Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Mecânica

CONTROLE BASEADO EM COMPORTAMENTO APLICADO A ROBÓTICA MÓVEL USANDO HARDWARE RECONFIGURÁVEL

Mario Andrés Pastrana Triana Autor

DISSERTAÇÃO DE MESTRADO

PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS MECATRÔNICOS

Brasília

2022

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Mecânica

CONTROLE BASEADO EM COMPORTAMENTO APLICADO A ROBÓTICA MÓVEL USANDO HARDWARE RECONFIGURÁVEL

Mario Andrés Pastrana Triana Autor

Dissertação de Mestrado submetida ao Departamento de Engenharia Mecânica da Universidade Brasília como parte dos requisitos necessários para a obtenção do grau de Mestre

Orientador: Prof. Dr. Daniel Mauricio Muñoz Arboleda

Brasília
2022

Autor, Mario Andrés Pastrana Triana.
A769c CONTROLE BASEADO EM COMPORTAMENTO APLICADO A ROBÓTICA MÓVEL USANDO HARDWARE RECONFIGURÁVEL / Mario Andrés Pastrana Triana Autor; orientador Daniel Mauricio Muñoz Arboleda. -- Brasília, 2022.
88 p.

Dissertação de Mestrado (Programa de Pós-Graduação em Sistemas Mecatrônicos) -- Universidade de Brasília, 2022.

1. Aprendizagem por demonstração. 2. Robótica Móvel. 3. Redes neurais artificiais. 4. Soc FPGA. I. Muñoz Arboleda, Daniel Mauricio, orient. II. Título

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Mecânica

CONTROLE BASEADO EM COMPORTAMENTO APLICADO A ROBÓTICA MÓVEL USANDO HARDWARE RECONFIGURÁVEL

Mario Andrés Pastrana Triana Autor

Dissertação de Mestrado submetida ao Departamento de Engenharia Mecânica da Universidade Brasília como parte dos requisitos necessários para a obtenção do grau de Mestre

Trabalho aprovado. Brasília, 15 de setembro de 2022:

Prof. Dr.Daniel Muñoz Arboleda,
UnB/FGA
Orientador

Prof. Dr.Carlos Llanos Quintero,
ENE/UnB
Examinador interno

Prof. Dr. Leandro dos Santos Coelho,
PUCPR
Examinador externo

Prof. Dr. Jones Yudi Alves da silva,
ENM/UNB
Suplente

Brasília
2022

Este trabajo es dedicado a mi pueblo Albania-Caquetá,
un pueblo pequeño, con un potencial grande.

Agradecimentos

Agradezco a Dios padre todo poderoso por ayudarme con el entendimiento y la sabiduría para llegar hasta este momento, a la virgen María por cubrirme con su manto y acompañarme en todo momento y en todo lugar, también a todos los santos de manera especial a San Agustín al cual me he encomendado en todos mis trabajos y el cual me ha ayudado en los procesos de discernimiento. Agradezco también de manera especial al espíritu santo, el cual me ha dado momentos de iluminación a lo largo de este caminar.

Adicionalmente les doy las gracias de manera especial al Padre Floresmiro Pastrana Medina el cual siempre me invito a apreciar el conocimiento por encima de cualquier cosa. Gracias a mi Madre por el apoyo económico que me ha brindado a lo largo de mi vida, sin el cual no podría estar donde me encuentro, además de sus oraciones las cuales siempre me han protegido y me han ayudado a tener aquellos momentos de clareza que han sido fundamentales para realizar esta maestría.

Agradezco de manera muy especial al Doctor Sergio Andrés Pertuz Mendez, el cual fue el primero en darme la oportunidad para demostrar mis capacidades en la Universidad de Brasilia, fue mi primer apoyo y la primera persona que me ayudó en este país extranjero que ahora llamo mi segunda patria. Agradecer también al Doctor Oscar Eduardo Anacona, con el cual filosofábamos en el laboratorio y me acompaño con mis primeros pasos en el ámbito de las FPGA.

Por último, agradecer a todos los profesores que acompañaron este camino de formación, de manera especial al profesor Carlos Llanos, y de manera muy especial al profesor Daniel Mauricio Muñoz Arboleda el cual además de mi orientador, lo considero como un Padre en este camino de maestría, todas sus sugerencias para mi eran palabras de Dios, y bueno, gracias a él logramos sacar este proyecto adelante, que al comienzo estaba lleno de penumbras y hoy son solo un recuerdo valioso en este camino por adquirir conocimiento.

Gracias Juancho (mi hermano), todo bien, todo bonito.

Resumo

A navegação de robôs moveis é uma área de pesquisa com diferentes desafios tecnológicos, desde desenvolvimento mecânico, sensoriamento, até em termos de processamento computacional. Em particular, o controle de robôs móveis envolve sistemas complexos que para ser modelados é necessário ter conhecimentos em áreas correlatas a sistemas de controle. A metodologia de aprendizagem por demonstração permite projetar controladores mediante a imitação de comportamentos. Esta área é relevante devido a capacidade de solucionar problemas complexos de controle de robôs e às facilidades que a metodologia apresenta. Entretanto, a aplicação desta metodologia requer sistemas embarcados com uma adequada capacidade de processamento e baixo consumo energético. Nesse sentido, poucas pesquisas têm sido desenvolvidas visando explorar soluções em *hardware* no intuito de acelerar os algoritmos envolvidos no processo de treinamento da metodologia de aprendizagem por demonstração. Esta dissertação descreve o desenvolvimento da metodologia de aprendizagem por demonstração embarcada em hardware reconfigurável baseado num sistema em chip (SoC do inglês *Sistem on Chip*), aplicado a robótica móvel. Para o processo de aprendizagem foi usada uma rede neural de camada simples com capacidade de adaptação e para o treinamento da mesma foi usado o algoritmo de otimização por enxame de partículas. Para realizar uma validação metodológica foi desenvolvida uma plataforma robótica móvel de tração diferencial de pequeno porte chamada *Maria*, que esta constituída por um SoC FPGA (*Field Programmable Gate Array*), quatro sensores de distancia infravermelhos, dois encoders, dois micro-motores, entre outros componentes. Seguidamente, três micro-comportamentos foram ensinados, a saber, avançar para frente, girar no sentido horário e girar no sentido anti-horário. Apos o processo de treinamento, o estágio de imitação foi implementado e dezesseis experimentos foram conduzidos para cada um dos micro-comportamentos, obtendo dados estatísticos de cada comportamento ensinado. Adicionalmente, um protocolo experimental foi proposto para testar o robô em cenários desconhecidos, afim de avaliar as capacidades de abstração da rede neural adaptativa. O erro de trajetória para o micro-comportamento girar no sentido horário foi de 4.830cm^2, com um 100% de sucesso, do micro-comportamento girar no sentido anti-horário foi de 6.872,4cm^2, com um 75% de sucesso e 25% de falhas, e o erro para o micro-comportamento de avançar para frente foi de 202cm, com um 81,25% de sucesso e um 18,75% de falhas. Adicionalmente o consumo total de recursos da implementação da metodologia em *hardware* foi de 13.662 *LookUp Table*, 12 processadores digitais de sinais e 8.993 *Flip Flops*, uma dissipação de potencia de 1,336W com um tempo de execução para cada micro-comportamento de 9,1344s.

Palavras-chave: Aprendizagem por demonstração. Robótica Móvel. Redes neurais artificiais. Soc FPGA.

Abstract

The navigation of mobile robots is a research area with different technological challenges, from mechanical development, sensing, and even in terms of computational processing. In particular, the control of mobile robots involves complex systems that, to be modeled, it is necessary to have experience in control systems. The learning from demonstration methodology allows designing controllers by imitating behaviors. This area is gaining strength because of its ability to solve complex control problems and its easy implementation. However, this methodology requires the development of embedded systems with good computational performance and low energy consumption. In this sense, the community has not conducted research projects to explore reconfigurable hardware to accelerate the algorithms involved in the training process. This work describes the development of the learning from demonstration methodology using embedded hardware applied to mobile robotics. A single-layer neural network with adaptiveness was used along with the particle swarm optimization algorithm were used for the learning process. To carry out a methodological validation, a small mobile robotic platform of differential traction called *Maria* was developed, which is constituted by a SoC FPGA (Field Programmable Gate Array), four infrared distance sensors, two encoders, two micro-motors, among others. components. Three micro-behaviors were taught move forward, rotate clockwise, and rotate counter-clockwise, After the training process, the imitation stage was implemented and sixteen experiments were conducted for each of the micro-behaviours, obtaining statistical data for each taught behavior.Additionally, an experimental protocol was proposed to test the robot in unknown scenarios, in order to evaluate the abstraction capabilities of the adaptive neural network.The trajectory error for the microbehavior to rotate in the direction clockwise was 4.830cm^2 with a 100% success, the micro-behavior turning counterclockwise was 6.872,4cm^2 with a 75% success and 25% failure, and the error for the micro-behavior moving forward was 202cm with an 81,25% success rate and an 18,75% failure rate.Additionally, the total resource consumption of the methodology implementation in hardware was 13.662 LookUp Table, 12 digital signal processors and 8.993 Flip Flops, a power dissipation of 1,336W with a runtime for each micro-behavior of 9,1344s

Keywords: Learning from demonstration. Mobile Robots. Artificial Neural Networks. Soc FPGA.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

1.1 Contextualização

A navegação de robôs moveis é uma área de pesquisa com diferentes desafios tecnológicos, que vão desde o desenvolvimento mecânico, sensoriamento, até em termos de processamento computacional. Em particular, o controle de robôs móveis envolve sistemas multivariáveis do tipo múltiplas entradas e múltiplas saídas (MIMO do inglês *Multi-input, Multi-output*) (CHEN, S.; BAI, 2022)(CARLUCHO; DE PAULA; ACOSTA, 2020), já que frequentemente os robôs tem dois ou mais atuadores e mais de dois sensores. Consequentemente, estratégias de controle tais como o controle adaptativo (PETROV; GEORGIEVA, 2018), controle preditivo baseado em modelos (WANG, H. et al., 2019), e inclusive estrategias de controle inteligente, tais como controle neural (ZHANG; JING, 2020) e controle nebuloso (MOHANTA; KESHARI, 2019)(HACENE; MENDIL, 2019), são comumente utilizadas para estes sistemas.

Ensinar robôs mediante a demonstração humana permite que pessoas sem conhecimentos nem habilidades na área de controle ou projeto eletrônico possam programar robôs mediante o ensinamento de comportamentos (QIAN et al., 2022), que são posteriormente imitados pelo robô, permitindo a disseminação desta tecnologia nos diversos setores produtivos. Aprendizagem por demonstração (do inglês *Learning From Demonstration*, LFD) é uma técnica de aprendizagem de máquina que permite aos robôs imitar tarefas a partir das observações de demonstração humana (SAKR et al., 2022) (XU et al., 2019) (WANG, Y. et al., 2021).

Um dos principais problemas do LFD é ensinar e imitar comportamentos complexos, devido ao custo computacional dos modelos envolvidos, que além de incluir o tempo, podem incluir a estimação de estados passados do sistema. Outro problema conhecido do LFD é o ensinamento e imitação de mais de um comportamento. Para contornar esses desafios, neste trabalho foi usado o princípio de seleção de comportamentos da técnica de controle baseado em comportamento ou BBC (BERANEK; AHMADI, 2016) (do inglês *Behavior Based Controller*). Nesta técnica, comportamentos complexos são divididos em micro-comportamentos mais simples de serem ensinados e aprendidos, os quais podem ser sequenciados no tempo. Cada micro-comportamento pode ter um modelo matemático para sua realização, sendo que esse modelo pode ser obtido analiticamente ou obtido por meio de técnicas de aprendizagem de máquina.

A metodologia de LFD é dividida em três estágios. a) *Estágio de Demonstração*, onde o comportamento desejado é feito por um operador humano e os dados de entrada e saída

são armazenados em memória; b) *Estágio de Treinamento*, no qual uma solução baseada em aprendizagem de máquina é mapeada utilizando os dados de entrada e saída; c) *Estágio de imitação*, no qual o sistema está habilitado para imitar o comportamento ensinado. Os estágios são ilustrados na Figura 1.

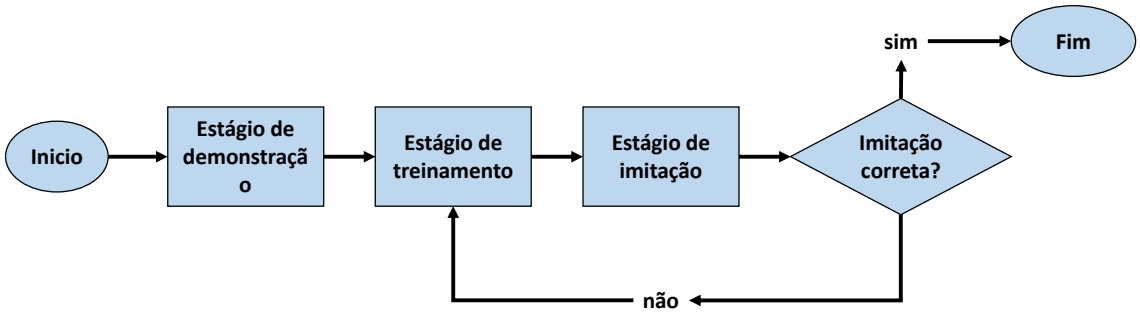


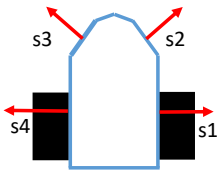
Figura 1 – Estágios da metodologia de aprendizagem por demonstração.

