Universidade de Brasília – UnB
Campus Gama – FGA
Engenharia Eletrônica

Aprendizagem por demonstração baseada em redes neurais artificiais aplicada à robótica móvel

Bruno da Costa Motta

Orientador: Dr. Daniel Mauricio Muñoz Arboleda

Bruno da Costa Motta

Aprendizagem por demonstração baseada em redes neurais artificiais aplicada à robótica móvel

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Orientador: Dr. Daniel Mauricio Muñoz Arboleda

Brasília, DF
2015

FGA/UnB – Universidade de Brasília, Campus Gama

Aprendizagem por demonstração baseada em redes neurais artificiais aplicada à robótica móvel

Bruno da Costa Motta

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do título de Bacharel em Engenharia Eletrônica.

APROVADA POR:

Prof. Daniel Mauricio Muñoz Arboleda, PhD
(Orientador)

Prof.ª Mariana Costa Bernardes, PhD
(Examinadora)

Prof. Maurício Serrano, PhD
(Examinador)

Resumo

Este trabalho visa a implementação de uma rede neural artificial do tipo perceptron múltiplas camadas. Com aplicação ao método de aprendizagem por demonstração em uma plataforma robótica móvel, mais especificamente em um robô Curumim. O processo de aprendizagem consiste em primeiramente coletar os dados dos sensores de distância do robô e das velocidades de cada roda do mesmo, enquanto o robô percorre um caminho guiado. Após a coleta de dados, é realizado o treinamento da rede neural artificial tipo perceptron e a obtenção dos pesos sinápticos da rede. A rede treinada é então utilizada na plataforma do robô Curumim e é responsável pelo processo de aprendizagem do robô. Diversos testes foram realizados para comprovar a eficiência e precisão da rede, como também diferentes topologias de RNA's foram testadas. Os treinamentos foram realizados em um ambiente de contorno fechado, onde diferentes trajetórias foram demonstradas e executadas pelo robô.

Palavras-chaves: LfD; Aprendizagem por Demonstração; RNA; Robótica Móvel; Robô Curumim;

Abstract

This work aims to implement a learning from demonstration algorithm, using an artificial neural network multilayer perceptron, applied in a mobile robotic platform, more specifically in a Curumim robot. The learning process occurs by first collecting data from the distance sensors of the robot and the velocities of each wheel, while the robot follows a guided path. After data is collected, the training of the neural network is conducted towards of obtaining the synaptic weights of the network. The trained network will be used in the Curumim mobile robot platform and is responsible for the learning process of the robot. Several tests were conducted to demonstrate the accuracy and efficiency of the network, different topologies of neural networks were also tested. The trainings were conducted in a closed loop environment, where different trajectories were demonstrated and performed by the robot.

.

Keywords: LfD; Learning from Demonstration; Neural Networks; Multilayer Perceptron; Mobile Robot; Curumim Robot;

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

O propósito de muitas aplicações em robótica móvel é reduzir o trabalho realizado por humanos em determinadas tarefas, e também aumentar a precisão e eficiência na realização dessas tarefas por meio da utilização de robôs. Baseados nesses requisitos, engenheiros projetam máquinas e escrevem programas com o intuito de suprir essa demanda, criando robôs que executam tarefas específicas.

Gradualmente, com o desenvolvimento da tecnologia e a demanda pela realização de diferentes tarefas, robôs com um número limitado de habilidades passam a se tornar ineficientes em determinadas aplicações onde a necessidade de aprender e realizar uma nova tarefa é requisitada. Robôs com algum grau de inteligência artificial passam então a serem requisitados para substituir aqueles que só são capazes de realizar tarefas pré-definidas ou pré-programadas. No entanto, projetar um robô com uma autonomia suficiente para auxiliar o ser humano na realização de novas tarefas sem a necessidade de uma reconfiguração ou reprogramação do robô, ainda é um desafio difícil de ser realizado.

1.1 Descrição do problema

Motivados pelo fato de que a imitação é um método que possibilita os seres humanos a aprender novas tarefas de uma forma fácil, engenheiros e cientistas passaram a desenvolver técnicas de aprendizagem com o intuito de ensinar novas tarefas para um robô por meio de um processo de imitação. Com isso foram criados sistemas de aprendizado autônomos para robôs, desenvolvendo o que hoje é chamado de aprendizagem por demonstração *(Learning from Demonstration*

- *LfD)* [4].

Diversas aplicações utilizando robótica móvel consistem em explorar e percorrer caminhos e campos muitas vezes de difícil acesso ou de difícil mapeamento. A utilização de aprendizagem por demonstração possibilita uma melhor exploração desses caminhos, de forma que o robô desenvolve a capacidade de aprender e se adaptar a novas trajetórias previamente demonstradas.

Existem diversas técnicas utilizadas no processo de LfD. Isso se dá pelo fato de que esse tipo de aprendizagem não possui soluções analíticas, não permitindo que o sistema aprenda uma tarefa de forma imediata. Geralmente essas soluções são baseadas em meta-heurísticas, dentre as quais as redes neurais artificiais tem sido amplamente utilizadas [1].

Levando em consideração as vantagens de aplicar técnicas de aprendizagem por demonstração na robótica móvel e considerando as inúmeras possibilidades que isso acarreta, esse trabalho visa a implementação de uma rede neural artificial tipo perceptron múltiplas camadas. Essa implementação será aplicada ao método de aprendizagem por demonstração em robôs móveis, objetivando o aprendizado de trajetórias por um robô Curumim.

1.2 Justificativa

Uma rede neural artificial (RNA) tem como objetivo principal adquirir e armazenar conhecimento, se assemelhando a um cérebro humano nesse quesito. Segundo Haykin[6], uma rede neural é um mecanismo projetado para modelar a maneira como o cérebro realiza uma determinada tarefa ou função. RNA's vem sendo bastante utilizadas no meio científico na solução de problemas de LfD por apresentarem uma facilidade de implementação e bons resultados se

comparados com outras técnicas.

O processo de utilização de RNA's na solução do problema de LfD consiste em modificar os chamados pesos sinápticos da rede neural, com o intuito de gerar uma saída desejada baseada nas entradas da rede. Levando em consideração o conceito de RNA's, e sua boa relação e aplicabilidade no processo de aprendizagem e armazenamento de conhecimento, legitima-se a utilização de redes neurais na implementação de um algoritmo de aprendizagem por demonstração aplicado em robótica móvel.

1.3 Objetivos

O objetivo geral desse trabalho consiste em implementar e aplicar uma rede neural artificial em um processo de aprendizagem por demonstração. É utilizada para isso a metodologia de LfD no aprendizado de trajetórias por robôs móveis através de um robô Curumim. Dessa forma o objetivo final é o aprendizado de diferentes trajetórias utilizando-se uma única ferramenta, sem a necessidade de uma reconfiguração ou reprogramação do robô.

Os objetivos específicos estão listados a seguir:

- Desenvolver um ambiente estruturado para coleta de dados e validação dos resultados.
- Propor experimentos simples que permitam a realização de uma análise de desempenho da metodologia de LfD e das RNAs aplicadas ao aprendizado supervisionado de trajetórias, visando o controle da movimentação do robô no intuito de desviar de obstáculos de acordo com a trajetória demonstrada.

- Realizar o treinamento da RNA utilizando uma rede neural perceptron de múltiplas camadas, variando o número de neurônios na camada escondida da RNA.

- Validar o comportamento das trajetórias obtidas, analisando as capacidades de generalização do conhecimento das RNAS de modo que a RNA mais complexa (com mais neurônios) seja capaz de executar tanto as tarefas mais simples como as mais complexas.

1.4 Organização do trabalho

O documento está organizado da seguinte maneira: O Capítulo 1, trata da introdução do trabalho, como visto acima. O Capítulo 2 aborda a fundamentação teórica por trás do trabalho proposto, uma breve descrição de robótica móvel e topologias de robôs, assim como os conceitos e métodos de aprendizagem por demonstração utilizando redes neurais artificiais. O Capítulo 3 descreve a implementação do trabalho e suas etapas metodológicas. O Capítulo 4 apresenta os resultados obtidos de simulação,de treinamento da rede neural artificial e de aprendizagem do robô Curumim. E por fim, o Capítulo 5 apresenta a conclusão do resultados e do trabalho como um todo, além de sugerir trabalhos futuros baseado no que foi desenvolvido.

2 Fundamentação teórica

Este Capítulo apresenta a fundamentação teórica do trabalho desenvolvido. O Capítulo está organizado da seguinte forma: Seção 2.1 aborda o conceito de robótica móvel, topologia de robôs e estratégias de controle de robôs móveis. A Seção 2.2 descreve a metodologia de aprendizagem por demonstração, e a Seção 2.3 apresenta os conceitos de redes neurais artificiais, rede neural tipo perceptron múltiplas camadas, e trata também do algoritmo de treinamento *backpropagation*.

2.1 Robótica Móvel

O estudo da robótica móvel é um tema bastante relevante e atual, de forma que essa área de estudo e pesquisa vem apresentando um grande salto em seu desenvolvimento nas últimas duas décadas. A aplicação prática de robôs móveis em diferentes atividades na indústria vem demonstrando o quão promissor é o futuro desta área [9].

A robótica móvel é utilizada muitas vezes no auxílio e execução de tarefas de difícil realização e ou exaustivas, que requerem muito tempo e repetições para serem completadas. Com o grande avanço tecnológico dos últimos anos, o desenvolvimento de novas técnicas e metodologias aplicadas a robótica móvel foram possibilitadas. Com isso existem novas áreas a serem exploradas e otimizadas, que visam uma melhor eficiência e qualidade na execução de tarefas por meio da utilização de robôs móveis.

2.1.1 Conceito de robô móvel

Os robôs móveis são dispositivos de transporte automático, ou seja, são plataformas mecânicas que possuem um sistema de locomoção capaz de navegar através de um determinado ambiente de trabalho. Robôs móveis são também dotados de um certo nível de autonomia para sua locomoção, que depende muitas vezes de sua aplicação [11].

Existem diversas topologias de robôs móveis, responsáveis por diferenciar e caracterizar os robôs visando aplicações específicas. Topologias de robôs móveis comumente variam na quantidade e disposição das rodas do robô e portanto na forma de locomoção do mesmo.

2.1.2 Topologia de robôs

Robôs móveis são caracterizados pelo número e tipo de suas rodas, que são responsáveis por definir o grau de manobrabilidade do robô. A seguir são apresentadas algumas das principais topologias de robôs móveis, suas diferenças e vantagens.

Robô de Tração Diferencial

Um robô de tração diferencial estabelece sua posição controlando a velocidade e orientação em cada roda. Configurações típicas incluem 3 rodas posicionadas nas vértices de um triângulo retângulo, comumente conectadas por um cilindro[3]. Em um robô de tração diferencial todas as rodas viram e andam em sincronia, todas as rodas apontam para a mesma direção e viram a uma mesma taxa. Um arranjo mecânico comum para um robô de tração diferencial utiliza dois motores independentes, um para controlar a movimentação para

frente das rodas, e outro para controlar a rotação das mesmas (movimento de curva do robô, tal como mostrado na Fig. 2.1). Todas as rodas de um robô de tração diferencial operam em paralelo, dessa forma o robô sempre rotaciona em torno de seu eixo central, com isso ele adquire a capacidade de controlar diretamente o ângulo de rotação θ de sua posição. A habilidade de controlar de forma independente a rotação e a velocidade para frente do robô, simplifica consideravelmente o controle do veículo e possibilita com que o robô sirva como um modelo conveniente de um robô pontual idealizado[3].

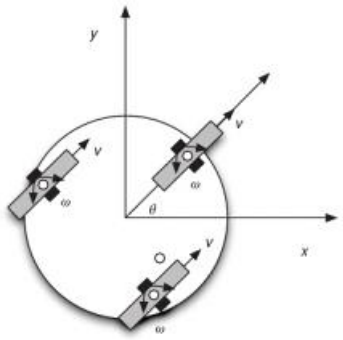

Figura 2.1. Robô de tração diferencial [11]

A equação cinemática que descreve a movimentação do robô de tração diferencial, conhecida como equação cinemática direta, é dada a seguir:

$$\begin{bmatrix} v \\ w \end{bmatrix} = 2\pi r \begin{bmatrix} 1/2 & 1/2 \\ -1/d & -1/d \end{bmatrix} \begin{bmatrix} \theta_L \\ \theta_R \end{bmatrix} \tag{2.1}$$

Onde v representa a velocidade linear do robô, w representa a velocidade angular,θ_L, θ_R são as velocidades de cada roda em revoluções por minuto, r é o raio da roda e d é a distância entre duas rodas do robô.

A equação cinemática inversa de um robô de tração diferencial pode ser derivada da equação anterior, invertendo-se sua matriz:

$$\begin{bmatrix} \theta_L \\ \theta_R \end{bmatrix} = 1/2\pi r \begin{bmatrix} 1 & -d/2 \\ 1 & d/2 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{2.2}$$

Robô Ackerman

O modelo de direção Ackerman é comumente utilizado em automóveis. Esse modelo possui quatro rodas, as rodas da frente rotacionam separadamente fazendo com que o robô tenha uma maior liberdade e variedade na rotação, as rodas traseiras não possuem liberdade de rotação (rodas fixas). Além disso os eixos das rodas frontais se interceptam em um ponto C pertencente ao eixo comum das rodas traseiras [11], formando um conjunto de arcos concêntricos, onde os vetores de velocidade instantânea das rodas são tangentes a estes arcos (vide Fig. 2.2).

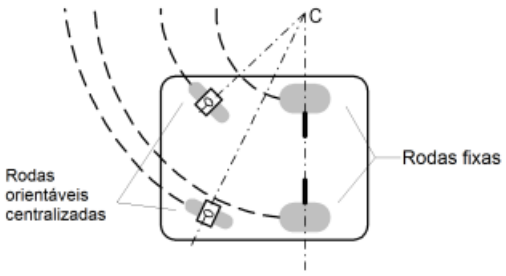

Figura 2.2. Robô Ackerman [11]

Esse tipo de estrutura proporciona uma maior estabilidade e tração ao robô, reduzindo com isso os erros relacionados a odometria (método utilizado para estimar a posição do robô baseado no giro das rodas)[11].

Em geral, o robô Ackerman tem a capacidade de variar a velocidade das

rodas de forma que se é necessário modificar a velocidade de apenas uma roda para variar a velocidade de todas as quatro.

A seguir tem-se a equação cinemática de um robô Ackerman:

$$\begin{bmatrix} v \\ w \end{bmatrix} = 2\pi r\theta \begin{bmatrix} 1 \\ sin(\alpha)/e \end{bmatrix} \tag{2.3}$$

Onde v é a velocidade linear do robô, w é a velocidade angular, r é o radio da roda, θ é a velocidade das rodas em rotações por segundo, α é o ângulo de curva do robô, e é a distância entre as rodas dianteiras e traseiras.

