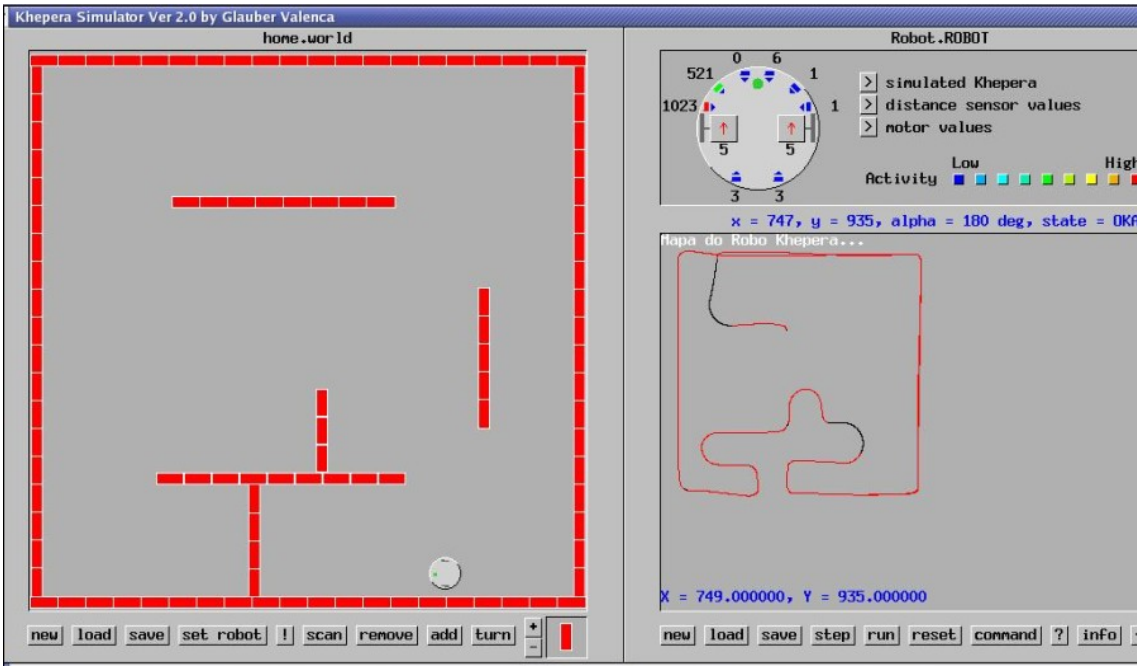
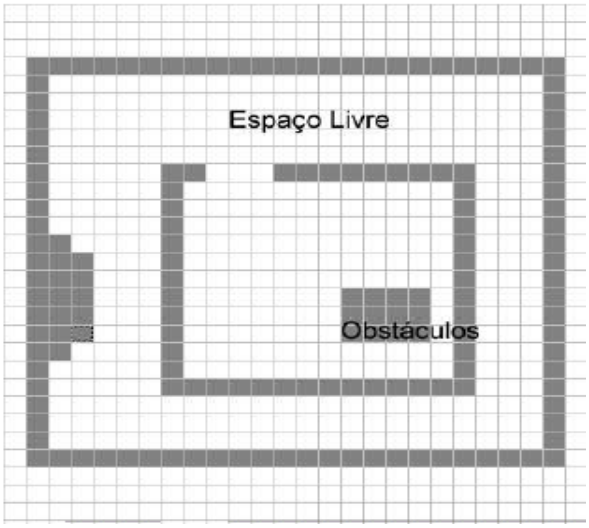


Figura 13 – Abordagem seguir paredes.

4.5 Mapeamento do Ambiente

A abordagem utilizada para a realização do mapeamento do ambiente é a de Grades de Ocupação, que representa o ambiente utilizando uma matriz bidimensional de tamanho fixo. Cada célula da matriz pode conter diferentes atributos que auxiliam na navegação do robô. Cada célula, por exemplo, pode indicar a presença de um obstáculo na região correspondente do ambiente, ou indicar a probabilidade desta determinada região estar ou não ocupada por um obstáculo. O ambiente é representado em uma matriz bidimensional, e a medida em que o robô navega pelo ambiente, é extraído o mapa métrico deste, isto é, é realizada a decomposição espacial do ambiente. Depois de estudos relacionados as diferentes técnicas envolvidas para realização do mapeamento do ambiente, chegou-se a conclusão que a técnica de Grades de Ocupação é uma das técnicas mais viáveis para a implementação do mesmo. Com esta técnica, a construção e a atualização do mapa se torna menos complexa com relação a outras técnicas, como por exemplo a técnica baseada em características, onde o ambiente é

representado da localização de todas as características relevantes do ambiente e suas propriedades geométricas, ou os mapas probabilísticos, sendo este um fator fundamental para sua escolha.



Fonte: [FARLEI 2002]. **Figura 14** - Exemplo de mapa representado por uma matriz bidimensional

Com a construção do mapa do ambiente é possível determinar a posição dos obstáculos inseridos no ambiente, como pode ser notado nas figuras 15 e 16, onde somente as informações sobre as posições dos obstáculos são apresentadas no mapa, no lado direito da figuras.

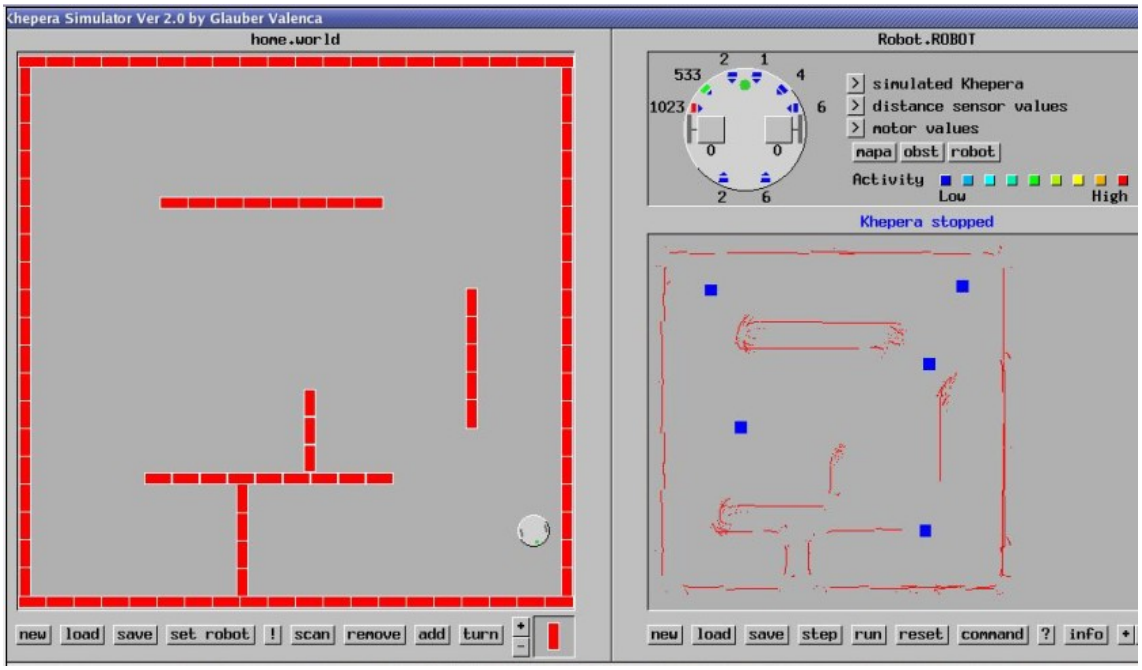


Figura 15 – Mapeamento dos Obstáculos.