Desenvolver um controlador para um robô móvel de pequeno porte, modelado como um sistema MIMO é complexo e precisa de experiência em diversas áreas do conhecimento, tais como modelagem de sistemas (PIDD, 2004) e projeto de controladores (OGATA, 2001). Com ajuda da metodologia LFD e da inteligência artificial, mais precisamente das redes neurais artificiais, é possível ensinar determinadas tarefas ou comportamentos a um robô móvel. Entretanto, tal abordagem gera um grande custo computacional, derivado do grande número de operações aritméticas requeridas para cada malha de controle e para o processo de aprendizagem. Esse problema poderia ser resolvido usando plataformas computacionais de alto desempenho, porém, as plataformas computacionais de pequeno porte geralmente possuem restrições de desempenho, de memória e de consumo energético.

Nesse sentido, a exploração de dispositivos reconfigurareis, tais como FPGAs (*Field Programmable Gate Arrays)*, permite paralelizar as operações e potencialmente atingir um desempenho computacional adequado para a aplicação.

1.2 Justificativa

Neste trabalho é apresentado o desenvolvimento de uma plataforma de um robô móvel de tração diferencial de pequeno porte, chamado *Maria*, a qual terá dois atuadores elétricos (motores de corrente continua CC), e quatro sensores de distância infravermelhos, compondo um sistema MIMO, como apresentado na figura 2. Adicionalmente, utilizará um módulo *bluetooth* e um *kit* de desenvolvimento Minized com um SoC (*System on Chip*) FPGA da família Zynq 7007S da Xilinx, e um controle de teleoperação criado em um telefono celular utilizando a ferramenta MIT App Inventor 2.



Sensor	Graus
S1	0
S2	30
S3	150
S4	180

Figura 2 – Esquemático do robô *Maria*, com as posições de cada sensor infravermelho.

Com o intuito de atingir as tarefas de aprendizagem dos comportamentos utilizando a metodologia LFD, propõe-se um sistema de controle neural de tipo perceptron unicamada (em inglês *Single Layer Perceptron*, SLP), o qual utilizará dois neurônios com os valores dos sensores infravermelhos como entrada (em centímetros), e os valores das velocidades das rodas direita e esquerda (em revoluções por minuto em implementação e centímetros sobre segundos em simulação) como saídas da rede. Uma metaheurística bio-inspirada denominada otimização por enxame de partículas (do inglês particle swarm optimization, PSO) foi usado para realizar o treinamento da rede.

Neste contexto, dispositivos FPGAs permitem mapear em *hardware* técnicas de controle das roda e os algoritmos de treinamento, explorando o seu paralelismo intrínseco. Essa abordagem visa alcançar os requerimentos da frequência de atualização do controlador na ordem de mili segundos.

A implementação da metodologia LFD no protótipo de robô móvel *Maria* permite que o robô possa executar comportamentos complexos, agregando micro-comportamentos previamente aprendidos, os quais são selecionado por um *referee*. Para que o robô possa implementar a metodologia LFD de maneira correta, precisa de uma capacidade computacional adequada para treinar a rede neural de forma online. Adicionalmente, durante o processamento dos sinais dos sensores são usados filtros para eliminação de ruído, os quais geralmente possuem paralelismo intrínseco que pode ser explorado por arquiteturas paralelas usando FPGAs.

1.3 Objetivos

1.3.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver um sistema embarcado baseado em arquiteturas reconfiguráveis aplicado à aprendizagem de comportamentos por demonstração em robôs móveis de pequeno porte.

1.3.2 Objetivos específicos

Para cumprir o objetivo geral deste trabalho, os seguintes objetivos devem ser alcançados:

- Desenvolver um modelo de aprendizagem por demonstração aplicado em robôs móveis de pequeno porte.
- Implementar uma plataforma móvel reativa de tração diferencial de pequeno porte que permita validar o método de aprendizagem por demonstração proposto.
- Usar algoritmos de otimização bioinspirada para sintonizar o controlador baseado em rede neural artificial aplicado na metodologia de aprendizagem por demonstração.
- Embarcar a estrategia de controle usando a técnica de co-projeto *hardware software* baseado em arquiteturas reconfiguráveis.

1.4 Aspectos metodológicos

Neste trabalho a revisão da literatura é realizada de maneira sistemática com o intuito de adquirir conhecimentos sobre os tópicos a tratar, além de obter familiaridade com as novas ferramentas que serão usadas no desenvolvimento do projeto, como o uso de estratégias de controle neural aplicado para aprendizagem por demonstração. A técnica utilizada nessa revisão se fundamenta em artigos publicados na literatura científica, usando a base de dados *Scopus* e *IEEE Xplore*, em áreas correlatas ao trabalho, tais como aprendizagem por demonstração, robótica móvel, controle e lógica programável. A ferramenta VOSViewer foi utilizada para visualizar o resultado do analise bibliométrico.

A pesquisa desenvolvida neste trabalho quanto à natureza é uma pesquisa aplicada, gerando conhecimentos para a aplicação pratica de algoritmos de otimização bioinspirada, dirigido à solução de aprendizagem por demonstração em robótica móvel de pequeno porte.

Quanto ao procedimento, este trabalho também é uma pesquisa de tipo experimental porque determina o aprendizagem por demonstração como objeto de estudo, realizando ensaios e testes de validação das técnicas de aprendizagem e de controle do robô móvel.

A estratégia de controle neural proposta foi inicialmente validada em simulação numérica usando a ferramenta V-rep, chamada de *CoppeliaSim* a partir de julho de 2020. Neste trabalho será utilizado o nome de V-rep devido a versão em que as simulações foram implementadas. No simulador foi verificado que o robô imita corretamente os comportamentos ensinados. Posteriormente, as estratégias de aprendizagem e de controle foram mapeadas em um SoC FPGA, realizando o treinamento online, permitindo executar ensaios reais a fim de avaliar o comportamento realizado pelo robô móvel.

Finalmente, os resultados obtidos neste trabalho foram analisados de maneira qualitativa e quantitativa, comparando de forma discursiva as caraterísticas físicas e comportamentais presentes em trabalhos correlatos desenvolvidos por outros autores.

1.5 Contribuição do trabalho

A principal contribuição do presente trabalho foi a implementação da metodologia LFD utilizando uma rede neural SLP adaptativa para treinar de forma online micro-comportamentos os quais são embarcados em um Soc FPGA Zynq 7007S para aplicações em robótica móvel.

As contribuições secundarias do presente trabalho são: (a) A sintonização online de controlador neural com algoritmos bioinspirados embarcado em um SoC FPGA; (b) Criação de um gerador de código em linguagem de descrição de hardware VHDL (do inglês *Very High Speed Integrated Circuit Hardware Description Language*) para realizar a aproximação da função Sigmoide mediante interpolação lineal; (c) Desenvolvimento de um gerador de código VHDL de conversores de inteiros para ponto flutuante de 27 bits; (d) Desenho do protótipo do robô móvel *Maria*. Uma contribuição adicional, embora não explorada nesta dissertação, foi o desenvolvimento em VHDL de um protocolo I2C para comunicação de sensores e atuadores usado em robótica móvel e de manipuladores.

1.6 Organização do trabalho

O restante do presente trabalho contém 6 capítulos e está organizado da seguinte maneira. O capitulo 2 apresenta o baseamento teórico sobre aprendizagem por demonstração, redes neurais artificias e hardware reconfigurável. O capítulo 3 apresenta o modelo de LFD proposto neste trabalho e a formulação do problema de otimização para treinamento da rede neural. Apresentam-se os resultados da simulação usando a ferramenta V-rep e também se apresenta o projeto mecânico, o projeto eletrônico e a montagem do protótipo do robô *Maria*. O capitulo 4 detalha o co-projeto *hardware software* do modelo LFD proposto mediante a utilização de um SoC FPGA Zynq. Este capítulo finaliza com uma proposta de protocolo experimental para validação física da solução integrada. O capitulo 5 apresenta os resultados e discussões da aplicação do protocolo experimental, e detalha uma análise de precisão numérica nos cálculos e de performance de execução dos algoritmos de controle embarcados. Finalmente, o capítulo 6 apresenta as conclusões e propostas de trabalhos futuros.

2 Fundamentação teórica

Neste capítulo, uma abordagem teórica sobre os tópicos a serem tratados nesta pesquisa será apresentada. Inicialmente, serão apresentados conceitos sobre aprendizagem por demonstração, redes neurais artificiais e algoritmos bioinspirados, os quais compõem o núcleo de conhecimentos aplicados nesta pesquisa. Adicionalmente, são apresentadas definições básicas sobre *hardware* reconfigurável e controladores proporcional e integral. Esses conceitos são usados para realizar o controle de comportamentos de robôs móveis. Finalmente, é apresentado um quadro sinótico de trabalhos correlatos encontrados na literatura científica, que resumem o estado da arte sobre uso de aprendizagem por demonstração em robótica móvel.

2.1 Aprendizagem por demonstração

A metodologia de aprendizagem por demonstração ou LFD vem se desenvolvendo desde os anos 1990 e muitos trabalhos tem se realizado nesta área de pesquisa (DAUTENHAHN; NEHANIV, 2002)(FRIEDRICH; DILLMANN, 1995)(KAISER; DILLMANN, 1996)(MÁNTARAS; PLAZA, 1997) e diferentes abordagens tem sido relatadas para modelar demonstrações humanas ao longo dos anos (ARGALL, B. D. et al., 2009)(IJSPEERT; NAKANISHI; SCHAAL, 2002)(BILLARD et al., 2008). Esta metodologia é um paradigma mediante o qual um usuário ensina habilidades para uma máquina, de forma que possa realizar uma atividade concreta sem necessidade de uma programação específica (AMBHORE, 2020). Geralmente, a metodologia de LFD apresenta três estágios em sua implementação:

- Ensinamento ou demonstração do comportamento desejado: No estagio de ensinamento, um usuário humano, também chamado de professor, apresenta ao robô os comportamentos desejados. Por sua vez, o robô realiza a aquisição dos sinais dos sensores e atuadores para compor um conjunto de dados de treinamento. O estágio de ensinamento pode ser realizado de diferentes formas, como apresentado na figura 3.

  A demonstração mediante movimentação do robô permite que o professor implemente o comportamento desejado diretamente no robô, o qual pode ser feito de forma cinestésica (movimentando o robô manualmente), tele-operada (utilizando um controle como um *joystick*, teclado do computador, etc) ou mediante sombreamento, na qual um sistema de visão externo coleta os movimentos realizados pelo professor e o robô replica os mesmos (SMART; PACK KAELBLING, 2002) (SMYS; RANGANATHAN, 2019) (SCHÖLKOPF; PLATT; HOFMANN, 2007) (CHEN, J.; ZELINSKY, 2003).

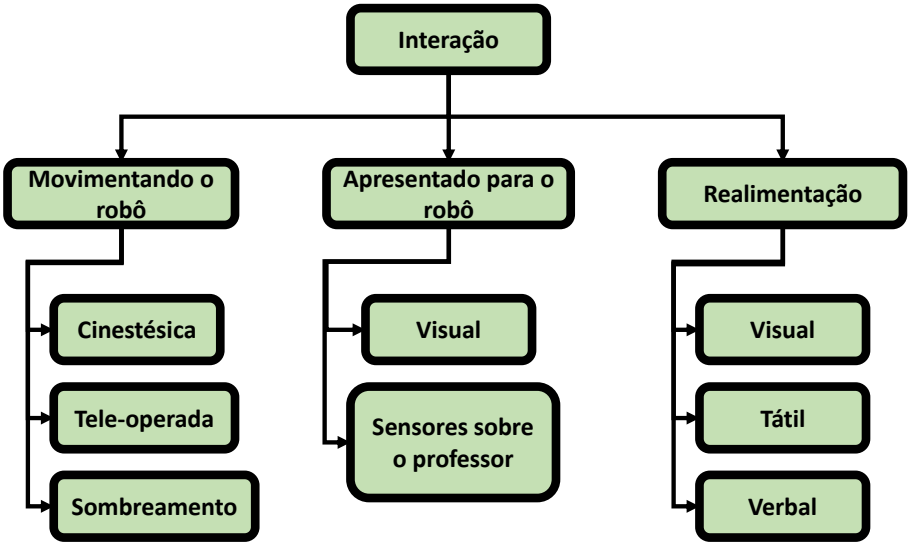


Figura 3 – Tipos de caracterização para implementar a metodologia LFD

A demonstração mediante apresentação, permite que um professor utiliza leis de controle para fornecer uma demonstração desejada (ARGALL, B.; BROWNING; VELOSO, 2007) (GROLLMAN; JENKINS, 2007) (LIU, K.; WAN; LI, 2018). O robô detecta as ações mediante a utilização de câmeras e visão computacional (PARDOWITZ et al., 2007), dispositivos portáteis (KOENEMANN; BURGET; BENNEWITZ, 2014), dispositivos de captura de movimento (KULIĆ et al., 2012), entre outros. Em algumas de estas técnicas o robô consegue acompanhar vagamente o comportamento do professor.

Na demonstração mediante realimentação o robô explora uma tarefa baseando-se no retorno do professor humano. Este retorno fornece as informações sobre até que ponto a tarefa tentada pelo robô atende ao objetivo desejado. Esta técnica pode ser utilizada para uma melhor obtenção dos dados de treinamento. (SUTTON; BARTO, 2018).

- Treinamento do comportamento: Neste estágio, utilizando o conjunto de dados de treinamento e mediante a utilização de alguma técnica de aprendizagem de máquina, como podem ser as redes neurais artificiais, o robô modela matematicamente o comportamento ensinado.
- Imitação do comportamento e avaliação: No último estagio da metodologia LFD, o robô faz a imitação do comportamento ensinado. Neste estagio pode-se ter uma avaliação do comportamento ensinado, e no caso que a imitação não seja satisfatória o usuário pode voltar ao estagio de treinamento.

