NeuroFSM: aprendizado de Autômatos Finitos através do uso de Redes Neurais Artificiais aplicadas à robôs móveis e veículos autônomos

Daniel Oliva Sales

NeuroFSM: aprendizado de Autômatos Finitos através do uso de Redes Neurais Artificiais aplicadas à robôs móveis e veículos autônomos

Daniel Oliva Sales

Orientador: **Prof. Dr. Fernando Santos Osório**

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

USP – São Carlos
Setembro de 2012

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

Sales, Daniel Oliva
S163n NeuroFSM: aprendizado de Autômatos Finitos através do uso de Redes Neurais Artificiais aplicadas à robôs móveis e veículos autônomos / Daniel Oliva Sales; orientador Fernando Santos Osório. -- São Carlos, 2012.
58 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2012.

1. Robótica Móvel. 2. Navegação Autônoma. 3. Redes Neurais Artificiais. 4. Autômatos Finitos. I. Osório, Fernando Santos, orient. II. Título.

Agradecimentos

Agradeço primeiramente a Deus pela vida, pelas oportunidades que me deu e por permitir que meus planos se realizassem.

Agradeço ao meu orientador, Fernando Osório pela confiança depositada em mim, pelo conhecimento compartilhado e pela ajuda no desenvolvimento deste trabalho.

Agradeço a toda a minha família, especialmente a meus pais, Wagner Sales e Esther Patrícia pelo constante apoio, carinho e dedicação. Agradeço também pela educação que me deram, os conselhos sempre corretos e o incentivo em concluir os estudos e ingressar na carreira acadêmica.

Agradeço minha namorada Elisangela pelo carinho e confiança, pela compreensão nos momentos de dificuldade, pelos momentos felizes que temos vivido juntos e por ter me apoiado e motivado a atingir meus objetivos.

Agradeço os meus amigos, pelo companheirismo, pelas horas de descontração e por estarem sempre dispostos a ajudar e ouvir. Agradeço especialmente o Mauricio, que desde antes do início do mestrado tem me ajudado, tanto nas atividades acadêmicas quanto pessoais.

Resumo

A Navegação autônoma é uma tarefa fundamental na robótica móvel. Para que esta tarefa seja realizada corretamente é necessário um sistema inteligente de controle e navegação associado ao sistema sensorial. Este projeto apresenta o desenvolvimento de um sistema de controle para a navegação de veículos e robôs móveis autônomos.

A abordagem utilizada neste trabalho utiliza Redes Neurais Artificiais para o aprendizado de Autômatos Finitos de forma que os robôs possam lidar com os dados provenientes de seus sensores mesmo estando sujeitos a imprecisões e erros e ao mesmo tempo permite que sejam consideradas as diferentes situações e estados em que estes robôs se encontram (contexto). Dessa forma, é possível decidir como agir para realizar o controle da sua movimentação, e assim executar tarefas de controle e navegação das mais simples até as mais complexas e de alto nível.

Portanto, esta dissertação visa utilizar Redes Neurais Artificiais para reconhecer o estado atual (contexto) do robô em relação ao ambiente em que está inserido. Uma vez que seja identificado seu estado, o que pode inclusive incluir a identificação de sua posição em relação aos elementos presentes no ambiente, o robô será capaz de decidir qual a ação/comportamento que deverá ser executado. O sistema de controle e navegação irá implementar um Autômato Finito que a partir de um estado atual define uma ação corrente, sendo capaz de identificar a mudança de estados, e assim alternar entre diferentes comportamentos previamente definidos. De modo a validar esta proposta, diversos experimentos foram realizados através do uso de um simulador robótico (Player-Stage), e através de testes realizados com robôs reais (Pioneer P3-AT, SRV-1 e veículos automatizados).

Abstract

Autonomous navigation is a fundamental task in mobile robotics. In order to accurately perform this task it is necessary an intelligent navigation and control system associated to the sensorial system. This project presents the development of a control system for autonomous mobile robots and vehicles navigation.

The adopted approach uses Artificial Neural Networks for Finite State Machine learning, allowing the robots to deal with sensorial data even when this data is not precise and correct. Simultaneously, it allows the robots to consider the different situations and states they are inserted in (context detection). This way, it is possible to decide how to proceed with motion control and then execute navigation and control tasks from the most simple ones until the most complex and high level tasks.

So, this work uses Artificial Neural Networks to recognize the robot's current state (context) at the environment where it is inserted. Once the state is detected, including identification of robot's position according to environment elements, the robot will be able to determine the action/-behavior to be executed. The navigation and control system implements a Finite State Machine deciding the current action from current state, being able to identify state changes, alternating between different previously defined behaviors. In order to validade this approach, many experiments were performed with the use of a robotic simulator (Player-Stage), and carrying out tests with real robots (Pioneer P3-AT, SRV-1 and autonomous vehicles).

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO

1

Introdução

As pesquisas relacionadas à aplicação de técnicas de Inteligência Artificial em robôs móveis e veículos autônomos sempre estiveram em posição de destaque junto à comunidade científica internacional (Bishop, 2000a) (Bishop, 2000b).

Uma das características mais desejáveis em um robô móvel é a capacidade de navegação autônoma. Como exemplo de um dos primeiros e mais famosos trabalhos nesta área, de acordo com Dudek e Jenkin (2000), podemos citar o projeto ALVINN (*Autonomous Land Vehicle In a Neural Network*) desenvolvido por Dean Pomerleau junto ao CMU nos anos 80. Este trabalho serviu de inspiração para inúmeros outros projetos de pesquisa e atualmente podemos citar os trabalhos de Thrun *et al.* (2006) e de outros pesquisadores no *Darpa Desert Grand Challlenge* e *Darpa Urban Challenge* (DARPA - Depto. de Defesa Americano), como sendo os projetos de maior visibilidade em termos do desenvolvimento de veículos autônomos.

A tarefa de navegação autônoma geralmente se divide nas seguintes operações básicas: localização, mapeamento, planejamento de trajetórias e controle de navegação (Wolf *et al.*, 2009). A tarefa de localização está associada com a estimativa da posição do robô em um ambiente conhecido, com base nos dados obtidos dos sensores. O mapeamento é a criação de um modelo para representar o ambiente com base na posição atual do robô. O planejamento de trajetórias consiste na definição de uma rota a ser seguida no mapa conhecido a partir da posição atual (inicial) até a posição desejada. E por fim, o controle de navegação consiste em obter informação do ambiente através dos sensores, processá-la e atuar sobre os motores fazendo com que o robô

se mova de forma segura, buscando alcançar a posição alvo ao mesmo tempo que desvia e evita colidir com os obstáculos presentes em seu caminho.

O foco principal deste trabalho é o controle em alto nível da tarefa de navegação autônoma de robôs móveis, com tomada de decisão sobre a atuação do robô em um ambiente conhecido. Sendo assim, é utilizado um método de aprendizado de máquina para que o robô aprenda os possíveis estados e respectivas ações em um sistema e dessa forma tenha capacidade de realizar a tarefa de navegação autônoma de uma forma segura.

1.1 Motivação

Há uma enorme quantidade de aplicações para robôs móveis. As aplicações desenvolvidas neste trabalho se relacionam principalmente ao monitoramento e segurança de ambientes internos e à condução autônoma e/ou assitida visando à prevenção de acidentes, no caso de veículos em ambientes urbanos.

Sistemas para navegação autônoma em ambientes urbanos (*outdoor*) e ambientes internos (*indoor*) requerem um sistema de controle e navegação inteligente e resistente a imprecisão nos dados sensoriais. O trabalho desenvolvido nesta dissertação busca justamente criar métodos computacionais que permitam a implementação deste tipo de sistema.

No caso das aplicações para ambientes internos, o sistema proposto nesta dissertação é adequado para tarefas de patrulhamento e monitoramento, tanto com um único robô como em sistemas multirrobóticos, auxiliando na segurança de setores estratégicos bem como atuando em missões não-tripuladas em ambientes hostis à presença humana.

Já no cenário de aplicações para ambientes urbanos, o sistema desenvolvido pode vir a integrar sistemas de auxílio ao motorista e direção autônoma, viabilizando a implementação de diversas aplicações comerciais.

1.2 Objetivo

O objetivo dessa dissertação é desenvolver um sistema de controle para navegação autônoma. Esse sistema deve ser adequado tanto para aplicações com robôs de pequeno porte como para aplicações com veículos autônomos, possuindo tolerância à imprecisão e erro nos dados de entrada. Além disso, o sistema deve ser capaz de lidar com o contexto em que o robô se encontra, escolhendo ações adequadas de acordo com cada situação.

Desta forma, o sistema desenvolvido é caracterizado como um sistema híbrido (reativo/-deliberativo), já que o robô não responde simplesmente aos dados de entrada, mas toma suas

decisões com base em um planejamento prévio, definido através de uma sequência de estados e ações.

O sistema é composto de um módulo de Inteligência Artificial responsável por permitir que o robô aprenda a reconhecer os possíveis estados de um autômato que descreve pontos chave do ambiente e as respectivas ações relacionadas. Com isso, o robô deve ser capaz de perceber as diferentes situações e reagir corretamente.

O trabalho desenvolvido nesse projeto de mestrado se encaixa em um projeto maior realizado no Laboratório de Robótica Móvel (LRM[1]) do Instituto de Ciências Matemáticas e de Computação (ICMC) ligado ao INCT-SEC[2] (Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos). O INCT-SEC possui atualmente quatro grupos de trabalho principais, sendo dois deles envolvendo desenvolvimento de pesquisas sobre controle e navegação de robôs e veículos autônomos: GT1 - "Robôs Táticos para Ambientes Internos"; e GT2 - "Veículos Terrestres Autônomos" (Figura 1.1).

(a) Robôs Táticos (b) Veículos Autônomos

Figura 1.1: Equipamentos usados nas pesquisas em desenvolvimento pelo LRM e INCT-SEC.

Como objetivos secundários, espera-se aplicar o método de navegação desenvolvido em algumas tarefas específicas como o patrulhamento de ambientes internos com um ou mais robôs e a navegação em ambientes urbanos (vias de trânsito reais), validando a metodologia proposta nesse projeto com diferentes sistemas sensoriais.

1.3 Estrutura da Dissertação

Esta dissertação está organizada da seguinte forma:

[1] `http://www.lrm.icmc.usp.br`
[2] `http://www.inct-sec.org`

- Capítulo 2: Descreve brevemente trabalhos relacionados que serviram como referência para este projeto.
- Capítulo 3: Apresenta a composição do sistema proposto, e a forma como os módulos envolvidos se relacionam .
- Capítulo 4: Descreve os experimentos realizados e os resultados obtidos.
- Capítulo 5: Apresenta a conclusão do trabalho e sugestões para possíveis trabalhos futuros.

CAPÍTULO

2

Trabalhos Relacionados

Muitas abordagens diferentes têm sido desenvolvidas para navegação autônoma. Esta tarefa requer o uso de sensores como os de posicionamento, referencial inercial, detecção de obstáculos, imagem e orientação individualmente ou combinados (Wolf *et al.*, 2009)(Buehler *et al.*, 2007)(Goebl *et al.*, 2008). É através das informações sensoriais que se constrói o mapa do ambiente e se realiza a auto-localização, assim como o planejamento de trajetórias, que se baseia no mapa e posições inicial e final conhecidas. Se o mapa do ambiente e a localização do robô não forem conhecidos, na maioria dos casos apenas uma navegação reativa seria possível.

Segundo Grassi e Okamoto (2012), as arquiteturas de controle de navegação podem ser divididas em três tipos: puramente reativas, puramente deliberativas e híbridas (reativas/deliberativas). Nesse contexto, a deliberação está associada ao processo de tomada de decisão ou planejamento das ações e movimentos do robô utilizando um modelo interno do mundo para que se possa alcançar um determinado objetivo. Por exemplo, um robô que precisa se locomover de um ponto "A" para um ponto "B", conhece o mapa do ambiente e sua localização nesse mapa, planeja a sua trajetória com base em uma sequência de movimentos. Já a reatividade, está associada a execução de ações pré-definidas em resposta a uma informação sensorial obtida localmente, como um robô que segue em frente e desvia para um dos lados toda vez que um objeto for detectado à sua frente, em uma distância inferior a um valor estabelecido.

De forma geral, a abordagem deliberativa é voltada para a formulação de um plano de ação definido, e a abordagem reativa está relacionada com a execução das ações do robô em

tempo real em um ambiente dinâmico. As arquiteturas híbridas combinam estas características principais das abordagens deliberativa e reativa, utilizando planejamento de ações juntamente com a execução destas de forma reativa.

O uso de um sistema puramente reativo não é adequado para o tipo de aplicações que este trabalho visa atender, uma vez que a reação imediata à informação obtida dos sensores não garante uma navegação segura e correta quando se trata de um caminho mais complexo.

Além disso, podem ocorrer situações onde nem sempre a informação sensorial é suficiente para determinar precisamente a localização e a situação exata em que o robô realmente se encontra. Sendo assim, uma possibilidade promissora é a de adotar um plano global mais amplo, responsável pela troca de contexto, com influência sobre o comportamento do robô. O uso de autômatos finitos permite a descrição das tarefas do robô como estados, onde cada estado está associado a um conjunto de ações (comportamento reativo local). Isto motivou um estudo sobre o uso de autômatos finitos em Robótica, apresentado na seção 2.1.

2.1 Autômatos Finitos em Robótica

Em Robótica, abordagens baseadas em Autômatos Finitos (Hopcroft e Ullman, 1979) são frequentemente utilizadas, como por exemplo em *Situated Automata*(Medeiros, 1998) e *Reactive Deliberation Architecture* (Sahota, 1994).

Em (Marino *et al.*, 2009), um autômato finito foi utilizado para tarefa de patrulhamento multirrobótico. Neste trabalho, um autômato simples e pré-definido foi implementado, de forma a representar os possíveis comportamentos dos robôs patrulheiros. As tarefas de localização e navegação eram baseadas em mapas pré-estabelecidos criados usando-se um laser SICK e um método de SLAM baseado em filtro de partículas.

Diversas equipes participantes do *DARPA Urban Challenge* também utilizaram abordagens baseadas em autômatos finitos, como a equipe Oshkosh (Beck *et al.*, 2007), que utilizou um esquema de controle para supervisão de eventos para modelar as regras e restrições da corrida bem como os comportamentos e táticas do veículo. O planejamento de trajetórias foi feito através de uma modificação do algoritmo de Dijkstra (Russell e Norvig, 2003), e o sistema sensorial composto de um sistema de visão computacional combinado com sensores laser.

O uso de um autômato finito (*FSM - Finite State Machine*) se mostra interessante porque os sistemas podem ser facilmente descritos como uma sequência de estados (trocas de contexto) que levam em consideração as entradas (dados dos sensores) e assim fazem a troca entre um estado (situação) para outro definindo-se para cada estado uma ação específica.

Desta forma, o uso de autômatos finitos provê noção de etapas (sequência) e de contexto, evitando que o robô gere uma reação equivocada diante de uma leitura sensorial inesperada.

Pode-se afirmar portanto que o uso de autômatos finitos em robótica está diretamente relacionado ao controle deliberativo do comportamento dos robôs. Este conceito deliberativo pode ser aplicado no planejamento de trajetórias em alto nível, soluções baseadas em *Navegação Topológica*.

Este método de navegação utiliza mapas topológicos (grafos que representam pontos-chave do ambiente) e é usado principalmente porque estes modelos já armazenam implicitamente informações semânticas sobre o ambiente. Além disso, são robustos à imprecisão nos dados de entrada já que possuem a vantagem de não precisarem de informação métrica precisa, e gerenciam muito bem comportamentos reativos, sem necessitar de algoritmos complexos e computacionalmente custosos para localização e mapeamento.

Em (Wang *et al.*, 2009), este conceito é utilizado para a navegação autônoma de robôs em ambientes internos, utilizando feixes de laser para detecção dos nós do mapa topológico (grafo), que é construído em tempo de execução. Essa abordagem permite inclusive a navegação em ambientes parcialmente desconhecidos. Esta idéia é também utilizada em (Lee *et al.*, 2006), onde é apresentado um sistema de baixo-custo baseado em Sonar para navegação topológica autônoma em corredores.

Os estados que representam cada nó são obtidos através do reconhecimento de padrões na estrutura do ambiente, e como os sensores estão sujeitos a erros e imprecisão, especificar manualmente os parâmetros que caracterizam cada estado com base na leitura sensorial não é uma tarefa fácil, mesmo nos autômatos mais simples. Um método de aprendizado de máquina pode dispensar a necessidade de uma descrição detalhada dos estados pelo programador, e ainda acrescentar maior tolerância a imprecisão dos sensores, como será apresentado na seção 2.2.