Robô Omnidirecional

Robôs omnidirecionais são caracterizados pela habilidade de se mover em qualquer direção, sem a necessidade de mudarem sua orientação. Para que isso seja possível, cada roda tem a capacidade de rotacionar a diferentes velocidades e em diferentes direções, sendo necessário um motor para cada roda. Devido a essas propriedades, é comum dizer que o robô omnidirecional se movimenta em diferentes ângulos, sendo que para que o robô se movimente em um ângulo específico cada motor precisa operar a uma certa velocidade em relação aos outros[11].

Um robô omnidirecional normalmente possui três rodas montadas em uma estrutura triangular, porém existem também robôs omnidirecionais com quatro rodas, como mostrado na Fig. 2.3. A maior vantagem de se utilizar três rodas, é na economia de material, especialmente pela necessidade de um motor para cada roda nesse tipo de configuração.

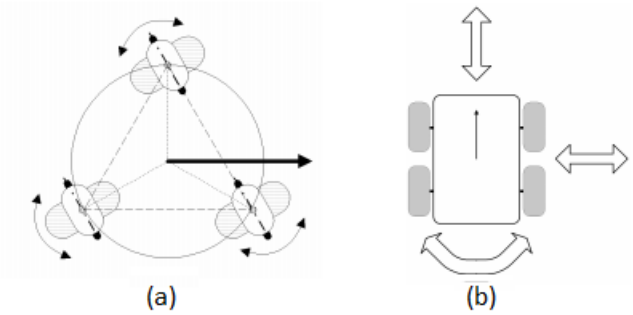

Figura 2.3. Robô Omnidirecional - (a)Topologia com três rodas, (b)Topologia com quatro rodas [11]

A utilização de quatro rodas porém, apresenta algumas desvantagens, como no caso de terrenos irregulares, onde não se é possível garantir que as quatro rodas estarão tocando o mesmo plano, acarretando movimentos imprevistos na trajetória, e em alguns casos até mesmo o atolamento do robô.

Um robô omnidirecional comumente apresenta rodas do tipo *Mecanum* (também conhecidas como rodas *Suecas*), ou rodas do tipo *Castor*. Rodas *Suecas* são envolvidas por *cilindros de rotação livre* (vide Fig 2.4), que conectados por rolamentos esféricos permitem uma livre rotação das rodas em torno de seus eixos, possibilitando assim a movimentação do robô em todas as direções sem a necessidade de uma mudança de orientação.

Figura 2.4. Rodas Mecanum (Suecas)[2]

A equação cinemática de um robô omnidirecional é uma matriz que especifica qual direção o robô irá seguir. A equação cinemática de maior interesse para este trabalho, é a equação cinemática inversa de um robô omnidirecional de

três rodas, pois essa equação expressa a movimentação das rodas de um robô Curumim que será utilizado neste contexto. Essa equação é representada a seguir:

$$\begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = 1/r \begin{bmatrix} -sin(\theta) & cos(\theta) & R \\ -sin(\theta+\alpha_2) & cos(\theta+\alpha_2) & R \\ -sin(\theta+\alpha_3) & cos(\theta+\alpha_3) & R \end{bmatrix} \begin{bmatrix} x \\ y \\ \Theta \end{bmatrix} \tag{2.4}$$

Onde ϕ_1, ϕ_2 e ϕ_3 são as velocidades angulares de cada roda, r é o raio de cada roda, θ é o ângulo de rotação do robô, α_2 e α_3 são respectivamente os ângulos que definem as posições das rodas 2 e 3 a partir de um plano local de coordenadas. R é a distância do centro de gravidade do robô até as rodas do mesmo, e x, y e Θ, são as velocidades linear e angular do robô.

2.1.3 Robô Curumim

O robô Curumim é parte de um kit de desenvolvimento educacional e aprendizagem. Essa plataforma possibilita estudantes de diversas áreas e níveis de conhecimento aplicarem conceitos de robótica e engenharia em geral, promovendo o desenvolvimento de tecnologia aplicados à robótica.

O robô Curumim possui software livre e uma arquitetura de hardware aberta em prol do desenvolvimento de novas aplicações. Possui também cinco sensores infravermelhos e uma câmera analógica frontal (vide Fig. 2.5). Utiliza a biblioteca openCV para o processamento de imagens e pode ser programado em C e C++. O robô apresenta a topologia omnidirecional com três rodas, além de um microcontrolador MSP430, um Arm Cortex A9 e uma comunicação via rádio wireless que trabalha na frequência de 2.4Ghz digitais. O sistema é originalmente alimentado através de duas baterias de Ni-Mh capazes de fornecer uma

tensão de 14.4V e uma corrente nominal de 2500mA/h. Porém o kit utilizado neste trabalho não possui as baterias originais, sendo que foi utilizada então uma fonte de alimentação DC de 15V com uma corrente nominal de 2500mA/h com o intuito de suprir a ausência das baterias.

Figura 2.5. Robô Curumim[15]

A Figura 2.6 apresenta um diagrama de blocos da unidade móvel do robô Curumim. Neste diagrama pode-se observar os componentes constituintes do robô, assim como a direção do fluxo da comunicação entre os mesmos[15].

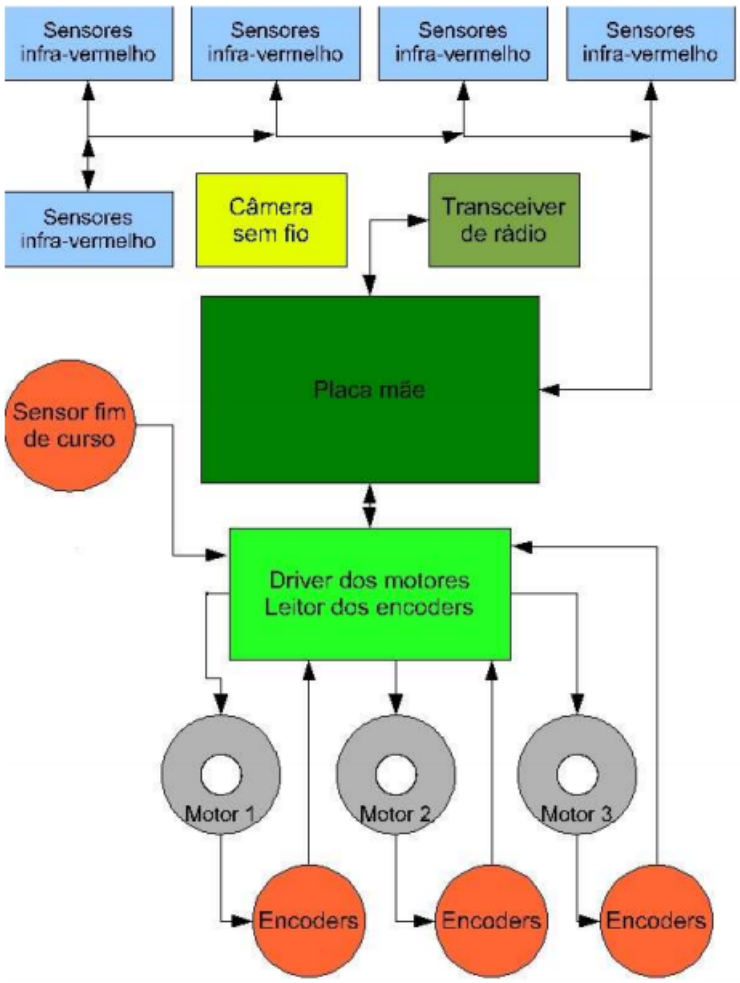

Figura 2.6. Diagrama de bloco da Unidade Móvel [15]

O robô se locomove a partir de três motoredutores responsáveis por controlar cada uma de suas rodas omnidirecionais. Com a devida combinação de acionamento das rodas, é possível executar qualquer tipo de manobra. A Figura 2.7 ilustra os motoredutores e as rodas montadas em suas devidas posições na plataforma do robô.

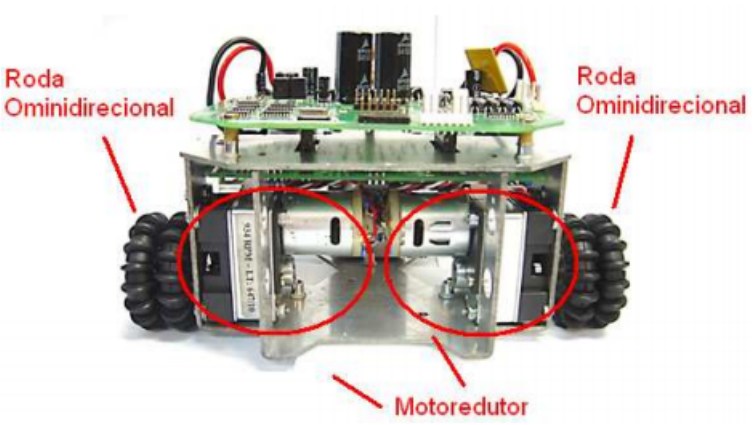

Figura 2.7. Rodas e motoredutores montados [15]

O robô Curumim é composto por 6 rodas agrupadas em pares. Essas rodas são relativamente simples, sendo compostas por roletes envolvendo toda a roda paralelos ao seu eixo de rotação. Isso permite com que a roda possa se deslocar na direção perpendicular ao eixo do motor. A Figura 2.8 ilustra um par de rodas Omnidirecionais utilizadas no robô Curumim.

Figura 2.8. Rodas omnidirecionais [15]

Devido a desgastes apresentados nas rodas do robô Curumim utilizado neste trabalho, a habilidade de andar de lado, característica interessante de um robô Omnidirecional, foi prejudicada e portanto não foi explorada no desenvolvimento deste trabalho. Considerando que a habilidade de andar de lado não será explorada no contexto em questão, e objetivando uma maior simplificação dos experimentos que serão aqui tratados, optou-se por não utilizar a roda

central do robô. Foram utilizadas portanto apenas combinações de movimentos das duas rodas laterais (paralelas entre si) do robô Curumim, tanto para realizar movimentos retos como curvilíneos. Com a modificação na utilização das rodas do robô Curumim, passa-se a ter uma topologia de tração diferencial (vide Seção 2.1.2), ao invés de uma topologia omnidirecional, previamente estabelecida. Desta forma os modelos cinemáticos que realizam o controle por meio da velocidade do robô são expressados pelas Equações 2.1 e 2.2.

Os sensores de proximidade do robô são do tipo infravermelho reflexivo, possuem um alcance de aproximadamente 860mm e abrangem toda a região ao redor da plataforma do robô.

Figura 2.9. Sensor de Infravermelho [15]

A plataforma Curumim dispõe também de um software responsável por intermediar a comunicação entre o usuário e o robô. Este software permite a programação através de blocos ou utilizando linguagem C/C++. A Figura 2.10 mostra a interface principal do software, dividida em cinco áreas: 1. Área de Menu e Barra de Ferramentas; 2. Área de Programação em blocos; 3. Área de Imagem; 4. Área de Programação C/C++; 5. Área de Mensagens.

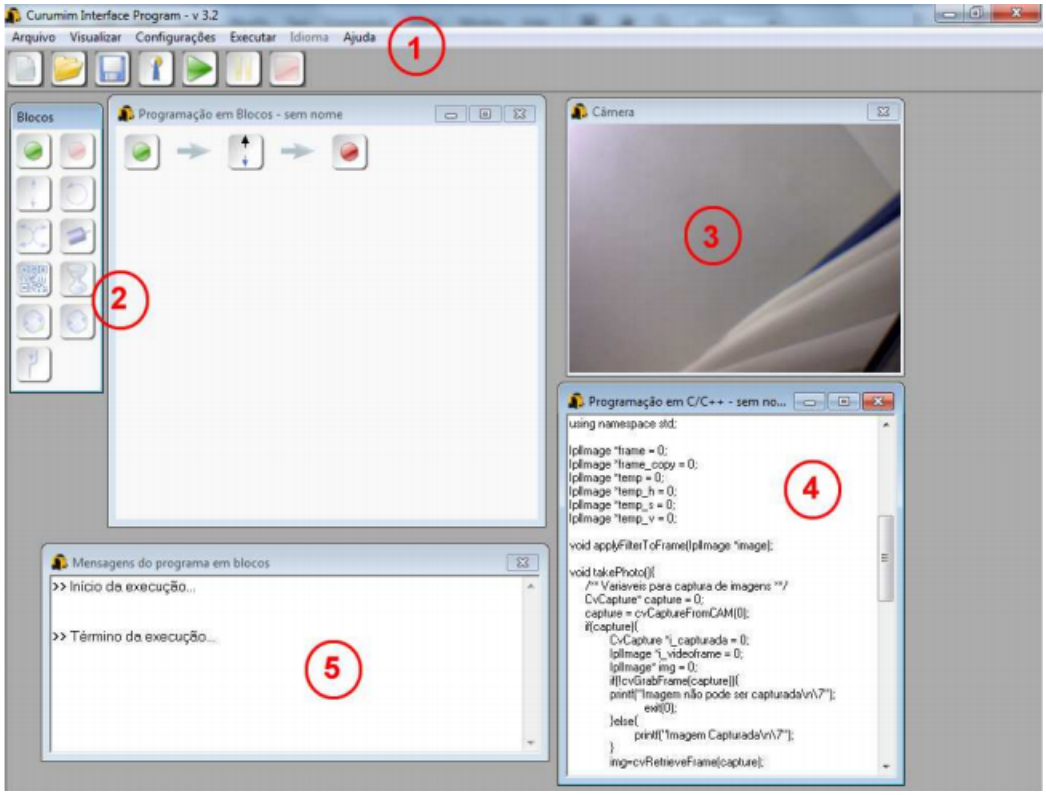

Figura 2.10. Divisão da interface do software Curumim [14]

2.1.4 Simulador EyeSim

O simulador EyeSim é um simulador 3D de robôs móveis que permite com que se teste e execute programas antes de utilizá-los em robôs reais. Essa prática é muito útil durante o desenvolvimento de novas aplicações, pois é mais prático e cômodo simular primeiramente o trabalho em desenvolvimento, para depois, quando consolidado, aplicá-lo no robô real.

O simulador dispõe de diferentes topologias de robôs, além de seis sensores de posição, câmera e comunicação a rádio. Para este trabalho as simulações foram realizadas utilizando-se o robô omni 2 (omnidirecional) de quatro rodas. As simulações foram realizadas unicamente como prova de conceito da metodologia de LfD aqui estudada, o fato de simular uma topologia com quatro rodas não irar afetar ou comprometer os resultados utilizando o robô Curumim, pois todo

o processo metodológico será refeito para o robô físico. O simulador dispõe também de um painel de controle, onde se pode visualizar os parâmetros do robô, como os dados dos sensores. Diferentes mapas e ambientes são também fornecidos pelo simulador. Com isso pode-se testar a eficiência do algoritmo utilizado para diferentes ambientes, e dessa forma verificar e validar a correta implementação do projeto.