Nesta pesquisa o ensinamento dos comportamentos foi feito movimentando o robô de forma tele-operada. Adicionalmente, o robô recebe uma realimentação por parte do

usuário quando são detectados erros no processo de imitação, sendo solicitado ao robô realizar novamente o treinamento. A etapa de treinamento, foi realizada usando redes neurais artificiais ou ANN (do inglês *Artificial Neural Networks*), as quais são apresentadas na seção a seguir.

2.2 Estrutura geral das redes neurais artificiais

As redes neurais artificiais são um grupo de algoritmos que simulam o comportamento do cérebro humano, e podem ser usadas para três principais problemas: a) regressão de funções, b) classificação de padrões, e c) predição de valores (HAYKIN et al., 2007), (SHAIK; ISLAM; HOSSAIN, 2021).

A figura 4 apresenta uma estrutura de uma rede neural artificial com uma camada de entrada S com n valores de entrada, pesos sinápticos W, uma camada de saída O com k neurônios e uma camada oculta H com j neurônios (PROFILLIDIS; BOTZORIS, 2019).

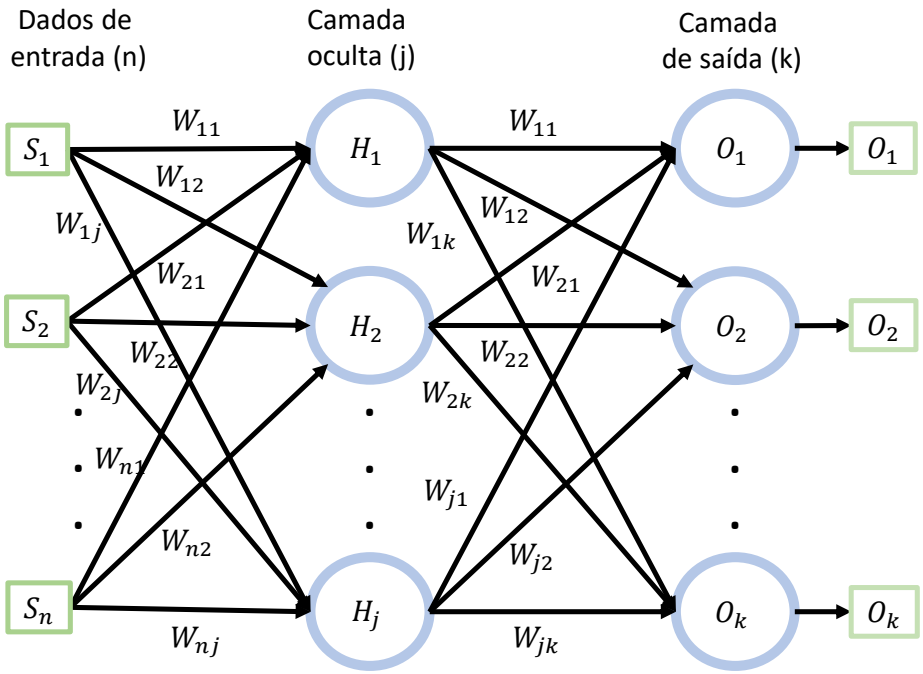
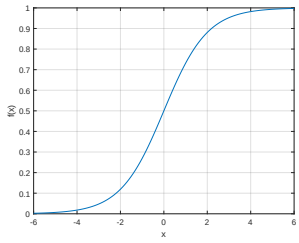
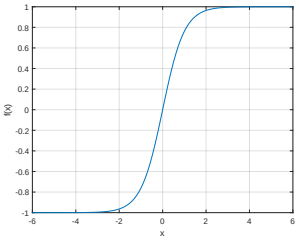


Figura 4 – Estrutura geral da rede neural artificial.

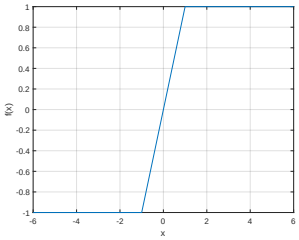
A figura 6 descreve o funcionamento de um neurônio que é a unidade básica de uma rede neural. O primeiro estágio é a soma das diferentes entradas S_n multiplicadas por seus pesos de conexão W_n. O produto vetorial $\vec{W} \cdot \vec{S}$ é então alimentado em uma função de ativação de escolha do projetista. A figura 5 apresenta as funções de ativação mais comuns, tais como a Sigmoide (HOPFIELD, 1988), Linear saturada em [-1 1] e a Tangente hiperbólica (Tanh) (ABE; GEE, 1995)



(a) Função Sigmoide $\frac{1}{1+e^{-x}}$



(b) Função Tanh $\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$



(c) Função Linear $FL = -1 \ \ se \ \ x <= -1, FL = x \ \ se \ \ x > -1 \ \ e \ \ x < 1, \ \ FL = 1 \ \ se \ \ x >= 1$

Figura 5 – Funções de ativação

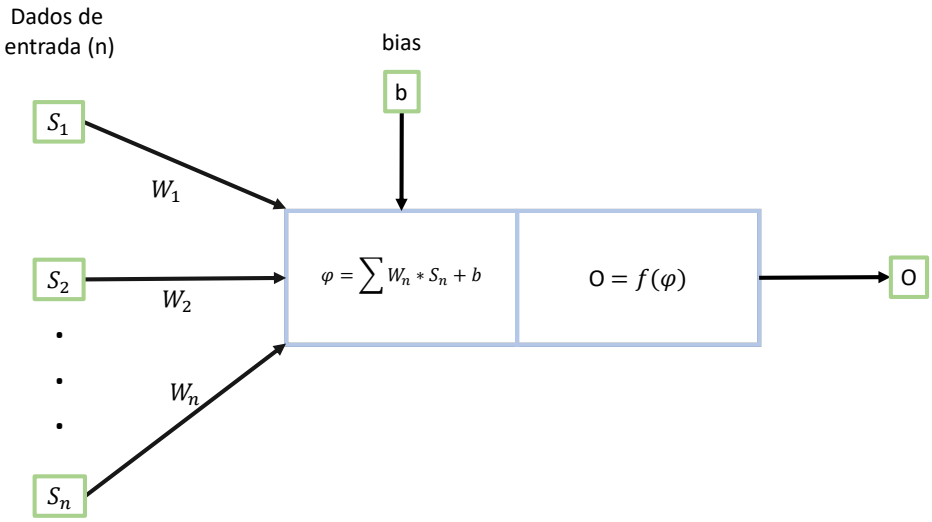


Figura 6 – Função básica de um neurônio artificial.

Nesta pesquisa foi implementada uma rede neural perceptron de camada única (SLP) com uma função de ativação Sigmoide, como mostrado na figura 7, onde S apresenta as entradas, n é o número total de entradas, O são os neurônios na camada de saída e k são o número total de neurônios de saída. O calculo de cada neurônio pode ser realizado mediante a utilização da equação (2.1)

$$O_i = f\left(\sum_{j=1}^{N} S_n W_{nk} + B_n\right), \tag{2.1}$$

onde N é o número de entradas, S_n são os valores de entrada, W_{nk} e B_n são os pesos sinápticos e o viés (*"bias"*) no neurônio k, os quais são ajustados por um algoritmo de treinamento. As redes SLP são conhecido como um classificador linear, ou seja, permitem classificar dados de entrada linearmente separáveis (BISHOP, 2006).

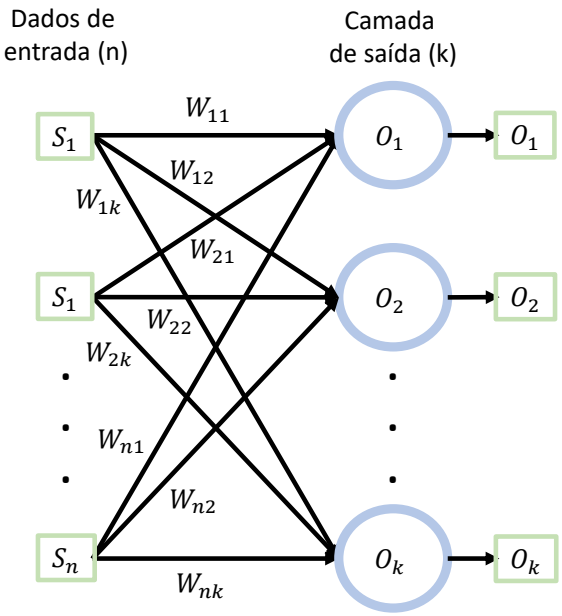


Figura 7 – .Topologia da rede neural SLP

O algoritmo de treinamento mais comumente usado é o algoritmo de retropropagação do erro (*error backpropagation*) (CHOI; LEE, 1993), o qual ajusta os pesos e *bias* dos neurônios das camadas de saídas para a camada de entrada. Isto é feito seguindo uma regra de minimização de uma função de erro (vede equação 3.6), a qual envolve o cálculo do gradiente. Como será explicado no capítulo 3, uma proposta de uma rede SLP com capacidade de adaptação será exposta. Esta rede neural será treinada usando o algoritmo bioinspirado de otimização por enxame de partículas (PSO), o qual será detalhado na seguinte seção.

2.3 Algoritmos bioinspirados

A otimização é um principio que sempre há estado na natureza, geralmente os organismos procuram melhorar continuamente suas estruturas para otimizar os recursos presentes em seu entorno, e com isto fazer que sua descendência perdure e se desenvolva de uma melhor forma. Os algoritmos bioinspirados se baseiam em alguns comportamentos dos seres da natureza (DANIEL et al., 2020), como podem ser colônias de formigas (DORIGO; DI CARO, 1999), colônias de abelhas (KARABOGA; BASTURK, 2007), enxames de peixes e aves (KENNEDY; EBERHART, 1995), entre outros (KASHIF et al., 2019) (DOKEROGLU et al., 2019), os quais baseiam-se nos princípios de compartilhar informação para sua sobrevivência.

Em um problema de otimização, o objetivo é encontrar os valores ótimos de um vetor de decisão x^* em um espaço de busca X que minimizem (ou maximizem) uma ou mais funções de custo $F = f_1, f_2, ..., f_n$, sujeitas a uma ou mais funções de restrição $G = g_1, g_2, ..., g_m$ (WEISE, 2009).

Nesta pesquisa será utilizado o algoritmo bioinspirado PSO para resolver o problema de otimização presente na sintonização dos pesos e bias de uma rede neural SLP adaptativa. A formulação do problema de otimização será apresentado no capítulo 3.

2.3.1 Algoritmo de otimização por enxame de partículas

O algoritmo de otimização utilizando exame de partículas é uma técnica de otimização baseada em população, desenvolvida por Kennedy e Eberhart em 1995 e é inspirada no comportamento de exames de aves e peixes (KENNEDY; EBERHART, 1995) (EBERHART; KENNEDY, 1995).

O algoritmo PSO utiliza as equações (2.2) e (2.3) para movimentar as partículas no espaço de busca. A velocidade da partícula i na dimensão j é atualizada por (2.2), sendo w o fator de inercia, o qual tipicamente diminui com o tempo, *c1* é o coeficiente cognitivo, *c2* é o coeficiente social os quais são ajustados de forma empírica deixando com mais pesso o coeficiente cognitivo (vai confiar mais da solução individual), e U_1 e U_2 são números aleatórios uniformemente distribuídos no intervalo [0,1]. Dessa forma, A velocidade da partícula i dependerá de sua velocidade atual v_i, da melhor posição individual (y_i) e da posição da melhor partícula no enxame (y_s) no espaço de busca.

Cada partícula atualiza sua posição usando a equação (2.3), que depende diretamente de sua posição atual e da velocidade na próximo instante de tempo. O algoritmo PSO é iterativo. Nas primeiras iterações as partículas tem uma velocidade maior, realizando uma exploração global no espaço de busca. Nas últimas iterações, a velocidade das partículas diminui, realizando uma exploração local ao redor do ponto da solução encontrada.

$$v_{ij}^{(t+1)} = w v_{ij}^{(t)} + c_1 \overbrace{U_{1j}(y_{ij}^{(t)} - x_{ij}^{(t)})}^{\text{componente cognitivo}} + c_2 \overbrace{U_{2j}(y_{sj}^{(t)} - x_{ij}^{(t)})}^{\text{componente social}}, \tag{2.2}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)}. \tag{2.3}$$

O algoritmo PSO pode ser aplicado a funções objetivo de forma unimodal (um único mínimo global) ou multimodal (múltiplos mínimos locais) e possui vários atributos desejáveis, tais como a simplicidade, fácil implementação, baixo custo computacional e a possibilidade de ser paralelizado (MUÑOZ et al., 2014). O algoritmo 1 apresenta o pseudo código do PSO, onde N é o número de dimensões do problema de otimização, S é o número de partículas, max_{iter} e $thres$ são o critérios de parada do algoritmo, que podem ser configurado como o máximo número de iterações ou um valor de threshold, e X_{max} e X_{min} são, respetivamente, o valor máximo e mínimo do espaço de busca.

Algorithm 1 Pseudocódigo do algoritmo PSO.