2.2 Redes Neurais Artificiais em Robótica

O uso de um método de aprendizado de máquina como Redes Neurais Artificiais (Haykin, 1998) apresenta-se como um método bastante adequado para processar os dados obtidos dos sensores, identificar e classificar os possíveis estados de um autômato e determinar as ações a serem tomadas.

Diversos trabalhos como o projeto ALVINN (Pomerleau, 1996) e outros (Foedisch, 2004) (Shinzato, 2010) utilizam RNAs como classificador dos dados de entrada, detectando regiões navegáveis no ambiente. Shinzato (2010) utiliza um método de navegação baseado em visão computacional para *Road Following*. Ele utiliza um comitê de Redes Neurais para classificar

os atributos obtidos da imagem e assim identificar as regiões navegáveis do ambiente. A Figura 2.1 exemplifica a classificação feita para um frame. São mostrados nessa figura (da esquerda para a direita, e de cima para baixo) o frame original, a classificação final obtida pelo sistema, uma representação da matriz de navegabilidade e a classificação obtida em cada uma das 5 RNAs.

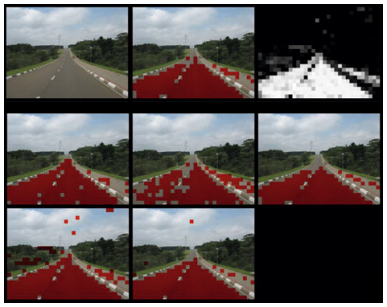


Figura 2.1: Comitê de 5 RNAs classificando um frame capturado pela câmera.

As Redes Neurais Artificiais permitem uma boa tolerância ao ruído e a imprecisão nos dados de entrada, assim como são capazes de identificar estados e o momento em que deve ser realizada a transição de um estado a outro. As redes neurais também são muito eficientes em generalizar conhecimentos e adaptar suas saídas em relação a diferentes entradas, mesmo que estas situações não tenham sido explicitamente ensinadas a rede (capacidade de generalização) (Haykin, 1998). Sendo assim, as redes neurais possuem diversas propriedades que são bastante interessantes e adequadas para a implementação de máquinas de estados, identificando transições de estado e gerando ações como saídas da rede.

2.3 Aprendizado de Autômatos Finitos por Redes Neurais Artificiais

O uso de Redes Neurais Artificiais para o aprendizado de autômatos finitos não é uma proposta totalmente nova, pois esta questão já vinha sendo discutida desde os anos 90 (Giles *et al.*, 1995) (Omlin e Giles, 1996) (Frasconi *et al.*, 1996) (Cleermans *et al.*, 1989), época em que os modelos de redes neurais se desenvolveram, amadureceram e ocuparam um importante papel junto às demais pesquisas das áreas de Inteligência Artificial e de Aprendizado de Máquina. Os trabalhos desenvolvidos por pesquisadores como Lee Giles apontavam nesta direção, a do uso de redes neurais para o aprendizado (e extração) de autômatos finitos. Entretanto, muitos destes trabalhos de aprendizado de autômatos acabaram aplicados no aprendizado de gramáticas e

aplicados em problemas de reconhecimento de seqüências (Siegelmann e Giles, 1997), onde poucos trabalhos foram realizados no sentido de aplicar o aprendizado de autômatos finitos por redes neurais em problemas de robótica.

Trabalhos mais recentes sobre o aprendizado de autômatos finitos por redes neurais aplicados em problemas de robótica (veículos autônomos) foram desenvolvidos principalmente por Fernando Osório, Farlei Heinen, Milton Heinen e Luciane Fortes (Heinen *et al.*, 2007) (Osorio *et al.*, 2002). Estes trabalhos, no entanto, foram mais focados em uma aplicação específica, o estacionamento de um veículo autônomo em uma vaga paralela. Pouco foi feito para avaliar as possibilidades de extensão destes trabalhos para outras aplicações, ou, sobre o uso de outras configurações de redes neurais no aprendizado de autômatos, motivando assim o desenvolvimento das aplicações propostas nesta dissertação.

No trabalho desenvolvido em (Heinen *et al.*, 2007), a tarefa de estacionamento em vagas paralelas foi caracterizada na forma de uma sequência de estados e ações que deveriam ser controlados por uma RNA (Figura 2.2). Em função disto, foi adotado um modelo de RNA com recorrência (Figura 2.3), denominado J-CC.

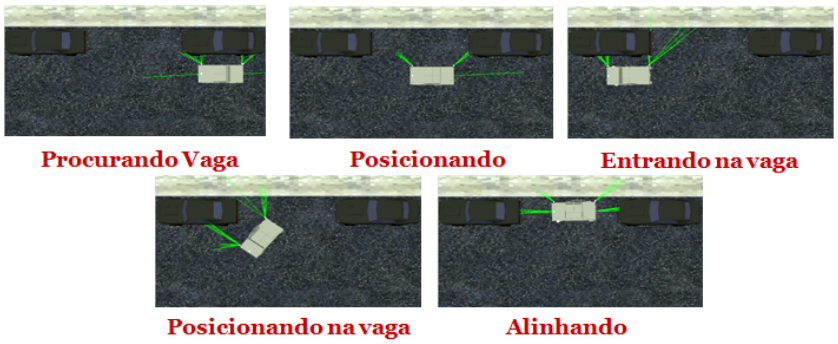


Figura 2.2: Estados do veículo na tarefa de estacionamento (Heinen *et al.*, 2007).

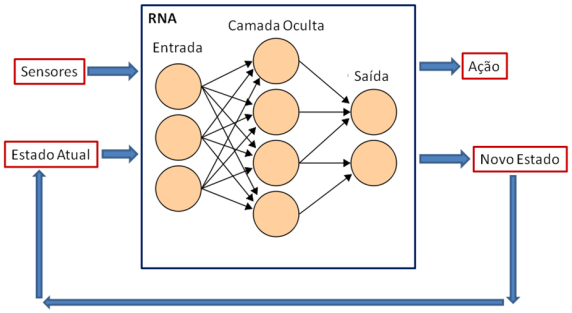


Figura 2.3: Rede Neural Recorrente para aprendizado de sequências de estados (Heinen *et al.*, 2007).

A forma de funcionamento da rede J-CC é através do uso de um conjunto de entradas que indicam o estado atual da rede (estado) juntamente com um conjunto de entradas que indicam a situação em que se encontra o robô no momento (dados dos sensores). A rede gera um conjunto de saídas que indicam o novo estado que a rede deve assumir em função de suas entradas (percepção externa) e do seu estado anterior (estado atual), e consequentemente a nova ação relacionada a este novo estado. O próximo estado, que pode inclusive permanecer igual ao estado atual, é obtido na saída da rede e então é re-injetado na entrada da própria rede, formando assim a recorrência. O diagrama de estados utilizado neste trabalho está representado na Figura 2.4.

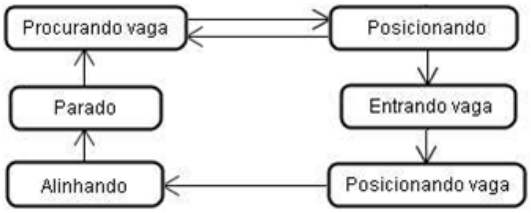


Figura 2.4: Diagrama de estados (Heinen *et al.*, 2007).

Esta dissertação apresenta o desenvolvimento de um sistema de navegação autônoma baseado nesta abordagem, porém foi utilizado um tipo de RNA sem recorrência, visando atender uma quantidade maior de aplicações conforme descrito mais detalhadamente na seção 3.2.

2.4 Considerações Finais

Robôs móveis autônomos podem executar tarefas de alto nível e sequências de ações, ao mesmo tempo em que necessitam de um comportamento local capaz de evitar colisões e reagir aos elementos do ambiente. Autômatos Finitos tem características que permitem este tipo de abordagem, e por isso tem sido muito utilizados em robótica.

Dada essa escolha, há a necessidade de reconhecer os estados do autômato e trocar de estados com base nos dados dos sensores. Essa tarefa de reconhecimento de padrões pode ser complexa levando-se em consideração o ruído nos dados de entrada e erros relacionados aos sensores. As RNAs aparecem como uma solução adequada para esta implementação graças a sua capacidade de aprendizado através de exemplos e de generalização desse aprendizado.

Neste trabalho são combinadas essas duas abordagens, criando um sistema capaz de aprender os estados de um autômato de forma similar ao trabalho realizado por Heinen *et al.* (2007), porém com o diferencial de atender a diversas aplicações e permitir que os autômatos sejam gerados na etapa de planejamento de trajetória, e não na etapa de treinamento da RNA apenas.

CAPÍTULO

3

Metodologia

O sistema de navegação autônoma baseado em RNA e Autômatos finitos apoia-se na idéia de que a tarefa de navegação pelo ambiente pode ser descrita na forma de um autômato finito onde as diferentes características desse ambiente (pontos-chave) são interpretadas como estados e transições. Sendo assim, uma Rede Neural é treinada para aprender a reconhecer todos os possíveis estados desse autômato, de tal forma que de acordo com os dados obtidos dos sensores ela tenha capacidade de definir qual é o estado atual.

Cada estado atual tem um ou mais comportamentos específicos associados (comportamentos reativos ou deliberativos). A navegação é realizada com base em um planejamento de trajetória, de tal forma que o robô conhece a sequência de estados que descreve um determinado caminho. Desta forma, a RNA passa a atuar como um classificador dos dados de entrada permitindo ao robô conhecer o contexto em que se encontra e percorrer qualquer trajeto. Em outras palavras, o robô deve se deslocar pelo ambiente buscando alcançar determinadas posições ou situações segundo uma sequência de estados que descreve a sua tarefa/rota.

Este capítulo apresenta portanto a metodologia de desenvolvimento do Sistema de Navegação Autônoma, discutindo os aspectos envolvidos nesta tarefa de navegação e detalhando as etapas de implementação utilizadas. A Seção 3.1 mostra como é representado o ambiente, e como a sequência de ações a serem executadas (tarefa ou trajetória) pode ser destrita na forma de um autômato. A Seção 3.2 descreve como funciona a etapa de preparação e classificação dos dados pela RNA para definição do estado atual, introduzindo alguns conceitos utilizados

na construção e utilização de RNAs . A Seção 3.3 explica como é realizado o controle híbrido, que associa planejamento de trajetória à comportamentos reativos no controle local dos robôs, apresentando uma visão geral do sistema. Por fim, a Seção 3.4 trata dos métodos de avaliação dos resultados obtidos nos experimentos além de apresentar a ferramenta utilizada na realização dos testes iniciais.

3.1 Representação do Ambiente através de Autômatos Finitos

O primeiro aspecto a ser considerado no desenvolvimento de um sistema de navegação autônoma é o tipo de representação do ambiente utilizada. Essa escolha define o conjunto de técnicas e algoritmos de localização e mapeamento que serão adotados e desenvolvidos no decorrer do projeto, sendo de extrema importância. Pode-se dividir os modelos de representação em duas categorias principais: modelos métricos e modelos topológicos.

A principal diferença entre esses dois modelos é que nos modelos métricos as proporções e distâncias entre todos os elementos do ambiente são determinantes; são criados mapas que descrevem com precisão a disposição dos elementos no espaço, em duas ou três dimensões. Já nos modelos topológicos o ambiente é representado de uma forma estrutural, onde a semântica é priorizada; são criados grafos que representam a interligação entre os elementos do ambiente.

A Figura 3.1 apresenta os dois modelos de representação para um mesmo ambiente interno, o mapa métrico está representado à esquerda e o mapa topológico à direita.

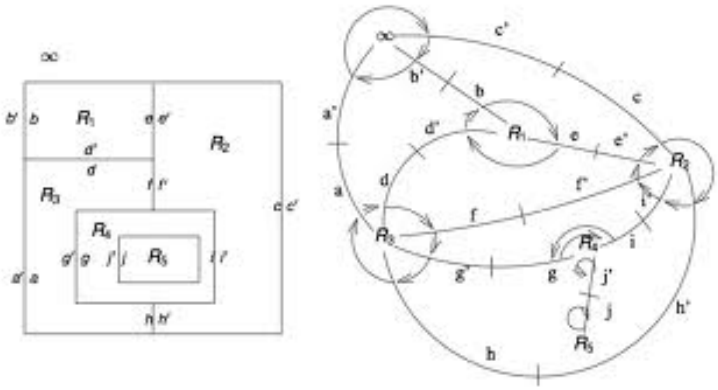


Figura 3.1: Mapas métrico e topológico de um ambiente.

Como neste projeto a trajetória/tarefas do robô são descritas na forma de autômatos finitos, é desejável adotar um modelo que tenha características favoráveis a esse tipo de implementa-

ção. Essa descrição de trajetória através de autômatos não é trivial quando se tem o ambiente representado em forma de mapa métrico ou quando não há representação prévia. Sendo assim, foi adotado para este sistema o modelo baseado em mapas topológicos devido a uma importante característica: tanto os mapas topológicos como os autômatos são representados através de grafos, ou seja, o mapa do ambiente e os autômatos de controle são representados com a mesma estrutura de dados.

Considerando ainda que o controle de navegação é baseado justamente em uma sequência de etapas a serem seguidas (trechos a serem visitados), e que o mapa topológico armazena partes específicas do ambiente e pontos de referência, a correspondência entre o autômato e o mapa topológico é imediata. Isso significa que nesta representação o autômato finito é o próprio mapa topológico. Cada estado do autômato corresponde a uma parte específica do ambiente, e a transição entre os estados acontece quando o robô se desloca para uma nova área do ambiente mapeado.

Desta forma, cada possível caminho neste ambiente é um sub-grafo desse autômato. A navegação de um nó a outro ocorre tomando-se como estado inicial o nó onde o robô se encontra, e como estado final o nó de destino no mapa topológico.

A Figura 3.2 exemplifica essa abordagem. Dado o mapa topológico de uma residência, um caminho entre sala (posição incial do robô) e área de serviço poderia ser descrito pela sequência {Sala-Corredor-Cozinha-Área de Serviço}, ou {Sala-Jardim-Área de Serviço}.

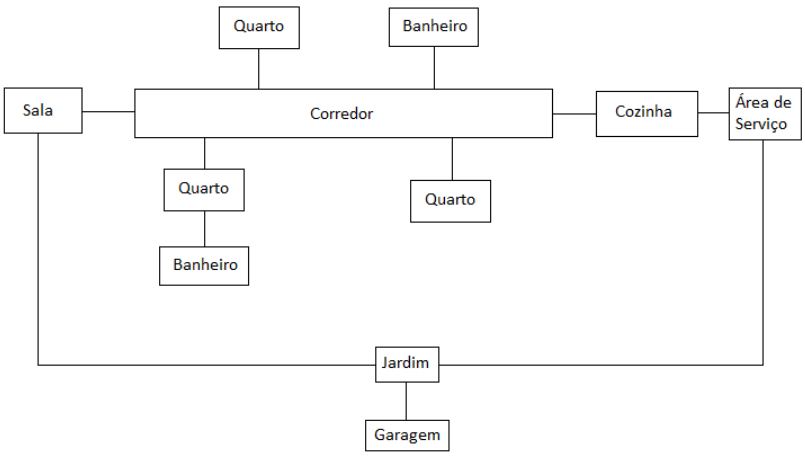


Figura 3.2: Exemplo de mapa topológico de uma residência.

Para que seja realizada a navegação de um ponto a outro usando este tipo de abordagem, o autômato (caminho) entre os pontos origem/destino no grafo deve ser estabelecido e armazenado em memória para que o robô planeje sua trajetória com base nessa sequência.

Uma grande vantagem deste tipo de notação é eliminar a necessidade de algoritmos precisos de localização e mapeamento. O robô consegue estimar sua posição global (em qual ponto do mapa se encontra), e com isso consegue executar a tarefa de localização ao mesmo tempo em que controla a navegação.

A percepção do estado atual (parte do mapa onde o robô se encontra) se dá pelo reconhecimento de certos padrões estruturais no ambiente, como por exemplo a presença de paredes em posições específicas ou o formato da área navegável, minimizando o impacto causado por possíveis erros e imprecisão nos sensores, o que seria crítico em um modelo métrico. Para este reconhecimento é utilizado um classificador neural, descrito detalhadamente a seguir.

3.2 Classificador Neural

O classificador é o módulo responsável por determinar qual é o estado atual (classe) dado um conjunto de valores de entrada obtido do sistema sensorial do robô em um certo instante. Esse classificador foi construído utilizando-se um método de aprendizado de máquina conhecido como "*Redes Neurais Artificiais*".