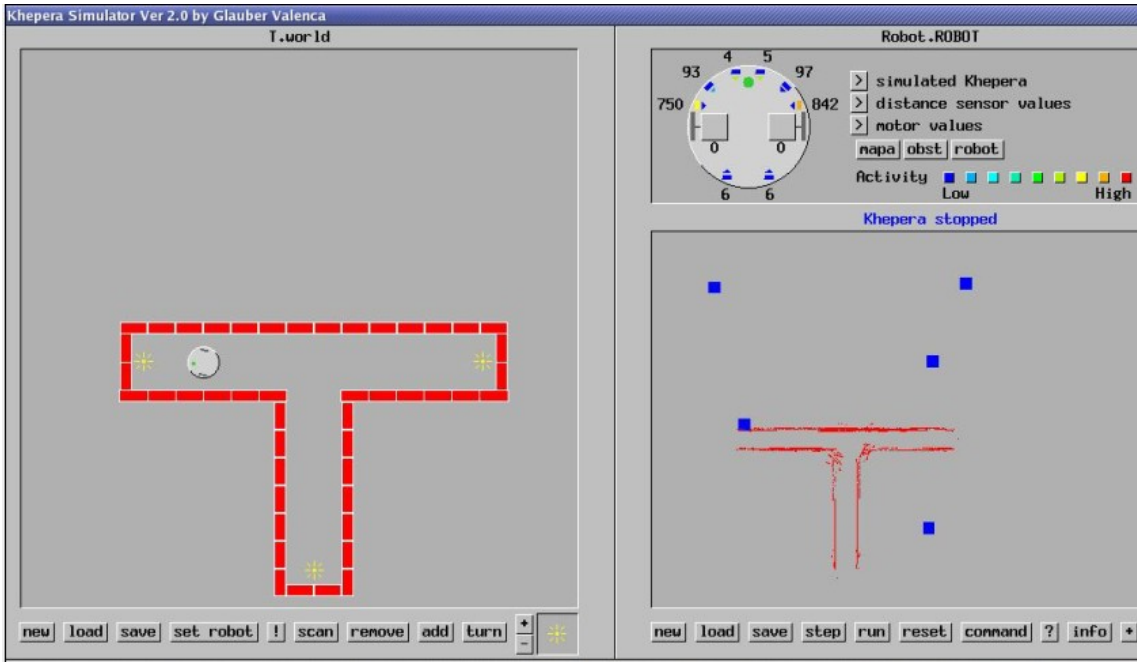


Figura 16 – Mapeamento dos Obstáculos.

A figura 17 descreve a estrutura definida para a implementação do mapa, bem como a estrutura dos pontos inseridos no ambiente e a cor representando a presença ou não de obstáculos no ambiente.

```
#ifndef MAPAROBOT_H
#define MAPAROBOT_H
struct Ponto
{
      float X,Y;
      int Visitado;
      int DistanceRobot;
};
struct Cores
{
      char Cor[10];
      float X,Y;
};
struct MatCores
{
      struct Cores *MatrizCor[1000][1000];
};
struct Mat
{
      float XObst,YObst,XRobot,YRobot;
      int gtdCaminho;
      int DistanceValue;
};

struct MatrizRobot
{
      struct Mat *Mapa[1000][1000];
};
#endif
```

Figura 17 – Estruturas para Pontos, Cores e Mapa.

4.5.1 A Struct Ponto

A estrutura de cada ponto é utilizada para determinar a posição de cada ponto inserido no ambiente, e é definido pelos seguintes valores:

- X,Y → determina a posição (x,y) de cada ponto no ambiente.
- Visitado → variável que indica se determinado ponto foi visitado pelo robô durante sua navegação no ambiente.
- DistanceRobot → variável utilizada para determinar a distância de cada ponto em relação ao robô.

4.5.2 A Struct MatCores

A estrutura de cores é utilizada para realizar o desenho do caminho percorrido pelo robô dentro do ambiente, e possui a seguinte característica:

- É formada por uma matriz bidimensional 1000x1000, sendo que cada célula da matriz, corresponde a posição (x,y) no ambiente, bem como a cor correspondente naquele pixel, sendo a cor vermelha indicando a presença de obstáculos e a cor preta a ausência do mesmo (figura 18).

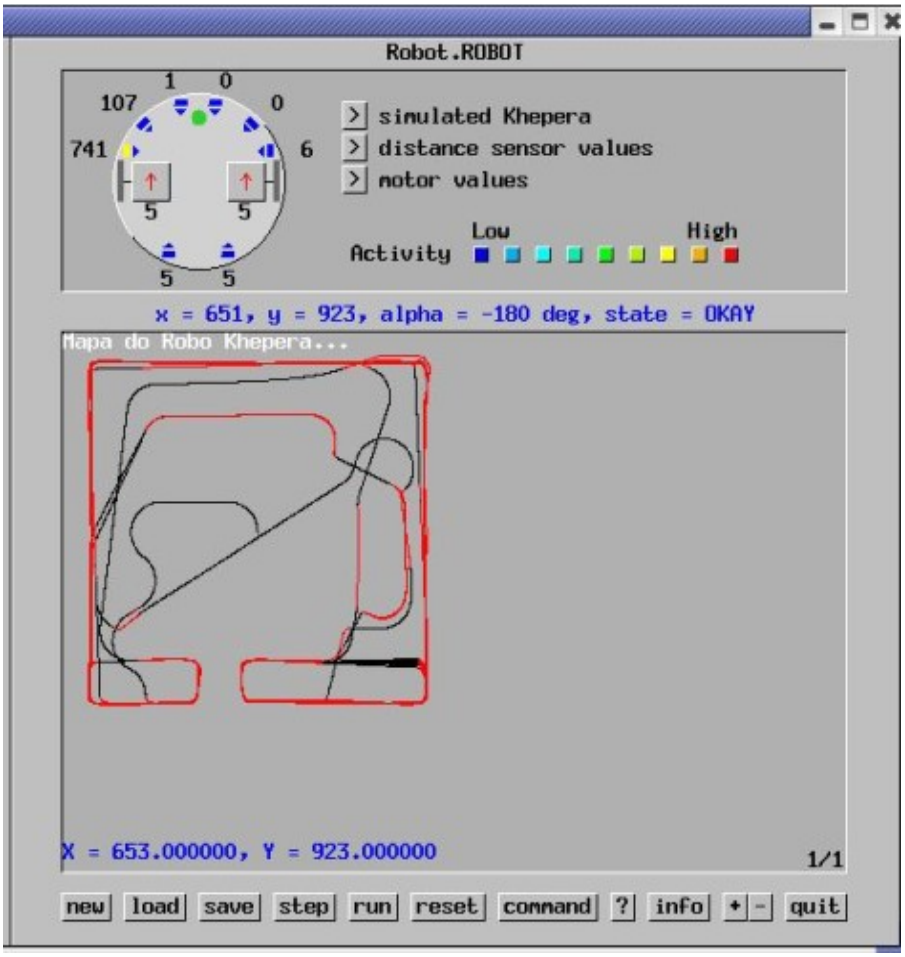


Figura 18 – Cores representando a ausência ou presença de obstáculos.

4.5.3 A Struct Mat

A struct Mat tem por características armazenar os valores relevantes do ambiente. A seguir são descritas as variáveis que compõem a struct Mat:

Os valores (XRobot,YRobot) indicam a posição x,y do robô dentro do ambiente, enquanto que os valores (XObst,YObst) indicam a posição x,y de um determinado obstáculo dentro do ambiente, o valor de qtdCaminho indica quantas vezes o robô passou pela posição x,y do ambiente, a variável de DistanceValue assume dois valores:

- 1: para a presença de obstáculos na posição XObst,Yobst.
- 0: para a ausência de obstáculos na posição XObst,Yobst.

Inicialmente esta estrutura é inicializada com valores pré-determinados(-1) e a medida que a exploração é efetuada pelo robô, seus valores são carregados com os dados relevantes do ambiente, fazendo com que ao longo de sua iteração com o ambiente o mapa venha a ser utilizado para verificação da posição atual do robô, para posterior tomada de decisão com relação ao próximo passo a ser tomado.