1: **função** PSO(S,N,c_1,c_2,Max_{iter},$thres$)
2: Inicialização do exame;
3: $iter = 1$
4: **repetição**
5: **para** i ← 1 **até** S **faça**
6: **se** $f(x_k) \leq f(y_{ik})$ **então**
7: $y_{ik} \leftarrow x_k$;
8: **fim se**
9: **fim para**
10: avaliação de y_s utilizando os S valores de aptidão $f(y_{ik})$
11: **para** i ← 1 **até** S **faça**
12: **para** j ← 1 **até** N **faça**
13: $v_{ij}^{(t+1)} \leftarrow w v_{ij}^{(t)} + c_1 U_{1j}(y_{ij}^{(t)} - x_{ij}^{(t)}) + c_2 U_{2j}(y_{sj}^{(t)} - x_{ij}^{(t)})$
14: $x_{ij}^{(t+1)} \leftarrow x_{ij}^{(t)} + v_{ij}^{(t+1)}$
15: **fim para**
16: **fim para**
17: $iter = iter + 1$
18: **until** $(f(y_{y_s}) < threshold)||(Iter <= Max_{iter})$
19: **devolve** Posição da melhor partícula x e sua aptidão $f(x)$
20: **fim função**

Avaliação e deteção da melhor posição individual

Melhor Global

Atualização

2.4 Hardware reconfigurável

Os FPGAs (*Field Programmable Gate Arrays*) são dispositivos de hardware reconfigurável. A sua arquitetura interna esta definida pela sua característica de granularidade, refletindo sob o bloco mais pequeno que compõe o dispositivo, estes blocos são chamados Blocos Lógicos Configuráveis (CLB). Nos dispositivos com arquiteturas de granularidade fina os CLBs estão compostos por memórias baseadas em LUTs (*LookUp Tables*) que implementam funções combinatórias e alguns flip-flops (FFs), os quais podem ser usados para implementar lógica sequencial. Um FPGA pode ter milhares de CLBs e operam a frequências de até 1GHz. Além dos CLBs e FFs, outros elementos básicos fazem parte de um FPGA: (a) elementos de conexão e de roteamento; (b) blocos de entrada e saídas; (c) blocos de processamento digital de sinais (DSPs), (d) blocos de memória RAM, e (e) circuitos especiais tais como PLLs, ADC, entre outros. Os CLBs são conectados através da rede de interconexão programável constituída pelos blocos de conexão e de roteamento, conforme mostrado na figura 8 e na figura 9, de forma a construir circuitos digitais de maior complexidade (MASSELOS; VOROS, 2005).

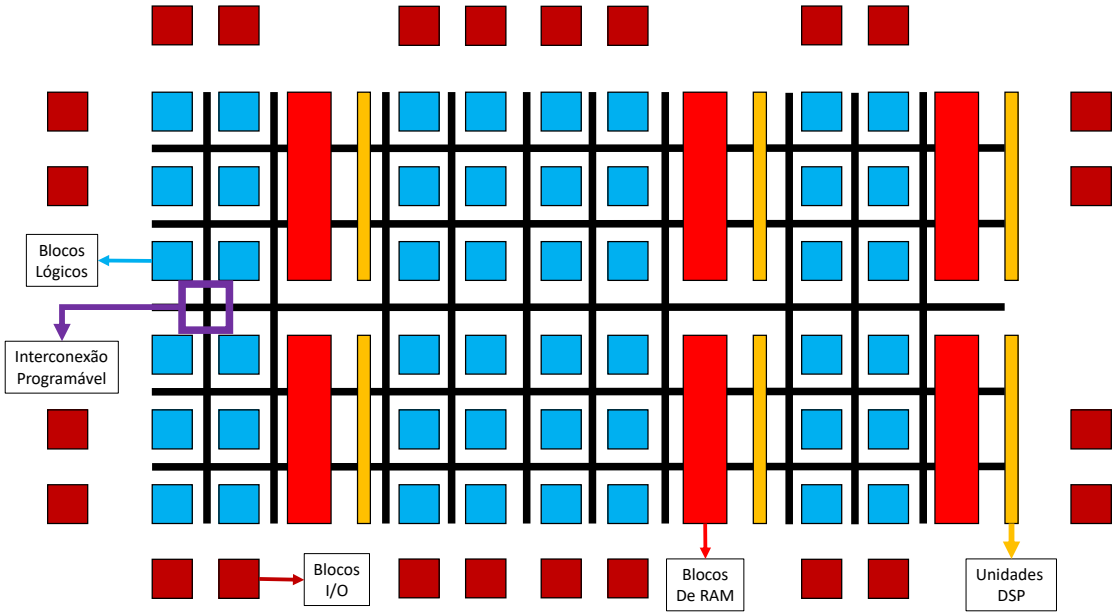
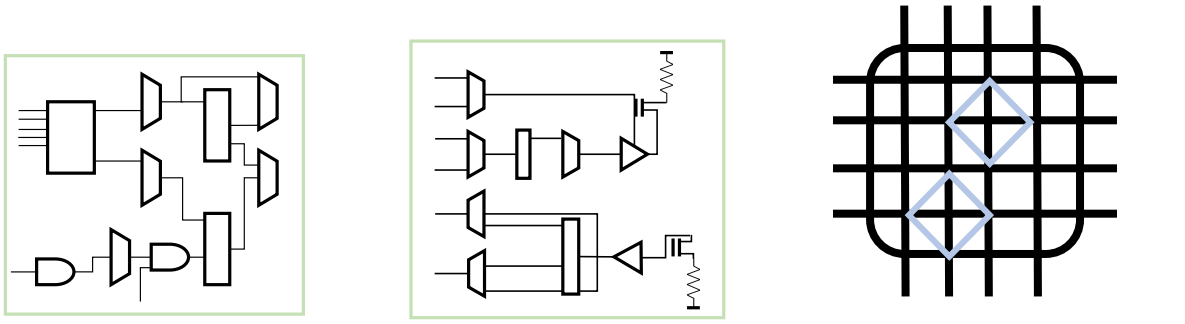


Figura 8 – Estrutura interna de uma FPGA.



(a) Blocos Lógicos (b) Blocos de I/O (c) Interconexões Programáveis

Figura 9 – Estrutura interna dos FPGAs

Na figura 10 pode-se observar a arquitetura interna do Soc (*System on Chip*) Zynq-7007S (XILINX, 2018) que esta presente na placa de desenvolvimento da Minized a qual foi utilizado nesta pesquisa. Neste Soc tem-se um processador ARM (*Advanced RISC Machine* e *RISC* do inglês *Reduced Instruction Set Computer*) Cortex A9, 512KB de Cache, 256 KB de OCM (do inglês *On-Chip Memory*), entre outros componentes. O SoC adicionalmente tem comunicação I2C (a qual foi utilizada nesta pesquisa), SPI, UART (Também utilizada para comunicação Bluetooth), assim como portas de entrada e saída (GPIO do inglês *General Purpose Input Output*).

Entre as vantagens de usar um SoC-FPGA se comparado com um circuito integrado de aplicação específica (ASIC), podem-se mencionar (HAUCK; DEHON, 2007):

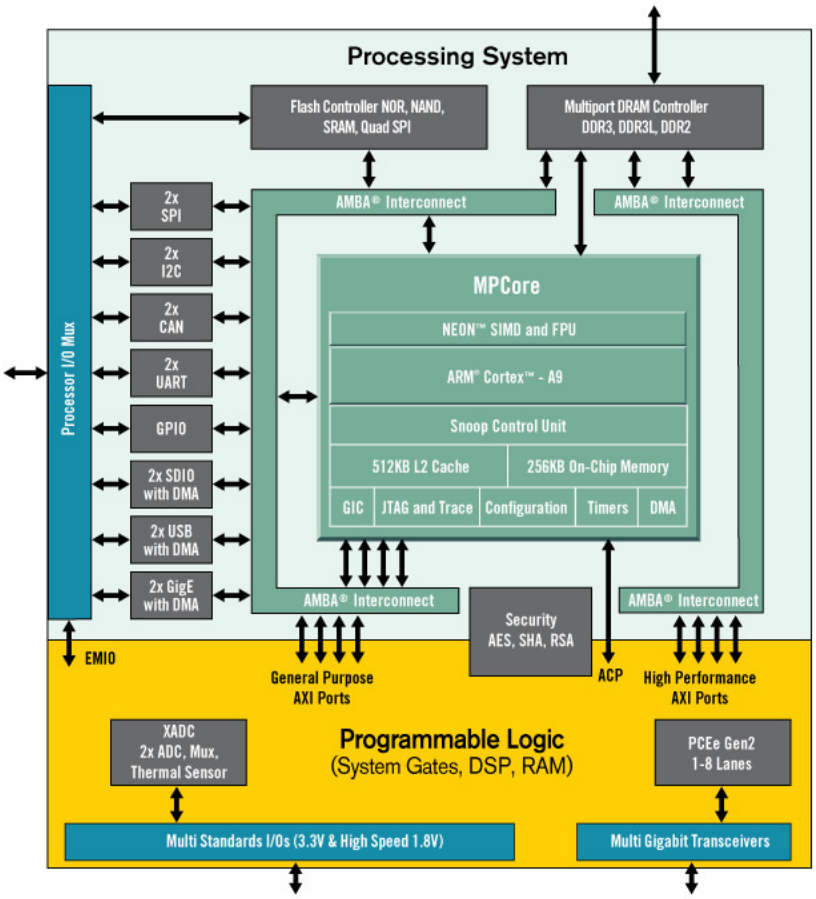


Figura 10 – Arquitectura interna do Soc-7007S Xilinx (XILINX, 2018)

- O sistema é reconfigurável.
- Custo de projeto em FPGAs é bem menor em termos de dinheiro e tempo de implementação.
- Time to market é menor para implementações em FPGAs.
- Facilidade e tempo de desenvolvimento.

Porém, o uso de FPGA acarreta as seguintes desvantagens:

- Área em silício maior.
- Consumo de potência maior.
- Custo de produção em massa maior.
- Desempenho (frequência de operação) restrito à tecnologia disponível pelo fabricante.

2.5 Controlador integral e proporcional (PI)

O controlador PI (do inglês *Proportional and Integral*) (BHANU et al., 2011), é comumente utilizado em diversas aplicações (SHARMA; GOEL; CHACKO, 2018) (KUNJITTIPONG; KONGKANJANA; KHWAN-ON, 2020) (SANT; RAJAGOPAL; SHETH, 2011). Este controlador tem uma popularidade a nível industrial e acadêmico devido a sua fácil implementação. No intuito de realizar o controle das velocidades das rodas do robô, dois controladores PI foram embarcados (um controlador PI para cada roda). O modelo matemático do controle PI em tempo continuo pode ser observado na equação 2.4

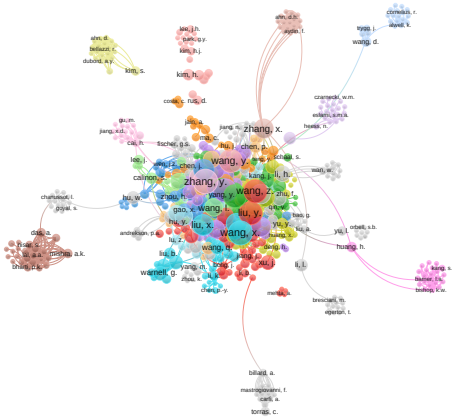
$$Out_{PI} = K_p * e(t) + K_i * \int_0^t e(\tau) * d\tau, \tag{2.4}$$

onde K_p e K_i são as constantes proporcional e integrativa, respectivamente, $e(t)$ é o erro da malha de controle (a substração do ponto de referencia com respeito ao valor adquirido pelo sensor) e $\int_0^t e(\tau) * d\tau$ é a integral do erro da malha de controle e Out_{PI} é a saída do controlador PI. Nesta pesquisa as contantes foram determinadas mediante a utilização do lugar geométrico das raízes (OGATA, 2001).

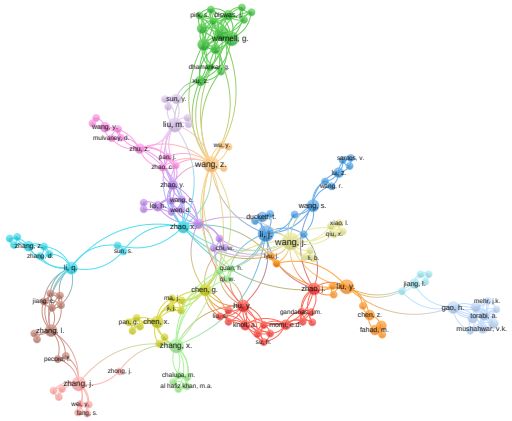
2.6 Trabalhos relacionados

Nesta seção são apresentados os trabalhos correlatos sobre aprendizagem por demonstração aplicado a robótica móvel, apresentando uma comparação entre os mesmos, tendo em conta varias caraterísticas das diferentes abordagens. Inicialmente é apresentado os trabalhos correlatos utilizando o software de *Vosviewer*, onde se inicia desde o mais global deste trabalho (aprendizagem por demonstração) até o mais especifico (aprendizagem por demonstração aplicado a robótica móvel, mediante a utilização de redes neurais artificiais e Socs FPGAs). Por último, se faz a comparação do estado da arte na tabela 1, onde as linhas representam os trabalhos correlatos e as colunas as caraterísticas mais relevantes.

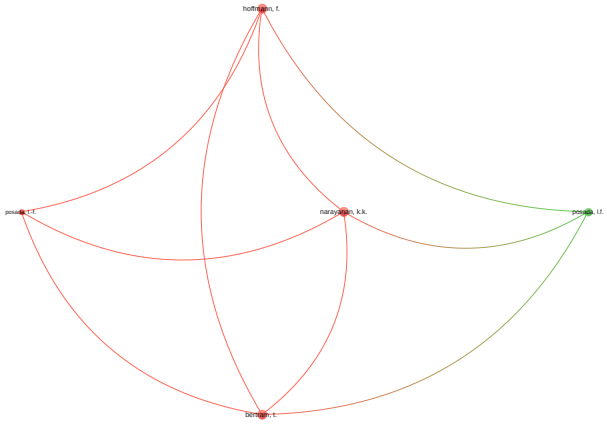
A figura 11 apresenta no software de *Vosviewer*, os resultados de trabalhos correlatos a esta dissertação, na figura 11a são apresentados os trabalhos de aprendizagem por demonstração, a figura 11b apresenta os trabalhos de aprendizagem por demonstração aplicados em robots moveis, a figura 11c apresenta os trabalhos correlatos desenvolvidos nas áreas de aprendizagem por demonstração em robótica móvel mediante a utilização de redes neurais artificiais. Por último, não tem-se trabalhos desenvolvidos em aprendizagem por demonstração, aplicado em robótica móvel, mediante redes neurais artificiais embarcados em Socs, FPGAs. É nesta areá que vai ser desenvolvida esta pesquisa.