O aprendizado de máquina se divide em três categorias: aprendizado supervisionado, aprendizado não-supervisionado e aprendizado por reforço (ou semi-supervisionado) (Baranauskas e Monard, 2003) (Mitchell, 1997) (Carbonell *et al.*, 1984).

O método adotado neste trabalho é o de aprendizado supervisionado, onde é dado para a rede um conjunto de pares de valores (entradas e saídas) que representam as possíveis situações (estados) em um determinado ambiente e suas respectivas classificações. O modelo precisa então passar por uma fase de treinamento, onde recebe os valores de entrada e verifica se para cada um deles foi gerada a saída esperada. Se isto não ocorrer, o modelo é reestruturado de forma a produzir o resultado correto (Mitchell, 1997).

As RNAs tem seu funcionamento inspirado nos mecanismos e estruturas do cérebro humano, sendo formadas por neurônios e conexões entre os neurônios responsáveis pelo processamento de informações. O neurônio artificial, ilustrado na Figura 3.3 é composto de 3 elementos básicos: conexões que permitem propagação de dados para outros neurônios, um somador que efetua a somatória dos dados que chegam até esse neurônio, e uma função de ativação/transferência, que determina o valor do dado de saída de acordo com o resultado do somador.

A forma como esses neurônios se conectam é chamada de topologia, e influencia diretamente no aprendizado da rede. Tradicionalmente, as topologias das RNAs apresentam seus neurônios dispostos em camadas. Há três grupos principais: *Perceptron*, *MLP* e *Recorrente*. O

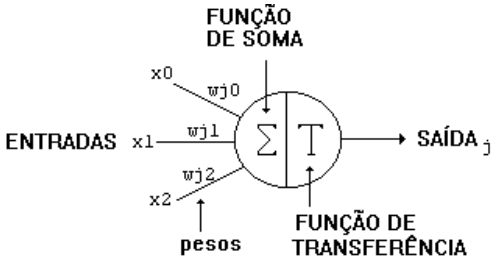
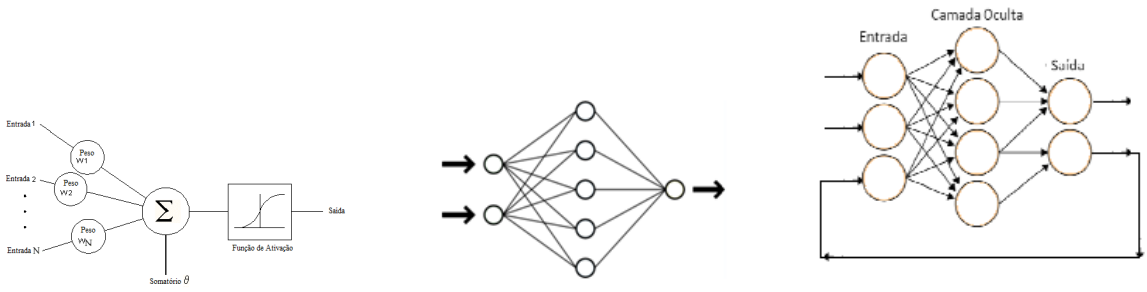


Figura 3.3: Representação de um Neurônio Artificial.

Perceptron (Figura 3.4(a)), também conhecido como *single-layer feedforward* é a configuração mais básica de RNA. Possui apenas uma camada de entrada e outra de saída, sem ciclos entre as conexões dos neurônios. Se restringe à resolução de problemas lineares (Braga *et al.*, 2000). O *Multi-layer Perceptron* ou *multi-layer feedforward* (Figura 3.4(b)) adiciona camadas ocultas ao modelo *perceptron* tradicional, estendendo sua capacidade para resolução de problemas não-lineares, além de possibilitar o aumento na precisão dos resultados (Haykin, 1998). Na topologia *Recorrente* (Figura 3.4(c) há a presença de um laço que conecta a saída da rede à sua entrada, de forma a realimentar a RNA. As redes recorrentes possuem a capacidade de simular uma memória de curto prazo.



(a) Perceptron (b) MLP (c) Recorrente

Figura 3.4: Diferentes topologias de RNAs.

Embora em (Heinen *et al.*, 2007) a topologia adotada tenha sido a Recorrente, neste trabalho foi adotada a topologia MLP. Essa escolha foi feita com base no fato de que em (Heinen *et al.*, 2007), o estado atual detectado torna-se um dos dados de entrada da rede através da recorrência, influindo na detecção. Como no modelo proposto busca-se criar uma rede neural que possa ser usada (e reaproveitada) no controle da navegação de robôs para diferentes trajetórias, optou-se por não criar uma rede neural treinada especificamente para uma sequência de estados. Isso permite que a sequência dos estados seja definida na etapa de planejamento de trajetória (em

tempo de execução), e não na etapa de treinamento, de forma que uma única rede neural já treinada possa ser aplicada nas diferentes configurações possíveis de autômatos finitos dentro de um mapa topológico.

O treinamento da rede é realizado antes do início da execução do sistema. O algoritmo de aprendizado escolhido influi tanto na correção da resposta, quanto no tempo de treinamento. Durante o treinamento, os pesos das conexões são ajustados para a melhor correspondência entre as entradas e saídas dos exemplos utilizados como base de treinamento (Russell e Norvig, 2003). Alguns dos algoritmos de aprendizado mais utilizados são o algoritmo *backpropagation* (Rumelhart *et al.*, 1986) e suas derivações como o *resilient backpropagation* (Riedmiller e Braun, 1994). O algoritmo utilizado neste trabalho é o *resilient backpropagation* (R-Prop). Este algoritmo tem alcançado bons resultados para redes *feedforward* em comparação a outros algoritmos em convergência e tempo de treinamento.

Os pares entrada/saída utilizados no treinamento devem ser gerados a partir de uma coleta de dados feita com o robô no ambiente alvo. A saída esperada (classificação) de cada exemplo deve ser especificada manualmente pelo supervisor do sistema nesta etapa. São utilizados como dados de entrada a informação obtida do sistema de percepção (sensores), e como saída uma codificação binária para representação do estado detectado. A Figura 3.5 ilustra esta definição.

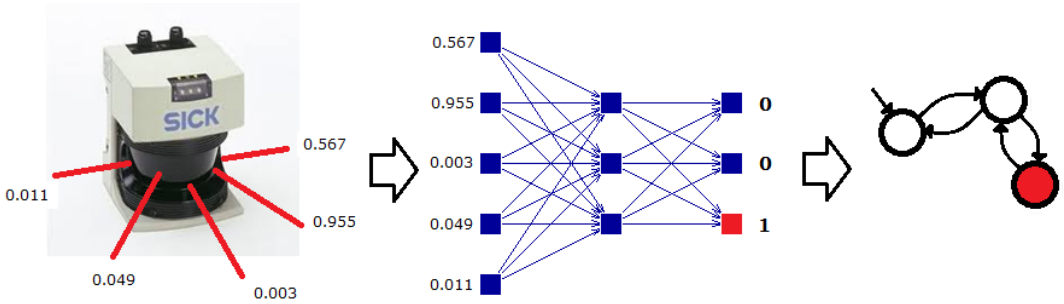


Figura 3.5: Formato adotado de entradas e saída da RNA.

A classificação manual pode ser realizada de uma forma bastante simples, apesar do grande volume de dados adquiridos, uma vez que o supervisor precisa apenas identificar em uma sequência de dados coletados o momento em que houve uma mudança de estado (indicando o novo estado). A Figura 3.6 apresenta três instantes em uma coleta de dados. Nos instantes 1 e 2, o sistema está associando automaticamente a classificação “reta” aos dados detectados. Entre os instantes 2 e 3, o supervisor precisa acionar uma tecla indicando que as próximas coletas estão associadas ao estado “cruzamento”, e assim a base de dados já é criada com cada exemplo associado a uma saída esperada.

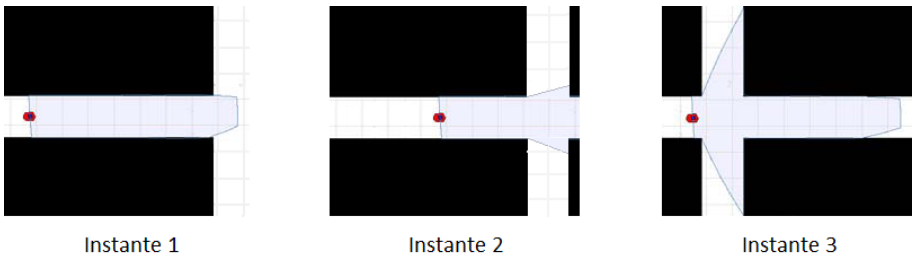


Figura 3.6: Exemplo de navegação para coleta de dados em ambiente simulado.

É importante ressaltar que para obter resultados satisfatórios no treinamento da rede, deve ser coletada uma quantidade suficiente de exemplos para que a base de dados esteja balanceada, isto é, os dados devem ser coletados na mesma proporção para cada classe.

Uma vez coletados e classificados esses dados, é realizado o treinamento em uma determinada quantidade de ciclos até que se alcance uma convergência, e depois é feita uma validação que define se a RNA está pronta para o uso. Essa validação é feita pelo método chamado de *cross-validation* (Haykin, 1998), onde parte dos dados é destinada ao treinamento, e parte utilizada como teste. Este método permite avaliar se a rede realmente aprendeu as diferentes classes ensinadas e tem capacidade de generalizar ou se apenas "decorou" os dados utilizados no treinamento.

A implementação da RNA foi feita com a ferramenta Stuttgart Neural Network simulator (SNNS) (Snns, 2010), com criação da topologia e treinamento da rede no software JavaNNS. Uma vez que o treinamento tenha sido concluído com sucesso e os pesos das conexões ajustados, a rede é convertida então para linguagem C com a ferramente SNNS2C para que seja incluída no código fonte do programa de controle do robô. A RNA está a partir desse ponto pronta para ser usada como classificador dos dados de entrada no reconhecimento de estados.

3.3 Navegação Topológica Híbrida

Uma vez que a Rede Neural já foi treinada para classificar os dados sensoriais reconhecendo os padrões que descrevem os estados, o sistema já é capaz de iniciar a etapa de navegação autônoma. As fases iniciais de preparação do sistema descritas até aqui são necessárias para que com um único treinamento o sistema seja capaz de reconhecer as diferentes características do ambiente e assim navegar por qualquer caminho (sub-grafo do mapa topológico). A Figura 3.7 apresenta estas etapas de configuração do sistema, até a geração do autômato para um caminho estabelecido.

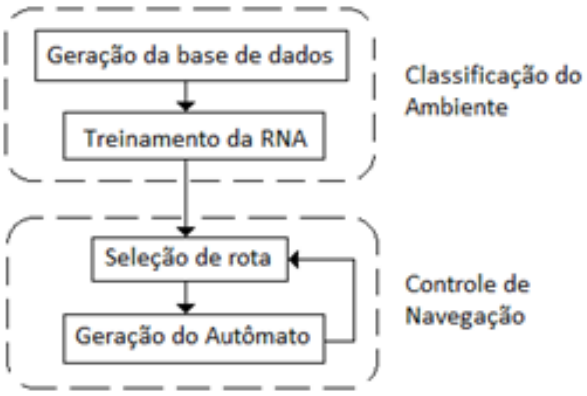


Figura 3.7: Etapas de preparação do sistema antes do início da navegação.

Dada uma rota desejada (nó de destino e posição inicial/atual do robô), a geração do autômato de navegação é feita selecionando-se um caminho entre esses dois nós (sub-grafo) no mapa topológico, e associando a cada estado intermediário ações específicas que levem o robô a alcançar o estado seguinte. Essa sequência de estados (pontos-chave a serem "visitados") e ações relacionadas (comportamentos) é armazenada em memória para que seja executada na fase de navegação.

Com a lista de estados/ações armazenada, a Navegação Topológica Híbrida pode ser iniciada. Esta etapa consiste em seguir a sequência pré-estabelecida de estados, ativando para cada estado um comportamento reativo adequado para aquela situação. Por exemplo, ao se detectar que o robô está em um corredor, é associado a este estado uma ação de seguir as paredes avançando pelo corredor até detectar o final deste.

O sistema sensorial fica responsável por capturar dados do ambiente que são processados pelo classificador, indicando qual é o estado atual. A partir do momento em que as detecções correspondem ao próximo estado armazenado na lista, há uma troca de contexto, alternando o estado atual e consequentemente o comportamento associado. A Figura 3.8 ilustra este processo.

O comportamento deliberativo por si só não é suficiente para garantir uma navegação segura em cada parte do ambiente, já que os mapas topológicos não oferecem informações detalhadas sobre a distância e posicionamento absoluto dos elementos no ambiente. É por este motivo que comportamentos reativos são associados a cada estado, permitindo ao robô realizar uma navegação local segura e adequada para cada diferente configuração do ambiente (ex. manter-se no centro de um corredor ao seguir em frente, permanecer dentro da faixa correta em uma via de trânsito, e assim por diante).

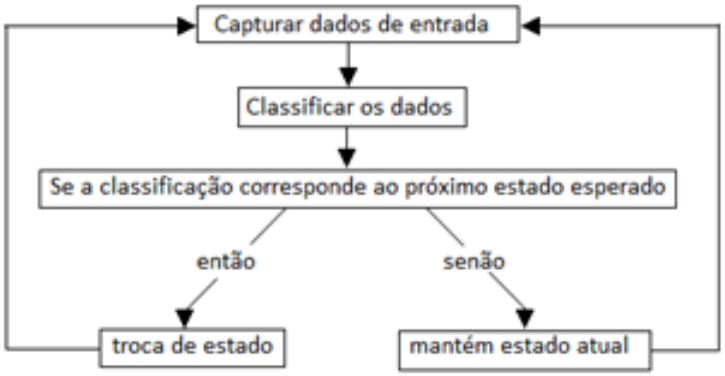


Figura 3.8: Fluxograma do controle de navegação.

No caso de situações que exigem tomada de decisão como cruzamentos, esta é realizada na etapa de planejamento de trajetória (na geração do autômato), deliberadamente associando ao estado o comportamento correto desejado (curva à direita, esquerda ou seguir em frente).

Conforme citado anteriormente, a tarefa de auto-localização é realizada simultaneamente à própria navegação, já que a posição aproximada do robô no ambiente sempre corresponde ao estado atual do autômato.

A detecção de estados não se restringe apenas à identificação de características estruturais do ambiente para navegação. É possível reconhecer situações relacionadas com outras tarefas executadas paralelamente à navegação. Em uma aplicação de patrulhamento por exemplo, podem ser inseridos dados correspondentes a detecção de intrusos no treinamento da rede. Em aplicações de trânsito podem ser inseridos dados correspondentes a detecção de obstáculos e pedestres, e assim por diante, definindo comportamentos específicos do robô para essas situações.

3.4 Implementação Inicial e Avaliação de Desempenho

Antes de implementar o sistema em um robô real, é necessário validar o método proposto em ambiente simulado. Isso é possível graças à ferramenta Player-Stage (Collett *et al.*, 2005), uma ferramenta que permite a implementação de algoritmos de controle robótico e também faz a interface para conexão com os robôs, de tal forma que o mesmo código utilizado na simulação pode ser reutilizado posteriormente nos robôs reais, sem necessidade de alterações severas. É interessante validar o sistema inicialmente em ambiente simulado porque o ambiente real possui uma quantidade bem maior de fatores que poderiam interferir na classificação (como ruído e imprecisão dos sensores). Uma vez que o sistema mostra-se funcional em ambiente simulado,

o código pode ser reaproveitado para o experimento com o robô real, passando apenas por pequenas adaptações.

Uma vez que o sistema foi projetado e implementado, é desejável adotar métodos para analisar e avaliar os resultados obtidos. Essa avaliação é feita de duas maneiras:

- Análise da taxa de acerto do classificador neural;
- Análise do comportamento do robô na tarefa de navegação.

Para a análise da taxa de acerto da RNA, deve ser adotado um modelo de validação antes da etapa de treinamento, gerando as diferentes bases de dados necessárias a partir da base de dados inicial coletada. Um método bastante utilizado para este fim é o *Stratified N-Fold Cross Validation*. Trata-se de um método de validação cruzada que particiona a base de dados, utilizando uma parte dos dados para treinamento e a outra para teste. Cada partição deve manter a proporção dos dados de cada classe sem repetição. São então realizados N treinamentos, de tal forma que todos os dados sejam usados para teste. Por fim, a média da taxa de acerto nas N execuções é considerada como a taxa final de acerto da RNA gerada. A Figura 3.9 ilustra a preparação das bases de dados para 5 partições em um sistema de 3 classes.