4.5.4 A Struct MatrizRobot

A struct MatrizRobot é a estrutura principal para a construção e representação do ambiente. É formada por uma matriz bidimensional 1000x1000, sendo que cada célula da matriz corresponde a um ponteiro para a estrutura struct Mat.

4.6 Determinação dos Pontos

Quando a navegação é iniciada, o robô segue em direção ao ponto mais próximo de sua posição inicial, esta característica tem por principal finalidade a obtenção de todas as características do ambiente, ou seja, fazer com que o robô

navegue por todo o ambiente para a realização do mapeamento do mesmo, onde os pontos são previamente definidos e inseridos no ambiente. Para a determinação destes pontos os seguintes passos são executados:

1. A estrutura de cada ponto (Struct Ponto) é inicializada com valores fixos, sendo que os pontos estão na seguinte forma:

```
typedef struct Ponto P[MAX];
typedef P *PPoint;
```

2. Para cada ponto é realizado o cálculo da distância entre a posição (x,y) do robô com a posição (x,y) do ponto. Dada pela seguinte fórmula:

→ $d = | (Xr - Xp) | + | Yr – Yp |$

d = distância entre o robô e o ponto determinado.
Xr = posição X do robô.
Xp = posição X do ponto.
Yr = posição Y do robô.
Yp = posição Y do ponto.

3. Calculada a distância de cada ponto em relação à posição atual do robô, o vetor contendo a estrutura do tipo Ponto é organizada de forma crescente baseado no valor da distância de cada ponto dado pelo valor “DistanceRobot”.

4. A cada passo do robô é calculado o valor do ângulo determinado pelas posições (x,y) do robô e (x,y) do ponto, sendo que o ponto em questão é o mais próximo ao robô, dado que este cálculo é realizado somente no momento da localização do ponto mais próximo, ao passo que enquanto este ponto não for localizado, somente o ângulo é calculado. Quando este

ponto for encontrado pelo robô, é realizado o cálculo para a obtenção do ponto mais próximo seguinte (passo 2).

O cálculo do angulo do robô com relação ao ponto é dado pela seguinte fórmula, como visto na figura 19:

$$\rightarrow \theta = \arctan (Yr - Yp / Xr - Xp)$$

θ = angulo do robô em relação ao ponto.

Xr = posição X do robô.

Xp = posição X do ponto.

Yr = posição Y do robô.

Yp = posição Y do ponto.

ArcTang = Arco Tangente.

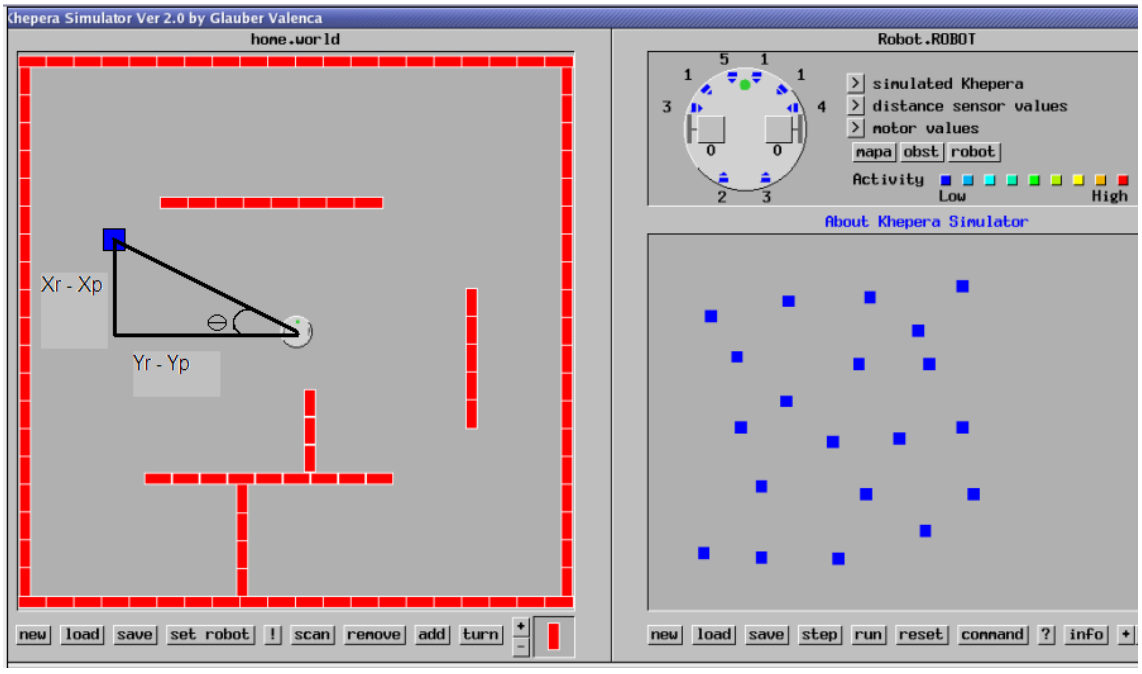


Figura 19: Determinação do ângulo em direção ao ponto

5. Determinado o ângulo do robô em relação ao ponto, verifica-se o quadrante onde o ponto está localizado em relação à posição do robô no ambiente, as seguintes situações são encontradas:

a. Caso o ponto esteja no primeiro ou quarto quadrante o ângulo é mantido com seu valor atual.
b. Caso o ponto encontra-se no segundo quadrante o valor do ângulo será a soma com o valor de PI, com seu valor invertido.
c. Caso o ponto encontra-se no terceiro quadrante é realizado à subtração de PI com o ângulo encontrado.

6. Calculado o valor do ângulo do robô em relação ao ponto, este é verificado para a atribuição aos valores dos motores do robô. Se a diferença entre o angulo do robô com relação ao mundo, e o ângulo calculado pelo ponto em relação ao robô for negativo, o robô vira para esquerda, caso contrário o robô vira à direita, fazendo assim que o robô se mova em direção ao ponto mais próximo.

7. Enquanto os valores dos sensores do robô não estiverem ativos, ou seja, se os valores lidos forem menor que 90, o passo 5 é executado, caso contrário a rede neural toma controle dos atuadores do robô, determinando qual o próximo passo a ser tomado.

8. Para a determinação se o robô encontrou o ponto, é verificado a posição (x,y) do robô, com a posição (x,y) do ponto, mas como os valores retornados pelos sensores do robô nem sempre são valores precisos, é necessário realizar um enquadramento, ou seja, uma normalização dos valores dos sensores do robô, neste caso é somado aos valores da posição (x,y) do robô o seu diâmetro que corresponde a 52 mm, para a verificação da posição do robô em relação a posição do ponto. Neste caso a precisão necessária para encontrar a posição (x,y) relacionada ao ponto será diminuída, ou seja, não preciso que, necessariamente o robô passe pela posição (x,y) do ponto para a verificação do se a sua posição com relação ao ponto é a mesma.

9. Os pontos estão localizados nas seguintes posições no ambiente:

- **Ponto 1→** x == 760, y == 350.
- **Ponto 2→** x == 250, y == 520.
- **Ponto 3→** x == 850, y == 140.
- **Ponto 4→** x == 750, y == 800.
- **Ponto 5→** x == 170, y == 220.
- **Ponto 6→** x == 150, y == 860.
- **Ponto 7→** x == 380, y == 180.
- **Ponto 8→** x == 380, y == 180.
- **Ponto 9→** x == 600, y == 170.
- **Ponto 10→** x == 306, y == 872.
- **Ponto 11→** x == 850, y == 520.
- **Ponto 12→** x == 374, y == 450.
- **Ponto 13→** x == 680, y == 550.
- **Ponto 14→** x == 514, y == 874.
- **Ponto 15→** x == 590, y == 700.
- **Ponto 16→** x == 570, y == 350.
- **Ponto 17→** x == 240, y == 330.
- **Ponto 18→** x == 306, y == 680.
- **Ponto 19→** x == 500, y == 560.
- **Ponto 20 →** x == 880, y == 700.
- **Ponto 21→** x == 730, y == 260.