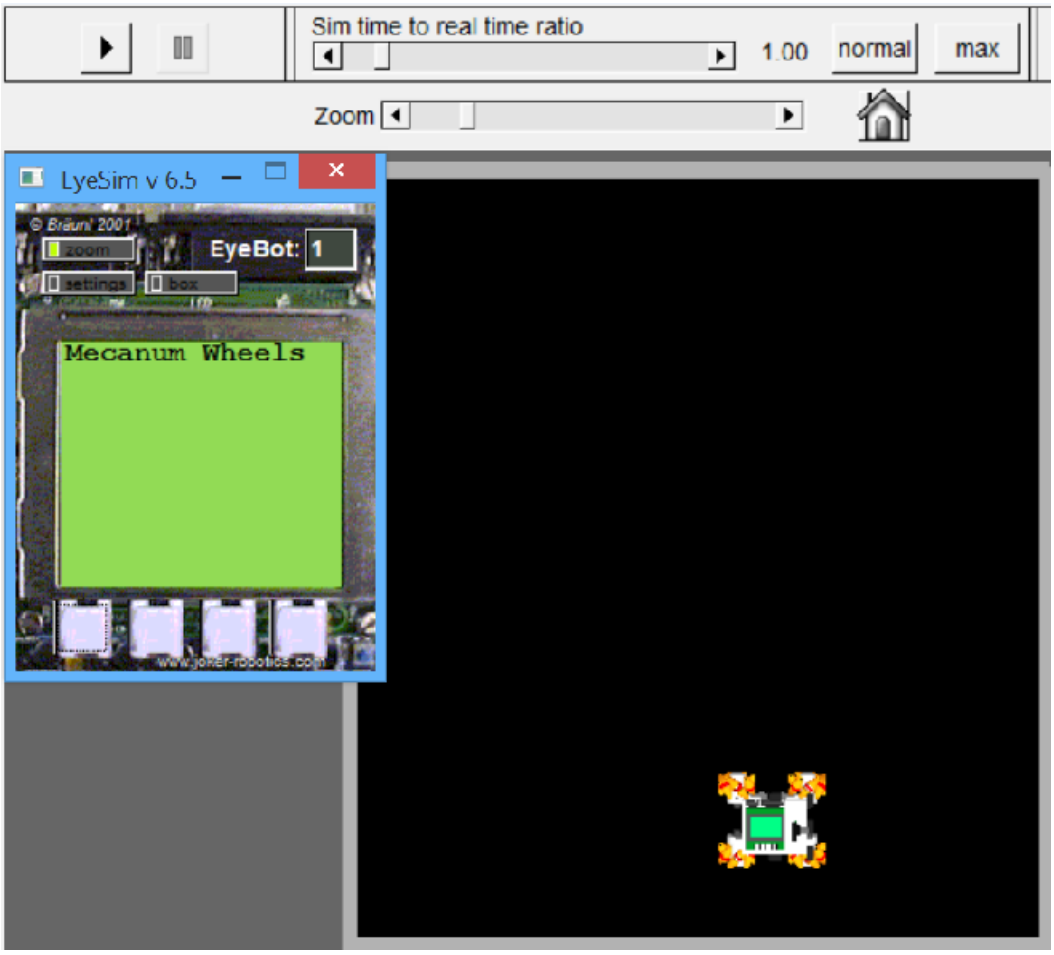

Figura 2.11. Simulador EyeSim

2.1.5 Estratégias de controle de robô móvel

Existe uma série de recursos que podem ser utilizados para controlar e determinar as decisões de um robô móvel. Serão tratados a seguir os recursos mais importantes e relevantes para o desenvolvimento deste trabalho.

Camada de sensoriamento

Existe uma grande variedade de sensores que podem ser utilizados em robótica móvel, e a escolha desses sensores desempenha um papel muito importante no desenvolvimento do projeto ou da aplicação em que estão inseridos. Portanto é essencial saber escolher os sensores certos para uma aplicação em particular.

Alguns fatores importantes na escolha de um sensor são o tipo de medida que se pretende obter, o tamanho e peso do sensor, a temperatura de operação, o consumo de potência desse sensor e também a viabilidade monetária do mesmo.

Existem diversas categorias de sensores, os critérios de classificação desses sensores é baseado em seus sinais de saída e também em suas aplicações. Para robôs móveis é importante distinguir os sensores em locais e globais, sendo que sensores locais e globais são também distinguidos em internos e externos. Sensores locais internos são aqueles acoplados ao robô e que são responsáveis por monitorar as funções internas do mesmo, alguns exemplos são:

- sensores de bateria
- sensores de temperatura
- acelerômetros
- giroscópios

Sensores locais externos são também acoplados ao robô, porém eles são responsáveis por monitorar o ambiente externo ao robô. Alguns exemplos são vistos a seguir:

- câmeras
- sensores sonares
- sensores infravermelhos
- sensores de luminosidade

Sensores globais se encontram no ambiente em que o robô está inserido, como câmeras, GPS e outros sistemas de posicionamento global.

Camada de tomada de decisões

As decisões tomadas por um robô móvel dependem de diversos fatores, tanto internos como externos. Fatores externos estão diretamente ligados ao dados de entrada coletados pelo robô através de seus sensores. Esses dados variam de acordo com o ambiente e a situação em que o robô se encontra em determinado momento. Os fatores internos dependem da forma com que o robô foi treinado para lidar ou responder aos fatores externos, dependem portanto do tipo de treinamento e orientação pré-programados no robô. Esses fatores combinados são responsáveis por gerar a saída do sistema.

No trabalho em questão, os fatores determinantes de tomada de decisões são os sensores de distância do robô, as velocidades de suas rodas e a configuração da rede neural do mesmo, responsável por determinar as saídas do sistema (decisões). Essa etapa é essencial para o correto e esperado funcionamento do sistema, qualquer erro ou inacurácia nessa etapa, pode acarretar problemas

drásticos na resposta do robô. Portanto essa é uma, se não a etapa mais importante no processo de controle do robô móvel.

Camada de controle de baixo nível

A camada de controle de baixo nível é responsável por executar fisicamente as saídas geradas na etapa de tomada de decisões. Essa etapa envolve a interpretação e processamento das decisões a nível de máquina, utilizando-se dos recursos de hardware do robô para controlar sua velocidade e orientação.

2.2 Aprendizagem por demonstração

O princípio de aprendizagem por demonstração (*learning from demonstration* - LfD) se baseia em ensinar novas tarefas a robôs sem a necessidade de programação.

Tomando em conta um cenário clássico de programação, se faz necessário primeiramente programar todas as tarefas que se deseja que o robô realize, sendo necessário cobrir todas as possibilidades e adversidades que possam ocorrer nesse cenário. Esse processo porém, envolve muitas etapas e testes, e caso erros ou novas circunstâncias ocorram depois da implementação no robô, muitas e em alguns casos todas as etapas do processo precisam ser refeitas[4].

Técnicas e métodos de aprendizagem por demonstração permitem ao usuário final comandar e especificar tarefas a serem realizadas pelo robô, sem nenhuma necessidade de programação, apenas demonstrando fisicamente como realizá-las. Dessa forma, quando algum erro ou adversidade ocorrer, será necessário apenas fornecer mais demonstrações para o robô, evitando assim a necessidade de reprogramação do mesmo.

Existem diversas técnicas utilizadas no processo de LfD. Uma bastante difundida e que será utilizada nesse trabalho se baseia na utilização de redes neurais artificiais no processo de aprendizagem e de reprodução das tarefas pelo robô. O processo de aprendizagem pode ser classificado em supervisionado e não supervisionado.

2.2.1 Aprendizagem não supervisionada

Aprendizagem não supervisionada ou *aprendizagem sem um professor* como é também chamada, se caracteriza por não possuir exemplos ou demonstrações da função a ser aprendida pela rede neural [6]. Esse tipo de aprendizagem é dividida em aprendizagem por reforço e aprendizagem auto-organizada.

Na aprendizagem por reforço o aprendizado é realizado através da interação contínua do sistema com o ambiente, objetivando minimizar um índice de desempenho. Na Figura 2.12 é ilustrado um diagrama de blocos de um tipo de aprendizagem por reforço. Esse sistema em específico é denominado aprendizagem por reforço atrasado, onde se observa uma sequência temporal de estímulos recebidos pelo ambiente através de vetores de estado. O objetivo dessa aprendizagem é minimizar uma função de custo para poder avançar, onde essa função é definida como a expectativa do custo cumulativo de ações tomadas ao longo de uma sequência de passos [7].

Na aprendizagem auto-organizada, ou simplesmente *não-supervisionada*, não existe a presença de um professor para auxiliar o processo de aprendizagem do robô. São dadas então condições para se realizar uma medida da qualidade das tarefas que a rede deve aprender através de dados de entrada do ambiente, e com isso parâmetros livres da rede neural são otimizados em relação a esta

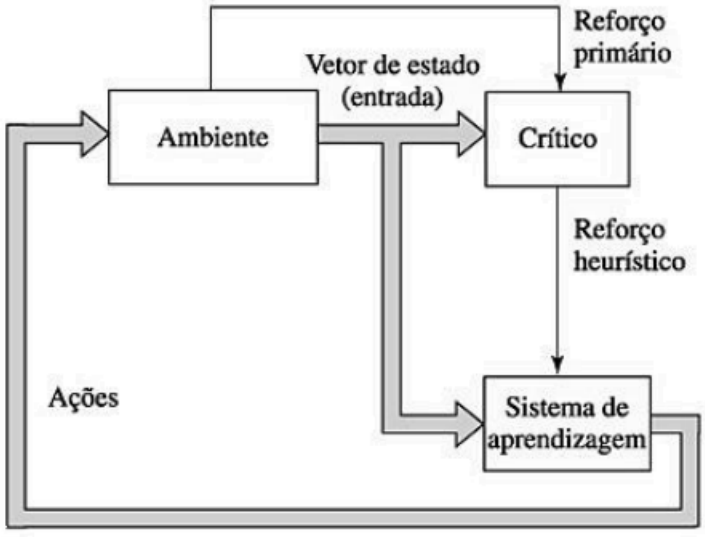

Figura 2.12. Diagrama em blocos da aprendizagem por reforço [6]

medida. Uma vez que a rede consiga se ajustar às regularidades estatísticas dos dados de entrada, ela é capaz de criar representações internas das características de entrada da rede e com isso gerar suas respectivas saídas.

2.2.2 Aprendizagem Supervisionada

A aprendizagem supervisionada ou *aprendizagem com um professor*, será utilizada no processo de aprendizagem por demonstração realizado neste trabalho. Para melhor clarificar e explicar seu conceito e aplicabilidade no contexto em questão, esse processo será dividido em três etapas, descritas a seguir:

- Demonstração - A etapa de demonstração consiste em, por meio de um professor, executar as atividades que se deseja que o robô aprenda. Assume-se que o professor possui conhecimento sobre o ambiente, que é representado por um conjunto de exemplos de *entrada-saída* [6], sendo que esse mesmo ambiente é desconhecido pelo robô e consequentemente pela rede neural. Considera-se então, que o robô e o professor são expostos a um vetor de

treinamento. De modo que o professor, baseado em seu conhecimento sobre o ambiente, tem a capacidade de fornecer uma resposta à rede neural, que representa a ação ótima a ser realizada pela RNA de acordo com o vetor de treinamento.

- Aprendizagem - Em uma aprendizagem supervisionada, os parâmetros da rede neural artificial são ajustados baseados no vetor de treinamento e no sinal de erro, que é definido pela diferença entre a resposta desejada e a resposta real da rede. Esse ajuste está ilustrado na Fig 2.13, e objetiva com que a rede neural simule o professor, transferindo o conhecimento do ambiente para a rede neural. Essa emulação é considerada ótima em um sentido estatístico [7], de modo que o conhecimento do ambiente é transferido para a rede de uma forma completa.

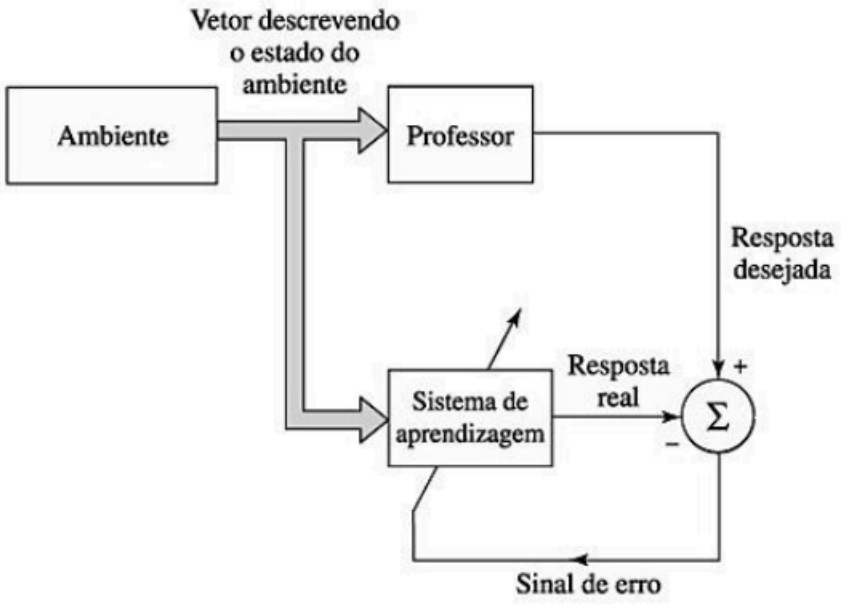

Figura 2.13. Diagrama de blocos da aprendizagem supervisionada [6]

- Imitação - Quando a condição de aprendizagem completa é estabelecida, significa que a rede neural terminou seu processo de treinamento. Dessa

forma o auxílio do professor não é mais necessário, e a rede neural é capaz de reconhecer e lidar com o ambiente sem a necessidade de ajuda externa. A esse processo é dado o nome de Imitação, onde a rede neural artificial é capaz de responder ao ambiente de forma completamente independente, de acordo com o que foi ensinado e esperado na fase de aprendizagem supervisionada.

2.3 Conceitos básicos sobre redes neurais artificiais

Uma rede neural artificial é um modelo matemático que objetiva reproduzir o funcionamento do cérebro. Segundo Haykin[7], uma rede neural artificial é um processador paralelamente distribuído constituído de unidades de processamento simples, que tem a propensão natural de armazenar conhecimento e torná-lo disponível para o uso. Uma rede neural se assemelha ao cérebro humano em dois aspectos: O primeiro aspecto está relacionando à aquisição de conhecimento, onde o conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem [7]. O segundo está relacionado ao armazenamento do conhecimento, onde os pesos sinápticos, que representam forças de conexões entre os neurônios cerebrais, representados por neurônios artificiais, são utilizados para armazenar esse conhecimento obtido pela rede.