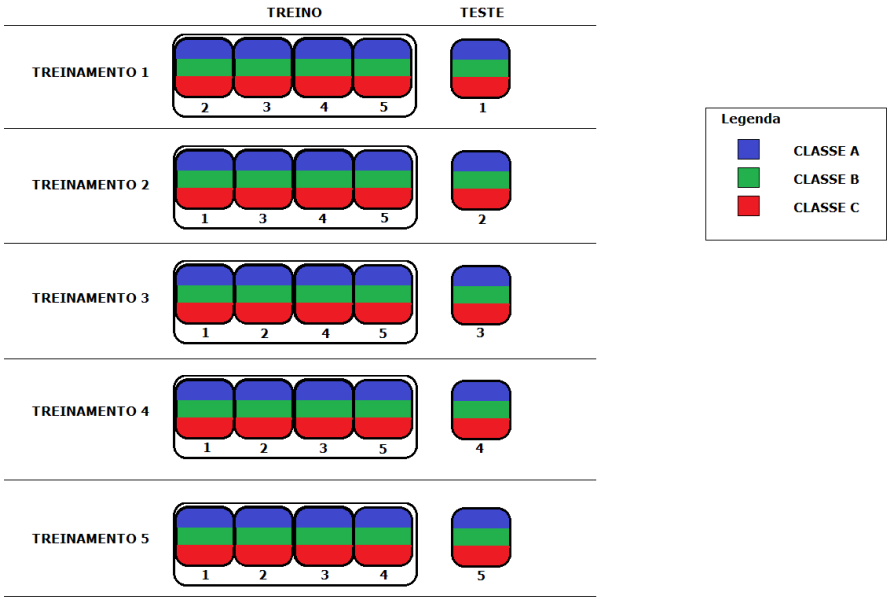


Figura 3.9: Exemplo de divisão dos dados utilizando *Stratified 5-Fold Cross Validation* em um sistema com 3 classes.

Nos experimentos realizados, optou-se por utilizar o método com 5 partições para que fossem usados 80% dos dados para treinamento e 20% para teste. Consequentemente, a taxa média

de acerto é sempre obtida após 5 treinamentos, representando a capacidade de generalização da RNA utilizada. É desejável calcular a taxa total de acerto da RNA assim como observar as matrizes de confusão para avaliar o erro por classe. Uma discrepância muito grande entre os erros por classe pode ser o indicativo de uma base de dados ruim, com uma ou mais classes mal representadas.

A análise do comportamento do robô por sua vez, consiste em verificar se houve sucesso na tarefa de navegação, com conclusão do trajeto e execução de tarefas conforme o planejamento. Nesta etapa avalia-se ainda se os comportamentos reativos foram bem sucedidos no desvio de obstáculos e prevenção de acidentes. Avalia-se portanto os aspectos funcionais do sistema.

3.5 Ajuste de Parâmetros e Limitações

A implementação bem sucedida da metodologia proposta em aplicações de navegação autônoma depende do ajuste de alguns parâmetros e da adoção de algumas práticas, observadas nos experimentos realizados durante a execução deste trabalho.

A primeira observação é com relação à quantidade de estados implementados. Eles devem representar os principais *tipos* de situação possíveis em um determinado ambiente, definidos por suas características estruturais, e não toda a sequência que compõe o caminho propriamente dito. Desta forma, uma pequena quantidade de estados básicos pode se repetir no autômato do caminho, simplificando a etapa de construção do classificador neural e produzindo melhores resultados. De acordo com os experimentos realizados, observou-se que quanto menor a quantidade de estados maior a taxa de acerto da RNA, e consequentemente a confiabilidade na transição de estados.

Outro aspecto importante é a criação da base de treinamento. A coleta deve ser realizada não através de uma trajetória suave e constante, mas sim de uma captura baseada em caminhos sinuosos e movimentos variados, permitindo que sejam coletados exemplos do robô nas mais diversas posições possíveis dentro de cado estado, prevendo possíveis posições que o robô possa alcançar durante o controle reativo e diminuindo portanto as chances de uma troca de estados acidental em uma situação isolada, onde o controle reativo possivelmente teria condições de retomar a orientação correta na pista.

Ainda tratando do treinamento da RNA, cabe destacar aqui que como o método baseia-se em aprendizado supervisionado, a preparação da base de treinamento é uma etapa fundamental para o sucesso da etapa de navegação, sendo necessário analisar cuidadosamente as situações possíveis no ambiente para geração de uma base abrangente, e ao mesmo tempo com potencial para generalização.

Já no âmbito do controle, o gargalo é o ajuste dos ângulos de direção e velocidade. Estes, devem permitir um movimento suave, sem movimentos bruscos, evitando assim desviar demais do trajeto esperado, o que prejudicaria a detecção de estados.

A troca de estados é portanto um ponto crítico do sistema. Se em alguma situação um novo estado não for detectado a tempo, o robô manterá o comportamento do estado anterior, da mesma forma que se uma transição ocorrer cedo demais, o novo comportamento será inadequado para a situação atual. Em ambos os casos, a tarefa de auto-localização é sériamente afetada, comprometendo toda a execução do sistema, já que as referências de contexto e posição global serão perdidas.

Apesar desse risco, bons resultados foram alcançados após a inclusão de um contador de repetições que atua como um filtro, descartando detecções inesperadas e garantindo assim que somente um volume mínimo de observações consecutivas do estado esperado ative de fato a transição. Essa quantidade mínima deve ser ajustada de acordo com a taxa de aquisição de dados do sensor, para que a troca de estados ocorra no momento mais adequado.

3.6 Considerações Finais

Neste capítulo foi apresentada a metodologia adotada para desenvolvimento do sistema de navegação autônoma através da descrição dos principais módulos e conceitos utilizados. Foram discutidos os aspectos estruturais do sistema, como forma de representação do ambiente e classificação dos dados sensoriais.

A metodologia proposta baseia-se em uma abordagem topológica para representação do ambiente, aproveitando as características desse modelo para implementar o aprendizado neural de estados que representam partes de um mapa topológico.

A navegação é implementada através de uma arquitetura de controle híbrida, que combina a execução de um plano deliberativo para geração do trajeto a ser percorrido em sequência (considerando o mapa topológico), com o uso de comportamentos reativos para controle da navegação local dentro de cada estado (parte do ambiente).

Foram apresentadas formas de avaliação dos sistemas gerados, e discutiu-se sobre a necessidade de utilizar ferramentas de simulação robótica que permitem a implementação de experimentos utilizando robôs simulados e robôs reais. Estes experimentos estão descritos no capítulo a seguir.

CAPÍTULO

4

Experimentos e Resultados

Para testar e avaliar a metodologia proposta e cumprir os objetivos deste trabalho, diversos experimentos foram realizados aplicando a metodologia descrita nesta dissertação em diferentes situações. Os testes foram conduzidos tanto em ambiente simulado como em robôs reais. Diversos tipos de sensores foram utilizados, entre eles câmera RGB monocular, laser SICK e sensor kinect. O método foi avaliado tanto em ambientes internos como em vias reais de trânsito. Todos os softwares foram desenvolvidos utilizando a linguagem C++ com a biblioteca SNNS (Snns, 2010) para geração das RNAs.

Os trabalhos realizados foram complementados com a elaboração e submissão de artigos relatando os resultados parciais, o que ajudou a direcionar e organizar as pesquisas ao longo do mestrado, bem como consolidar os conhecimentos adquiridos.

Este capítulo se divide em sete experimentos principais, apresentados por ordem cronológica. A Seção 4.1 apresenta o primeiro experimento utilizando um autômato para controle da navegação. Neste experimento, não foi utilizada ainda uma RNA para aprendizado do autômato de controle, apenas para o reconhecimento de padrões em imagens. Em seguida, na Seção 4.2 é descrita uma outra forma de implementação de autômatos no controle de navegação, desta vez como filtro, diminuindo a oscilação na classificação dos dados de entrada. Na Seção 4.3 é descrito o primeiro experimento onde a o aprendizado neural de autômatos foi implementado, com uma RNA aprendendo um autômato para navegação em um ambiente interno e sistema sensorial baseado em laser. Na Seção 4.4 é descrito um experimento que propõe uma evolução

ao modelo anterior, utilizando um algoritmo de busca de menor caminho no planejamento de trajetórias, em uma tarefa de patrulhamento multirrobótico. Na Seção 4.5, é apresentado um sistema de navegação baseado em visão 3D para ambientes internos utilizando sensor kinect. Por fim, nas Seções 4.6 e 4.7 é discutida a implementação do sistema de navegação topológica em vias de trânsito reais, aplicando a metodologia proposta com sistema sensorial baseado em visão.

4.1 Sistema de Navegação Baseado em Visão com controle por Autômatos Finitos

O experimento descrito nesta seção foi o primeiro experimento realizado envolvendo o uso de um autômato finito para controle de trajetória. Embora tenha sido utilizado um classificador neural para os dados de entrada, esse classificador não informava o estado atual diretamente, ele era utilizado apenas para detectar as áreas navegáveis. Era necessário portanto um pós-processamento da saída da rede para definição do estado atual.

Foi usado um robô de pequeno porte, o SRV-1Q (Figura 4.1) equipado com uma câmera de vídeo e sistema Wi-Fi de comunicação. O objetivo desse experimento era que o robô pudesse navegar por um trajeto que continha diversas situações, como retas, curvas suaves e de 90 graus. O sistema sensorial adotado, baseado em visão apresenta características muito vantajosas como baixo custo, peso e consumo de energia, viabilizando seu uso nas mais diversas aplicações.

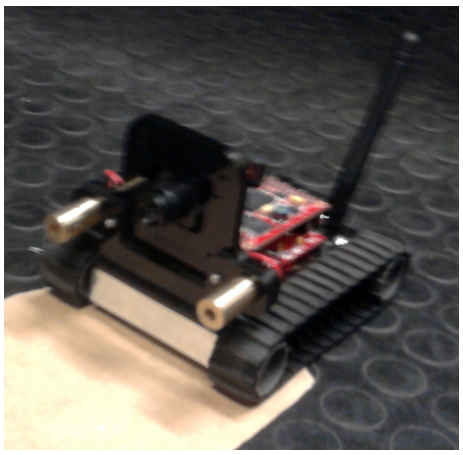


Figura 4.1: Robô Surveyor SRV-1Q utilizado no experimento.

O uso de um autômato finito para planejamento de ação é necessário porque há algumas situações especiais (como as curvas de 90 graus, por exemplo) onde o sistema de visão detecta

a curva antes do ponto onde o robô precisa realmente virar, e no ponto onde deve ser realizada a curva, a pista já não é mais visível. Sendo assim, é preciso adicionar uma informação de contexto em memória (sequenciamento de ações que vai além do comportamento puramente reativo) para que ao chegar nessa situação onde a pista não é mais visível o sistema saiba para qual lado deve virar para continuar a navegação.

Considerando as características de um sistema baseado em visão, o primeiro passo a ser executado é a segmentação da imagem, usando um algoritmo que realize a detecção das áreas navegáveis. Foi usada para este fim uma versão simples do classificador desenvolvido por Shinzato (2010) para extrair da imagem informações que definem quais são os trechos navegáveis e não-navegáveis, obtendo como resultando uma *matriz de navegabilidade*, estrutura que é então utilizada como entrada para o sistema de controle do robô.

O classificador utilizado é composto de quatro RNAs, cada uma recebe quatro ou cinco atributos da imagem como entradas (características de RGB e HSV) e gera como saída um mapa de navegabilidade: uma matriz que representa o grau de certeza de que um bloco da imagem representa uma área navegável ou não.

A imagem capturada pela câmera possuia 320×240 pixels, portanto para reduzir o tempo de processamento essa imagem era segmentada em blocos de 10×10 pixels, de forma que eram processados pela RNA esses 768 blocos, e não todos os 76800 pixels individualmente. A matriz de navegabilidade gerada possui portanto dimensões 32×24. A Figura 4.2 ilustra a divisão da imagem original.

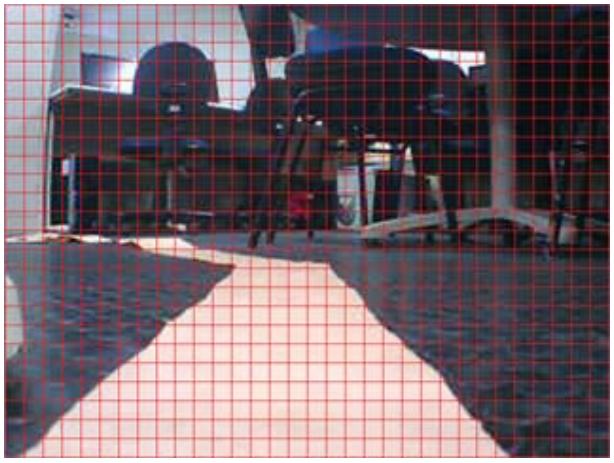


Figura 4.2: Frame dividido em blocos de 10×10 pixels.

Os atributos utilizados como entrada de cada rede são apresentados na Tabela 4.1, e a Figura 4.3 apresenta a estrutura do classificador utilizado.

Tabela 4.1: Atributos de entrada para cada RNA.

RNA	*Atributos*
RNA1	Média de R, média de B, média de H, entropia de V e energia de HSV
RNA2	Média de R, média de H, entropia de H e entropia de V
RNA3	Média de B, entropia de S, entropia de V, energia de S e entropia de HSV
RNA4	Média de B, entropia de V, energia de S, variância de S e entropia de RGB

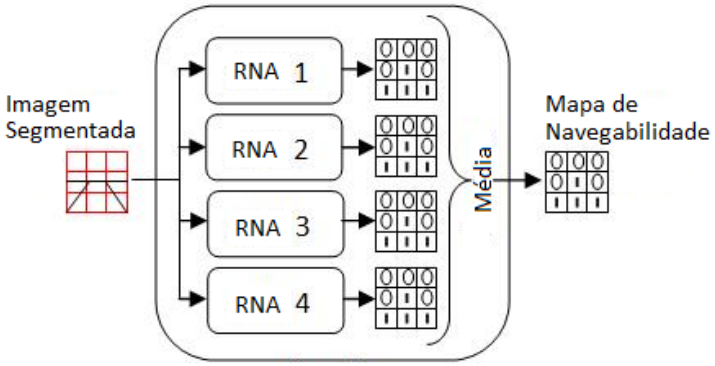


Figura 4.3: Estrutura do classificador. Valor no mapa de navegabilidade é a média entre as saídas das RNAs.

Como os valores finais na saída variam entre 0.0 e 1.0, um *threshold* foi aplicado de forma a classificar os blocos como não-navegável (0.0), navegável (1.0) e incerteza (0.5).

O treinamento do classificador foi feito utilizando uma interface interativa onde são selecionados os blocos navegáveis e não-navegáveis em um frame para calibração do sistema. Como a matriz de navegabilidade por si só não armazena informações de contexto, um pós-processamento foi realizado para o reconhecimento de padrões, aproveitando-se de uma característica interessante em ambientes estruturados: situações semelhantes apresentam uma forma visível semelhante, apresentando mapas de navegabilidade muito parecidos. Essa situação é ilustrada na Figura 4.4, que mostra como diversas estradas em linha reta apresentam características em comum diante da câmera, mesmo em ambientes diferentes.

Dada essa propriedade, foi utilizado um método para avaliar áreas de interesse na matriz (mapa de navegabilidade) para definir o estado atual. A figura 4.5 apresenta a forma como isso foi feito. A área A0 está sempre acima da linha do horizonte, portanto não é utilizada no processamento. As áreas A1, A2 e A3 estão relacionadas com a forma da área navegável detectada, permitindo indentificar qual o estado atual (curva, reta, etc). Uma vez que a matriz está preenchida com 0s e 1s, basta calcular a média em cada área de interesse para definir qual forma está representada na matriz.

Figura 4.4: Fotos de diferentes retas. Após processadas todas resultam em mapas de navegabilidade parecidos.

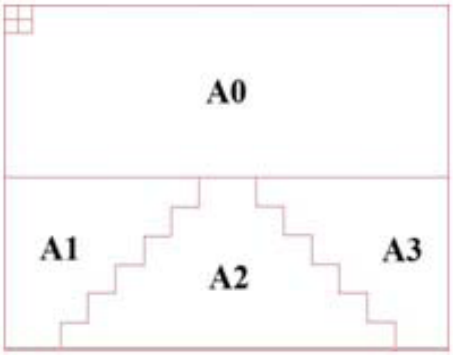


Figura 4.5: Areas de interesse na matriz de navegabilidade.