A figura 20 mostra o trajeto percorrido pelo robô pra a localização dos pontos espalhados pelo ambiente, este mundo e conhecido com o nome de “first”, e por sua vez não possui nenhum obstáculo a não ser a parede em volta do ambiente.

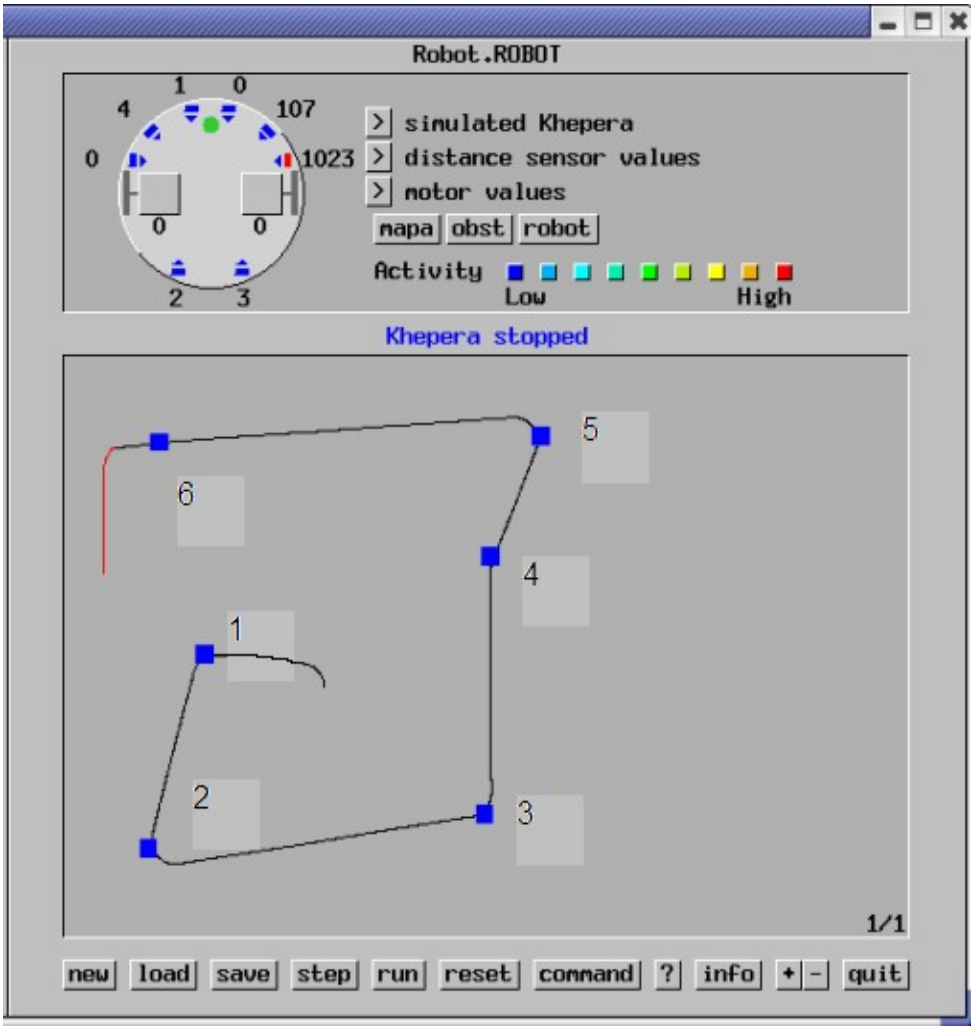


Figura 20 – Exemplo de trajeto percorrido pelo robô em busca dos pontos.

4.7 Determinação da Posição dos Obstáculos

A medida em que navega pelo ambiente, é necessário que o robô realize a construção do mapa, e nesse mapa é necessário saber a posição dos obstáculos dispostos no ambiente bem como outras informações relevantes do mesmo como foi descrito na seção 4.4. Para a obtenção da posição dos obstáculos no ambiente os seguintes cálculos são realizados:

1. Primeiramente é necessário definir qual o sensor mais ativo do robô, ou seja, qual a direção em que a probabilidade de existir obstáculo é maior.

2. Logo após é necessário definir um ângulo Alpha que é dado pela seguinte forma:

   → $\alpha = \theta + \beta$

   θ = ângulo do robô em relação ao mundo (ambiente).

   β = ângulo do sensor mais ativo em relação ao robô.

3. Após calculado este ângulo Alpha, é calculado o valor de (x,y) correspondente ao obstáculo no ambiente, dado pela seguinte formula:

   → XObst = Xrobot + coseno (Alpha) * d;

   → YObst = Yrobot + seno (Alpha) * d;

   XObst = posicao x do Obstáculo no ambiente.
   Yobst = posicao y do Obstáculo no ambiente.
   Xrobot = posição x do robô no ambiente.
   Yrobot = posição y do robô no ambiente.
   Alpha = α.
   d = distância do robô ao Obstáculo.

4. Para o cálculo da distância do robô ao obstáculo (d), foram realizados experimentos, para a obtenção do mesmo, sendo que a medida em que o valor dos sensores aumentava a distancia do robô ao obstáculo diminuía, e esta distância foi parametrizada para a obtenção do valor d, ou seja, dependendo do valor de ativação do sensor mais ativo do robô o dado (d) possui um valor correspondente.

4.8 Carregando e Atualizando o Mapa

A estrutura do mapa possui uma variável chamada MapaRobot que é um ponteiro para a matriz que representa o ambiente, sendo que cada célula da matriz é um ponteiro para a estrutura de dados chamada struct Mat. Esta estrutura, por sua vez, possui os dados relevantes oriundos dos sensores do robô, que podem ser posteriormente utilizados para a implementação de algoritmos de navegação do robô.

Inicialmente o mapa é carregado com valores iniciais pré-determinados, como por exemplo o valor de DistanceValue é inicializado com menos um (-1) e qtdCaminho é inicializado com zero (0). A medida em que o robô explora o ambiente o mapa é carregado com os valores oriundos dos sensores do robô, sendo que a função responsável por carregar e atualizar o mapa do ambiente é a função LoadMap().

A função LoadMap além de executar a atualização do mapa, verifica a presença ou ausência de obstáculos no ambiente indicado pelo valor de DistanceValue, além de realizar o desenho da navegação do robô nos diferentes mundos.

A medida em que o robô explora o ambiente o mapa é preenchido, ou seja, a respectiva posição (i,j) da matriz (Matriz[i][j]), é correspondente a posição (x,y) do ambiente onde os dados relevantes do mesmo estão armazenados. Quando o robô passar novamente pela posição (x,y), o mapa será atualizado com os valores oriundos dos sensores do robô, neste caso se houver alguma mudança com relação ao ambiente o robô atualizará seu mapa referente a nova mudança que ocorreu no ambiente. Como descrito anteriormente o cálculo referente a determinação da posição dos obstáculos é realizado pela função LoadMap, sendo esta uma função determinante para a implementação do trabalho.

4.9 Construção do Mapa *Fuzzy*

Quando encerrado a busca por todos os pontos inseridos no ambiente, é realizada a construção do mapa *fuzzy* do mesmo, ou seja, os obstáculos inseridos no ambiente serão transformados em obstáculos *fuzzy*, onde as coordenadas de cada obstáculo serão inseridas em um vetor, e este vetor por sua vez, conterá os obstáculos mapeados pelo robô durante sua navegação, sendo que o conjunto de todos os obstáculos mapeados forma o mapa *fuzzy* do ambiente.