Neurônios artificiais são unidades de processamento fundamentais para o funcionamento da rede neural. Eles possuem pesos sinápticos, utilizados para ponderar os dados de entrada, um somador, responsável por somar todos os valores de entrada ponderados, e uma função de ativação, responsável por restringir o sinal de saída do neurônio.

Um neurônio artificial pode ser descrito pela seguinte equação matemática:

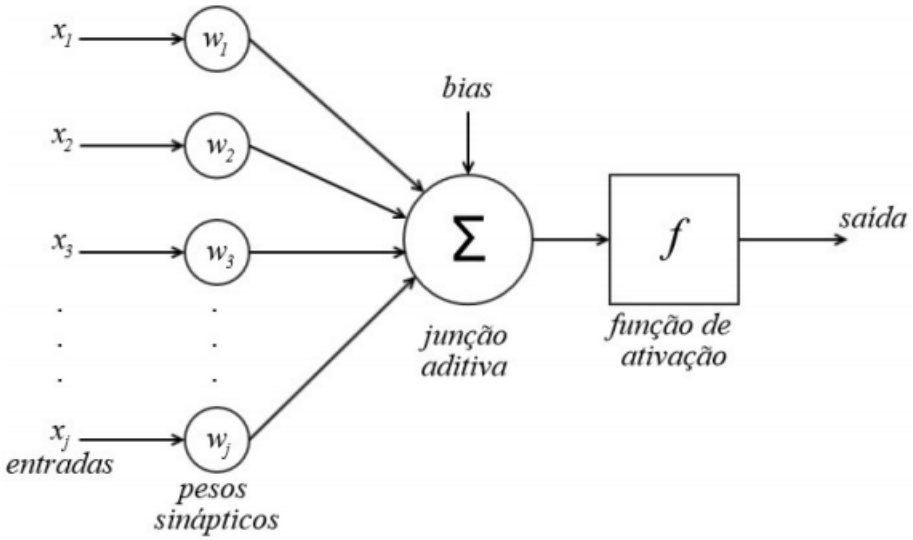

Figura 2.14. Representação de um neurônio artificial [6]

$$y = F(\sum_{i=1}^{\mathrm{m}} w_i x_i + b) \tag{2.5}$$

onde *w1, w2, ..., wi* são os pesos sinápticos, *x1, x2, ..., xi* são os sinais de entrada, F a função de ativação, b o bias, e y o sinal de saída do neurônio [6]

A função de ativação é responsável por restringir os valores de saída do neurônio. Para os propósitos desse trabalho, será utilizada a função tangente hiperbólica. Ela restringe a saída do neurônio e consequentemente a saída da rede neural, em valores de -1 e 1. Esses valores serão responsáveis por controlar as rodas do robô.

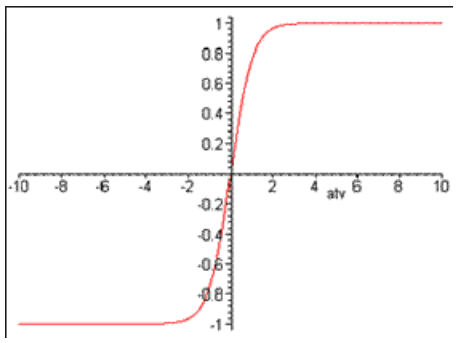

Figura 2.15. Função de ativação Tangente Hiperbólica [7]

O processo de aprendizagem utilizado em redes neurais, é realizado a partir de um algoritmo denominado algoritmo de aprendizagem. Esse algoritmo estabelece e modifica os pesos sinápticos da rede neural de acordo com as entradas e com os resultados esperados para determinada aplicação. A partir daí, com os pesos sinápticos estabelecidos, se é capaz de gerar as saídas esperadas a partir de novas entradas, estabelecendo dessa forma o processo de aprendizagem.

Um fator importante relacionado à habilidade de aprendizagem em uma rede neural artificial, é a generalização. A generalização está relacionada com a capacidade da rede de gerar saídas adequadas a partir dos pesos sinápticos e das entradas recebidas, sendo que essas entradas não fizeram parte do processo de aprendizagem da rede (geração dos pesos sinápticos), e portanto, são dados absolutamente novos para a rede.

Redes neurais artificiais podem apresentar diferentes características e também oferecer diferentes propriedades, algumas delas apresentadas a seguir:

1- Não-linearidade. Neurônios artificiais podem apresentar características lineares ou não-lineares. Redes neurais compostas por conexões entre neurônios não-lineares, são também não-lineares. Essa característica não-linear da rede é importante, principalmente nos casos em que a aplicação envolve dados de entrada com uma natureza não-linear.

2- Mapeamento de entrada e saída. O processo de aprendizagem por demonstração utilizado nesse trabalho envolve a modificação dos pesos sinápticos da rede neural baseado em um conjunto de dados de entrada e de saída, comumente chamados de amostras de treinamento. Cada amostra de treinamento possui um sinal de entrada e uma saída desejada. Apresentam-se esses dados para a rede a fim de modificar os pesos sinápticos com o intuito de minimizar a

diferença entre a resposta desejada e a resposta real da rede. A partir da realização desse processo, cria-se um mapeamento de entrada e saída da rede, onde são geradas as saídas desejadas de acordo com as entradas dadas, aplicando-se os pesos sinápticos modificados.

3- Adaptabilidade. É a capacidade da rede de adaptar os pesos sinápticos de acordo com variações ou modificações do ambiente em que a rede foi aplicada. Isso significa que uma rede neural pode facilmente ser readaptada ou retreinada para responder e operar em um novo ambiente, simplesmente modificando seus pesos sinápticos.

Esses e outros fatores demonstram a importância e utilidade das redes neurais artificiais em geral, como também sua relevância no processo de aprendizagem por demonstração estudado neste trabalho.

2.3.1 Rede neural tipo perceptron de múltiplas camadas

Redes neurais artificiais tipo perceptron múltiplas camadas possuem um conjunto de unidades sensoriais que constituem a camada de entrada, camada oculta e camada de saída da rede, onde cada camada é constituída por nós computacionais, responsáveis por processar os dados de entrada da rede, aplicando os pesos sinápticos e gerando a saída desejada.

Redes perceptron múltiplas camadas são utilizadas para resolver diversos tipos de problemas complexos. Essas redes utilizam um algoritmo bastante difundido, chamado de algoritmo de retropropagação de erro (*error back-propagation*). Este algoritmo é baseado na regra de aprendizagem por correção de erro [6], é também considerado uma generalização do algoritmo do mínimo quadrado médio (LMS), utilizado em redes com um único neurônio linear.

Uma rede perceptron de múltiplas camadas possui três características mais importantes:

1- Cada neurônio da rede possui uma função chamada de função de ativação não-linear, onde a não-linearidade é suave (diferenciável em qualquer ponto). Uma forma de não-linearidade que satisfaz essa questão, é dada pela função sigmóide:

$$y = \frac{1}{1 + exp(-x)} \tag{2.6}$$

Onde x é dado pela soma ponderada das entradas sinápticas (com bias) de um neurônio e y é a saída do mesmo.

2- A rede possui uma ou várias camadas de neurônios ocultos, que se localizam entre a camada de entrada e a camada de saída da rede. Os neurônios ocultos possibilitam a rede a aprender tarefas mais complicadas, auxiliando na extração das características mais importantes dos dados de entrada da rede.

3- Uma rede perceptron de múltiplas camadas apresenta um alto grau de conectividade, determinado pelo número de sinapses da rede. Desta forma, uma mudança na conectividade da rede acarreta em uma mudança no número de conexões sinápticas.

É devido a essas características que uma rede neural artificial do tipo perceptron de múltiplas camadas apresenta uma grande capacidade de processamento computacional. e é por isso utilizada na resolução de problemas de maior complexidade. Porém existem algumas dificuldades relativas ao total conhecimento do comportamento da rede, como a análise teórica do modelo, devido a sua não-linearidade e alto grau de conectividade. A visualização do processo de aprendizagem é também dificultada devido a utilização de neurônios ocultos.

A seguir tem-se a representação de uma rede neural tipo perceptron de múltiplas camadas, com dois neurônios na camada de entrada, uma camada oculta com quatro neurônios e três neurônios na camada de saída:

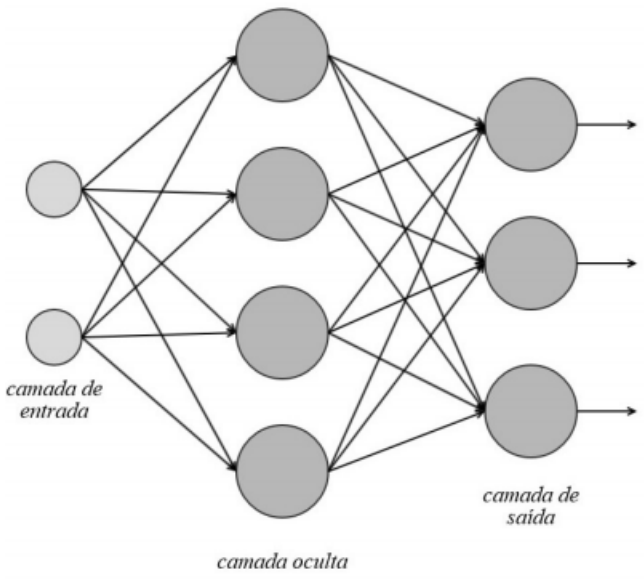

Figura 2.16. Rede Neural tipo perceptron múltiplas camadas [6]

2.4 Algoritmo de treinamento Backpropagation

Um algoritmo de treinamento backpropagation, ou retropropagação de erro, consiste basicamente de dois passos: um para frente, chamado propagação, e outro para trás, a retropropagação. Na propagação, um vetor de entradas é inserido na rede e percorre todas as camadas da mesma, gerando um vetor de saídas, definido como a resposta real da rede. Durante esse passo, os pesos sinápticos da rede são fixos [7].

No passo de retropropagação, os pesos sinápticos da rede são ajustados respeitando uma regra de correção, de forma que a resposta real da rede é subtraída de uma resposta desejada gerando um resultado chamado de sinal de erro. Este

sinal é propagado para trás (retropropagado) através da rede neural, ajustando os pesos sinápticos com o intuito de convergir a resposta real da rede com a resposta desejada, obtendo ao final do processo os pesos sinápticos otimizados.

A seguir tem-se a representação da equação do sinal de erro, definido pela soma dos erros quadráticos de cada neurônio:

$$E = \sum_{j,p} E_i^p = 1/2 \sum_{i,p} (d_i^p - O_i^p)^2 \tag{2.7}$$

sendo que E_i^p representa o erro no $i$ésimo elemento neural para o $p$ésimo padrão de entrada, d_i^p representa a saída esperada no $i$ésimo elemento neural para o $p$ésimo padrão de entrada, e O_i^p representa a saída produzida pelo neurônio, definida pela equação abaixo:

$$O_i^p = F(y_i^p) = F(\sum_j (w_i x_j^p - b_i)) \tag{2.8}$$

sendo que x_j^p é a $j$ésima entrada da rede neural.

O desenvolvimento do algoritmo backpropagation representa um marco nas redes neurais, pois fornece um método computacional eficiente para o treinamento de perceptrons de múltiplas camadas [6].

3 Implementação

Foi realizada neste trabalho a implementação e treinamento de uma rede neural artificial aplicada à metodologia de aprendizagem por demonstração. Essa implementação visa generalizar a solução do problema de aprendizagem de diferentes trajetórias para robôs móveis. Essa abordagem foi adotada considerando a problemática de se ter que reconfigurar ou reprogramar o robô sempre que se deseja realizar uma nova tarefa ou trajetória.

O processo de programar cada tarefa de um robô é relativamente caro, pois requer tempo e recursos humanos especializados. A técnica de LfD permite substituir o processo manual de programação de um robô por um processo automatizado, onde o robô é capaz de aprender novas trajetórias mediante demonstração.

Desta forma, esse trabalho objetiva a aplicação da metodologia de aprendizagem por demonstração no robô Curumim, uma plataforma robótica de aprendizagem utilizada para pesquisa e desenvolvimento em diversas áreas do conhecimento.

3.1 Metodologia

A metodologia utilizada no aprendizado, desenvolvimento e solução do problema proposto neste trabalho, está dividida a seguir:

1) Estudo teórico - Primeiramente foi realizado um estudo teórico sobre aprendizagem por demonstração e redes neurais artificiais.

2) Simulação EyeSim - Em seguida realizou-se o trabalho de desenvolvimento

e simulação do algoritmo de coleta dos dados de treinamento do robô, que consistem dos dados de distância dos sensores e velocidades das rodas do robô. Essa etapa de simulação foi realizada utilizando o simulador EyeSim. Foram realizados diferentes percursos e trajetórias pelo robô a fim de provar o conceito e aplicabilidade do problema de aprendizagem por demonstração aqui estudado. A Figura 3.1 ilustra as trajetórias demonstradas no simulador.

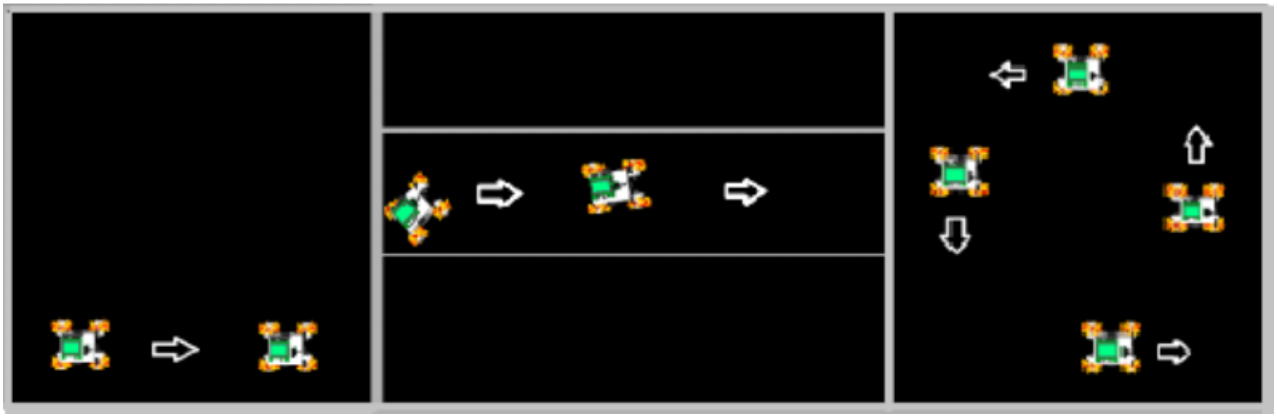

Figura 3.1. Trajetórias - Simulador EyeSim

Após a demonstração e coleta de dados dos experimentos utilizando o simulador EyeSim, iniciou-se a etapa de treinamento da rede neural artificial tipo perceptron múltiplas camadas, utilizando a ferramenta *nntool* do Matlab para implementar e testar a rede. Após o treinamento da rede neural artificial, foram realizadas simulações com a rede treinada, com o objetivo de verificar a precisão e eficiência do treinamento e consequentemente do aprendizado do robô simulado.