A Figura 4.6 contém o fluxograma do sistema gerado, resumindo seu funcionamento em duas etapas: módulo de percepção e módulo de controle. O controle estabelece para cada tipo de ação uma configuração pré-estabelecida de acionamento dos motores, o que permite por exemplo que o robô continue andando para a frente durante a distância certa após deixar de "enxergar" a curva.

Para realização do experimento, foi criada uma pista artificial em um ambiente interno que resultava em seis diferentes classificações (e portanto seis estados), descritos na Tabela 4.2. Alguns trechos interessantes do trajeto montado são exibidas na Figura 4.7.

Tabela 4.2: Estados do autômato e açoes relacionadas

Estado	*Ação*
Reta	Seguir em frente
Curva de 90 graus à esquerda	Armazenar "esquerda" na memória e seguir em frente
Curva de 90 graus à direita	Armazenar "direita" na memória e seguir em frente
Curva à esquerda	Virar suavemente para a esquerda
Curva à direita	Virar suavemente para a direita
Fim de pista	Se há alguma direção em memória, girar 90 graus para esta direção

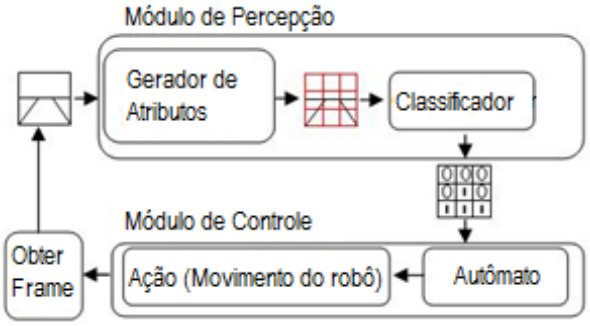


Figura 4.6: Fluxograma do sistema de visão desenvolvido.

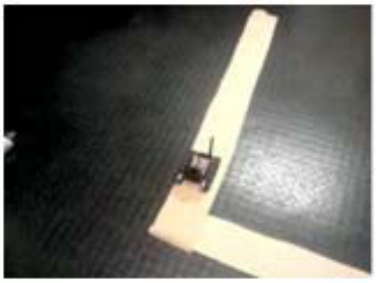
Este experimento apresentou bons resultados, com realização correta da tarefa de virar 90 graus nos pontos desejados mesmo sem informação suficiente e atualizada dos dados de entrada. A grande desvantagem de se usar o pós-processamento implementado para detecção de estados é que não há capacidade de generalização, ou seja, se aparecer alguma situação na pista não prevista pelo sistema a chance de erro na navegação é alta. Os resultados apresentados nesta seção foram publicados no IEEE LARS (Latin American Robotics Symposium) 2010 (Sales *et al.*, 2010).

4.2 Navegação Autônoma em ambientes urbanos utilizando *Template-Matching*

Foi realizado ainda mais um experimento utilizando Autômatos Finitos antes da implementação do aprendizado neural dos mesmos. Nesse experimento, realizado em co-autoria com Souza *et al.* (2011), foi utilizado o classificador desenvolvido no trabalho de Shinzato (2010) para navegação autônoma baseada em visão em um ambiente urbano, utilizando o veículo elétrico do LRM (Figura 4.8). Assim como no experimento descrito na seção 4.1, o estado atual era definido com base em um pós-processamento do mapa de navegabilidade gerado pelo classificador, desta vez utilizando um modelo baseado em *"template matching"*.

Este modelo também privilegia o aspecto geométrico representado no mapa de navegabilidade, porém ao invés de calcular as médias em áreas de interesse é feita uma comparação dos dados armazenados na matriz com um modelo pré-definido (template) do comportamento esperado da matriz para cada estado possível.

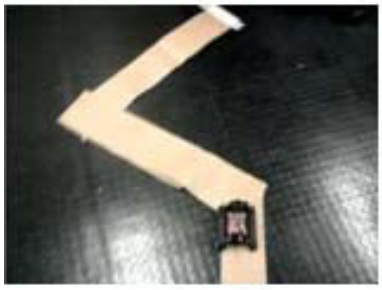
Sendo assim, algumas máscaras contendo a forma trapezoidal que imita a forma esperada da rua eram aplicadas sobre o mapa de navegabilidade, e a que apresentasse o melhor resultado era escolhida como representação o estado atual. A Figura 4.9 mostra um exemplo de processa-



(a) 90 graus à esquerda

(b) curvas à esquerda

(c) 90 graus à esquerda após curva suave



(d) Curva à direita com menos de 90 graus

Figura 4.7: Trechos da pista montada neste experimento.

mento. À esquerda está representado o mapa de navegabilidade em um dado instante, e à direita sua respectiva classificação, com a máscara que melhor se adaptou a ele sobreposta.

Como neste experimento o esterçamento da direção dependia da máscara aplicada, era indesejável que houvessem variações muito bruscas na classificação. Como no ambiente real a oscilação no resultado do classificador é constante, optou-se por utilizar um contador de repetições, atuando como um filtro para impedir uma variação grande de trocas de estados em um curto espaço de tempo.

No caso de sistemas baseados em visão, onde a taxa de coleta dos dados de entrada fica em torno de 30fps, uma classificação incorreta em um ou mais frames poderia representar trocas de contexto em intervalos menores que 1 segundo, inviabilizando a implementação do controle.

Sendo assim, a função deste filtro é de evitar de troca de contexto a cada detecção. É imposta portanto uma restrição de que ocorram pelo menos “N” classificações seguidas que sugerem uma troca de estado, para que efetivamente o sistema de controle de navegação troque de contexto e modifique a ação atual, indicando certeza sobre aquela transição.

Figura 4.8: Veículo elétrico executando trajeto com o sistema implementado.

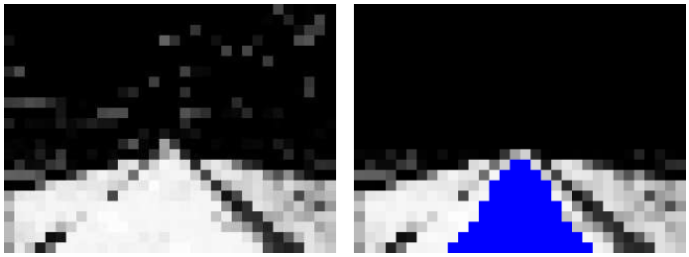


Figura 4.9: Mapa de navegabilidade em um dado instante e melhor máscara aplicada.

A implementação desse filtro é muito importante, integrando portanto todas as implementações seguintes, como forma de eliminar a incerteza associada às taxas finais de erro médio das RNAs treinadas.

Os resultados desse experimento como um todo foram publicados no ACM SAC (Symposium on Applied Computing) 2011 (Souza *et al.*, 2011).

4.3 Navegação Autônoma Topológica em Ambientes Internos utilizando RNA e Autômatos Finitos

O experimento descrito nesta seção apresenta uma evolução com relação ao modelo utilizado na Seção 4.1, uma vez que o autômato de controle não é mais especificado manualmente, e sim aprendido por uma RNA, sendo este portanto o primeiro trabalho a aplicar a metodologia completa apresentada nesta dissertação.

Para validação do sistema de navegação proposto, foi implementado o controle de navegação autônoma em uma tarefa de patrulhamento e vigilância em um ambiente interno: um andar de um prédio composto por diversas salas e corredores.

Foram realizados testes inicialmente em ambiente simulado, no player-stage para validar o funcionamento do método proposto, e em seguida utilizando um robô real de pequeno/médio porte, um Pioneer P3-AT equipado com um laser SICK e câmera termal (Figura 4.10).

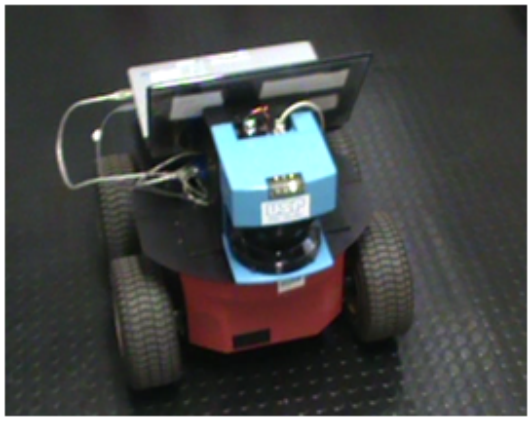


Figura 4.10: Robô Pioneer P3-AT com laser SICK usado no experimento.

O sistema sensorial (Figura 4.11) utiliza um laser SICK LMS291 como sensor principal, de tal forma que as entradas do classificador neural (RNA do tipo MLP) são os dados obtidos desse laser (feixes angulares), que possui abertura de 180 graus. A leitura de 20 feixes com intervalo de 10 graus entre si é a informação utilizada para classificação, definindo portanto uma camada de entrada com 20 neurônios. É utilizada ainda uma câmera termal Flir PathFindIR como sensor secundário, realizando uma detecção de agentes humanos no ambiente, porém à parte do sistema principal (a leitura dos dados desse sensor não é utilizada pelo classificador neural).

(a) Laser Sick LMS291 (b) Câmera termal Flir PathFindIR

Figura 4.11: Sensores utilizados no experimento.

A RNA possui como saída a representação binária que indica o estado atual detectado, seguindo o modelo proposto na metodologia. Foram adotados cinco estados para representação das situações possíveis nesse ambiente, conforme mostra a Tabela 4.3. Desta forma, a camada de saída da RNA é composta por 5 neurônios. Por fim, a quantidade de neurônios na camada escondida foi definida de forma empírica. Foram avaliadas as seguintes topologias: 20-20-5, 20-10-5, 20-20-20-5 e 20-10-10-5.

Tabela 4.3: Estados do autômato e açoes relacionadas para cada estado

Estado	*Ações Possíveis*
Reta	Seguir em frente com controle reativo
Curva à esquerda	Virar para a esquerda
Curva à direita	Virar para a direita
Interseção	Virar para esquerda; Virar para direita, Seguir em frente
Bifurcação	Virar para esquerda; Virar para direita

A geração da base de dados para treinamento foi realizada controlando-se o robô manualmente pelas diversas partes do ambiente e salvando um registro (arquivo de log) do estado dos sensores e as respectivas classificações de diversas posições do robô durante essa coleta, conforme o método descrito na Seção 3.2. Foram gerados no total 5429 pares entrada/saída desejada. O algoritmo de treinamento utilizado foi o *Resilient Propagation* (R-Prop), com seus parâmetros padrão. A convergência foi alcançada antes de 500 ciclos.

As 4 topologias de RNA avaliadas foram validadas através do método *Stratified 5-fold cross validation*. As médias de acerto das quatro RNAs são apresentadas na Figura 4.12. É possível observar que os resultados dos treinamentos foram muito parecidos. Sendo assim, dentre as que obtiveram médias acima de 99% foi escolhida a RNA 20-20-5 para integrar o sistema de controle do robô. As matrizes de confusão dos testes dessa RNA são apresentadas na Figura 4.13, onde é possível observar o alto índice de acerto do sistema em todas as classes.

PORCENTAGEM DE ACERTO DA RNA APÓS 500 CICLOS DE TREINAMENTO

	RNA	Particao 1		Particao 2		Particao 3		Particao 4		Particao 5		Media	
		Treino	**Teste**	Treino	**Teste**	Treino	**Teste**	Treino	**Teste**	Treino	**Teste**	Treino	**Teste**
1 Camada Oculta	20-10-5	99	**98**	99	**99**	98	**98**	99	**99**	98	**98**	98,6	**98,4**
	20-20-5	99	**99**	99	**99**	99	**99**	99	**99**	99	**99**	99	**99**
2 Camadas Ocultas	20-10-10-5	99	**99**	99	**99**	99	**99**	99	**99**	99	**99**	99	**99**
	20-20-20-5	99	**99**	99	**100**	99	**99**	100	**99**	100	**99**	99,4	**99,2**

Figura 4.12: Médias de acerto das 4 RNAs avaliadas.

De acordo com a metodologia proposta, o sistema desenvolvido utiliza um sistema de controle híbrido, que combina o controle deliberativo obtido do planejamento de trajetória com um controle reativo, que nesse caso avalia feixes das extremidades do laser para manter o robô alinhado, evitando colisão com as paredes e portas. A Figura 4.14 mostra o robô navegando por alguns pontos no cenário.

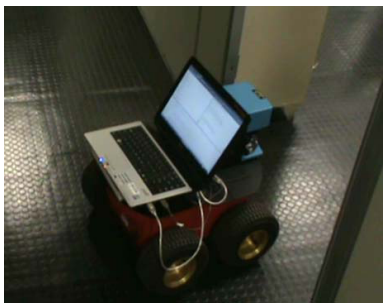
Teste 1	Teste 2	Teste 3	Teste 4	Teste 5
248 2 0 0 0 2 185 0 0 0 0 0 207 1 0 0 0 1 235 0 0 0 0 0 208	250 0 0 0 0 0 187 0 0 0 1 0 205 0 1 0 0 0 235 0 0 0 0 0 208	249 0 0 0 0 2 184 0 0 0 0 0 204 2 1 0 0 0 235 0 0 0 0 0 208	248 0 1 0 0 1 185 0 0 0 0 0 205 1 1 0 0 0 235 0 0 0 0 0 207	249 0 0 0 0 4 182 0 0 0 0 0 206 0 1 0 0 0 235 0 0 0 0 0 207
Acertos: 1083 Erros: 6	Acertos: 1085 Erros: 2	Acertos: 1080 Erros: 5	Acertos: 1080 Erros: 4	Acertos: 1079 Erros: 5

Figura 4.13: Matrizes de confusão nos testes da RNA 20-20-5.

(a) Curva à direita

(b) Interseção

(c) Reta (corredor)



(d) Curva à direita (porta)

Figura 4.14: Pontos estratégicos do cenário.

O uso da câmera termal Flir PathFindIR foi proposto como uma funcionalidade adicional ao sistema para detecção de agentes (humanos ou não). O frame capturado é processado utilizando OpenCV, e uma simples contagem de pixels permite avaliar se há um agente no ambiente ou

não, já que que a temperatura do agente (quente) é bem diferente da temperatura do ambiente (frio). Essa situação é ilustrada na Figura 4.15. Esta funcionalidade é bastante útil em tarefas de vigilância.

Figura 4.15: Exemplo de detecção de agente (humano).

Os resultados obtidos foram satisfatórios, tanto pela análise do desempenho do classificador como pela conclusão bem sucedida da tarefa de navegação, demonstrando a viabilidade da abordagem topológica adotada neste trabalho. Este experimento e os resultados obtidos geraram um artigo, publicado no I CBSEC (*Conferência Brasileira em Sistemas Embarcados Críticos*) 2011 (Sales *et al.*, 2011).

4.4 Sistema de Patrulhamento Autônomo Multi-Agente

Este experimento se destinou a implementar o sistema de navegação topológica obtido no trabalho anterior em um problema real, onde além da navegação os robôs tivessem que executar uma ou mais tarefas específicas. Foi implementado portanto um sistema multirrobótico para patrulhamento de um ambiente interno. Os experimentos foram conduzidos em um ambiente simulado (Figura 4.16) composto de corredores, interseções e curvas.

Os robôs representados na simulação bem como seus sistemas sensoriais são os mesmos utilizados no experimento anterior (Robô Pioneer P3-AT com Laser Sick de 180 graus). A única diferença está no sistema de percepção, com relação a quantidade de feixes angulares observados: são usados 180 feixes com intervalo de 1 grau entre si ao invés dos 20 feixes usados anteriomente.

O sistema foi composto por quatro robôs autônomos com comunicação entre si, porém sem comunicação com uma estação central, ou seja, o controle dos robôs é totalmente descentralizado. O comportamento de cada robô é determinado de acordo com a combinação do estado atual detectado com as mensagens enviadas pelos agentes aliados (que compõem o grupo robótico patrulheiro).

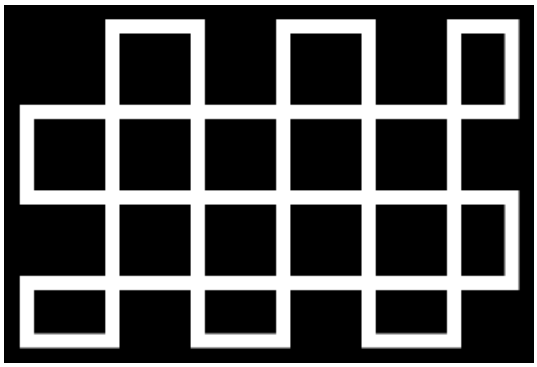


Figura 4.16: Ambiente simulado utilizado nos testes.