Após o término da busca pelos pontos, é realizado inicialmente um pré-processamento de filtragem do mapa do ambiente gerado pelo robô, para realizar a eliminação dos pontos esparsos no ambiente, após a realização da filtragem, uma varredura pelo mapa do ambiente é executado, sendo que para cada coordenada (i,j) do mapa onde for detectada a presença de um obstáculo, é realizada uma busca horizontal e vertical do obstáculo, para conhecer os pontos iniciais e finais do mesmo como se pode observar na (figura 21), fazendo assim com que o vetor de obstáculos *fuzzy* contenha as coordenadas de cada obstáculo mapeado durante a navegação do robô.

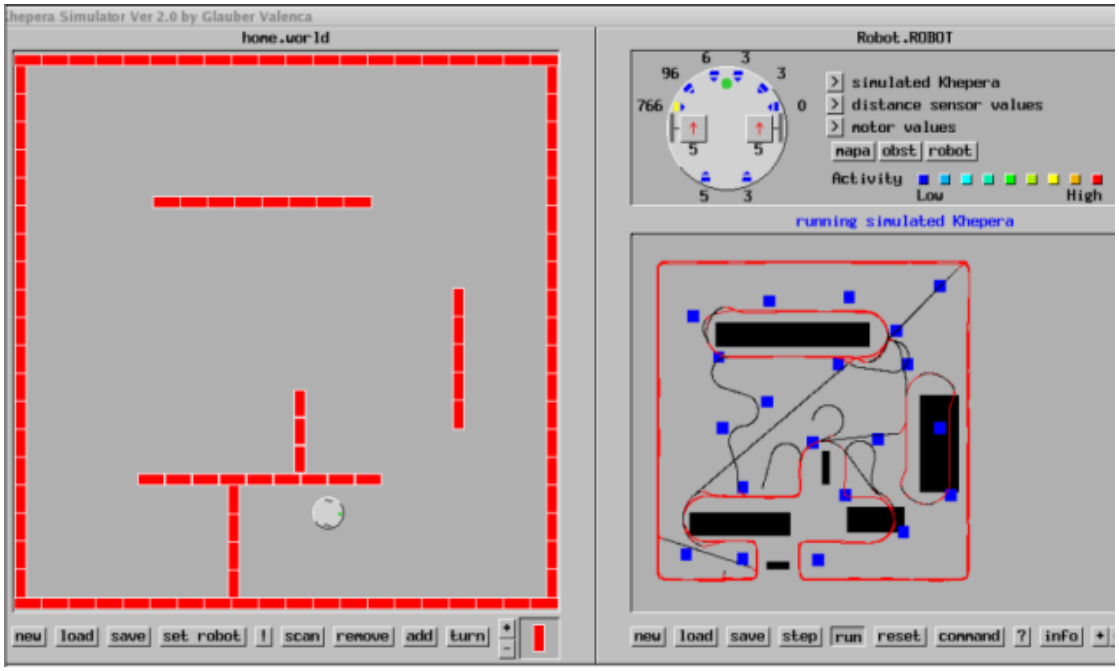
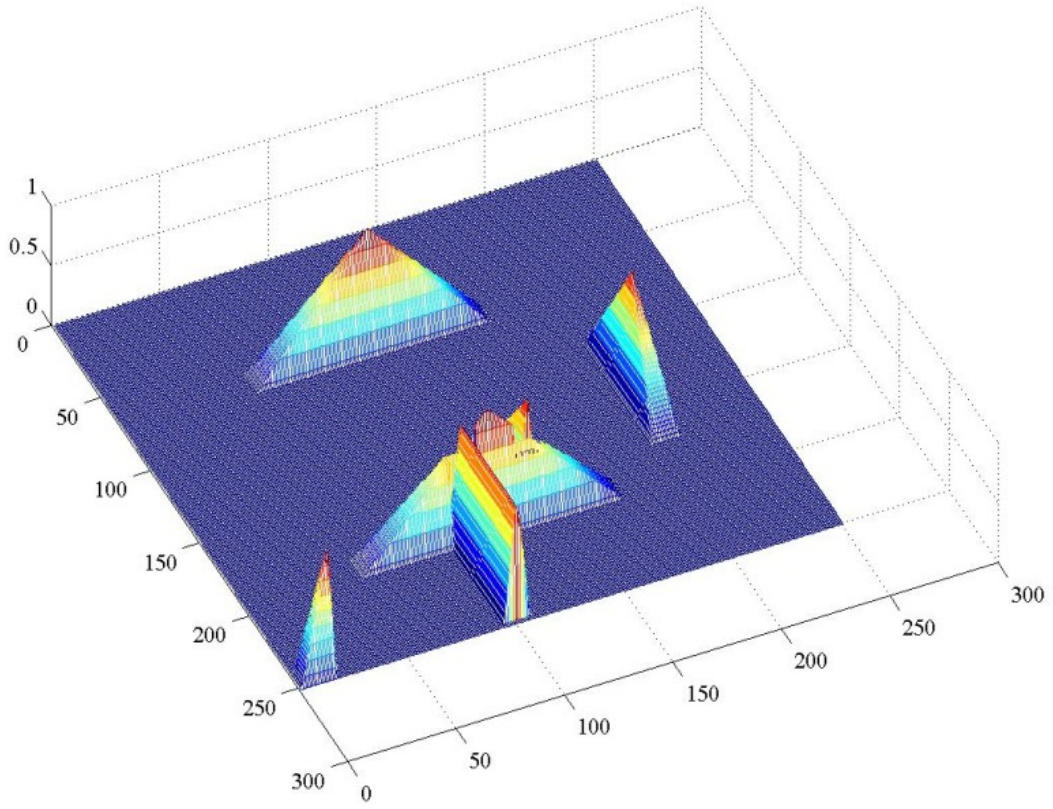


Figura 21 – Mapa ***Fuzzy*** do Ambiente.

4.9.1 Junção dos Obstáculos

Quando realizado a busca por obstáculos durante a leitura do mapa, é realizada a junção entre obstáculos, ou seja, se existirem dois obstáculos próximos entre si, e a distancia entre eles não for suficiente para garantir a passagem do robô entre os obstáculos, ou seja, a distancia entre os obstáculos for menor que 60 milímetros, a junção entre os obstáculos é realizado, formando assim um único obstáculo *fuzzy* presente no ambiente.

Quando formado um obstáculo *fuzzy* é calculado o meio deste obstáculo contendo assim um objeto com inicio, meio e fim, como pode ser observado na (figura 22), onde os obstáculos estão mapeados e o centro de cada obstáculo é o pico da pirâmide formada.



Fonte [FABRO 2001], FABRO, J. A. **Figura 22 – Obstáculos Mapeados.**

4.10 Comportamento Navegacional

A medida em que o robô navega pelo ambiente e dependendo dos valores obtidos pelos seus sensores o robô adquire um comportamento especifico dentro do ambiente.

Inicialmente o robô tende a seguir em direção ao ponto mais próximo de sua posição atual, ou seja, seu comportamento inicial é de buscar os pontos espalhados pelo mundo. Porém, quando são detectados obstáculos durante sua navegação, o módulo das RNA's é ativado, para que com isso o robô não venha a colidir com os obstáculos.

Quando detectado que o robô já passou pela determinada posição (x,y) do ambiente o comportamento de troca de RNA (intercalação de RNA) como descrito na seção 4.3 é acionado, ou seja, será determinado qual a RNA que entrará em execução, logo quando a RNA é desativada, ou seja, quando não existe nenhum sensor ativo, a busca pelo ponto mais próximo volta a tomar o controle do robô indicando a direção a ser tomada por este a fim de sair busca do ponto especificado.

A medida em que o robô navega pelo ambiente, e este por sua vez não encontra o ponto especificado, um contador é verificado para determinar a busca por outro ponto espalhado pelo ambiente, ou seja, podem existir casos em que o ponto inserido no ambiente fica inacessível pelo robô. Por exemplo, no caso de estar na mesma posição de um obstáculo no ambiente, este ponto se torna inacessível, e o próximo ponto é selecionado para a que o robô continue sua navegação.