(a) Uma palavra chave utilizada (*Learning from demonstration.*)



(b) Duas palavras chaves utilizadas (*Learning from demonstration. Mobile Robots.*)



(c) Três palavras chaves utilizadas (*Learning from demonstration. Mobile Robots. Artificial Neural Networks.*)

Figura 11 – Gráfica do *VosViewer* com as palavras chaves utilizadas nesta dissertação

Na tabela 1 a coluna *Tipo de controle* apresenta o tipo de controle utilizado na metodologia LFD implementada. A coluna *Atuadores* apresenta o tipo e o número e atuadores usados. A coluna *Sensores* apresenta quantos sensores foram utilizados e de qual tipo (por exemplo, sensores infravermelhos, de ultra-som ou câmeras, entre outros). A coluna *Plataforma* apresenta a plataforma computacional na qual foi implementada a metodologia LFD. Por último, a coluna de *Sistema de Locomoção* apresenta a tração do robô utilizado para implementar a metodologia LFD, note-se que geralmente são utilizados os sistemas de locomoção *Hackerman* e *Diferencial* devido a sua facilidade para ser controlados. A revisão da literatura foi realizada utilizando as palavras chaves desta dissertação nas bases de dados *Scopus* e *IEEE Xplorer*, onde o principal critério para filtragem dos resultados é a utilização da metodologia de aprendizagem por demonstração.

Tabela 1 – Comparação dos trabalhos correlatos de desenvolvimento da metodologia LFD aplicado à robótica móvel

Autor	Ano	Técnica de Aprendizagem	Atuadores	Sensores	Plataforma	Sistema de Locomoção
D. Muñoz et all (MUÑOZ et al., 2014)	2014	Redes Neurais	2 motores CC	5 Sendores infravermelhos	FPGA xc5vlx110t 30F6 (Eye-Sim and FPGA)	Diferencial
H. Tan (TAN, 2015)	2015	Algoritmo computacional Evolucionario	2 motores CC	7 Sendores de luz	dsPIC 30F6014A (robô E-Puck)	Diferencial
N.Vuković et all (VUKOVIĆ; MITIĆ; MILJKOVIĆ, 2015)	2015	Redes Neurais Bioinspiradas	2 motores CC	5 Ultrasom	Motorola 68331 (robô Khepera II)	Diferencial
L. Jiang et all (JIANG et al., 2018)	2018	Controle de servo visual usando a teoria de controle de espaço não vetorial	4 motores CC	1 Câmera	Computador (robô)	manipulador paralelo móvel
D.G Sillas et all (GARCIA et al., 2018)	2018	Processos Gaussianos	NA	Arquivo	ARM cortex A7 (simulação)	NA
J. Liu et all (LIU, J. et al., 2021)	2020	Redes Neurais Spiking	2 motores CC	3 Sensores de distância	Computador (Simulação)	Diferencial
Omar Gamal et all (GAMAL; CAI; ROTH, 2020)	2020	Controle Nebuloso	2 motores CC	5 Infravermelho	Computador (Simulação)	Hackerman
F. Jefferson (BORGES, 2020)	2020	Redes Neurais	2 motores CC	5 Ultra-som	Aduino Uno e Arduino Mega	Diferencial
Y. Wei et all (WEI et al., 2021)	2021	Redes Neurais Convolucionais	NA	1 Sensor Lidar, odometria e sensor de inercia	Computador (ROS)	Diferencial
Este trabalho	2022	SLP adaptativa + PSO	2 motores CC	4 sensores infravermelhos	SoC FPGA robô *Maria*	Diferencial

Aqui cabe destacar que os trabalhos encontrados não embarcaram as soluções de LFD em SoCs FPGAs, só o trabalho de (MUÑOZ et al., 2014) embarca o controle neural em FPGA e faz a simulação no *software* de Eye-sim. Adicionalmente (BORGES, 2020) realizou um controle neural para a metodologia LFD que tem correlação ao controle implementado nesta pesquisa, este controlador neural foi implementado em Arduino, gerando uma solução inteiramente em *software*. Cabe salientar que ele também crio uma plataforma robótica diferencial. Além disto as outras soluções foram simuladas em computadores ou embarcadas em microcontroladores com uma plataforma robótica já pronta. Os sensores utilizados variam, desde sensores infravermelhos, ultra-som, até câmaras e Lidar. Pode-se observar que a locomoção mais utilizada foi a diferencial, devido a sua facilidade de controle. Por último nota-se uma generalização na utilização de dois atuadores (dois motores CC) na maioria de trabalhos correlatos.

Neste trabalho, a técnica de co-projeto *hardware-software* foi usada para implementar a rede SLP adaptativa em um SoC FPGA. Foi utilizada a técnica de seleção no tempo de micro-comportamentos, treinados através do algoritmo de otimização por meio do algoritmo PSO. Adicionalmente, um robô de tração diferencial de pequeno porte com quatro sensores de distância infravermelhos foi construído para propósitos de validação experimental.

3 Projeto do robô *Maria*

3.1 Requisitos do projeto do robô *Maria*

Para o desenho e construção do robô *Maria* foram levados em conta alguns parâmetros de desenho essências nesta pesquisa, os seguintes requisitos apresentam estos parâmetros de desenho:

- utilizar tração diferencial(devido a sua facilidade de fazer controle).
- O desenho mecânico em software deve ser facilmente construído utilizando manufatura aditiva (baixo custo de fabricação e fácil replicabilidade do modelo).
- Tamanho do robô máximo de 17 cm de diâmetro.
- Compatível para utilizar a placa Minized da Xilinx.
- Adaptável para seis sensores infravermelhos com referencia Sharp GP2Y0A21YK0F.
- Autonomia de mais o menos trinta minutos.
- Conexão bluetooth.
- Treinamento do robô de forma online.
- Permitir a localização em tempo real.
- Solução escalável para futuras aplicações em robótica multiagente.

Na seguinte seção é apresentado o resultado do desenho mecânico do robô *Maria* em Solidworks.

3.2 Desenho mecânico e eletrônico do robô *Maria*

3.2.1 Desenho assistido por computador (CAD) do robô *Maria*

Para o projeto do robô foi utilizada a ferramenta de projeto CAD Solidworks, a qual salva o modelo em formato *stl* que *à posteriori* é usada em uma impressora 3D Prussa I3. O desenho do chassis do robô *Maria* pode ser observado na figura 12.

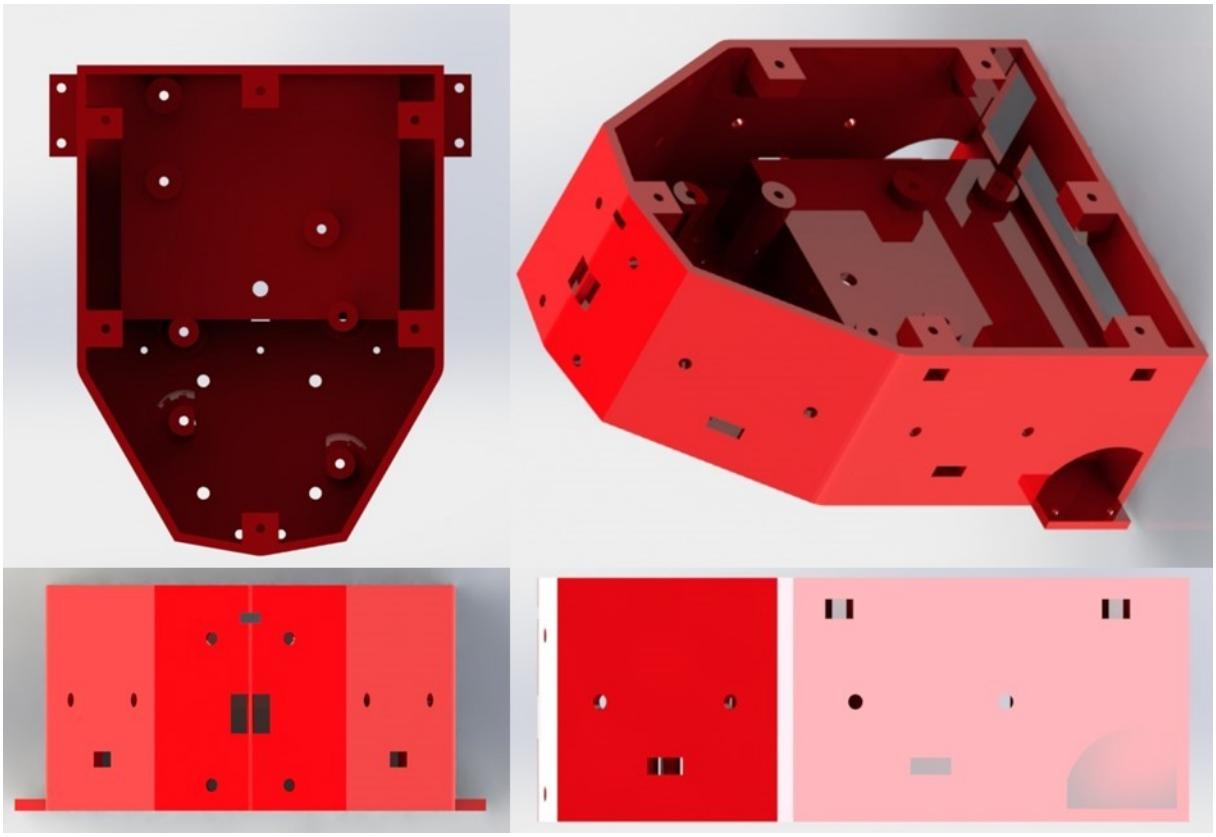


Figura 12 – CAD do robô *Maria*. Dimensões: largura de 120 mm, comprimento de 160 mm e altura de 70 mm.

3.2.2 Conexões eletrônicas do robô *Maria*

No intuito de otimizar a eficiência computacional do FPGA utilizada nesta pesquisa, a parte de potência (motores, encoders, driver LM298) e a parte de controle (FPGA, Arduino, Bluetooth e sensores infravermelhos) foram alimentadas separadamente, com terra comum. Os componentes eletrônicos utilizados são listados a seguir:

- Quatro sensores Sharp GP2Y0A21YK0F infravermelhos com uma distancia mínima de 10cm e máxima de 90 cm.
- Dois conjuntos de roda, encoder e motores Pololu de 6V, 220 RPM, torque de 1,2Kgcm e relação de 150:1.
- Um kit de desenvolvimento SoC FPGA Minized da Avnet.
- Um Arduino UNO.
- Um driver de potência LM298N.
- Um módulo bluetooth compatível com os Pmods da Minized.
- Quatro baterias Ion-Li de 4,2V, ref 18650.

- Um regulador de tensão DC-DC LM317.
- Uma roda caster.

O diagrama de conexões é apresentado na figura 13. Destaca-se o SoC FPGA Minized, oa qual chega a informação dos encoders para estimar a velocidade de cada roda, o sinal do bluetooth, e o sinal digital dos sensores, enviados pelo Arduino através de comunicação I2C. A Minized envia a ação de controle para o driver LM298 que aciona as rodas do robô. É importante salientar que o Arduino UNO exclusivamente realiza a conversão analógico-digital dos sensores infravermelho.

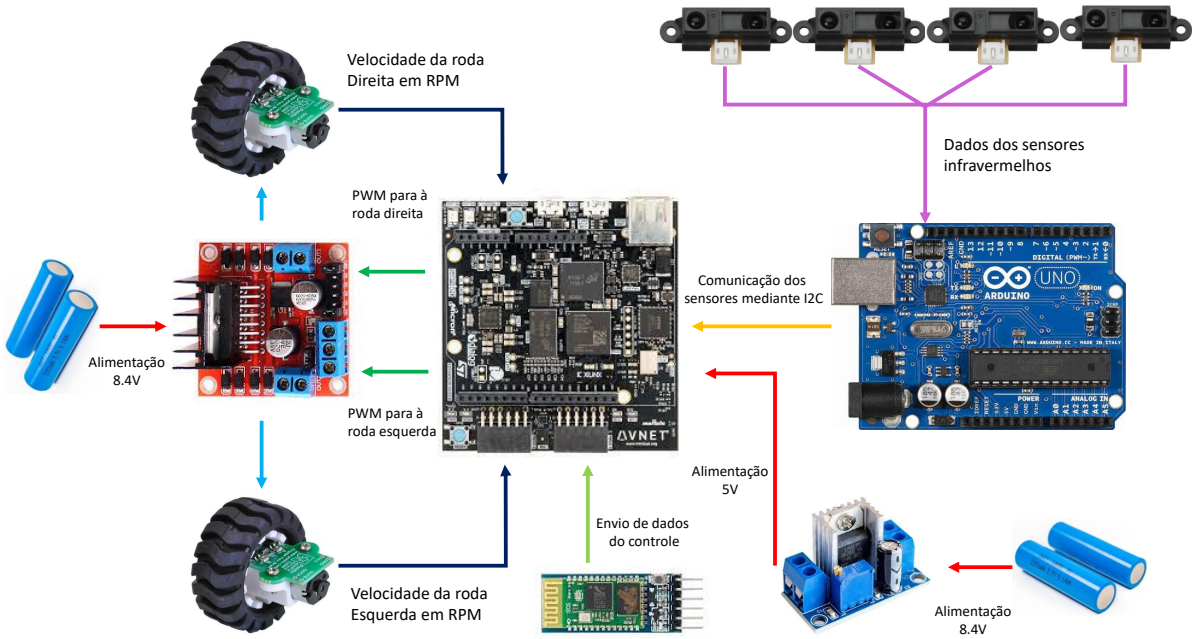


Figura 13 – Conexões eletrônicas do robô *Maria*.

Por último se fez a integração da parte eletrônica com o desenho mecânico (em software) para corroborar que todos os componentes encaixavam da melhor forma. A figura 14 apresenta as vistas superior, frontal, lateral e isométrica do robô *Maria*.