Dois comportamentos principais foram implementados: *"Patrulhando"* e *"Cercando inimigo"*. Na etapa de patrulhamento, os robôs podiam navegar pelo ambiente de forma aleatória ou seguindo um caminho cíclico pré-estabelecido (rota de patrulha). Se durante a patrulha um agente qualquer for detectado por um robô, uma mensagem é enviada em *broadcast* para o grupo com a posição do agente encontrado no mapa topológico. Cada robô responde com sua posição atual, e se nenhum dos robôs aliados é esse agente, o comportamento "cercar inimigo" é acionado. Caso contrário, mantém-se o comportamento de patrulha.

O planejamento de trajetórias nesse experimento ocorreu de duas formas: planejamento da rota a ser seguida por cada robô em sua patrulha (cíclica ou aleatória); planejamento do menor caminho entre cada robô e o inimigo detectado para neutralizá-lo. No comportamento de patrulha, a geração do caminho é realizada da mesma forma que no experimento anterior, definindo-se um ponto de destino para criação do autômato de navegação.

Já no comportamento de neutralização do inimigo o caminho é calculado com o algoritmo A* (Russell e Norvig, 2003). Cada robô calcula o menor caminho entre ele e o inimigo levando em consideração a posição dos aliados de forma que eles possam cercar esse inimigo. Para isto, o mapa do ambiente é convertido em um grid conforme mostra a Figura 4.17. Um algoritmo é responsavel por analisar esse caminho obtido com o A* e gerar a partir dele um autômato de navegação. Essa implementação mostrou que é possível gerar caminhos topológicos que priorizam a distância mínima entre os pontos origem/destino.

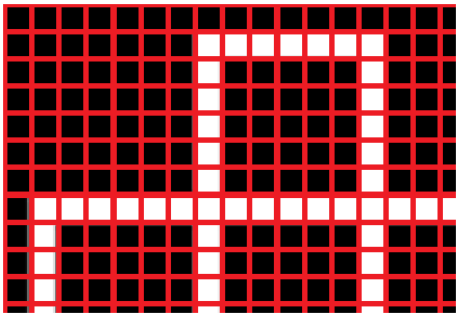


Figura 4.17: Representação de um trecho do mapa original na forma de grid de ocupação.

Com base nas características estruturais do ambiente, cinco estados foram implementados para este experimento: *Detecção de agente*, *Reta* (corredor), *Curva para a esquerda*, *Curva para a direita* e *Interseção*. Estas situações são apresentadas na Figura 4.18.

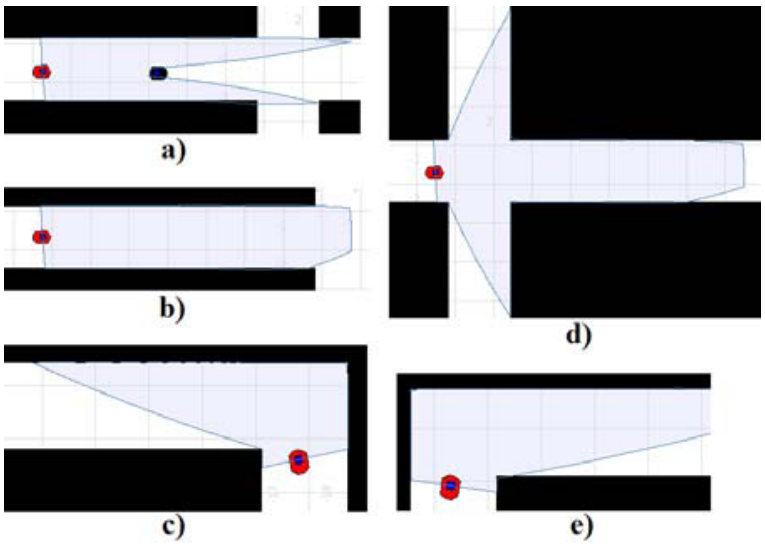


Figura 4.18: Estados do autômato. a)detecção de agente, b)reta, c)esquerda, d)interseção e e)direita.

O treinamento da RNA foi realizado da mesma forma que no experimento anterior, uma base de dados para treinamento e validação foi gerada a partir de um controle manual de um dos robôs no ambiente, e então aplicado o algoritmo de treinamento. Como neste experimento foram usados todos os feixes do laser (180 feixes com intervalo de 1 grau), a camada de entrada da rede possuía 180 neurônios. A quantidade de neurônios na camada escondida (90) foi baseada nos resultados conhecidos do experimento anterior. A tabela 4.4 apresenta os dados do treinamento da rede implementada. Como os resultados obtidos nas matrizes de confusão chegaram a 100%

de acerto, não houve necessidade de testar outras topologias, adotando a RNA 180-90-5 como classificador final.

Tabela 4.4: Configuração da RNA

Parâmetro	*Valor*
Topologia da RNA	180-90-3
Padrões na base de dados	2835 exemplos
Algoritmo de aprendizado	R-Prop
Método de treinamento e validação	Stratified 5-Fold Cross Validation
Erro médio de teste	inferior a 0.025(MSE)

Após o treinamento da RNA, na etapa de execução cada um dos quatro robôs foi colocado em uma posição oposta no mapa, e um agente inimigo (desconhecido) colocado intencionalmente em pontos específicos desse mapa. Ao ser encontrado por um dos robôs, esperava-se que os demais conseguissem navegar de forma autônoma até essa posição. Essa tarefa foi cumprida corretamente conforme ilustrado na Figura 4.19, onde o robô preto (inimigo) aparece cercado pelos demais.

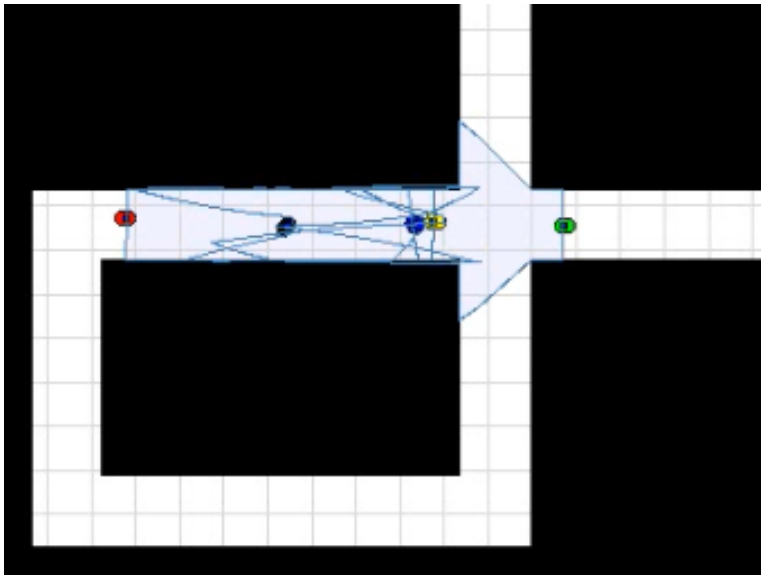


Figura 4.19: Screenshot do final de uma execução do algoritmo.

Os resultados deste experimento foram publicados no II CBSEC (*II Conferencia Brasileira em Sistema Embarcados Críticos*) 2012 (Sales *et al.*, 2012b).

4.5 Sistema de Navegação Autônoma Baseado em Visão 3D utilizando sensor Kinect

Os experimentos anteriores realizados com sistema sensorial baseado em laser tiveram um importante papel nesta pesquisa, validando a metodologia proposta tanto nas simulações como no ambiente real. Uma vez que o sistema de navegação topológica autônoma tinha se tornado uma proposta madura, foram projetados experimentos para avaliação da aplicação dessa metodologia com outros sistemas sensoriais, ambientes e estados, como é o caso do experimento descrito nesta seção.

Neste experimento, foi implementado um sistema de navegação autônoma para robôs de serviço em ambientes internos. Procurou-se adotar um sistema sensorial de baixo-custo (em oposição ao modelo baseado em laser), mas que possuísse bom volume de dados a cada detecção e diversidade no tipo de informação extraída desses dados.

Foi adotado para este fim o sensor Kinect (Figura 4.20), um sensor 3D de baixo custo desenvolvido pela Microsoft para o vídeo-game XBOX 360 que permite que o jogador use seu próprio corpo como controle nos jogos. Ele consiste em uma câmera RGB associada a um emissor e receptor de infravermelho, permitindo uma estimação da profundidade dos elementos capturados no ambiente.

Figura 4.20: Sensor kinect.

Assim que percebido seu potencial como sensor diversas pesquisas independentes surgiram com o objetivo de explorar as vantagens deste equipamento em aplicações que variam desde a área de saúde até a robótica. O grande diferencial deste sensor é a possibilidade de construção de mapas de profundidade em 3D, que permitem uma estimação bastante precisa da distância dos diferentes tipos de obstáculos detectados à frente do robô, ao mesmo tempo em que captura imagens RGB convencionais. Seu uso porém é voltado apenas à aplicações em ambientes internos, devido às limitações do hardware.

As leituras do kinect são armazenadas em mapas de profundidade (matrizes) de dimensões 640x480. Cada valor armazenado representa a distância entre o sensor e o objeto representado

no pixel correspondente. Sendo assim, a utilização deste sensor para detecção de profundidades fornece 307200 pontos por leitura. A correspondência direta entre dados sensoriais e entradas da RNA para a tarefa de navegação neste caso inviabilizaria o treinamento da rede.

Portanto, tornou-se necessária uma etapa de pré-processamento dos dados sensoriais, na qual é selecionada uma área de interesse na captura do Kinect cujos pontos forneçam informação suficiente para classificação e consequente detecção dos estados. Isso é feito aproveitando uma propriedade interessante do Kinect: cada linha da matriz de distâncias pode ser comparada ao corte planar que um sensor laser convencional representa. Em outras palavras, uma única captura do Kinect contém informação equivalente à detecção obtida com 480 lasers. Como os dados de entrada estão sujeitos à ruído, é interessante usar então a combinação de algumas linhas dessa matriz como base para o reconhecimento dos estados.

São selecionadas como área de interesse um conjunto de 80 linhas da parte central da detecção do Kinect, como mostra a Figura 4.21. É calculada então a média entre os valores de cada coluna desse conjunto, resultando em um vetor que representa toda essa área de interesse em apenas uma linha. A saída do sistema sensorial portanto é dada por esse vetor de 640 elementos.

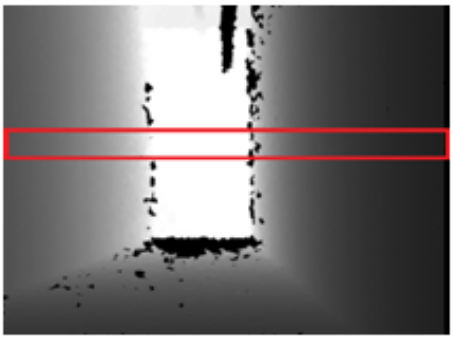


Figura 4.21: Zona de interesse na detecção do Kinect.

Cada neurônio da camada de entrada recebe um valor desse vetor, porém 8 valores não são representações de profundidade, e são descartados. A camada de entrada da RNA é composta portanto por 632 neurônios.

O ambiente utilizado nos testes foi representado na forma de 8 estados, ilustrados na Figura 4.22. Os estados definidos foram: “corredor livre”, “curva esquerda”, “curva direita”, “curva direita e esquerda”, “curva esquerda e corredor”, “curva direita e corredor”, “fechado” (fim de corredor) e “cruzamento”. O robô utilizado no experimento foi o Pioneer P3-AT.

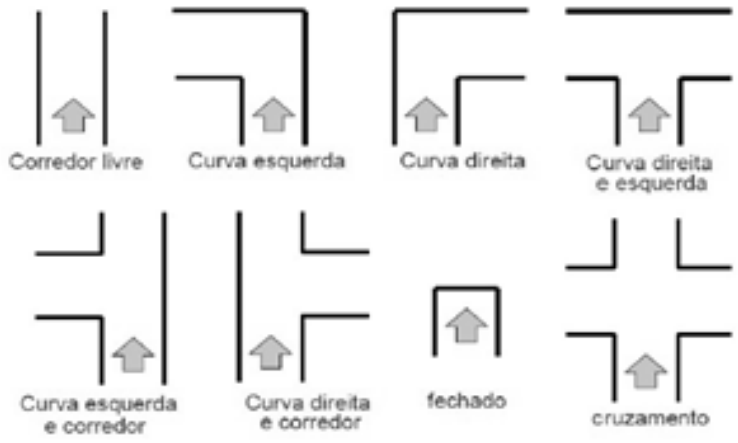


Figura 4.22: Representação dos 8 estados implementados.

Assim como nos experimentos anteriores, a topologia da RNA foi definida de forma empírica. Foram testadas as seguintes topologias: 632-16-8, 632-80-8, 632-316-8, 632-16-16-8 e 632-80-80-8. Todo o processo de treinamento e validação das RNAs seguiu os padrões estabelecidos na metodologia, sendo gerada uma base de dados com 1421 pares entrada/saída (aproximadamente 180 exemplos por classe). O algoritmo de treinamento mais uma vez foi o R-Prop, com parâmetros padrão e critério de parada de 1000 ciclos.

Conforme apresentado na Figura 4.23, os resultados foram ligeiramente parecidos, com desempenho maior obtido pela RNA 632-316-8. A principal diferença perceptível ocorre na análise dos tempos de treinamento, devido ao grande número de neurônios na rede. Enquanto a rede 632-16-8 gastou apenas 20 minutos para cada treinamento na máquina utilizada, a rede 632-316-8 gastou 8 horas aproximadamente. Isto não chega a ser um problema, já que a etapa de treinamento é executada *offline*.

PORCENTAGEM DE ACERTO DA RNA APÓS 1000 CICLOS DE TREINAMENTO

	RNA	Particao 1		Particao 2		Particao 3		Particao 4		Particao 5		Media	
		Treino	**Teste**	Treino	**Teste**	Treino	**Teste**	Treino	**Teste**	Treino	**Teste**	Treino	**Teste**
1 Camada Oculta	632-16-8	89	**88**	95	**87**	92	**81**	92	**86**	91	**85**	91,8	**85,4**
	632-80-8	97	**89**	96	**91**	97	**90**	97	**92**	96	**89**	96,6	**90,2**
	632-316-8	99	**92**	99	**93**	98	**92**	98	**91**	98	**93**	98,4	**92,2**
2 Camadas Ocultas	632-16-16-8	90	**80**	93	**85**	94	**85**	90	**86**	94	**86**	92,2	**84,4**
	632-80-80-8	98	**92**	99	**91**	98	**91**	99	**92**	99	**94**	98,6	**92**

Figura 4.23: Médias de acerto das RNAs avaliadas.

As matrizes de confusão resultantes dos testes com a RNA 632-316-8 são apresentadas na Figura 4.24. É possível observar que o erro por classe é bem pequeno, porém como a taxa total de acerto está próxima a 92% é desejável a inclusão do filtro implementado no experimento da Seção 4.2 para garantir uma navegação segura.

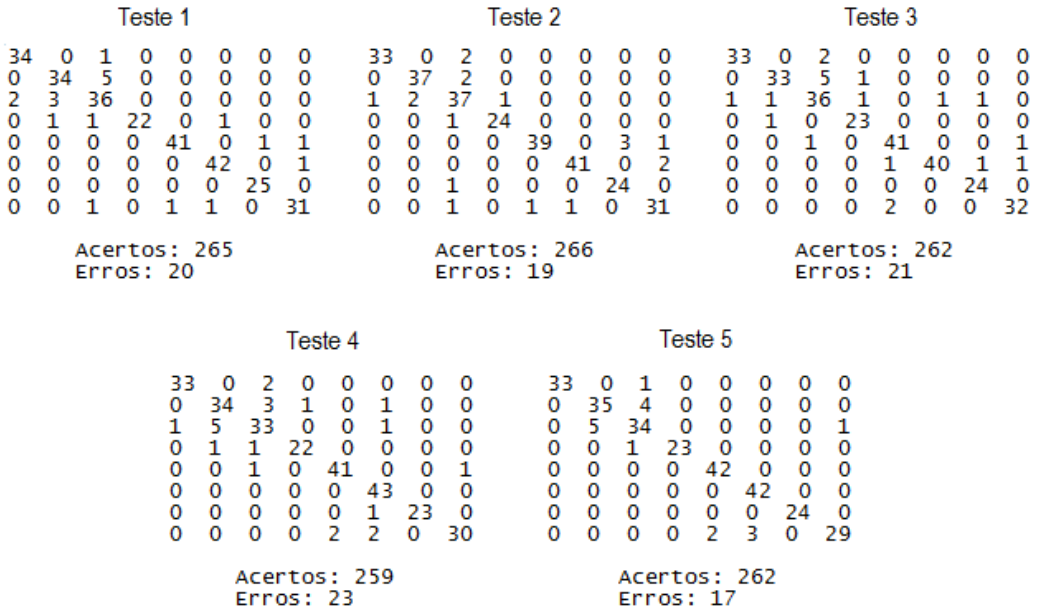


Figura 4.24: Matrizes de confusão dos testes com a RNA 632-316-8.