Logo após a determinação de todos os pontos inseridos no ambiente, é realizada a construção do mapa *fuzzy* do ambiente como descrito na seção 4.9.

Quando são encontrados todos os pontos espalhados pelo ambiente e não existem obstáculos próximos ao robô e a construção do mapa *fuzzy* foi realizada, simplesmente o robô vaga pelo ambiente. O comportamento navegacional pode ser expresso pelo diagrama de estados apresentado na figura 23.

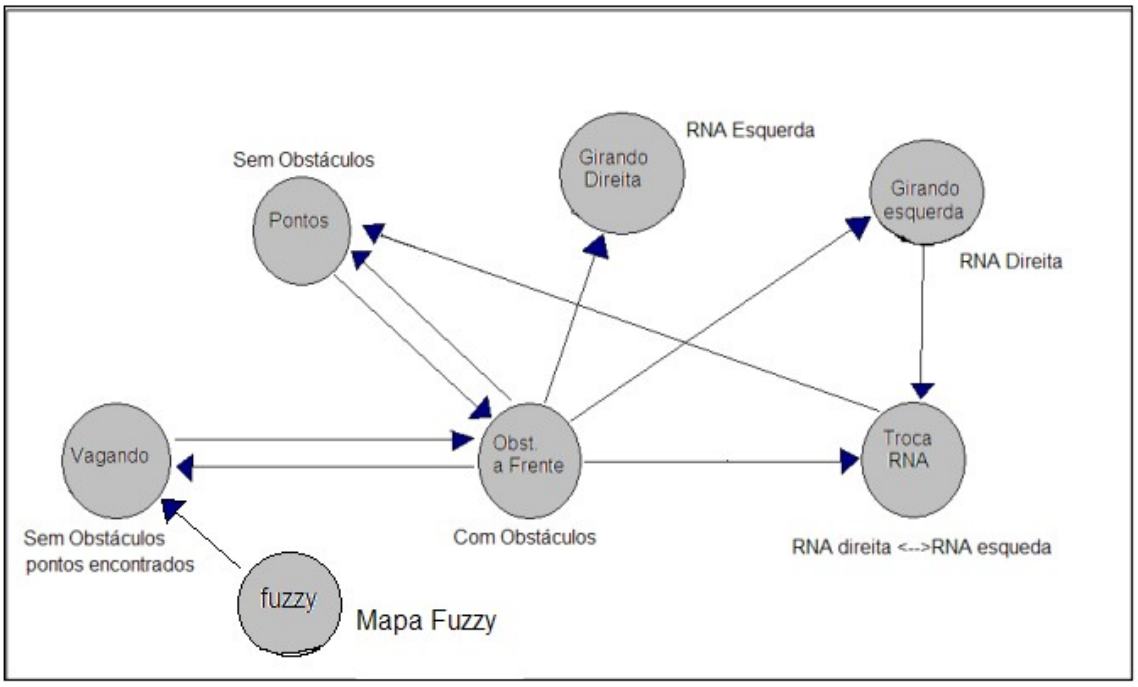


Figura 23 – Comportamento Navegacional

5 EXPERIMENTOS REALIZADOS

Inicialmente foram realizados diversos experimentos para a formulação das RNA's direita e esquerda, para que o robô viesse a realizar uma navegação estável livre de colisões. Para determinar se os padrões de treinamento estavam coerentes, diversas situações dentro do ambiente foram simuladas para a verificação da reação do robô em decorrência da situação em que se encontra, quando a maioria das possibilidades foram mapeadas e satisfeitas em virtude da reação do robô em cada uma delas, os padrões de treinamento estavam prontos.

Primeiro foram estabelecidos os padrões de treinamento para desvio de obstáculos à esquerda, logo após com estes padrões prontos foi possível determinar com mais facilidade os padrões de treinamento à direita, pois na maioria dos casos os padrões eram recíprocos.

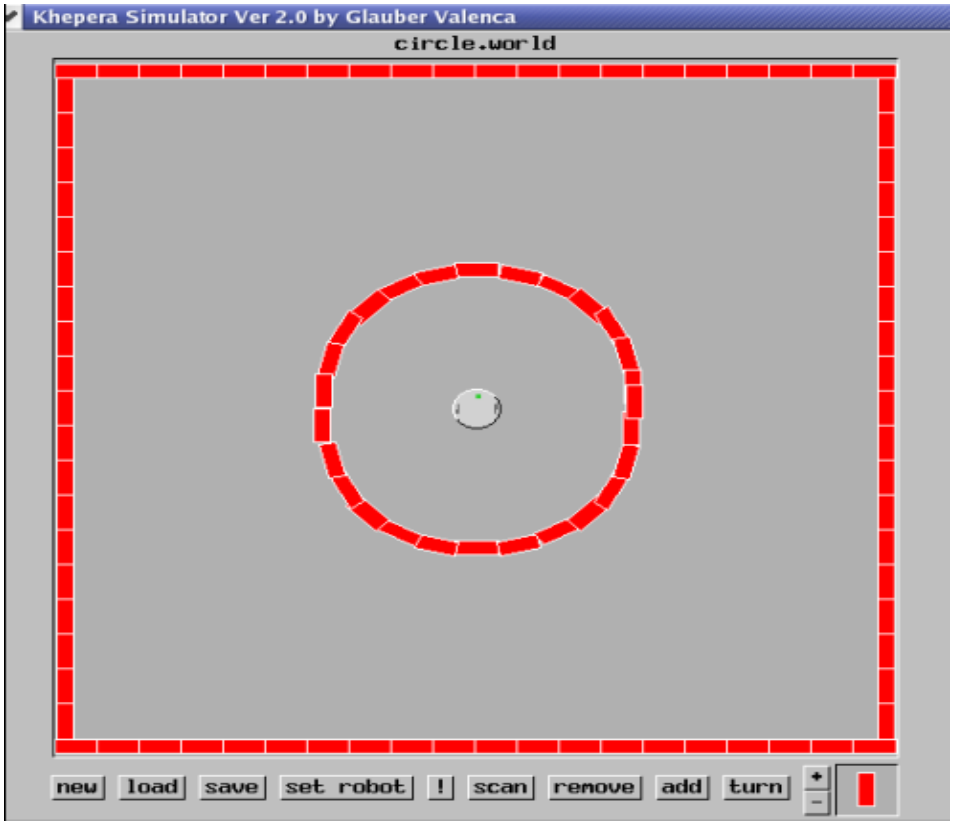
Terminada a fase de desenvolvimento dos padrões de treinamento, foi possível desenvolver o módulo de mapeamento do ambiente, pois, possuindo uma navegação estável, é possível realizar a extração do mapa métrico do ambiente. Foram realizados diversos experimentos em diferentes mundos (ambientes), sendo que na maioria deles o robô se manteve estável, ou seja, livre de colisões, realizando assim o mapeamento do ambiente.

Outra funcionalidade implementada com sucesso foi a busca por pontos espalhados pelo mundo, caso o robô não tenha acesso a determinado ponto, o robô recalcula o novo ponto a ser alcançado, e sai em busca do mesmo. Por exemplo, quando o robô está inserido em um ambiente onde sua locomoção fica restrita, como no caso do ambiente "*circle*"(figura 24), o robô não pode encontrar todos os pontos, fazendo assim a busca somente nos pontos onde o robô tem acesso.