3) Preparação da plataforma Curumim e do ambiente estruturado - Após a realização das simulações, iniciou-se a programação do robô Curumim e a preparação do ambiente de treinamento a fim de se realizar os experimentos reais.

4) Experimentos práticos - foram realizados experimentos utilizando a me-

todologia estudada neste trabalho na plataforma do robô Curumim. Isso foi realizado com o intuito de atestar e validar a utilização de RNA's na solução do problema de LfD, e de sua aplicabilidade prática na solução do problema de aprendizagem de novas trajetórias por robôs móveis.

4.1) Fase de demonstração - A primeira etapa experimental realizada em um processo de LfD consiste na demonstração da tarefa a ser ensinada e na coleta dos dados referentes a essa tarefa. No caso em questão, as tarefas correspondem as trajetórias ensinadas ao robô, e os dados coletados correspondem aos dados de distância dos sensores e aos valores de velocidade e distância percorrida por cada motor do robô. Lembrando que, como mencionado na Seção 2.1.3, foram utilizadas apenas as duas rodas paralelas do robô, obtendo com isso 4 valores de saída(2 velocidades e 2 distâncias percorridas). Isso foi feito com o intuito de simplificar os experimentos e consequentemente o treinamento da RNA, levando em conta também que a terceira roda se dá necessária apenas na realização de movimentos laterais e essa habilidade não foi explorada devido a dificuldades mecânicas do robô.

Foram planejados 5 experimentos para aprendizagem de trajetória, cada experimento com um grau diferente de dificuldade. Os experimentos foram planejados de forma a explorar a capacidade de aprendizagem da rede assim como a capacidade de aprendizagem de diferentes situações práticas. Foram exploradas a aprendizagem através de trajetórias não contínuas (Vide Seção 3.2) de curvas com obstáculos frontais, curvas com obstáculos frontais e laterais, situações de parada com obstáculos frontais, como também a realização da correção da orientação inicial do robô. A seguir são ilustradas as trajetórias demonstradas utilizando o robô Curumim durante a realização deste trabalho.

Experimento 1 - A primeira trajetória demonstrada consistiu de andar em linha reta com o robô e parar próximo a parede frontal. Deseja-se verificar com esse experimento a capacidade do robô de parar a uma determinada distância da parede, de forma que diferentes valores de distância podem ser lidos pelos sensores em uma mesma posição, isso ocorre devido a imprecisões dos sensores e também a variações na trajetória de uma imitação para outra. Objetiva-se portanto verificar a capacidade da RNA de parar o robô a uma distância relativamente próxima a distância indicada na fase de demonstração. A Figura 3.2 apresenta uma estimativa da trajetória realizada.

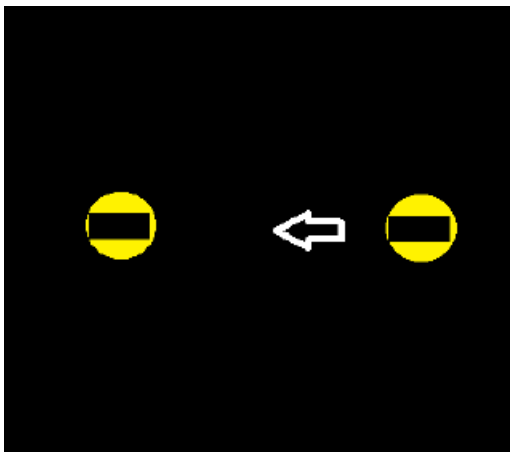

Figura 3.2. Trajetória - Experimento 1

Experimento 2 - O segundo trajeto estudado consistiu de inicialmente realizar uma curva com o robô, que se encontrava deslocado cerca de 45 graus da trajetória pretendida. Após realizar a curva andou-se com o robô até próximo a parede frontal. Objetiva-se com este experimento verificar a capacidade da RNA de aprender três movimentos diferentes (sequência de giro, deslocamento frontal e parada). Os erros de odometria e de leitura dos sensores podem influenciar o processo de imitação de forma que o robô não gire exatamente 45°, afetando com isso a continuidade da trajetória. Dessa forma este experimento

apresenta um nível de dificuldade de aprendizado da RNA maior em relação ao experimento anterior. A figura a seguir ilustra o trajeto realizado.

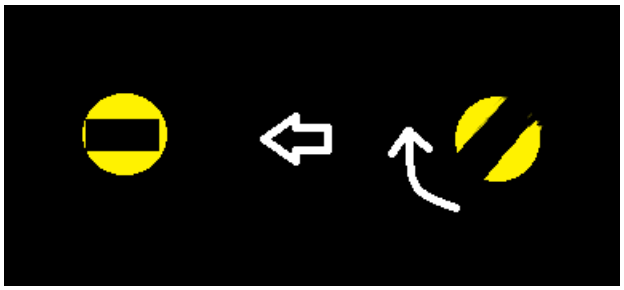

Figura 3.3. Trajetória - Experimento 2

Experimento 3 - A terceira trajetória consiste da realização de um contorno completo no sentido horário no ambiente onde o robô se encontra. Esse contorno foi demonstrado mantendo o robô relativamente próximo da parede a sua esquerda. Este experimento objetiva avaliar a capacidade da RNA de aprender um percurso recursivo, de modo que erros de odometria e dos sensores desempenham uma grande influência na continuidade da trajetória e na repetição do contorno. Além disso, pela trajetória não ser contínua, diversos pontos diferentes da demonstração podem ser encontrados pelos sensores a medida que a trajetória se repete, tornando o aprendizado mais complexo pela RNA. A Figura 3.4 ilustra a trajetória.

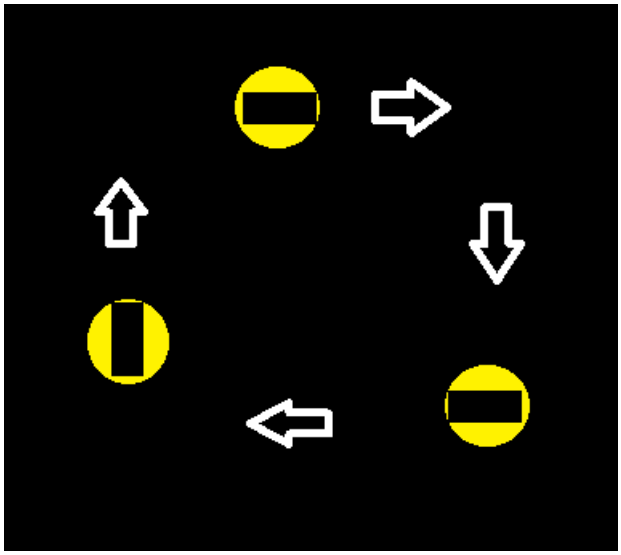

Figura 3.4. Trajetória - Experimento 3

Experimento 4 - A quarta demonstração foi realizada exatamente como a terceira, porém colocou-se um obstáculo no centro do ambiente, como pode ser visto na Figura 3.5. Este experimento visa estudar o comportamento da RNA na situação de contorno com uma menor possibilidade de movimentação devido ao obstáculo central. Isto trás uma maior dificuldade para imitação devido a que um menor erro de odometria é tolerado, pois um erro razoavelmente grande nas curvas pode fazer com que o robô colida com o obstáculo.

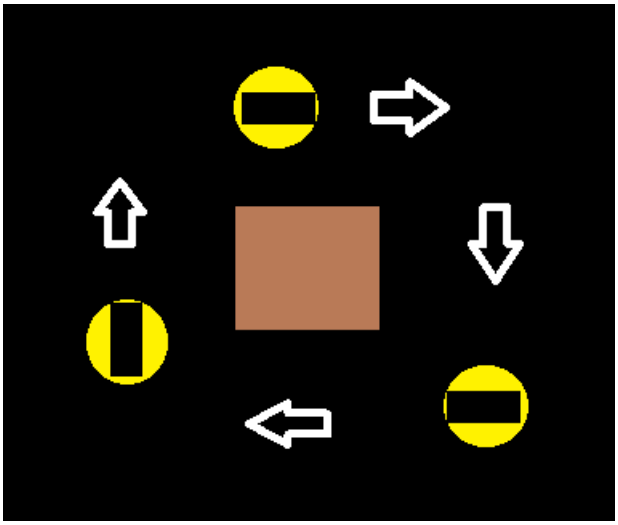

Figura 3.5. Trajetória - Experimento 4

Experimento 5 - A quinta trajetória demonstrada consistiu da realização do contorno do robô sobre o ambiente alternando movimentos para frente e movimentos de curvas de 45 graus. Objetivou-se com este experimento avaliar a capacidade da RNA em alternar suas saídas, de modo que se deseja em suas saídas sempre uma movimentação para frente seguida de uma movimentação de curva. Além disso visa-se com este experimento analisar a capacidade do robô de girar em torno do ambiente considerando seus erros intrínsecos de odometria. Essa trajetória está ilustrada na Figura 3.6.

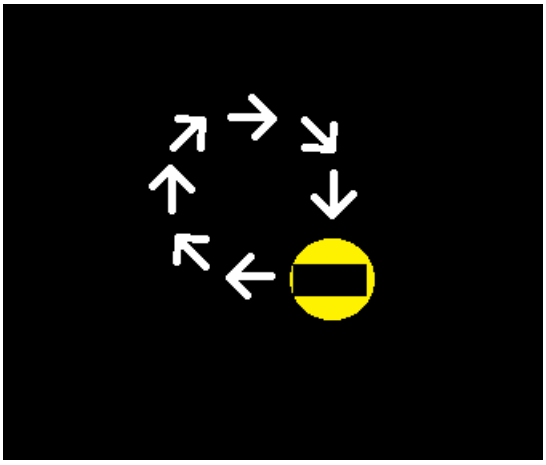

Figura 3.6. Trajetória - Experimento 5

É importante notar que, os experimentos realizados neste trabalho são relativamente simples, de forma que uma lógica condicional ou a utilização de máquinas de estados seria suficiente para solucioná-los. Porém justifica-se a utilização de RNA's na solução desses problemas devido a sua natureza meta-heurística, onde uma meta-heurística é um método utilizado no intuito de resolver problemas de otimização de uma forma genérica. Portanto o objetivo aqui, é o aprendizado de diferentes trajetórias utilizando-se uma única ferramenta, sem a necessidade de uma reconfiguração ou reprogramação do robô.

Como mencionado na Seção 2.1.3, devido a desgastes físicos das rodas do robô Curumim utilizado, a habilidade de andar de lado (característica de um robô omnidirecional) foi prejudicada, acarretando a execução de trajetórias errôneas quando a mesma foi solicitada. Portanto essa habilidade não foi explorada nas trajetórias demonstradas utilizando o robô Curumim.

4.2) Fase de aprendizagem - após a demonstração e coleta dos dados utilizando o robô Curumim, foi realizado o treinamento da RNA através da ferramenta *nntool* do Matlab. Foram testadas diferentes topologias de RNA para cada experimento, sendo que experimentos mais simples necessitaram de uma topologia com um menor número de neurônios em relação a experimentos mais complexos. Espera-se que a topologia mais complexa utilizada, seja capaz de treinar e gerar resultados satisfatórios para todos os experimentos demonstrados.

4.3) Fase de imitação - A fase de imitação consistiu em aplicar os pesos sinápticos treinados no Matlab, na plataforma do robô Curumim. Para isso foi implementada a topologia da RNA utilizada em C++. A fase de imitação objetiva uma verificação visual do comportamento aprendido pelo robô.

5) Análise de resultados - Após a realização das etapas de demonstração, coleta de dados e imitação, foi realizada uma análise subjetiva da trajetória aprendida. Foi realizada também uma análise do erro de treinamento gerado pela ferramenta do Matlab, como também uma comparação dos resultados simulados pela ferramenta e os resultados reais.

6) Documentação do Trabalho - Após a implementação das etapas anteriores, realizou-se o trabalho de documentação do projeto em questão.

3.2 Fase de demonstração e coleta de dados

A primeira etapa realizada no desenvolvimento e treinamento da rede neural artificial, é a de coleta de dados. Esse processo é realizado através da obtenção dos dados dos sensores do robô, como também o armazenamento das velocidades e das distâncias a serem percorridas por cada roda do robô naquele momento em questão. Primeiramente, a coleta de dados foi realizada a nível de simulação, utilizando o simulador EyeSim. A coleta de dados em ambientes físicos foi realizada utilizando o robô Curumim, a partir de um ambiente projetado para tal (vide Fig. 3.8), considerando as limitações físicas do robô.

A coleta de dados foi realizada com o intuito de obter os dados necessários para se realizar o treinamento da rede neural, onde os dados obtidos através dos sensores são os dados de entrada da rede. As velocidades das rodas do robô juntamente com a distância a ser percorrida em milímetros, são os dados de saída desejados pela rede. Esse tipo de procedimento é realizado com o intuito de se criar e fornecer uma base de conhecimento para a rede neural artificial.

É importante destacar que a coleta de dados não é contínua, isso ocorre devido a que a movimentação do robô também não é contínua. O robô se

movimenta a "passos"discretos, definidos pelos parâmetros de distância e velocidade enviados a cada roda do mesmo. Esse procedimento foi adotado levando em consideração o funcionamento das funções disponíveis no ambiente de desenvolvimento do robô.

Um filtro de média de 16 amostras foi implementado com o intuito de filtrar os sinais de entrada dos sensores. Essa medida foi adotada a fim de minimizar possíveis ruídos interferentes no sinal desses sensores. Dessa forma, foram coletadas 16 amostras de dados em cada posição demonstrada, e apenas a média aritmética dessas amostras foi utilizada para treinamento.

Realizando-se a coleta de dados e os treinamentos da RNA, constatou-se a necessidade de coletar mais dados durante movimentos de curva e parada, pois apenas uma coleta em cada um desses pontos não se apresentou suficiente para um correto aprendizado. Considerando isso, foram coletados 5 pacotes de dados a cada movimento(passo) de curva ou parada, e apenas 1 coleta(1 pacote de dados) para movimentos de "caminhar"com o robô.

3.3 Fase de aprendizagem

A fase de aprendizagem corresponde ao processo de adaptação dos pesos sinápticos da rede neural através do treinamento da RNA utilizando os dados previamente coletados pelo robô. Essa etapa é muito importante, pois a precisão do pesos sinápticos gerados determinará a eficiência no aprendizado e execução das tarefas pelo robô. São gerados diferentes pesos sinápticos para diferentes trajetórias ensinadas ao robô, sendo que após a realização do treinamento de determinada tarefa e a geração de seus pesos sinápticos, basta aplicar esses pesos sinápticos no algoritmo de demonstração do robô para que o mesmo

imite a trajetória ensinada.