O classificador obteve bom nível de acerto, permitindo uma navegação sem trocas indesejadas de estados após implementação do filtro. O sensor kinect se mostrou adequado para a classificação de características de ambientes internos, viabilizando a construção de robôs com sistema de locomoção baseados em visão 3d de baixo custo. Este trabalho originou um artigo publicado no EANN (Engineering Applications of Neural Network) 2012 (Sales *et al.*, 2012a).

4.6 Sistema de Detecção de Estados Baseado em Visão em Ambientes Urbanos

Uma vez que a metodologia proposta foi avaliada em aplicações para ambientes internos, foram realizados testes para aplicação da mesma em ambientes urbanos. O experimento descrito nesta seção consiste na proposta de um sistema de detecção de estados baseado em visão monocular, para análise da viabilidade da implementação deste em um sistema de controle de navegação autônoma.

Optou-se por adotar um modelo sensorial que apresentasse características adequadas para percepção dos elementos do ambiente urbano, levando em consideração as suas principais limitações. Foi adotado então um modelo sensorial baseado em visão monocular para aquisição dos dados de entrada, devido a características como baixo custo, peso e consumo de energia.

O sistema desenvolvido neste experimento divide-se em duas etapas principais: detecção das áreas navegáveis do ambiente e reconhecimento de estados através do modelo topológico desenvolvido. A primeira etapa consiste na geração de mapas de navegabilidade através da classificação do frame capturado pela câmera, e a segunda consiste no reconhecimento de padrões nesses mapas de navegabilidade para detecção de estados. A Figura 4.25 apresenta uma visão geral desse sistema.

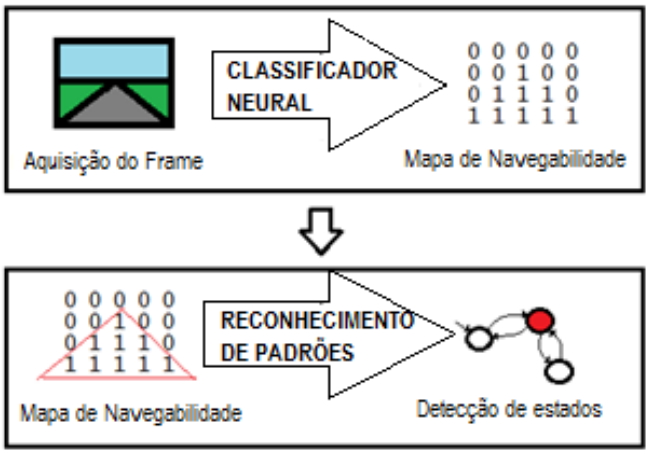


Figura 4.25: Visão geral do sistema desenvolvido neste experimento.

Inicialmente foi preciso processar o frame capturado para determinar quais eram as áreas navegáveis e não-navegáveis. Para isto, foi utilizado o classificador neural desenvolvido por Shinzato (2010), uma versão aprimorada do classificador utilizado no experimento da Seção 4.1. Este classificador consiste em um comitê de 5 RNAs, onde cada RNA avalia atributos diferentes da imagem. A Figura 4.26 evidencia a eficiência desse classificador na detecção da pista em diversas condições diferentes.

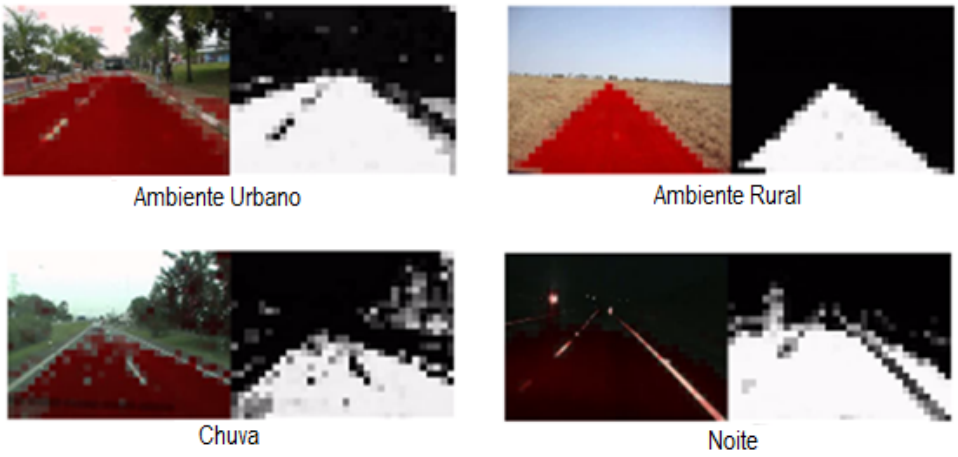


Figura 4.26: Exemplos de classificação em diversos cenários.

Assim como no experimento anterior, a imagem original de 320×240 pixels foi divida em blocos de 10×10 pixels para otimizar o processamento, gerando como saída uma matriz de navegabilidade com dimensões 32×24. Este mapa de navegabilidade gerado é a entrada considerada pela RNA, de tal forma que essa etapa inicial pode ser vista como parte do próprio sistema de percepção.

Apesar do ganho de desempenho obtido com a segmentação da imagem em blocos, a quantidade total de elementos na matriz de navegabilidade (768) ainda é muito grande para ser utilizada diretamente como entrada da RNA na etapa de navegação topológica. Como os primeiros 384 elementos da matriz estão relacionados geralmente a elementos acima da linha do horizonte na imagem, estes foram desconsiderados pois eles dificilmente seriam relevantes na classificação dos estados. Foram usados portanto os outros 384 elementos como entradas da RNA.

Quatro estados foram definidos a partir da observação do formato da pista identificada pelos mapas de navegabilidade, e um estado pela detecção de elementos dinâmicos na cena (como outros veículos). São eles “Reta”, “curva à direita”, “curva à esquerda”, “bifurcação” e “obstáculo à frente”. Exemplos de frames associados a cada estado são apresentados a seguir, na Figura 4.27.

(a) Reta (b) Direita (c) Esquerda

(d) Bifurcação (e) Obstáculo

Figura 4.27: Estados definidos para o ambiente urbano.

Foram treinadas e avaliadas as seguintes RNAs: 384-96-5, 384-192-5 e 384-384-5. A base de dados para treinamento e teste foi formada através da coleta de vídeos de aproximadamente 1 minuto e 25 segundos a 30fps de cada estado, o que resultou em aproximadamente 2500 exemplos por classe. Foram gerados 11186 exemplos no total para formação das bases de dados. O método de treinamento e validação utilizado foi mais uma vez o *Stratified 5-fold cross validation*, e o algoritmo de treinamento R-Prop, com critério de parada de 500 ciclos.

As três redes avaliadas apresentaram bons resultados, sendo a rede 384-384-5 a que obteve melhores resultados, como pode ser observado na Figura 4.28. As matrizes de confusão relacionadas aos testes com essa RNA são apresentadas na Figura 4.29, onde pode-se analisar mais detalhadamente os bons índices de acerto obtidos nos treinamentos.

PORCENTAGEM DE ACERTO DA RNA APÓS 500 CICLOS DE TREINAMENTO

ANN	Partition 1		Partition 2		Partition 3		Partition 4		Partition 5		Mean	
	Train	**Test**	Train	**Test**	Train	**Test**	Train	**Test**	Train	**Test**	Train	**Test**
384-96-5	99	**98**	99	**98**	99	**99**	99	**98**	99	**99**	99	**98,4**
384-192-5	99	**99**	99	**98**	99	**99**	99	**98**	99	**99**	99	**98,6**
384-384-5	99	**99**	99	**99**	99	**99**	99	**99**	99	**99**	99	**99**

Figura 4.28: Índices de acerto das RNA testadas.

Teste 1

```
492   0   0   0   1
0   407   0   0   0
0   0   404   2   1
0   0   2   480   1
2   0   0   1   447

Acertos: 2230
Erros: 10
```

Teste 2

```
489   0   0   0   4
0   406   0   0   1
0   0   403   3   0
0   0   0   482   1
1   0   0   1   448

Acertos: 2228
Erros: 11
```

Teste 3

```
488   0   0   1   3
0   406   0   0   0
0   0   404   2   0
0   1   1   480   1
2   1   0   2   445

Acertos: 2223
Erros: 14
```

Teste 4

```
491   0   0   0   1
0   406   0   0   0
0   0   401   5   0
0   1   2   476   3
2   0   0   0   447

Acertos: 2221
Erros: 14
```

Teste 5

```
492   0   0   0   0
0   406   0   0   0
0   0   405   1   0
0   1   3   474   4
1   0   0   3   445

Acertos: 2222
Erros: 13
```

Figura 4.29: Matrizes de confusão dos testes com a RNA 384-384-5.

Os resultados obtidos foram promissores, sugerindo a viabilidade do uso da abordagem topológica no controle de veículos autônomos. O sistema sensorial baseado em visão se mostrou adequado para aplicações urbanas, com bom índice de acerto. Estes resultados foram descritos em um artigo, aceito para publicação no Latin American Robotics Symposium (LARS) 2012 (Sales e Osorio, 2012).

4.7 Sistema de Navegação Topológica baseada em Visão em Ambientes Urbanos

Uma vez testado o classificador de estados baseado em visão apresentado na seção anterior, foi realizado um experimento para avaliar a implementação dessa abordagem no controle de navegação de um veículo autônomo em vias de trânsito reais, complementando a gama de aplicações que faziam parte dos objetivos desta dissertação.

Desta forma, foi utilizado um veículo automatizado (Fiat Palio Adventure) equipado com uma câmera estéreo (PointGray Bumblebee2), em um ambiente urbano composto de retas, curvas e rotatórias. Para este experimento, optou-se por não implementar o estado "obstáculo", avaliando somente a tarefa de navegação no ambiente sem elementos dinâmicos. A Figura 4.30 apresenta frames reais coletados para cada classe.

(a) Reta (b) Direita (c) Esquerda

(d) Bifurcação

Figura 4.30: Estados definidos para implementação do sistema.

A câmera foi posicionada em um suporte acima do veículo, como mostra a Figura 4.31, com um ângulo de captura similar ao utilizado no experimento anterior. Esta configuração foi adotada para que a área à frente do veículo fosse corretamente observada, demandando uma nova ação de controle a cada novo *frame* detectado, permitindo reação imediata aos dados observados. Como o sistema de detecção de estados não foi desenvolvido para lidar com informação tridimensional, somente a imagem da esquerda da câmera foi utilizada para processamento.

O sistema de controle foi implementado no *Robot Operating System* (ROS), um *framework* que provê bibliotecas e ferramentas para o desenvolvimento de aplicações robóticas. Os módulos do sistema (como controle de navegação, detecção de estados e geração de mapas de navegabilidade) foram implementados como nós do ROS com comunicação entre si através de troca de mensagens específicas. A Figura 4.32 apresenta os módulos implementados e a interação entre eles a cada passo da execução.

Figura 4.31: Posição da câmera bumblebee2 nos testes realizados.

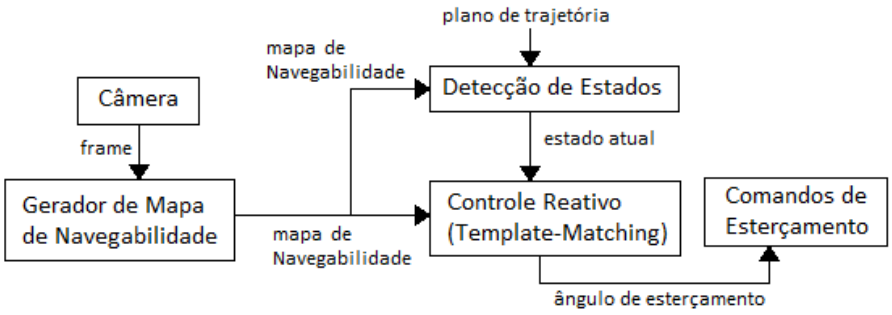


Figura 4.32: Visão geral do sistema implementado.

Foi feita uma nova coleta de dados, seguindo exatamente o modelo proposto no experimento anterior, com 2040 frames por classe, resultando em 8165 pares entrada/saída na base de dados. Desses frames, 15 foram utilizados para fazer o treinamento do gerador de mapas de navegabilidade. Uma vez treinado este primeiro classificador, este foi utilizado para processar cada frame coletado e produzir sua respectiva matriz de navegabilidade, classificada em um dos quatro estados possíveis para composição das bases de treinamento e teste. O ambiente completo onde os dados foram coletados é representado a seguir, na Figura 4.33.

Foi utilizada a mesma topologia de rede do experimento anterior, com 384 neurônios na camada de entrada, 384 neurônios na camada escondida e 4 neurônios na camada de saída. Os parâmetros e algoritmo de treinamento também foram os mesmos. Assim, obteve-se uma acurácia média de 99,14% nos 5 testes realizados. As matrizes de confusão para estes 5 testes podem ser observadas na Figura 4.34.

Figura 4.33: Ambiente percorrido para coleta de dados.

417	0	1	1	416	0	2	1	417	1	0	1	417	0	1	1	417	0	1	1
1	380	0	0	0	380	0	1	0	378	1	2	0	380	0	1	0	380	0	0
0	0	374	7	0	0	373	8	0	0	375	6	1	0	376	4	0	0	375	5
2	0	6	445	1	1	2	448	0	1	2	449	1	0	1	450	0	0	1	451
Teste 1				Teste 2				Teste 3				Teste 4				Teste 5			

Figura 4.34: Matrizes de confusão nos 5 testes da RNA 384-384-4.

Para o controle da movimentação do veículo, foi implementado um controle reativo baseado em *template-matching*, similar ao implementado em (Souza *et al.*, 2011). Cinco máscaras foram criadas para representar situações específicas, cada uma associada a um comando de esterçamento diferente. Para isto, o módulo de controle reativo utilizava como entrada a mesma matriz de navegabilidade utilizada pelo módulo de detecção de estados, comparando os valores da matriz com o das 5 máscaras criadas (Figura 4.35). A máscara que tivesse maior correspondência com a matriz de navegabilidade atual definia o esterçamento do volante naquele instante. O estado atual detectado também foi utilizado como entrada para este módulo. Era adicionado um viés aos templates mais adequados a cada estado atual, diminuindo as chances de uma mudança brusca de direção em momentos inadequados.

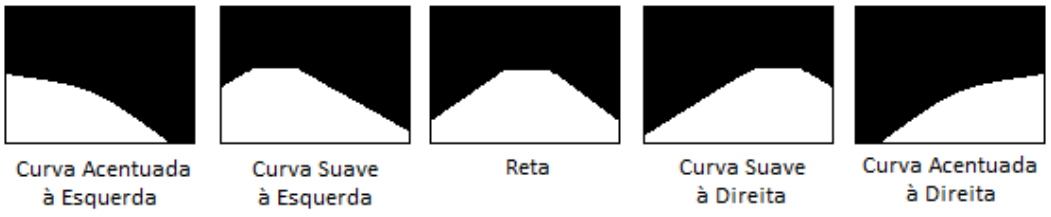


Figura 4.35: Representação das máscaras criadas para o módulo de controle reativo.

Os testes de navegação autônoma foram conduzidos em algumas partes específicas do ambiente, como o trecho representado na Figura 4.36. O mapa topológico neste caso é direcionado, levando em consideração o sentido de circulação permitido nessa via. O sistema foi testado em trajetórias pré-estabelecidas que incluiam o contorno da rotatória e saída da mesma, entrando em uma nova reta. A Figura 4.37 mostra o veículo percorrendo um trecho de uma rotatória de forma autônoma.