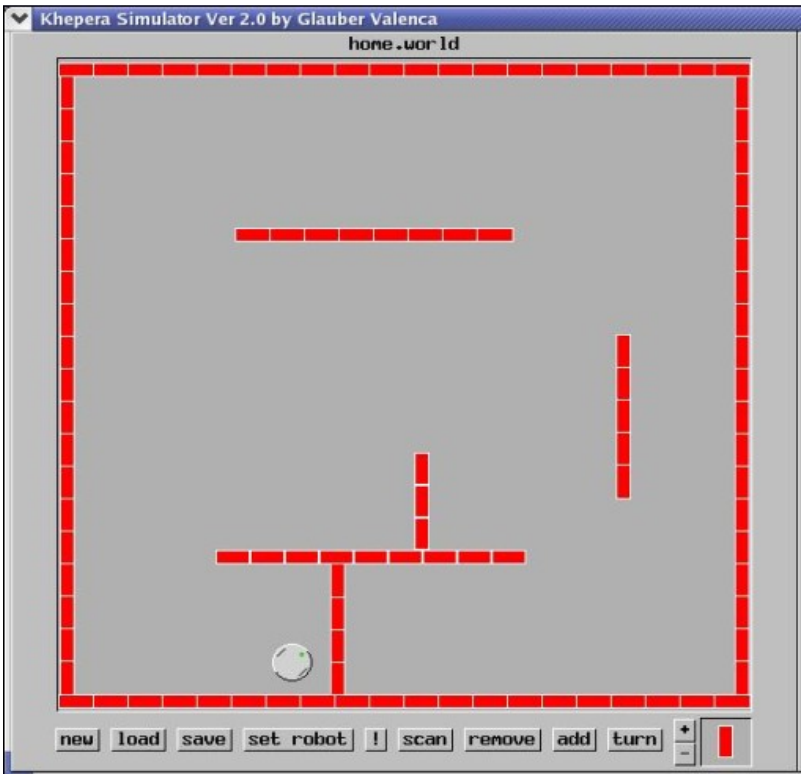
Para a construção do mapa *fuzzy,* o ambiente em que foram realizadas a maioria dos experimentos foi o ambiente denominado "home" (figura 25). A maior dificuldade encontrada para a construção do mapa *fuzzy*, é quando o mapa do ambiente construído pelo robô durante sua navegação não esta completo, ou seja, quando existe algum obstáculo no ambiente que não foi mapeado por completo.

Isto ocorre quando o robô não dá a volta completa em torno de algum obstáculo, fazendo com que a detecção deste obstáculo seja mais difícil de ser calculada. Outra dificuldade está relacionada ao mapa do ambiente propriamente dito, pois este possui pontos esparsos, pontos estes que não correspondem a obstáculos, porém sua leitura e detecção pode levar a criação de um obstáculo *fuzzy* inválido. Este comportamento aparece mesmo quando realizado um pré-processamento de filtragem do mapa do ambiente construído pelo robô.

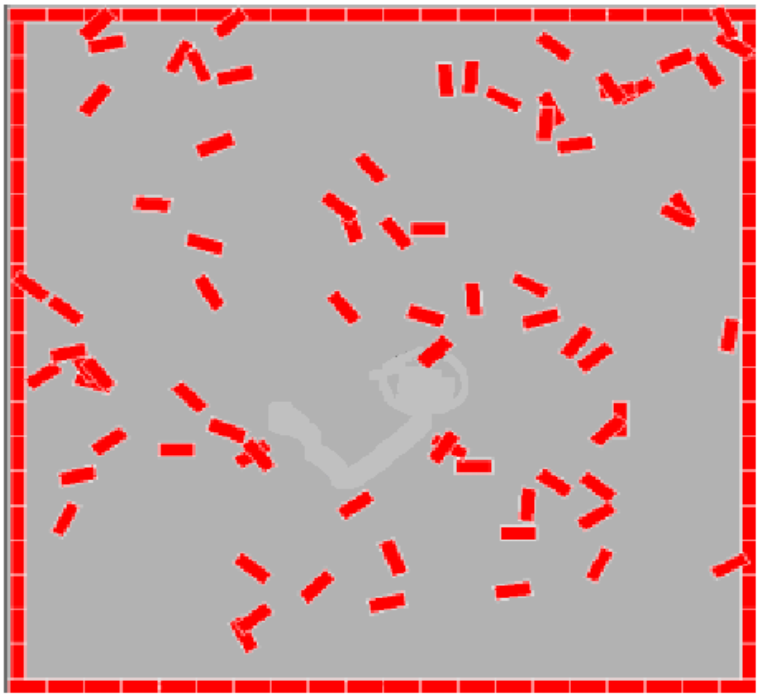
Com relação aos ambientes, o que proporcionou maior dificuldade foi o denominado “chaos” (figura 26), pelo fato de ser um ambiente completamente desestruturado, levando assim as leituras dos sensores do robô ao seu valor máximo (1023), e conseqüentemente paralisando o robô no ambiente. Porém a atuação do robô nos diferentes ambientes foi satisfatória, pois na maioria deles o robô conseguiu manter-se operando e alcançando seus objetivos.



Fonte: [MICHEL 1996]. **Figura 24** – Ambiente “circle”.



Fonte: [MICHEL 1996]. **Figura 25** – Ambiente “home”.



Fonte: [MICHEL 1996]. **Figura 26** – Ambiente “chaos”.

6 CONCLUSÃO

Neste trabalho foram apresentados os principais métodos utilizados para navegação autônoma de robôs móveis, e foram estudadas as principais técnicas de navegação, como por exemplo, a abordagem sensorial/reativa, e as técnicas referentes ao mapeamento, bem como as formas de representação de mapas do ambiente.

Este trabalho propõe uma arquitetura capaz de realizar a elaboração do mapeamento do ambiente a partir da sua interação e de leituras de sensores. Esta arquitetura é implementada, utilizando técnicas de navegação por redes neurais, construção de mapas métricos utilizando grades de ocupação, e construção do mapa simplificado do ambiente utilizando técnicas nebulosas.

O principal objetivo deste trabalho, que é a realização do mapeamento do ambiente a partir de leituras de sensores, foi atingido com sucesso, com a obtenção do mapa métrico do ambiente a medida em que o robô navega pelo mesmo.

Outra característica importante a ser ressaltada é a construção do mapa *fuzzy* do ambiente, baseado nas características obtidas durante o a etapa de navegação, características estas inseridas no mapa do ambiente. A maior dificuldade encontrada para a construção do mapa *fuzzy* é o tratamento dos ruídos obtidos pelo mapa do ambiente, onde inicialmente é realizada uma filtragem, para posterior construção do mapa *fuzzy.*

Dentre as contribuições trazidas pelo trabalho realizado, a principal está no conhecimento obtido ao longo de estudos realizados e a colocação deste estudo em prática com a implementação do sistema para a locomoção e mapeamento de ambientes desconhecidos em robótica móvel. Outra contribuição foi o desenvolvimento da arquitetura de mapeamento do ambiente, e sua implementação, gerando um sistema capaz de interagir e mapear qualquer ambiente desconhecido. A transformação do mapa métrico em um mapa *fuzzy* permite uma grande vantagem em termos de memória gasta para armazenar as

informações sobre o ambiente, facilitando sua utilização posterior, por algoritmos de planejamento de trajetórias.

Uma das maiores dificuldades encontradas ao longo do desenvolvimento deste trabalho está relacionada à própria área estudada, a área da robótica, que por sua vez está intimamente relacionado à área da IA, pois, por ser uma área nova, existem relativamente pouco material bibliográfico disponível. Outra dificuldade foi a própria plataforma de desenvolvimento utilizada (plataforma LINUX RED HAT 8.0), por falta de conhecimento sobre a mesma, além do tempo escasso para o desenvolvimento do que foi planejado anteriormente no que diz respeito à implementação do planejamento de trajetória. Um problema crítico encontrado durante a implementação do sistema foi a sua depuração, pois como foi implementado sem um ambiente gráfico de desenvolvimento, era necessário realizar impressões das saídas dos resultados para a depuração do código. Estas dificuldades, porém, foram superadas ao longo do decorrer deste trabalho.

6.1 Trabalhos Futuros

Uma proposta para um trabalho futuro é a implementação de um filtro utilizado para estimar os dados oriundos dos sensores do robô, pois, como estes dados nem sempre são confiáveis, é viável a utilização de alguma técnica para estimar estes dados, ou seja, de alguma forma transformar estes dados para dados mais confiáveis para sua utilização na implementação do sistema. Uma técnica amplamente utilizada para este objetivo é o Filtro de Kalman.