3.3 Calibração dos sensores infravermelhos

Nesta seção é apresentada a forma de calibração dos sensores infravermelhos do robô *Maria*. Inicialmente para obter a equação que descreve cada sensor foi assumido que todos os sensores são idênticos, permitindo replicar o modelo de um sensor nos outros. Tendo isto em consideração foi utilizado um Arduino UNO para adquirir os dados do conversor ADC. No procedimento de calibração foi usada uma trena como referência, o robô foi posicionado soba trena e um objeto feito de papel kraft foi colocado à frente do sensor. Uma primeira

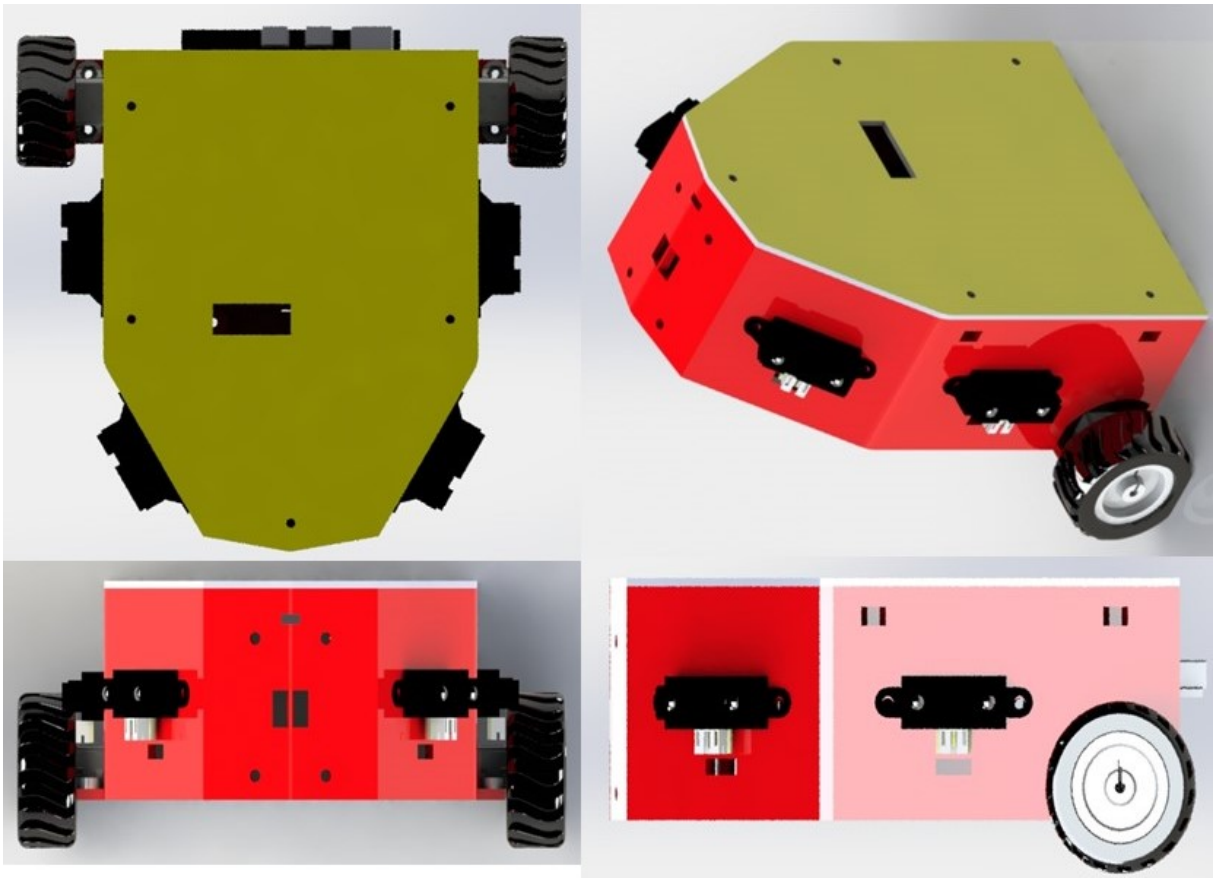


Figura 14 – Integração do chassis mecânico com os componentes eletrônicos.

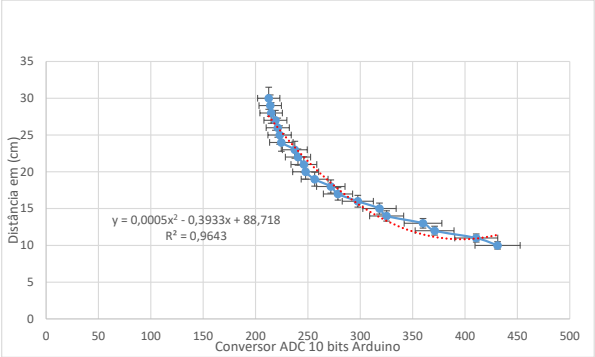
calibração foi implementada mediante coletadas de 16 amostras desde 10 cm até 30 cm com intervalos de 1 cm e o valor médio foi usado para realizar a identificação da equação do sensor, esta calibração deu como resultado a equação 3.1

$$ES = (0{,}0005) * val_{adc}^2 - (0{,}3933) * val_{adc} + 88{,}718. \tag{3.1}$$

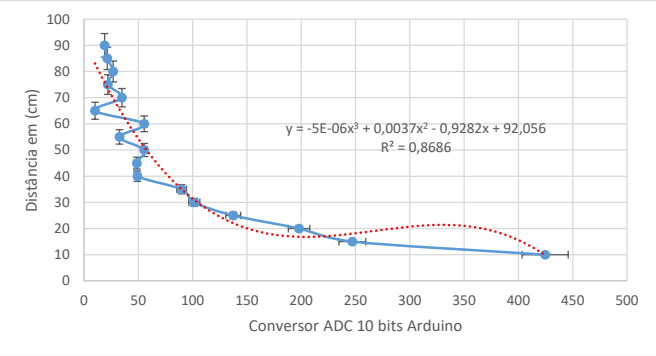
Nesta primeira calibração optou-se por ter uma melhor resolução em pequenas distâncias. Uma segunda calibração foi desenvolvida mediante coletas de 16 amostras desde 10 cm até 90 cm (distância máxima do sensor) com intervalos de 5 cm, o valor médio foi usado para realizar a identificação da equação do sensor, esta calibração deu como resultado a equação 3.2

$$ES = (-5 * 10^{-6}) * val_{adc}^3 + (0{,}0037) * val_{adc}^2 - 0{,}9282 val_{adc} + 92{,}059. \tag{3.2}$$

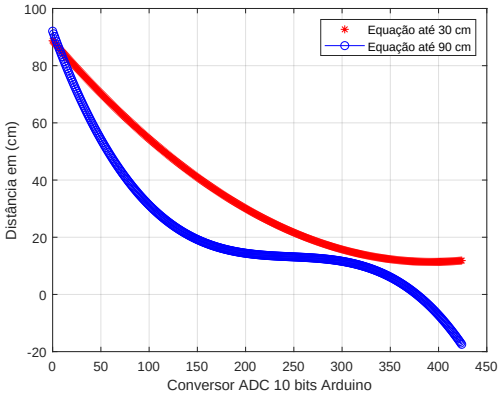
Nesta segunda calibração optou-se por ter uma medição melhor ao longo da faixa de medição do sensor, a qual vai desde 10 cm até 90 cm. Em 3.1 e 3.2 ES é o valor em centímetros, val_{adc} é o valor do conversor ADC do Arduino. A figura 15a e 15b apresentam os valor utilizados para a obtenção das equações 3.1 e 3.2. Note-se que a primeira equação tem um coeficiente R^2 maior que a segunda equação.



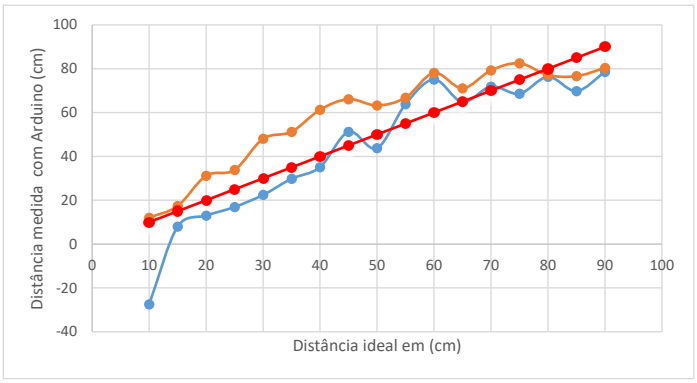
(a) Dados para a obtenção da equação 3.1



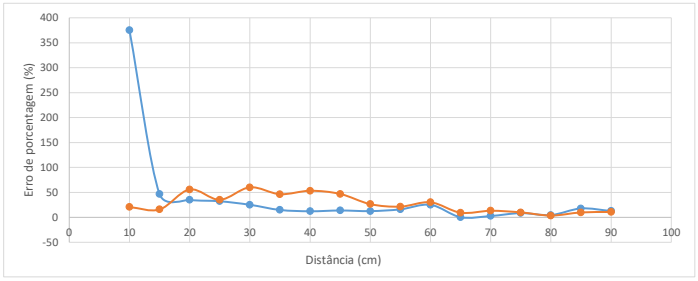
(b) Dados para a obtenção da equação 3.2



(c) Comparação da equação 3.1 e a equação 3.2



(d) Comparação das equações na medição de distância



(e) Comparação da equação 3.1 e a equação 3.2

Figura 15 – Calibração do sensor de distância infravermelho

Na figura 15c é apresentada a comparação das duas equações para uma entrada de conversor ADC de 10 bits simulada em Matlab, onde a gráfica vermelha é a equação 3.1 e a gráfica azul é a equação 3.2. Os valores utilizados para simular começam em 0 e vão até 430 com intervalos de 10. Note-se que para valores maiores a 350 o valor da equação 3.2 são negativos. Na figura 15d foram implementadas as duas equações em Arduino e colocou-se um objeto feito de papel kraft em distâncias desde 10 cm até 90 cm com intervalos de 5 cm. Em cada uma das distâncias do objeto foram adquiridas 16 amostras de distância e o valor médio foi utilizado para realizar cada um dos pontos da gráfica. A gráfica laranja apresenta os valores da equação 3.1, a gráfica azul apresenta os valores da equação 3.2 e a gráfica vermelhar apresenta os valores ideais que deveriam ser adquiridos. Note-se que para o valor

de 10 cm a equação 3.2 apresenta valores negativos de distância, esto é indesejável para o sistema de medição do robô devido a que não funcionaria em distancias vizinhas de 10 cm. Por último na figura 15e é apresentado o erro da figura 15d, este erro e calculado mediante a utilização da equação 3.5, onde é utilizado os valores da gráfica vermelha da figura 15d como valores de referencia. Nesta última gráfica (15e) pode-se observar que a equação 3.2 tem um erro significativo em 10 cm, já nos outros valores o erro é tolerável. Por outro lado a equação 3.1 não tem erros excessivos (como a equação anterior em 10 cm), mas desde 20 cm até 50 cm este erro é maior que a equação anterior e já nos últimos valores de distância, os valores das duas equações são bastantes similares.

A equação implementada no robô *Maria* foi a 3.1 devido a que a tem um melhor coeficiente de R^2, possui um polinômio de menor grau (pode-se implementar de uma forma mais fácil), não possui valores negativos para distancias pequenas e tem um erro similar para valores maiores à 30 cm se comparado com a equação 3.2.

A tabela 2 apresenta um resumo estatístico para a equação 3.1, nesta tabela foram utilizados os valores da primeira calibração, onde foram calculados os valores médios, o desvio padrão e a precisão mediante o erro relativo. O valor médio é calculada mediante a utilização da equação 3.3

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \tag{3.3}$$

onde $\bar{x}$ é o valor médio, n é o número de amostras e $\sum_{i=1}^{n} x_i$ é a somatória de todas as amostras, o desvio padrão que é calculado mediante a utilização da equação 3.4

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N}}, \tag{3.4}$$

onde σ é o desvio padrão, $\bar{x}$ é o valor médio e x_i é a amostra i. por último a precisão foi calculada mediante a utilização do erro relativo 3.5

$$Erro_r = (\frac{x_i - x_t}{x_i}) * 100\%, \tag{3.5}$$

onde x_i é a media do valor e x_t é o valor de referencia,nesta equação quando o valor é mais próximo de zero a medição tem maior precisão.

Tabela 2 – Calibração do sensor infravermelho

Distancia cm	$\bar{x}$ cm	σ	$Erro_r\%$
10	12,3828	0,7263	23,8284
11	11,7087	0,2490	6,4427
12	12,0113	1,2541	0,0938
13	12,0611	0,8633	7,2223
14	13,9962	2,1512	0,0271
15	14,2065	0,4816	5,2902
16	15,9843	0,7980	0,0983
17	18,0021	1,3493	5,8944
18	18,7651	0,5606	4,2503
19	20,8043	1,6152	9,4965
20	21,9607	0,2872	9,8034
21	22,3094	2,0470	6,2353
22	23,1563	2,1361	5,2559
23	23,9311	3,3953	4,0484
24	25,8226	2,9215	7,5942
25	26,0550	3,0619	4,2199
26	26,6460	4,5797	2,4846
27	27,2314	4,6197	0,8569
28	27,8822	4,8111	0,4206
29	28,0677	5,2872	3,2149
30	28,7216	5,7533	4,2612

Na seguinte seção se apresenta o modelo LFD usado neste trabalho, incluindo a formulação do problema de otimização para treinamento da rede neural artificial.