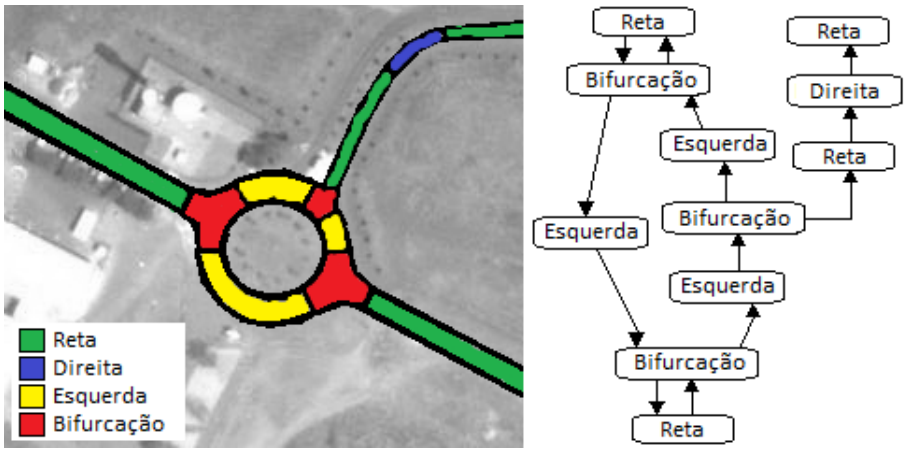


Figura 4.36: Trecho do ambiente utilizado nos testes de navegação autônoma.

Figura 4.37: Veículo autônomo percorrendo um trecho da rotatória.

O veículo percorreu corretamente os trechos determinados em velocidade constante, o que demonstra que a abordagem utilizada pode ser um método promissor para o controle de nave-

gação de veículos autônomos. Os resultados obtidos neste experimento foram descritos num artigo publicado no *II Workshop on Visual Control of Mobile Robots (ViCoMoR)*, evento integrante do *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS) 2012 (Sales *et al.*, 2012c).

4.8 Considerações Finais

A execução dos experimentos descritos neste capítulo possibilitou a avaliação parcial dos resultados obtidos e a implementação incremental de aprimoramentos no sistema proposto.

Os resultados apresentados mostraram que o método proposto é adequado para a Navegação Robótica Autônoma, nas diferentes aplicações escolhidas. Foram implementados diferentes tipos de sistemas sensoriais em robôs simulados e reais (baseados em Laser, kinect, câmera de vídeo e câmera termal), e apesar das características distintas de cada uma das abordagens, o uso associado de autômatos finitos e de Redes Neurais obteve sempre um bom desempenho. Os classificadores neurais implementados apresentaram um bom nível de acerto, com pequena taxa de erro por classe, o que viabilizou a implementação dos sistemas de controle, atingindo os resultados esperados.

Nos diferentes experimentos os algoritmos foram executados em tempo-real (respeitando as taxas de coleta de dados dos sistemas sensoriais), o que sugere a viabilidade do sistema e eficiência das técnicas adotadas.

CAPÍTULO

5

Conclusão

As pesquisas realizadas durante este projeto de mestrado possibilitaram o desenvolvimento de uma metodologia para implementação de Sistemas de Navegação Autônoma baseados no aprendizado de Autômatos Finitos por Redes Neurais Artificiais.

Esta metodologia sugere basicamente uma navegação Topológica Híbrida, que leva em consideração aspectos estruturais do ambiente para o reconhecimento de estados que permitem ao robô conhecer o contexto em que se encontra e a sequência de etapas a serem percorridas, realizando assim tarefas de navegação tanto em ambientes internos como externos.

A abordagem utilizada dispensa o uso de mapas métricos e de algoritmos de localização e de mapeamento muito precisos, complexos e com alto custo computacional, possibilitando a implementação do método mesmo em sistemas com recursos computacionais limitados.

Com os diversos experimentos realizados, foi ressaltada a capacidade de adaptação do método a diversos tipos de sensores, aplicações e ambientes, servindo de referência para o desenvolvimento de trabalhos futuros, tanto em implementações baseadas em sistemas sensoriais de baixo-custo como em implementações que privilegiam volume e riqueza de informação sensorial.

Os resultados obtidos são promissores, tanto do ponto de vista quantitativo (análise do nível de acerto das RNAs nas implementações desenvolvidas) como do ponto de vista qualitativo (avaliação da tarefa de navegação nos experimentos realizados). Os artigos resultantes deste

trabalho tiveram boa aceitação, sendo publicados em diversas conferências nacionais e internacionais.

Além disso, este trabalho contribuiu com os projetos em execução no Laboratório de Robótica Móvel do ICMC-USP em parceria com o INCT-SEC, uma vez que a metodologia desenvolvida pode ser empregada no controle de navegação tanto de robôs táticos para ambientes internos como de veículos terrestres autônomos.

Espera-se dar continuidade a este trabalho avaliando novas formas de aquisição de dados e pré-processamento das entradas, possibilitando assim a implementação de um sistema de navegação autônoma em ambientes urbanos mais complexos, onde a quantidade de elementos dinâmicos e a complexidade da informação a ser tratada são grandes.

Referências Bibliográficas

BARANAUSKAS, J. A.; MONARD, M. C. Combining symbolic classifiers from multiple inducers. *Knowledge-Based Systems*, v. 16, n. 3, p. 129 – 136, 2003.

BECK, J.; BROGGI, A.; SUNDARESWAN, V.; DERRICK, B. Team oshkosh - darpa urban challenge. disponivel em `http://www.darpa.mil/grandchallenge/Tech_Papers/Team_Oshkosh.pdf`, 2007.

BISHOP, R. Intelligent vehicle applications worldwide. IEEE Intelligent Systems - Intelligent Systems and their Applications, 2000a, p. 78–81.

BISHOP, R. A survey of intelligent vehicle applications worldwide. IEEE intelligent Vehicles Symposium, 2000b, p. 25–30.

BRAGA, A. P.; DE CARVALHO, A. P. L.; LUDERMIR, T. B. *Redes neurais artificiais - teoria e aplicacoes*. 2 ed. Rio de Janeiro: LTC, 2000.

BUEHLER, M.; IAGNEMMA, K.; SINGH, S. *The 2005 darpa grand challenge: The great robot race*. Springer, 2007.

CARBONELL, J. G.; MICHALSKI, R. S.; MITCHELL, T. M. An overview of machine learning. In: MICHALSKI, R. S.; CARBONELL, J. G.; MITCHELL, T. M., eds. *Machine Learning: An Artificial Intelligence Approach*, Berlin, Heidelberg: Springer, p. 3–23, 1984.

CLEERMANS, A.; SERVAN-SCHREIBER, D.; MCCLELLAND, J. Finite state automata and simple recurrent networks. In: *Neural Computation*, 1989, p. 372–381.

COLLETT, T. H.; MACDONALD, B. A.; GERKEY, B. P. Player 2.0: Toward a practical robot programming framework. In: *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005)*, 2005.

DUDEK, G.; JENKIN, M. *Computational principles of mobile robotics*. Cambridge, MA, USA: Cambridge University Press, 2000.

FOEDISCH, M. Adaptive real-time road detection using neural networks. In: *in Proc. 7th Int. Conf. on Intelligent Transportation Systems, Washington D.C*, 2004, p. 167 – 172.

FRASCONI, P.; GORI, M.; MAGGINI, M.; SODA, G. *Representation of finite state automata in Recurrent Radial Basis Function networks*, p. 5 – 32, 1996.

GILES, L.; HORNE, B.; LIN, T. *Learning a class of large finite state machines with a recurrent neural network*, p. 1359 – 1365, 1995.

GOEBL, M.; ALTHOFF, M.; BUSS, M.; FARBER, G.; HECKER, F.; HEISSING, B.; KRAUS, S.; NAGEL, R.; LEON, F.; RATTEI, F.; RUSS, M.; SCHWEITZER, M.; THUY, M.; WANG, C.; WUENSCHE, H. Design and capabilities of the munich cognitive automobile. IEEE intelligent Vehicles Symposium, 2008, p. 1101–1107.

GRASSI, V.; OKAMOTO, J. *Arquiteturas de controle: Tipos e conceitos. in: "robotica movel"*. Rio de Janeiro: Romero, Roseli A. F.; Osorio, F.S.; Prestes, Edson; Wolf, Denis F. (Editores). LTC Editora, 2012.

HAYKIN, S. *Neural networks: A comprehensive foundation (2nd edition)*. 2 ed. Prentice Hall, 1998.

HEINEN, M.; OSORIO, F.; HEINEN, F.; KELBER, C. *SEVA3D: Autonomous Vehicles Parking Simulator in a three-dimensional environment*, 2007.

HOPCROFT, J.; ULLMAN, J. *Introduction to automata theory, languages and computation*. Addison-Wesley, 1979.

LEE, K.; CHO, N.; CHUNG, W. K.; DOH, N. L. Topological navigation of mobile robot in corridor environment using sonar sensor. In: *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, p. 1–6.

MARINO, A.; PARKER, L.; ANTONELLI, G.; CACCAVALE, F. Behavioral control for multi-robot perimeter patrol: A finite state automata approach. IEEE International Conference on Robotics and Automation (ICRA 09), 2009.

MEDEIROS, A. *A survey of control architectures for autonomous mobile robots*, 1998.

MITCHELL, T. *Machine learning (mcgraw-hill international edit)*. 1st ed. McGraw-Hill Education (ISE Editions), 1997.

OMLIN, C.; GILES, L. *Constructing Deterministic Finite-State Automata in Recurrent Neural Networks*, p. 937 – 972, 1996.

OSORIO, F.; HEINEN, F.; FORTES, L. Autonomous vehicle parking using a finite state automata learned by j-cc artificial neural nets. In: *VI Brazilian Symposium on Neural Networks (SBRN)*, Porto de Galinhas, PE, Brasil, 2002.

POMERLEAU, D. Neural network vision for robot driving. In: ARBIB, M., ed. *The Handbook of Brain Theory and Neural Networks*, p. 161–181, 1996.

RIEDMILLER, M.; BRAUN, H. *Rprop – description and implementation details*. Relatório Técnico, Universitat Karlsruhe, 1994.
Disponível em `citeseer.ist.psu.edu/riedmiller94rprop.html`

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning internal representations by error propagation* Cambridge, MA, USA: MIT Press, p. 318–362, 1986.
Disponível em `http://portal.acm.org/citation.cfm?id=104279.104293`

RUSSELL, S. J.; NORVIG, P. *Artificial intelligence: A modern approach*. Pearson Education, 2003.
Disponível em `http://portal.acm.org/citation.cfm?id=773294`

SAHOTA, M. Reactive deliberation: An architecture for real-time intelligent control in dynamic environments. In: *AAAI-94*, 1994, p. 1303–1308.

SALES, D.; CORREA, D.; OSORIO, F. S.; WOLF, D. F. 3d vision-based autonomous navigation system using ann and kinect sensor. In: *13th Engineering Applications of Neural Network Conference (EANN2012)*, London, UK, 2012a.

SALES, D.; FEITOSA, D.; OSORIO, F. S.; WOLF, D. F. Multi-agent autonomous patrolling system using ann and fsm control. In: *II CBSEC: Conferencia Brasileira em Sistemas Embarcados Criticos*, Campinas, Brasil, 2012b.

SALES, D.; OSORIO, F.; WOLF, D. F. Topological autonomous navigation for mobile robots in indoor environments using ann and fsm. In: *I CBSEC: Conferencia Brasileira em Sistemas Embarcados Criticos*, Sao Carlos, Brasil, 2011.

SALES, D.; SHINZATO, P. Y.; ; PESSIN, G.; OSORIO, F.; WOLF, D. F. Autonomous vision-based navigation using fsm control. In: *Latin American Robotics Symposium*, Sao Bernardo do Campo, Brasil: IEEE, 2010.

SALES, D. O.; FERNANDES, L. C.; OSORIO, F. S.; WOLF, D. F. Fsm-based visual navigation for autonomous vehicles. In: *Workshop on Visual Control of Mobile Robots - IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Algarve, Portugall, 2012c.

SALES, D. O.; OSORIO, F. S. Vision-based autonomous topological navigation in outdoor environments. In: *Latin American Robotics Symposium*, Fortaleza, Brasil: IEEE, 2012.

SHINZATO, P. *Sistema de identificacao de superficies navegaveis baseado em visao computacional e redes neurais artificiais*. Dissertação de Mestrado, ICMC - Universidade de Sao Paulo, 2010.

SIEGELMANN, H.; GILES, L. *The complexity of language recognition by neural networks*, p. 327–345, 1997.

SNNS `http://www.ra.cs.unituebingen.de/SNNS/`, visitado em Agosto, 2010.

SOUZA, J.; SALES, D.; SHINZATO, P. Y.; OSORIO, F.; WOLF, D. F. Template-based autonomous navigation in urban environments. In: *26th SAC: ACM Symposium on Applied Computing*, TaiChung, Taiwan: ACM, 2011, p. 1–6.

THRUN, S.; MONTEMERLO, M.; DAHLKAMP, H.; STAVENS, D.; ARON, A.; DIEBEL, J.; FONG, P.; GALE, J.; HALPENNY, M.; HOFFMANN, G.; LAU, K.; OAKLEY, C.; PALATUCCI, M.; PRATT, V.; STANG, P.; STROHBAND, S.; DUPONT, C.; JENDROSSEK, L.-E.; KOELEN, C.; MARKEY, C.; RUMMEL, C.; VAN NIEKERK, J.; JENSEN, E.; ALESSANDRINI, P.; BRADSKI, G.; DAVIES, B.; ETTINGER, S.; KAEHLER, A.; NEFIAN, A.; MAHONEY, P. Winning the darpa grand challenge. *Journal of Field Robotics*, 2006.

WANG, Y.; CHEN, D.; SHI, C. Autonomous navigation based on a novel topological map. In: *2009 Asia-Pacific Conference on Information Processing*, 2009, p. 1–6.

WOLF, D.; OSORIO, F.; SIMOES, E.; TRINDADE, O. Robotica inteligente da simulacao as aplicacoes no mundo real. JAI - Jornada de Atualizacao em Informatica da SBC, 2009.

APÊNDICE

A

Publicações Obtidas Como Resultado Desse Trabalho

1. SALES, D. O.; SHINZATO, P. Y.; PESSIN, G.; OSÓRIO, F. S.; WOLF, D. F. Vision-Based Autonomous Navigation System Using ANN and FSM Control. In: *Proceedings of IEEE Latin American Robotics Symposium*, São Bernardo do Campo, Brasil, 2010.

2. SOUZA, J.; SALES, D. O.; SHINZATO, P. Y.; OSÓRIO, F. S.; WOLF, D. F. Template-based autonomous navigation in urban environments. In: *26th SAC: Proceedings of the 2011 ACM Symposium on Applied Computing*, TaiChung, China, 2011.

3. SALES, D. O.; OSÓRIO, F. S.; WOLF, D. F. Topological Autonomous Navigation for Mobile Robots in Indoor Environments using ANN and FSM. In: *Proceedings of the I CBSEC: Conferência Brasileira em Sistemas Embarcados Críticos*, São Carlos, Brasil, 2011.

4. SALES, D. O.; FEITOSA, D.; OSÓRIO, F. S.; WOLF, D. F. Multi-agent Autonomous Patrolling System using ANN and FSM Control. In: *Proceedings of the II CBSEC: CBSEC: Conferência Brasileira em Sistemas Embarcados Críticos*, Campinas, Brasil, 2012.

5. CORREA, D. S. O.; SALES, D. O.; SCIOTTI, D. F.; PRADO, M. G.; WOLF, D. F.; OSÓRIO, F. S. Mobile Robots Navigation in Indoor Environments Using Kinect Sensor. In: *Proceedings of the II CBSEC: CBSEC: Conferência Brasileira em Sistemas Embarcados Críticos*, Campinas, Brasil, 2012.

6. SALES, D. O.; CORREA, D. S. O.; OSÓRIO, F. S.; WOLF, D. F. 3D Vision-based Autonomous Navigation System using ANN and Kinect Sensor. In: *Conference Proceedings EANN 2012 – CCIS: Volume number 311.*, London, UK, 2012.

7. SALES, D. O.; OSÓRIO, F. S.; Vision-Based Autonomous Topological Navigation in Urban Environments. In: *Proceedings of IEEE Latin American Robotics Symposium*, Fortaleza, Brasil, 2012.

8. SALES, D. O.; FERNANDES, L. C.; OSÓRIO, F. S.; WOLF, D. F.; FSM-based visual navigation for autonomous vehicles. In: *Workshop on Visual Control of Mobile Robots - IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* , Vilamoura, Algarve, Portugal, 2012.