Com a utilização do mapa pode ser possível a implementação do planejamento de trajetórias de um determinado ponto a outro, ou seja, com a utilização do mapa é possível determinar qual o melhor caminho a ser percorrido entre dois pontos, pois o mapa possui as informações dos obstáculos no ambiente, possibilitando assim a determinação prévia do trajeto a ser percorrido pelo robô.

Outro exemplo de extensão deste trabalho pode ser a implementação dos algoritmos desenvolvidos em um robô real, para validar seu funcionamento. A

elaboração de um módulo de auto-localização do robô dentro do ambiente também poderia ser realizada com experimentações utilizando um robô real.

Um grande desafio também pode ser a elaboração de uma estratégia de exploração multi-nível, para ambientes cuja dimensão é maior que a capacidade de mapeamento do robô.

Bibliografia:

[BAI 2001] BAILEY, T. **Localization in Large Scale Environments,** Robotics and Autonomous Systems, 2001.

[BET 1994] BETKE, M. **Mobile Robot Localization Using Landmarks,** Internaltional joint Conference on Atrtificial Inteligence, Munich, Germany, 1994.

[BRAGA 2000] BRAGA, A.P., CARVALHO, A. C. P. L. F., LUDEMIR, T. B.; **Redes Neurais Artificiais: teoria e aplicações**. Livros Técnicos e Científicos (LTC), 2000.

[BROOKS 1986] BROOKS, R.; **A Robust Layered Control System for a Mobile Robot**. IEEE Journal of Robotics and Automation, Vol. 2, 1986.

[BRO 1991] BROOKS, R. A. **Intelligence Without Reason,** Artificial Inteligence, pg, 569 – 595, 1991.

[CRO 1985] CROWLEY, J. L. **Navigation et Modelisation pour un Robot Mobile.** Grenobe France, Laboratorie d' informatique fondamentale et d' inteligence artificialle, 1985.

[CHA 1996] CHATILA, R. **Autonomous Mobile Robot Navigation for Planet Exploration the Eden Project.** Laboratorie d'analise et d'Architecture des Systemes, 1996.

[DAM 99] DAM, J. V. **Environment Modelling for Mobile Robots,** Neural Learning for Sensor Fusion, University van Amsterdam,1998, Tese Apresentada.

[DEV 1995] DEVY, M. **On Autonomous Navigation in Natural Environment.** Robotics and Autonomous Systems, 1995.

[DIS 2001] DISSANAYKE, M. W. M. G. **A Solution to the Simultaneous Localization and Map Building Problem.** Transactions on Robotics and Automation. 2001.

[DUD] DUDEK, G. **Using Local Information in a Non-Local Way for Mapping Graph-Like Worlds.** Internaltional joint Conference on Atrtificial Inteligence, Los Altos, 1993.

[ELFES 1989] ELFES, A. **Unsign Ocupation Grids for Mobile Robot Perception and Navigation.** Computer Magazine, 1989.

[ELFES 1987] ELFES, A. **Sonar Based Real World Mapping and Navigation.** Journal of Robots and Automation, 1987.

[ENG 1994] ENGELSON, S. **Passive Map Learning and Visual Place Recognition.** Yale University, New Haven, Tese Apresentado, 1994.

[FABRO 2001] FABRO, J.A., LOPES, H.S., ARRUDA, L.V.R.; **Planejamento de Trajetos por Algoritmos Genéticos Baseados em um Mapa Fuzzy**. V Congresso Brasileiro de Redes Neurais , PUC Rio de Janeiro, 2001, Artigo apresentado.

[FARLEI 2002] FARLEI, J.H.; **Sistema de Controle Híbrido para Robôs Móveis Autônomos**, São Leopoldo, Universidade do Vale do Rio dos Sinos UNISINOS, 2002, Dissertação apresentada.

[FED 1999] FEDER, H. J. S. **Adaptive Mobile Robot Navigation and Mapping**. International Journal of Robotics Research, 1999.

[FOX 1999] FOX, D. **Probabilistic Methods for Mobile Robot Mapping**. Workshop on adptive spatial representations of dynamic environments, 1999.

[GUI 2001] GUIVANT, J. E. **Optmization of the Simultaneous Localizations and map-building Algortithm for real-time Implementation**, Transactions on robotics and Automations. 2001.

[HAL 1990] HALLAM, J.; **Intelligent Sensing and Control: Notes on Control Theory (part 2)**. Lecture Notes, Department of AI University of Edinburgh, 1990, Artigo apresentado.

[HEB 1995] HEBERT, P. **Probabilist Map Learning,** Toulouse Laboratorie d'Analyse et d'Architeture des Systemes, 1995.

[KUI 1991] KUIPERS, B. **A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations.** Robotics and Autonomous Systems, 1991.

[MAT 1990] MATARIC, M.; **Environment Learning Using A Distributed Representation**. Proceedings of the IEEE International Conference on Robotics and Automation. p. 402-406, 1990.

[MICHEL 1996] MICHEL, O.; **Khepera Simulator version 2.0 – User Manual.** University of Nice – Sophia Antipolis, Valbonne, France, 1996.

[MICHELON 2000] MICHELON, R.; Desvio de Obstáculos utilizando um Ambiente Simulado para o robô Khepera, Universidade Regional Integrada do Alto Uruguai e das Missões, Trabalho de Conclusão de Curso, Agosto, 2000.

[MCK 1995] MACKERROW, P. J.; **Introduction to Robotics**. Journal of Robots and Automation, 1995.

[PRESTES 2003] PRESTES, E.J.; **Navegação Exploratória Baseada nos Problemas de Valores de Contorno,** Porto Alegre, Universidade Federal do Rio Grande do Sul, 2003, Tese apresentada.

[PRESSMAN 1995] PRESSMAN, Roger S. **Engenharia de Software***,* Makron Books, 1995.

[RAO 1993] RAO, N. S. V. **Robot Navigation in Unknow Terrains,** Oak Ridge, Tenesse: Oak Ridge National Laboratory, 1993.

[ROM 2000] ROMERO, L.; **Contrucíon de Mapas e Localizacíon de Robots Móviles un Enfoque Híbrido**. Cuernavaca, TEC de Monterrey, 2000, Tese apresentada.

[ROY 1999] ROY, N.; **Coastal Navigation**, Conferência de robôs e automação. Artigo Apresentado.

[RUMELHART 1986] RUMELHART, D.E, HINTON, G.E., WILLIAMS, R.J.; *Learning internal representations by error propagation*, **Parallel Distributed Processing: Explorations in the Microstructure of Cognition**, Vol. 1, MIT Press, Cambridge, MA, 1986, pg. 318--362.

[RUSSEL 1995] RUSSELL, S. **Artificial intelligence: a modern approach**. Prentice-Hall, 1995.

[SMITH 1987] SMITH, R. **A Stochastic Map for Uncertain Spatial Relationships.** International Symposium on Robotics Research, 1987, Santa Cruz.

[THR 1999] THRUN, S.; **Learning Maps for indoor mobile robot navigation,** Jornal of Robots and Automation, 1999.

[WIDROW 1960] WIDROW, B., HOFF Jr, M.E.; **Adaptive switching circuits**. In IRE WESCON Convention Record, part 4, pages 96--104, 1960.

[YAM 1997] YAMAUCHI, B. A.; **A Frontier Based exploration for autonomous exploration**. Simpósio Internacional de inteligência computacional para robôs autônomos. Monterrey, 1997, Artigo apresentado.

[ZADEH 1988] ZADEH, L. A., ***Fuzzy Logic***, Computer, pg.83-93, Abril 1988.

[ZUBEN 1999] ZUBEN, J.V.; **Uma Caricatura Funcional de Redes Neurais Artificiais**. Disponível em http://www.inf.ufsc.br/~fischser/caricatura, acessado em 07/09/2003.