Alguns fatores que influenciam diretamente nesse aprendizado são a quantidade de dados coletados na fase de demonstração, pois quanto maior o número de dados em um determinado ponto ou situação, mais informação o robô terá sobre que decisão tomar nessa ocasião. Além disso, o tipo e a eficiência na convergência do algoritmo utilizado também representa um papel importante na geração dos pesos sinápticos, e portanto, na qualidade do aprendizado do robô [6].

A Figura 3.7 apresenta um diagrama ilustrando os passos realizados para cada experimento apresentado neste trabalho, mostrando as etapas desenvolvidas desde a demonstração da trajetória a ser ensinada até a imitação dessa trajetória pelo robô.

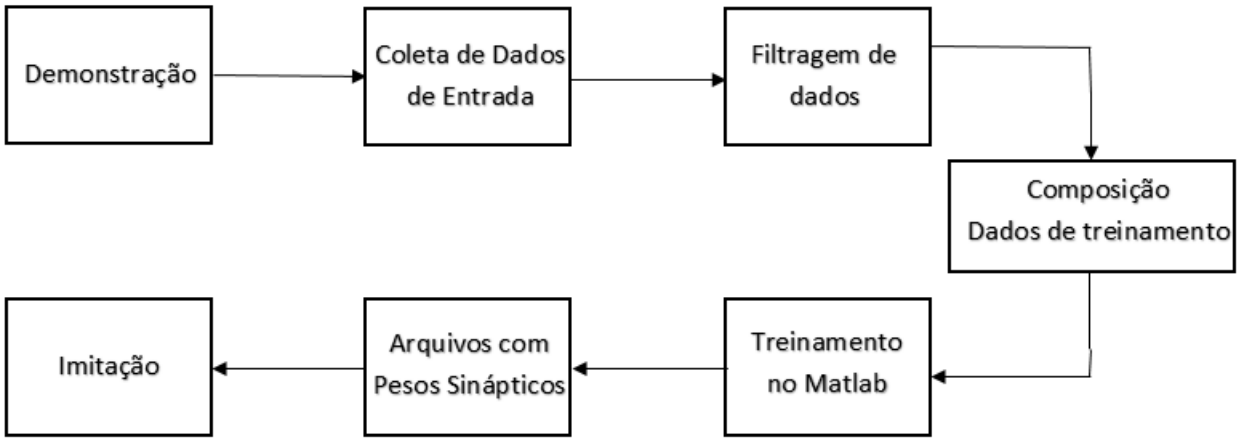

Figura 3.7. Diagrama - Fluxo de Desenvolvimento

3.4 Ambiente de treinamento

O ambiente utilizado para realizar os experimentos com o robô Curumim está ilustrado na Figura 3.8. Foi construído um ambiente retangular, com cerca de 107cm de largura e 11.449cm^2 de área.

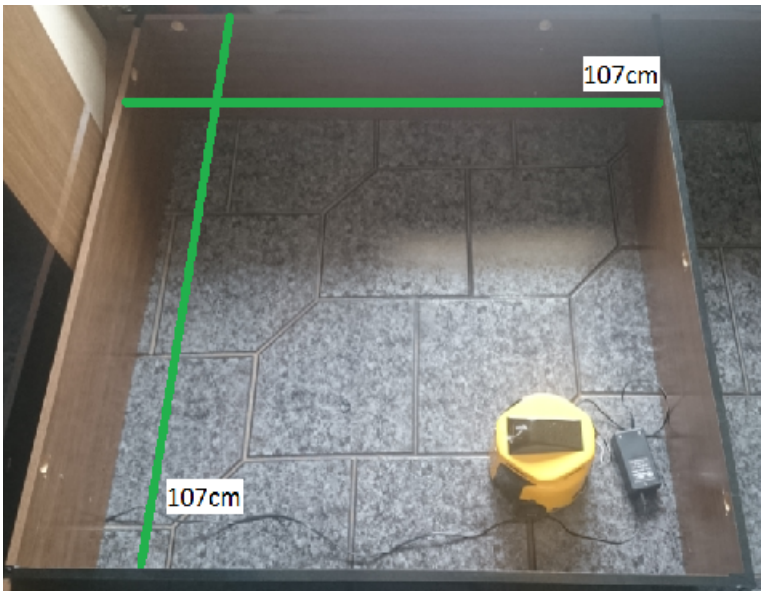

Figura 3.8. Ambiente de treinamento

Esse ambiente foi projetado considerando o alcance dos sensores infravermelhos do robô que variam de 10cm até 86cm,de modo que pode-se observar pela Figura 3.9 que o ambiente foi projetado visando maximizar a utilização desses sensores.

O piso utilizado na realização dos experimentos foi um piso cerâmico. Esse piso apresenta alguns pequenos desníveis, porém esses desníveis não foram suficientes para prejudicar as trajetórias experimentadas.

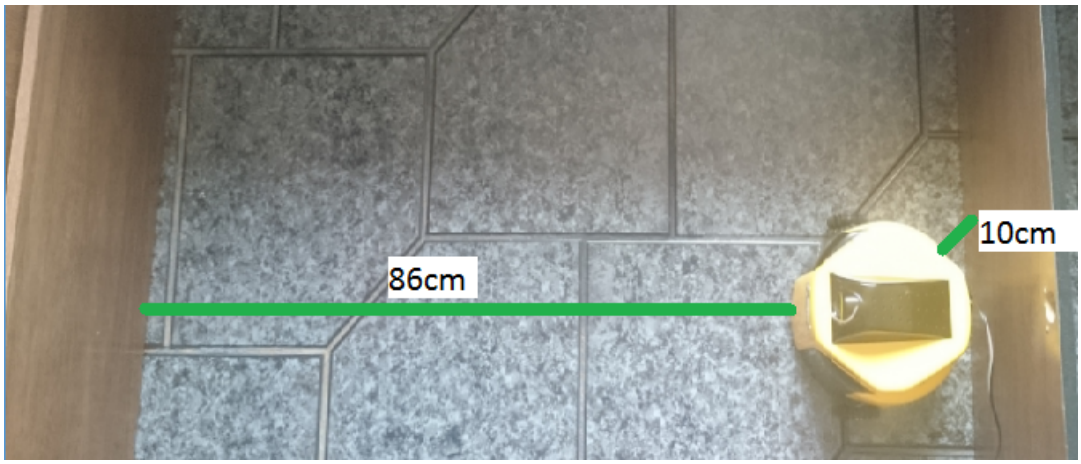

Figura 3.9. Alcance dos sensores infravermelhos

3.5 Plataforma experimental

A plataforma experimental utilizada foi composta de um Notebook com sistema operacional Windows, um robô Curumim e um rádio transmissor/receptor. Foi utilizado no notebook o software SwCurumim, fornecido pela empresa Xbot. Este software foi utilizado como ferramenta de aprendizado para entender o funcionamento das funções utilizadas pelo robô, como também para o entendimento e estabelecimento da comunicação e troca de informações entre o notebook e o robô. Após estabelecido esse conhecimento, foram implementados os algoritmos utilizados na realização do experimento. Estes algoritmos foram implementados em C/C++ utilizando a ferramenta de desenvolvimento Visual C++ 2005 Express Edition, compatível com a plataforma do robô Curumim.

A Tabela 3.1 ilustra as funções utilizadas para implementar o algoritmo responsável pela coleta e armazenamento dos dados de treinamento, como também o algoritmo de imitação.

Onde os parâmetros *dist1,dis2,vel1...,etc.* da função *turn* correspondem a distância a ser percorrida e a velocidade de cada um dos motores do robô.

Tabela 3.1. Funções - Robô Curumim

Função	Parâmetros	Retorno
turn	dist1 dist2 dist3 vel1 vel2 vel3	1
waitStopped		0
distanceSensors	sensor[ ]	true

O parâmetro *sensor[]* da função *distanceSensors* corresponde a um vetor de variáveis onde os valores de distância de cada sensor são armazenados.

A fim de simplificar o treinamento da rede neural, os valores de distância e velocidade de cada motor foram mantidos fixos, variando apenas o sentido de rotação dos motores de acordo com a direção que se deseja que o robô siga. Esses valores foram definidos em $60mm$ para a distância, e $100rpm$ para a velocidade dos motores. Esses valores foram escolhidos visando reduzir o número de movimentos necessários para se realizar uma mudança de orientação do robô. Dessa forma, com os valores utilizados, o robô realiza uma mudança de orientação de cerca de 45 graus em relação a sua orientação original em cada movimento de curva realizado.

4 Resultados

Neste Capítulo serão apresentados os resultados obtidos no processo de implementação do trabalho proposto. Em uma primeira etapa foram realizadas diversas simulações e treinamentos da rede neural artificial e do robô omnidirecional no simulador EyeSim. Posteriormente foram realizadas demonstrações e treinamentos da rede neural a partir de dados coletados utilizando o robô Curumim. Nessa seção é apresentada também uma análise dos pesos sinápticos gerados e seu erro associado, como também uma comparação com os resultados práticos obtidos.

4.1 Resultados de simulação - Simulador EyeSim

A simulação utilizando o simulador EyeSim, consistiu em realizar diferentes trajetórias com o robô, onde o controle dos movimentos foi realizado através do teclado do computador. Foram coletados dados em diferentes posições e trechos percorridos pelo robô, onde esses dados foram armazenados em um arquivo de texto e depois utilizados no treinamento da rede neural.

A quantidade de dados coletados está relacionada a trajetória demonstrada. Dessa forma, uma trajetória mais simples necessita de menos dados para realizar a aprendizagem, enquanto uma trajetória mais elaborada pode precisar de mais dados para estabelecer um correto aprendizado. A Tabela 4.1 apresenta a quantidade de dados coletados para cada trajetória realizada em simulação, ilustradas na Seção 3.1.

Tabela 4.1. Coleta de dados - EyeSim

Experimento	Quantidade de dados
1	25
2	67
3	1470

A seguir são apresentados os resultados das trajetórias simuladas:

1) Linha reta próximo a parede - O primeiro experimento consistiu em andar reto com o robô e parar próximo a parede frontal. Este experimento apresentou resultados satisfatórios em relação ao aprendizado do robô. Pode se observar os trajetos de demonstração(a) e imitação(b) na Figura 4.1.

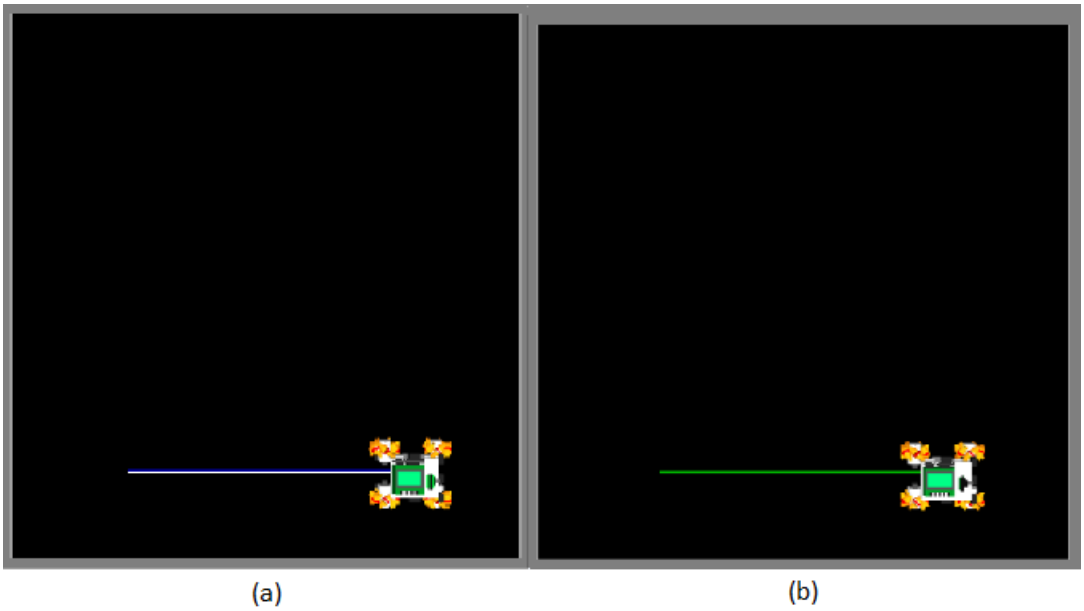

Figura 4.1. Resultados experimento 1 - (a)Trajetória ensinada, (b)Trajetória aprendida

Para este experimento foram utilizados apenas três dados de entrada dos sensores (frontal, traseiro e diagonal esquerdo), pois os outros sensores não sofrem alteração durante o percurso em questão. Portanto a rede neural consistiu de 3 neurônios de entrada, 3 neurônios na camada escondida, e 4 neurônios de saída.

2) Corredor - A Figura 4.2 apresenta os trajetos de demonstração(a) e imitação(b) realizados para este experimento. O experimento apresentou resultados satisfatórios no sentido de que o robô foi capaz de realizar (aprender) a tarefa desejada. Porém é interessante notar que a trajetória aprendida difere razoavelmente da trajetória ensinada. Isso pode acontecer por que durante a etapa de imitação, a rede neural está susceptível a novas entradas, que em alguns casos diferem das entradas utilizadas no treinamento. Com isso é razoavelmente aceitável e esperado que a saída da rede treinada seja diferente da saída demonstrada em alguns pontos, desde que o resultado final esperado, ainda seja alcançado pelo robô.