3.4 Estrategia de controle LFD em ambiente de simulação

Nesta secção é apresentada a metodologia de aprendizagem por demonstração utilizada para ensinar comportamentos simples ao robô móvel. As habilidades aprendidas podem ser combinadas, por multiplexação no tempo, para realizar tarefas complexas em cenários reais. A representação de baixo nível das habilidades desejadas, também chamadas de micro-comportamentos, foi realizada em V-rep por um operador humano usando o teclado do computador. Em particular, três micro-comportamentos foram demonstrados: *avançar para frente*, *girar em sentido horário* e *girar em sentido anti-horário*. O micro-comportamento de *avançar para frente* é realizado mantendo o robô no meio de um corredor. Os micro-comportamentos de girar são realizados entorno de um objeto circular de 200 cm de diâmetro.

Para cada micro-comportamento foi coletado um conjunto de dados composto por 600 amostras dos valores de seis sensores de distância e das velocidades esperadas das rodas esquerda e direita. Em seguida, uma rede neural perceptron de camada única foi treinada

usando o algoritmo PSO no Matlab, obtendo assim um controlador do robô para cada micro-comportamento. Para isso, o PSO foi configurado para minimizar o Erro Quadrático Médio (MSE) das velocidades absolutas das rodas esquerda e direita, conforme mostrado pela equação (3.6).

$$minf_1 = \frac{1}{n}\sum_{i=0}^{n}(\hat{SR}_i - SR_i)^2 + \frac{1}{n}\sum_{i=0}^{n}(\hat{SL}_i - SL_i)^2, \tag{3.6}$$

onde n é o número de amostras, $\hat{SR}_i$ e $\hat{SL}_i$ são as velocidades ensinadas da roda direita e esquerda, respectivamente, e SR_i e SL_i são as velocidades calculadas pela rede neural das rodas direita e esquerda, respetivamente.

Este trabalho propõe a utilização de um único neurônio, chamado de *referee*, que dependendo da medida atual da distância, irá alternar os pesos e bias, endereçando-os ao controlador de velocidade SLP, a fim de executar o micro-comportamento selecionado. Para treinar o *referee*, um cenário de teste composto pelos três micro-comportamentos foi desenvolvido, e manualmente o operador humano ensina o robô móvel quando alternar entre os micro-comportamentos. Neste trabalho, foi criado um padrão de saída manual, dividindo a saída esperada do árbitro ($\hat{P}$) em três níveis. Por exemplo, um padrão desejado de $\hat{P} = [0{,}25\ 0{,}5\ 0{,}0]$ indica a saída do árbitro desejada para os micro-comportamentos *avançar para frente*, *girar no sentido horário* e *girar no sentido anti-horário*, respectivamente.

A função objetivo para treinar o árbitro foi modelada conforme a equação (3.7),

$$minf_2 = \sum_{i=0}^{n}(\hat{P}_i - P_i)^2, \tag{3.7}$$

onde n é o número de amostras, $\hat{P}_i$ é o valor de padrão esperado, P_i é o padrão classificado, respectivamente.

Duas topologias de redes neurais são usadas no ambiente de simulação, a primeira é utilizada nos estágios de aprendizagem dos micro-comportamentos e a segunda é utilizada quando se faz a integração com o *referee*. Estas topologias são apresentadas na seguinte seção.

3.4.1 Arquitetura das redes neurais artificiais para LFD

Duas etapas vão ser abordadas nesta secção. A primeira para o processo de aprendizagem dos micro-comportamentos, e a segunda para integração dos micro-comportamentos com o *referee*.

Para aprendizagem dos micro-comportamentos foi utilizada a topologia perceptron de camada simples, como apresentado na figura 16. Observa-se que cada neurônio controla a velocidade de cada roda. As entradas são normalizadas.

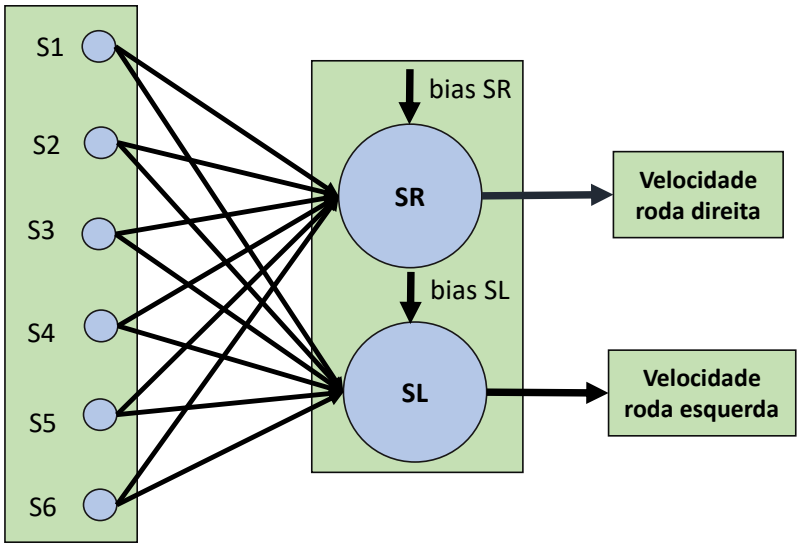


Figura 16 – Rede neural SLP para implementar a metodologia LFD para cada micro-comportamento.

O calculo de cada neurônio é realizado por uma soma de entradas ponderadas, e esse valor é aplicado a uma função de ativação Sigmoide f, conforme mostrado na equação (3.8),

$$y = f\left(\sum_{i=1}^{N} x_i w_i + b\right), \tag{3.8}$$

onde N é o número de entradas (número de sensores infravermelho), x_i são os valores de entrada, neste caso os valores dos sensores, w_i e b são os pesos sinápticos e o bias, respectivamente, os quais são ajustados pelo algoritmo de treinamento. Na equação (3.9) apresenta a função de ativação Sigmoide utilizada na rede.

$$f = \frac{1}{1 + e^{-x}}. \tag{3.9}$$

A integração do *referee* com a rede SLP resultou em uma rede neural adaptativa, mostrada na figura 17. Até a data da escrita deste documento, e no melhor entendimento do autor, esta topologia é uma contribuição original no estado da arte.

Nesta topologia, os pesos de cada micro-comportamento (três conjuntos de pesos e bias), previamente obtidos no processo de treinamento, estão armazenados em memória. Um terceiro neurônio, chamado *referee*, se encarrega de multiplexar no tempo os pesos da rede a partir da estimação do tipo de micro-comportamento que o robô deve executar. Esta estimação é feita a partir da medição atual das distâncias até os obstáculos.

Ressalta-se que nesta topologia o treinamento do *referee* não utiliza as velocidades como saídas. A saída esperada do *referee* é o padrão manualmente desenvolvido, exposto na figura 18. Este padrão indica quando deve-se utilizar cada um dos comportamentos, por

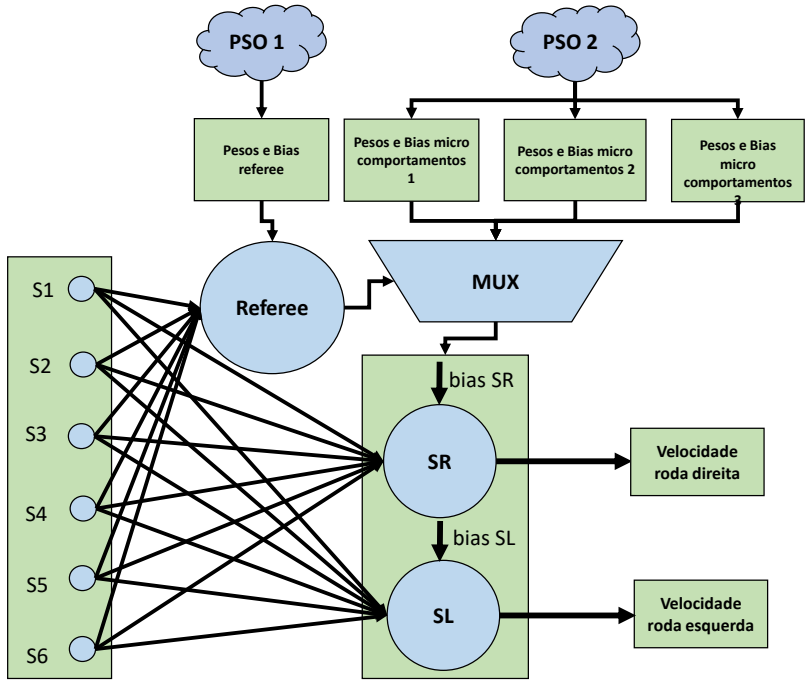


Figura 17 – Topologia SLP adaptativa para o controle neural do robô *Maria*

exemplo, a saída 0.25 indica que deve-se utilizar o comportamento de avançar para frente, a saída de 0.5 indica que deve-se utilizar o comportamento de girar no sentido horário, e a saída igual a zero indica que deve-se utilizar o comportamento de girar no sentido anti-horário.

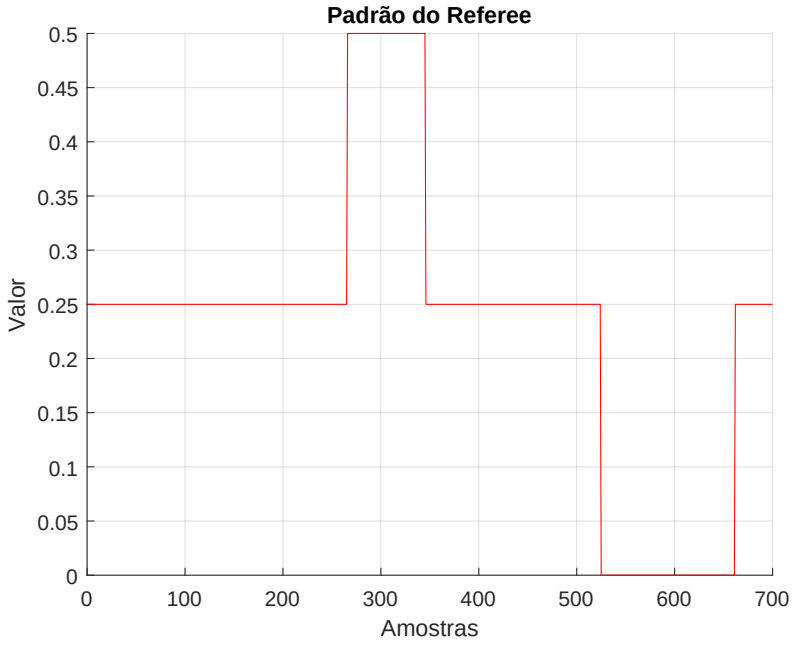


Figura 18 – Padrão de treinamento do *referee*.

A figura 17 mostra dois algoritmos de treinamento PSO. O primeiro (PSO1) é usado

para o treinamento do *referee*. O segundo (PSO2), é usado para o treinamento dos pesos e bias que emulam cada micro-comportamento. Na seguinte subseção se apresenta a configuração de parâmetros de cada PSO.

3.4.2 Algoritmo PSO para treinamento

Como foi dito no capitulo 2, o algoritmo de otimização por enxame de partículas será utilizado para minimizar as funções custo 3.6 e 3.7. Trata-se de dois algoritmos PSO executados em momentos diferentes e com configurações diferentes. Na proposta de LFD é necessário que o PSO2, que realiza o treinamento da rede neural SLP dos micro-comportamentos (vide figura 16), seja executado antes do PSO1 que ajusta os pesos e bias das entradas da rede para o *referee*.

A tabela 3 apresenta os parâmetros de configuração de cada PSO usados para treinar a rede SLP e o *referee*. O fator de inércia W diminui linearmente de 0,9 para 0,1, permitindo que as partículas realizem uma busca global durante as primeiras iterações e refinem a solução alcançada durante as iterações finais. Os coeficientes cognitivos e sociais foram ajustados empiricamente usando experimentos individuais com 500 iterações como critério de parada. Para a aplicação pretendida, c_1 e c_2 foram configurados para 3,05 e 1,05, respectivamente, obtendo partículas mais confiantes no conhecimento individual do que no conhecimento social, o espaço de busca de todas as variáveis para o PSO2 foi utilizando um limite máximo de 1 e um mínimo de -1, no caso do PSO1 o limite máximo do espaço de busca de todas as variáveis foi de 10 e o mínimo de -10. Para treinar a rede SLP, que determina a velocidade de cada roda, foram necessárias $N = 14$ variáveis de decisão, doze pesos de conexão e dois bias. Para treinar o *referee*, foram necessárias $N = 7$ dimensões, seis pesos de conexão e um bias. todos os valores foram escolhidos mediante prova e erro em experimentação simulada. Neste contexto não foi feito um analise estatístico, só foi utilizado um experimento para achar os valores das pesos e bias da rede neural artificial.

Tabela 3 – Parâmetros do PSO para treinamento do controlador SLP e do *referee*.

Parâmetros	**PSO2 (SLP)**	**PSO1 (*referee*)**
Número de partículas S	10	10
Dimensões N	14	7
Fator de inercia w	[0,9 to 0,1]	[0,9 to 0,1]
Coeficiente cognitivo c_1	3,05	3,05
Coeficiente social c_2	1,05	1,05
Limite máximo X_{max}	1	10
Limite mínimo X_{min}	-1	-10
Numero de iterações Max_{iter}	500	500

3.5 Testes de simulação

Com o intuito de fazer testes utilizando o desenho CAD do robô, no *software* de de V-Rep foi feita a importação de todas as peças, montando o robô *Maria* no *software*. Foram adicionados os sensores infravermelhos com uma distancia útil de 10cm a 90cm. Adicionalmente, foram incluídos dois motores CC, os quais têm internamente um controle de velocidade PID. O V-rep conta com funções próprias para estimação da velocidade das rodas, as quais são usadas para realizar o controle de velocidade. É importante salientar que o robô simulado não conta com um encoder, e que as simulações realizadas neste trabalho são cinemáticas, portanto, não são consideradas a carga do robô e o atrito entre componentes internos e entre o chão e as rodas.