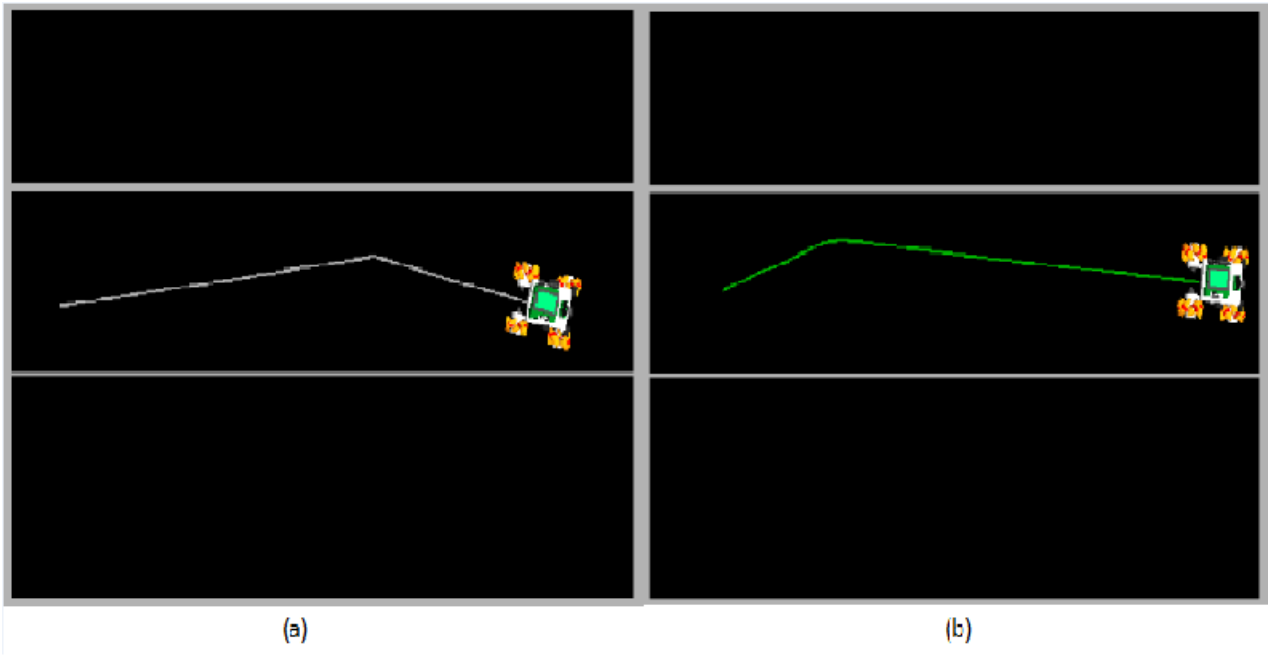

Figura 4.2. Resultados experimento 2 - (a)Trajetória ensinada, (b)Trajetória aprendida

3) Contorno - Os resultados do terceiro experimento foram bem sucedidos, observou-se porém, durante as fases de coleta de dados e de treinamento da rede, que a quantidade dos dados coletados influenciou significativamente na resposta do robô. Alguns comportamentos inesperados foram observados quando insuficientes dados foram coletados para treinamento (vide Fig 4.3).

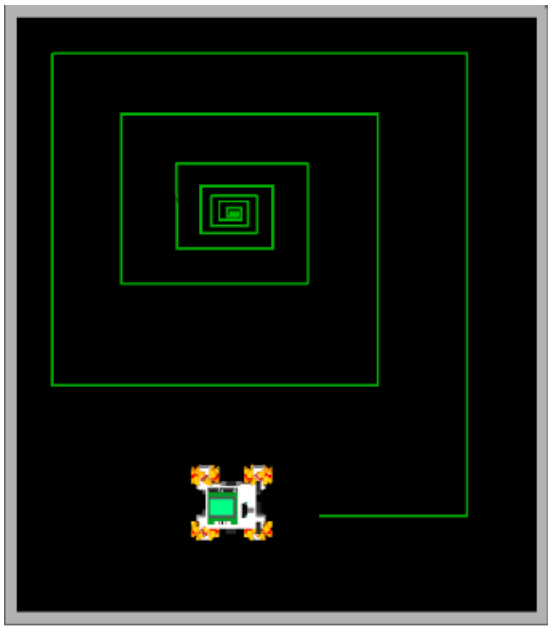

Figura 4.3. Resultado inesperado após treinamento da RNA

Resultados satisfatórios foram alcançados utilizando-se cerca de mil dados para treinamento, coletados em diferentes trechos do percurso ensinado. A Figura 4.4 ilustra os percursos de demonstração e imitação realizados.

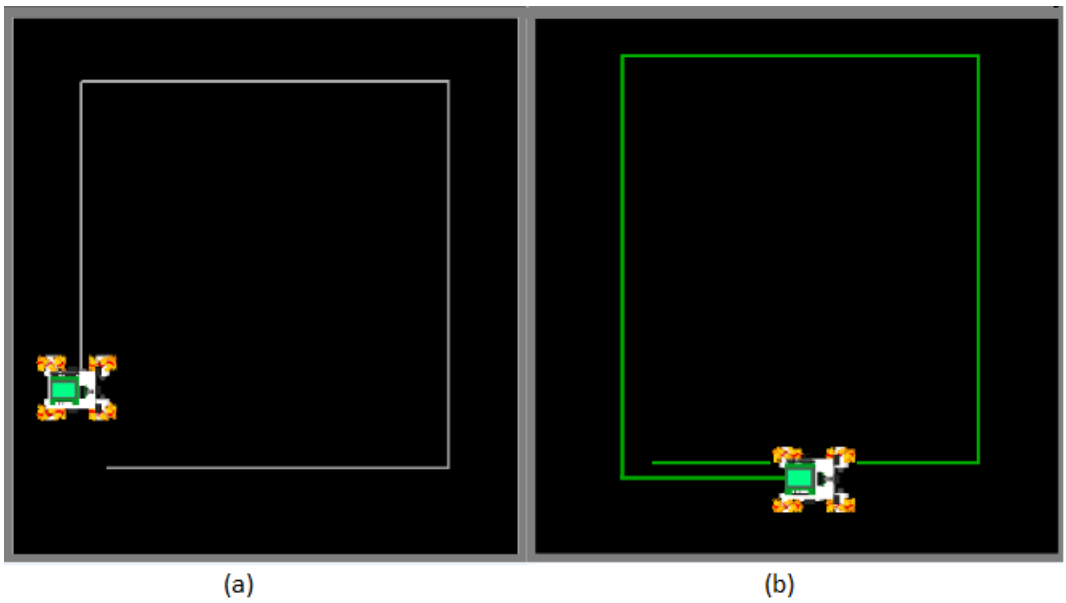

Figura 4.4. Resultados experimento 3 - (a) Trajetória ensinada (demonstração), (b) Trajetória aprendida (imitação).

As simulações foram importantes no sentido de permitir um primeiro aprendizado prático da aplicação da metodologia de LfD utilizando RNAs. Com essas simulações pôde-se ter uma melhor noção da capacidade da metodologia aplicada, como também verificar alguns fatores de maior relevância como a importância da quantidade de dados coletados na convergência do aprendizado pela RNA. É notável que muitos fatores como erros de odometria e dos sensores não são considerados nas simulações, tanto como possíveis desvios de trajetória devido a imperfeições do ambiente. Porém mesmo sabendo que as simulações apresentam resultados simplificados em relação ao mundo real, essa prática possibilitou um aprofundamento do aprendizado da metodologia de LfD juntamente com suas etapas de treinamento e configuração da rede neural artificial.

4.2 Coleta de dados utilizando o robô Curumim

A coleta dos dados de demonstração do robô Curumim foi realizada através da utilização das funções pré-definidas do robô. A quantidade de dados coletados varia com a trajetória demonstrada, de modo que em geral uma trajetória mais simples necessita de menos dados coletados para ser aprendida em relação a uma trajetória mais complexa. Cada coleta corresponde aos valores de distância dos 5 sensores do robô em determinada posição, dos valores da distância a ser percorrida e da velocidade de cada motor do robô.

A Tabela 4.2 a seguir apresenta a quantidade de dados coletados para cada trajetória realizada com o robô (descritas na Seção 3.1).

Tabela 4.2. Coleta de dados - Robô Curumim

Experimento	Quantidade de dados
1	11
2	25
3	61
4	57
5	14

4.3 Treinamento da rede neural no Matlab

O trabalho de treinamento da rede neural artificial foi realizado utilizando a ferramenta *nntool* do Matlab. Essa ferramenta permite a criação de diferentes redes neurais, o treinamento da mesma para a geração dos pesos sinápticos, e também a simulação dos resultados obtidos pela rede após o treinamento.

Dependendo da trajetória a ser ensinada, como também das variações nos valores de saída, o número de neurônios na camada escondida foi modificado com o intuito de se obter uma correta resposta do treinamento. Dessa forma, foram testadas diferentes topologias para cada trajetória ensinada, visando sempre a

obtenção de um resultado satisfatório utilizando o mínimo de recursos (neurônios) da rede.

A Figura 4.5 ilustra a estrutura das topologias de RNA's utilizadas na realização dos treinamentos. Sendo que os neurônios de entrada e saída permanecem fixos durante os treinamentos com valores 5 e 4 respectivamente, enquanto os neurônios da camada escondida foram ajustados para cada trajetória objetivando garantir um correto aprendizado. Os valores de entrada e saída foram mantidos fixos pois correspondem aos valores dos sensores (5) e aos valores de velocidade e distância percorrida (4), respectivamente.

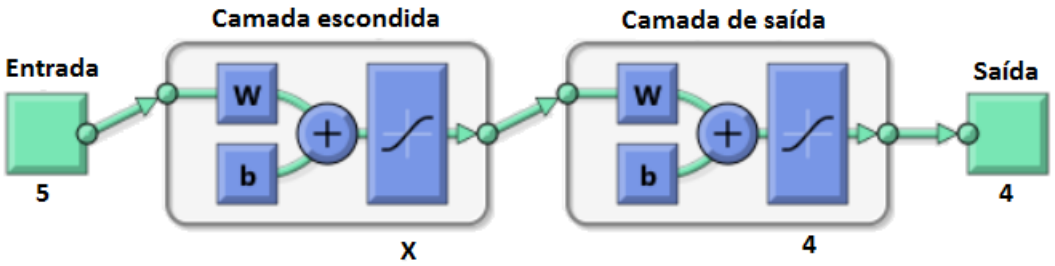

Figura 4.5. Topologias de RNA's - Matlab

A Tabela 4.3 apresenta as topologias de RNA utilizadas no treinamento de cada experimento que resultou em aprendizados satisfatórios das trajetórias, juntamente com o erro associado a geração dos pesos sinápticos de cada rede. É notável que a topologia mais complexa implementada utiliza 5 neurônios na camada escondida, dessa forma todos os experimentos podem ser realizados utilizando essa mesma topologia. Essa prática é realizada objetivando a utilização de uma mesma topologia para todos os experimentos, mesmo concluindo que alguns dos experimentos apresentam já bons resultados utilizando menos neurônios.

Tabela 4.3. Topologias de RNA's

Experimento	Topologia utilizada	Erro
1	5-2-4	1.6561e-08
2	5-3-4	2.8833e-12
3	5-5-4	2.2204e-10
4	5-5-4	0.048633
5	5-3-4	5.814e-09

Os resultados gerados após o treinamento da rede neural foram primeiramente validados através da própria ferramenta *nntool* no Matlab, onde se é possível simular a saída da rede neural utilizando os pesos sinápticos gerados no treinamento. Além disso é possível visualizar um gráfico de performance gerado pela ferramenta, onde 70% dos dados coletados são utilizados no treinamento da rede, e 30% dos dados são para validação e testes. É possível verificar a convergência do erro através do gráfico, onde o mesmo diminui com o aumento das iterações, como também o ponto de melhor performance, representado por um círculo no gráfico. A Figura 4.6 apresenta os gráficos de performance para cada um dos experimentos realizados.

As RNA's foram treinadas utilizando-se um algoritmo de treinamento do tipo FeedFoward backpropagation, utilizando como critérios de parada o número de iterações, estabelecido como mil, como também o número de testes de validação, definido no valor de seis. Esses valores são definidos como padrão pela ferramenta *nntool* do Matlab.

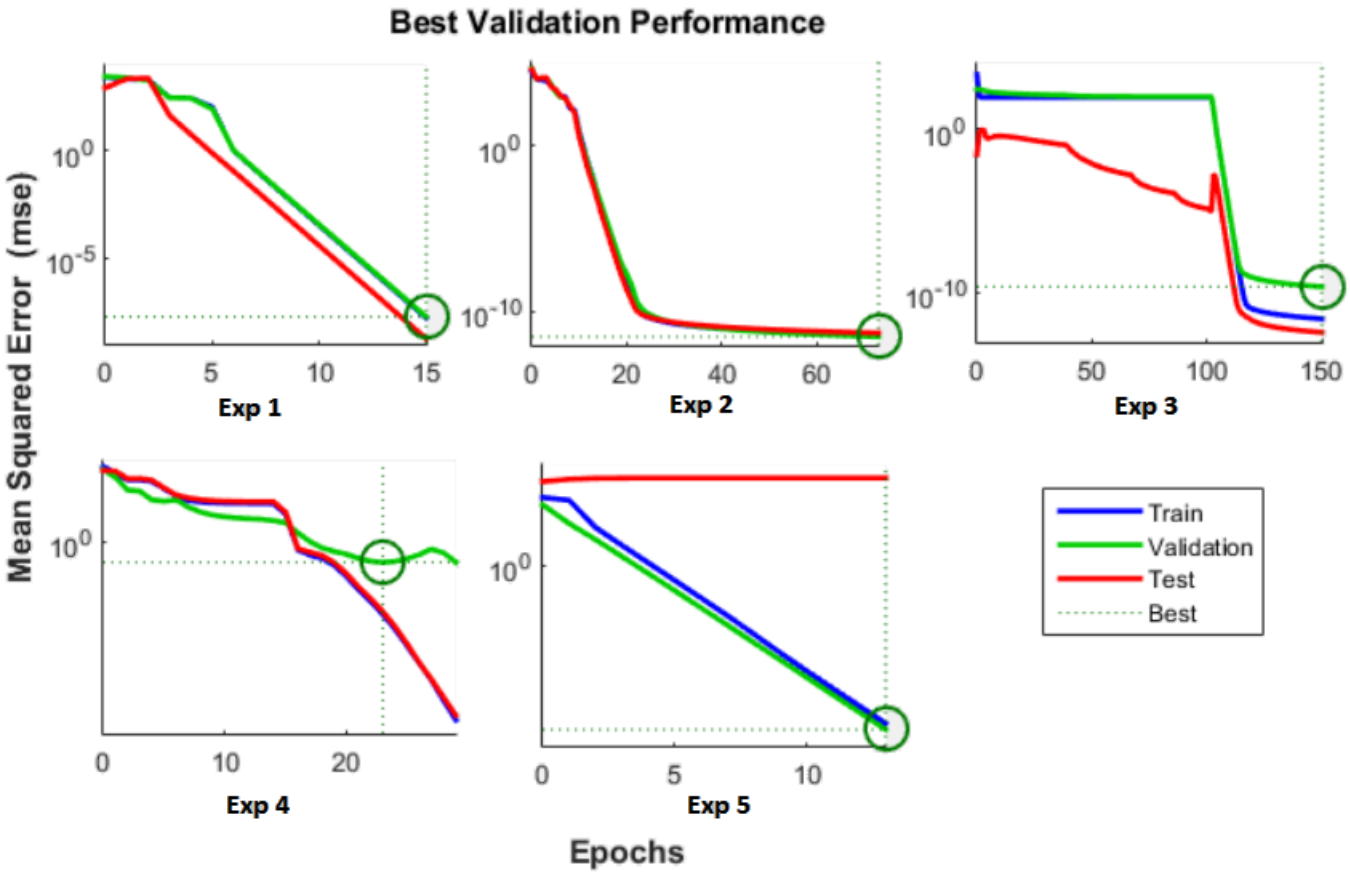

Figura 4.6. Gráfico de performance de treinamento - Matlab

4.4 Resultados Robô Curumim

Nesta seção serão apresentados os resultados obtidos com o treinamento da rede neural artificial aplicados na plataforma do robô Curumim.

Experimento 1 - Os resultados de imitação do experimento 1 foram satisfatórios utilizando 2 neurônios na camada escondida da RNA. Para 1 neurônio, o robô apresentou bons resultados simulados no matlab. Porém quando os pesos sinápticos gerados foram utilizados no algoritmo de backpropagation aplicado ao robô, o mesmo apresentou dificuldades para parar, continuando por andar reto indefinidamente. O treinamento foi também aplicado para topologias com 3 e 5 neurônios na camada escondida, ambos apresentaram bons resultados. A

figura a seguir ilustra uma estimativa da trajetória aprendida.

Figura 4.7. Trajetória - Experimento 1

Experimento 2 - O segundo experimento apresentou resultados interessantes a partir da utilização de 5 neurônios na camada escondida, executando a curva inicial de acordo com a demonstração. Porém o treinamento apresentou dificuldades em ensinar o robô a parar próximo a parede, ao final da trajetória. Foram realizados diversos treinamentos, utilizando 1,2,3,5,8 e 10 neurônios na camada escondida. A partir da utilização de 5 neurônios, a imitação da curva foi bem executada, porém a situação de parada não foi aprendida pelo robô.

Foram coletados a princípio 5 lotes de dados com informações da curva demonstrada, 1 lote de dados para cada movimento para frente demonstrado e 5 lotes de dados com informações da demonstração de parada próximo a parede frontal. Após detectado o problema do aprendizado da condição de parada, foram realizadas mais 5 coletas e posteriormente mais 10 coletas de dados com informações da demonstração de parada, porém ainda sim um resultado satis-

fatório não foi obtido utilizando o algoritmo de RNA utilizado no robô. Uma estimativa da trajetória demonstrada e imitada é ilustrada na Figura 4.8.

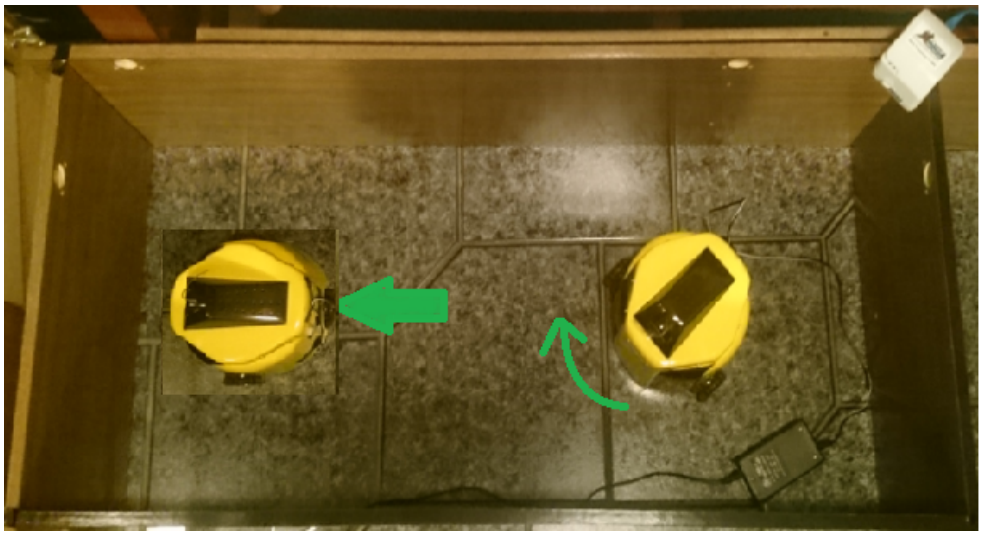

Figura 4.8. Trajetória - Experimento 2

Experimento 3 - Este experimento apresentou bons resultados no Matlab, porém a trajetória demonstrada foi imitada pelo robô com algumas ressalvas. O robô realizou o contorno no ambiente, porém apresentou dificuldade em seguir a trajetória paralela as paredes do contorno, como a princípio foi ensinado.

A fim de testar a capacidade de aprendizagem da RNA tipo perceptron utilizada, foram realizados dois tipos de treinamentos, o primeiro utilizando dados coletados em toda a trajetória demonstrada, e um segundo utilizando apenas metade desses dados, correspondendo a metade do percurso. Notou-se com esses treinamentos, que a utilização de apenas metade dos dados foi suficiente no que se refere ao aprendizado da trajetória ensinada ao robô. A Figura 4.9 ilustra uma estimativa das trajetórias demonstrada e aprendida.

Figura 4.9. Estimativa Exp 3 - (a)Trajetória demonstrada, (b)Trajetória aprendida

Experimento 4 - O quarto experimento não apresentou resultados satisfatórios, apesar de bons resultados simulados terem sido apresentados no Matlab. O robô aprendeu a realizar a primeira curva demonstrada, porém após essa situação a adaptação dos pesos sinápticos gerados não apresentou uma boa resposta, de modo que em algumas imitações o robô colidiu com o obstáculo central e em outras simplesmente gerou saídas errôneas(passou a girar em círculos). É interessante notar que, apesar da ferramenta no Matlab ter apresentado bons resultados de treinamento, este experimento foi o que apresentou um maior erro associado a sua resposa gerada em relação aos outros experimentos (vide Tabela 4.3). Mesmo o erro gerado sendo pequeno, diferenças entre o algoritmo de RNA utilizado no robô e o algoritmo do Matlab,utilizados para a convergência dos dados a partir dos pesos sinápticos, acaba por dificultar o aprendizado do robô em alguns casos, como foi obervado neste experimento. A figura a seguir ilustra a estimativa da trajetória demonstrada neste experimento.

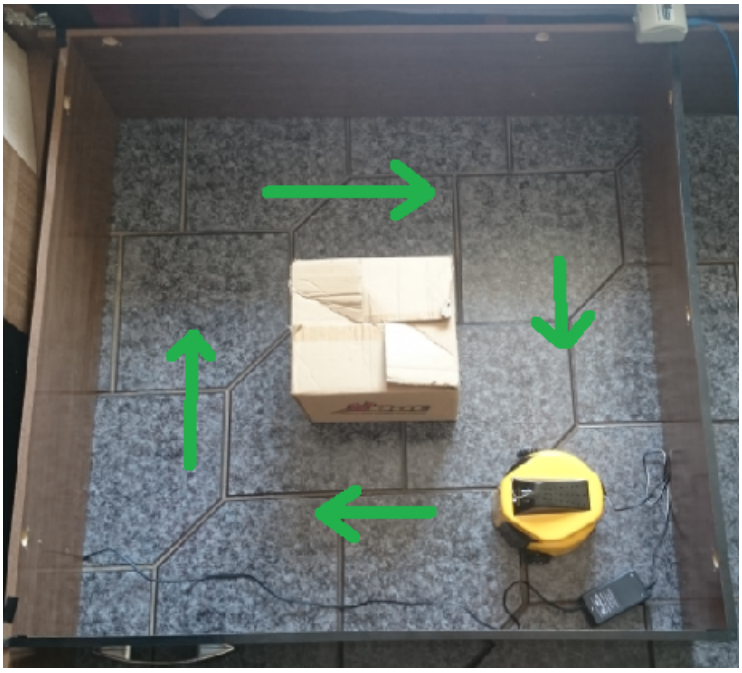

Figura 4.10. Trajetória - Experimento 4

Experimento 5 - A quinta trajetória estudada consistiu de alternar movimentos de andar para frente e virar 45 graus para direita. Foram coletados dados até o completo contorno do ambiente realizando esses movimentos. O aprendizado se deu satisfatoriamente no sentido de que o trajeto foi aprendido, porém a trajetória sofreu variações devido que essa alternância de movimentos de andar para frente e virar 45 graus não foi observada. O que ocorreu foi a realização de seguidas curvas de 45 graus, fazendo com que o robô completasse o contorno, porém seguindo uma trajetória de raio menor que a demonstrada.

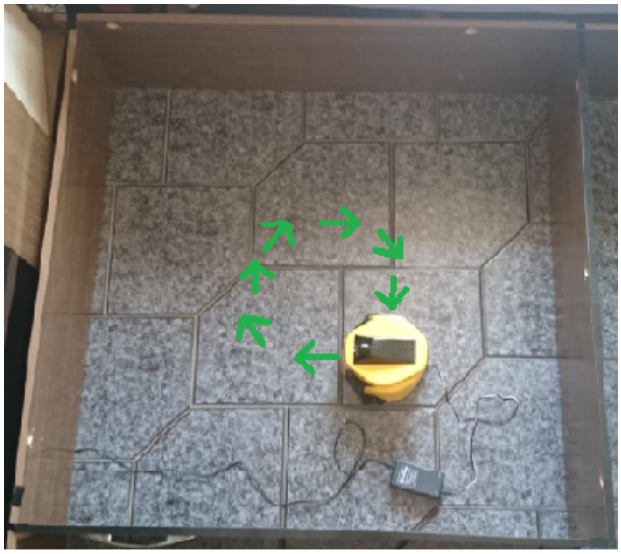

Figura 4.11. Trajetória - Experimento 5

Notou-se que alguns experimentos apresentaram desvios em suas trajetórias de algumas imitações para outras, utilizando os mesmos pesos sinápticos. Isso se deu pela leve irregularidade do piso em alguns pontos, como também as imprecisões intrínsecas do sistema do robô.

Durante a realização dos experimentos, notou-se que alguns resultados simulados no Matlab não condizeram com os resultados observados no robô. Em alguns casos foi necessária a realização de treinamentos utilizando mais neurônios do que se mostrava necessário pela ferramenta do Matlab, de modo que, por mais que o Matlab apresentasse um bom resultado de saída, o mesmo não acontecia com o robô. Esse tipo de ocorrência se deu por diferenças entre o algoritmo de treinamento de redes neurais artificias utilizado pelo Matlab, e o algoritmo implementado de propagação dos pesos sinápticos na rede neural do robô.

5 Conclusões e Trabalhos futuros

A partir do estudo feito sobre LfD e RNA's e dos experimentos realizados neste trabalho, chegou-se às seguintes conclusões:

O trabalho apresentou bons resultados considerando que o mesmo tinha como objetivo o aprendizado e a aplicação do método de aprendizagem por demonstração utilizando redes neurais artificiais em uma plataforma robótica móvel.

Observou-se a partir dos resultados obtidos, que o algoritmo de RNA implementado é relativamente simples, e apresentou algumas dificuldades no completo aprendizado de algumas das trajetórias demonstradas, como explicitado no Capítulo 4. Conclui-se desse tipo de resultado que, o fato de a ferramenta *nntool* do Matlab não permitir acesso ao seus algoritmos de treinamento, dificulta uma exata reprodução de seus resultados gerados. No entanto, resultados de aprendizagem satisfatórios ainda foram alcançados utilizando o algoritmo de RNA implementado juntamente com os pesos sinápticos gerados no Matlab, principalmente para trajetórias relativamente simples.

Os experimentos estudados neste trabalho foram realizados utilizando diversas topologias de RNA, objetivando ao final a utilização de uma única topologia geral. O experimento que necessitou de uma topologia com um maior número de neurônios para apresentar bons resultados foi o experimento 3, que necessitou de 5 neurônios na camada escondida. Os demais experimentos que apresentaram bons resultados, o fizeram com um número menor que 5 neurônios na camada escondida. Desta forma, todos os experimentos realizados podem ser

modelados utilizando a topologia de rede neural artificial com 5 neurônios na camada escondida.

Considerando as conclusões acima, é deixado como sugestão para trabalhos futuros o desenvolvimento do algoritmo de treinamento de redes neurais artificiais para a geração de seus pesos sinápticos. Dessa forma, a utilização da ferramenta no Matlab não será necessária, e o completo conhecimento do algoritmo utilizado para treinamento possibilitará uma melhor adaptação e aplicação dos pesos sinápticos da rede. Outro trabalho interessante é a implementação do ROS *(Robot Operating System)* para a plataforma do robô Curumim, pois através da utilização do ROS pode-se criar as próprias funções de controle e movimentação do robô com maior facilidade, pode-se utilizar por exemplo uma movimentação contínua do robô, através da implementação de uma função própria para isso. Outro trabalho interessante é a utilização de uma rede neural realimentada. A utilização de realimentação permite com que seja possível saber a localização do robô a partir de sua posição anterior, isso possibilita a utilização de alguns outros movimentos, como o movimento de ré, que só é praticável a partir de uma RNA realimentada.

Conclui-se por fim que os objetivos propostos do projeto foram alcançados, tais como o estudo e aprendizado teórico de redes neurais, como também o aprendizado e utilização de ferramentas de simulação,implementação e treinamento de redes neurais artificiais aplicadas à robótica móvel.

Referências Bibliográficas

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[2] T. Braünl. *Embedded robotics - 2 Ed., pages 454*. 2006.

[3] G. Dudek. *Computational Principles of Mobile Robotics - Gregory Dudek, Michael Jenkin - Google Books*. 2010.

[4] S. Ekvall and D. Kragic. Robot learning from demonstration: A task-level planning approach. *International Journal of Advanced Robotic Systems*, 5(3):223–234, 2008.

[5] M. Farber. Topology and Robotics - Artigo, pages 215, 2007.

[6] S. Haykin. *Redes Neurais - Princípios e Prática - 2 Ed, Bookman, pages 937*. 2001.

[7] S. Haykin. *Neural Networks and Learning Machines - 3 Ed, Bookman, pages 902*. 2008.

[8] V. G. Junior. Arquitetura Híbrida para Robôs Móveis Baseada em Funções de Navegação com Interação Humana - Tese. 2006.

[9] F. S. Osório, D. F. Wolf, and K. R. L. J. C. Branco. Robôs Móveis e Veículos Autônomos : Pesquisa , Desenvolvimento e Desafios na área da Inteligência Artificial - Artigo, pages 15 . 2010.

[10] L. Pereira, O. Chase, and E. Sobrinho. Método de desenvolvimento de um robô móvel diferencial didático. *Engenharia de Computação em Revista, 2009*, pages 0–4, 2009.

[11] H. Secchi. Robôs Móveis - Artigo, pages 79 . 2008.

[12] T. H. Silva, P. O. S. V. D. Melo, J. M. Almeida, A. C. Vianna, J. Salles, and A. a. F. Loureiro. Definição , Modelagem e Aplicações de Camadas de Sensoriamento Participativo. pages 353–366, 2014.

[13] C. D. O. Sul. Fábio de almeida guimarães, Desenvolvimento de robô móvel utilizado para a exploração de ambientes hostis - DissertaÃ§Ã£o - pages 245. 2007.

[14] Xbot. Apostila de Software- Robodeck. pages 1–143, 2012.

[15] C. Xbot. Apostila de hardware v2.0. pages 1–18, 2012.