3.5.1 Micro-comportamento 1: Avançar para frente

A figura 19 mostra o cenário usado para ensinar o robô a avançar no meio de um corredor (linha azul). O operador humano usa o teclado para manter o robô a uma velocidade linear constante, no entanto, pequenas variações (menos de 5%) nas velocidades das rodas foram observadas no conjunto de dados. Essas variações são esperadas durante uma operação no mundo real de um robô móvel. A linha verde ilustra a trajetória imitada após o processo de treinamento do controlador de velocidade SLP.

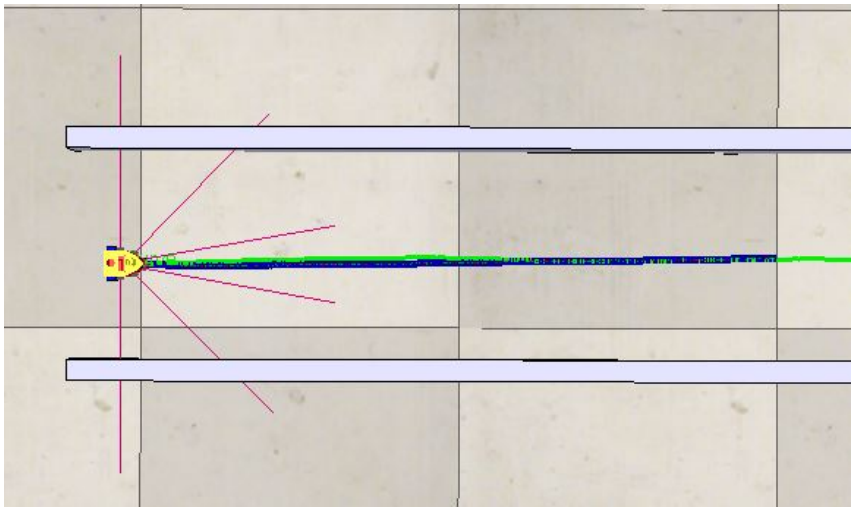


Figura 19 – Micro comportamento1: avançar no meio de um corredor. Azul: trajetória demonstrada. Verde: trajetória imitada.

Os erros de velocidade das rodas e o erro de trajetória são mostrados na figura 20. Pode-se observar que as velocidades das rodas imitadas são o valor médio das demonstradas. A trajetória imitada no plano $x-y$ acompanha a trajetória demonstrada, as quais apresentam uma leve inclinação devido ao ruído do conjunto de dados de treinamento. Entretanto, em $x = -2{,}0m$, o controlador SLP (linha verde) compensa o desvio e mantém o robô no meio do corredor durante o restante do cenário.

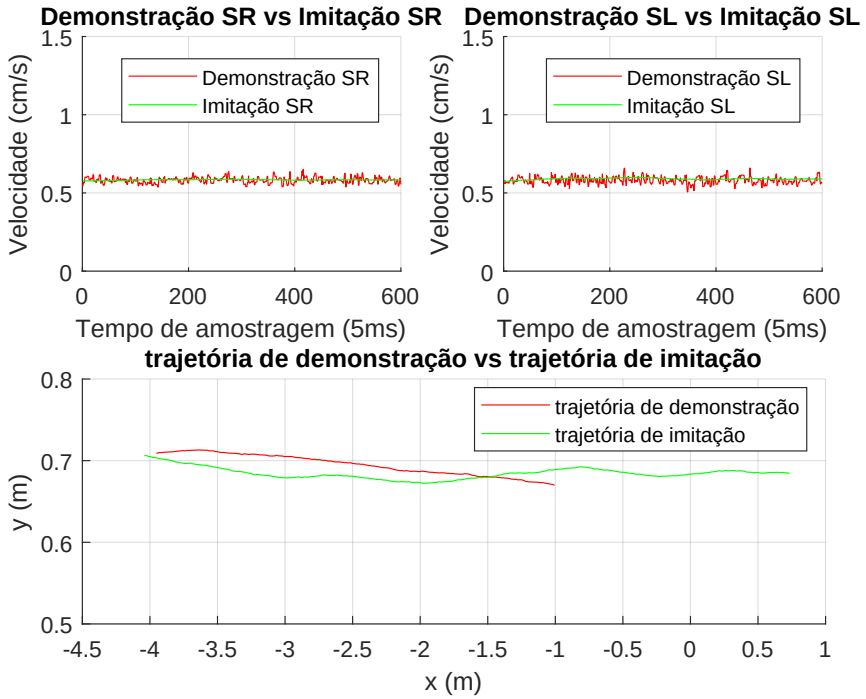


Figura 20 – Acima: Velocidades da roda direita (SR) e esquerda (SL) para o micro-comportamento1: avançar no meio de um corredor. Abaixo: trajetória demonstrada (vermelho) vs trajetória imitada (verde). Detalhes da simulação podem ser vistos em https://youtu.be/R5U8wYOh45c

3.5.2 Micro-comportamento 2: Girar no sentido horário

A figura 23 mostra o cenário usado para ensinar o robô a girar no sentido horário, mantendo uma distância de 14 cm de um obstáculo cilíndrico de 200 cm de diâmetro (linha azul). Observou-se que o conjunto de dados obtido da trajetória demonstrada possui variações normais de uma operação manual do robô, ou seja, o caminho realizado não é um círculo perfeito. No entanto, a trajetória imitada (linha verde) conseguiu ignorar esses desvios realizando o comportamento esperado.

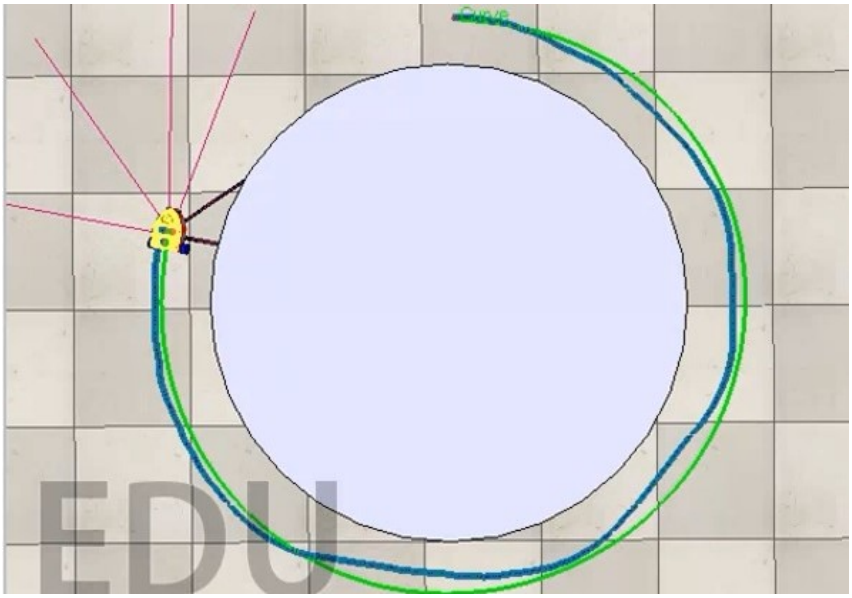


Figura 21 – Micro-comportamento2: girar no sentido horário. O robô se move no sentido horário mantendo uma distância de 14 cm de um cilindro com 2 metros de diâmetro. Azul: trajetória demonstrada. Verde: trajetória imitada.

Os erros de velocidade das rodas e o erro de trajetória são mostrados na figura 22. Observa-se que o conjunto de dados de treinamento tem uma grande variação para a velocidade da roda direita se comparada à velocidade da roda esquerda (vide linhas vermelhas). No entanto, ambas as velocidades imitadas à direita e à esquerda mantêm-se no respectivo valor médio (linhas verdes). Pode-se observar ainda que tanto as trajetórias demonstradas quanto as imitadas são semelhantes. Embora não relatado na figura, observamos que se a posição inicial for maior que 14 cm do obstáculo central, o robô se aproxima do obstáculo e então mantém a distância esperada do cilindro. Esse fato demonstra que a aproximação universal do controlador neural contribui para aprender a habilidade desejada mesmo que o conjunto de dados de treinamento não contenha essa informação específica.

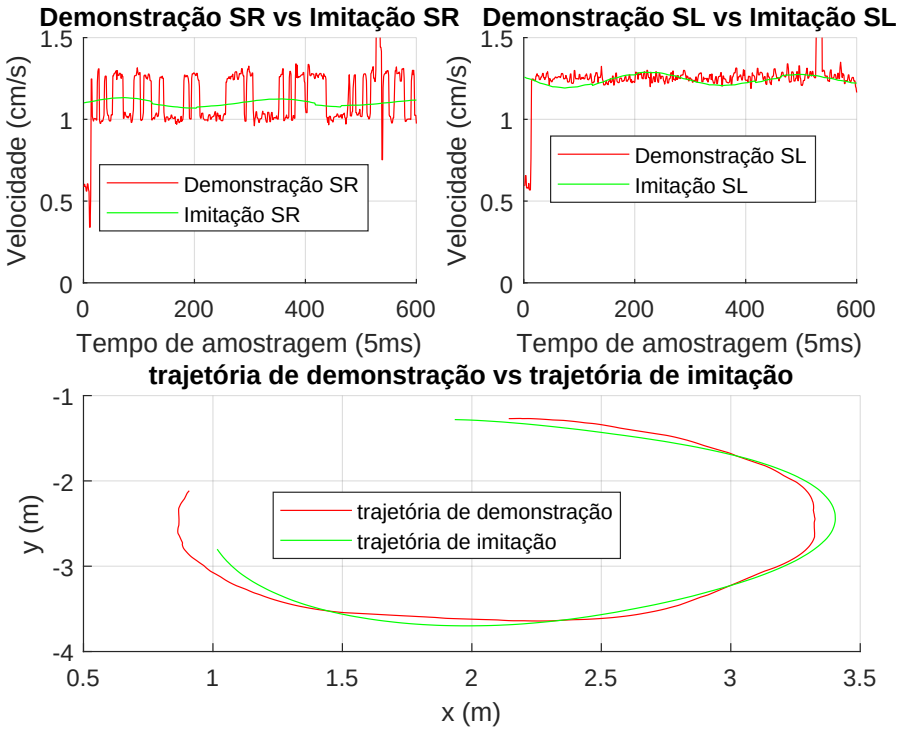


Figura 22 – Acima: Velocidades direita e esquerda para o micro- comportamento2: girar no sentido horário. Abaixo: trajetória demonstrada (vermelho) vs trajetória imitada (verde). Detalhes podem ser vistos em https://youtu.be/PeaMqcKctFA

3.5.3 Micro-comportamento3: Girar no sentido anti-horário

A figura 23 mostra o cenário usado para ensinar o robô a girar no sentido anti-horário, mantendo uma distância de 16 cm de um obstáculo cilíndrico de 200 cm (linha azul). Da mesma forma que no cenário anterior, alguns desvios nas velocidades das rodas foram efetivamente compensados, e a trajetória imitada (linha verde) demonstra que o robô realiza o comportamento esperado.

Os erros de velocidade da roda alcançados e o erro de trajetória são mostrados na figura 24. Nesse caso, o conjunto de dados de treinamento mostra que a velocidade da roda direita tem uma pequena variação se comparado com a velocidade da roda esquerda (vide as linhas vermelhas). Durante a fase de imitação, observou-se que ambas as velocidades das rodas mantêm um valor médio das velocidades demonstradas (linha verde). Além disso, também pode-se observar que tanto as trajetórias demonstradas quanto as imitadas são semelhantes, indicando que o robô aprendeu a habilidade esperada. Embora não relatado na figura, observou-se que se a posição inicial for maior que 16 cm do obstáculo central, o robô se aproxima do obstáculo e então mantém a distância esperada do cilindro.

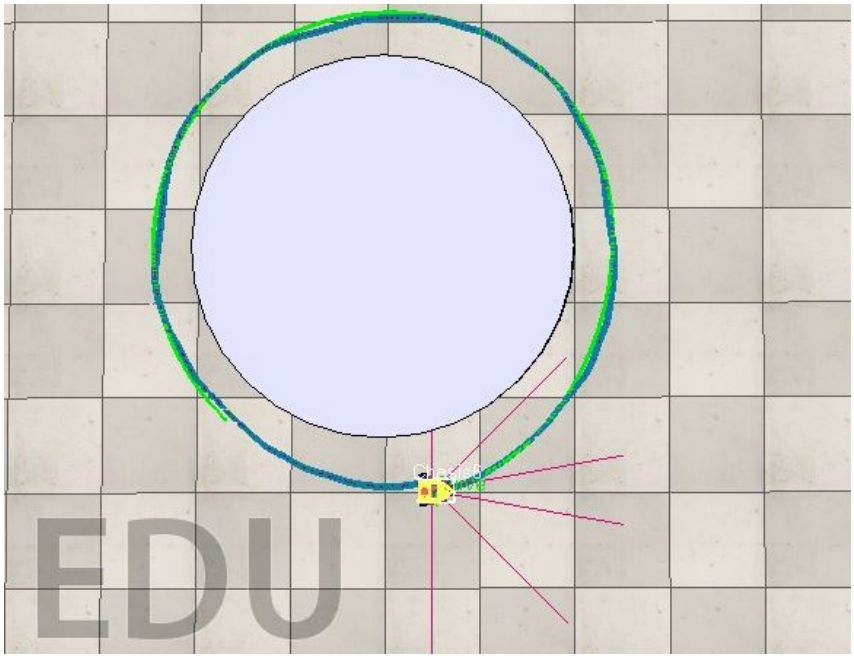


Figura 23 – Micro-comportamento3: girar no sentido anti-horário. O robô se move no sentido anti-horário mantendo uma distância de 16 cm de um cilindro com 2 metros de diâmetro. Azul: trajetória demonstrada. Verde: trajetória imitada.

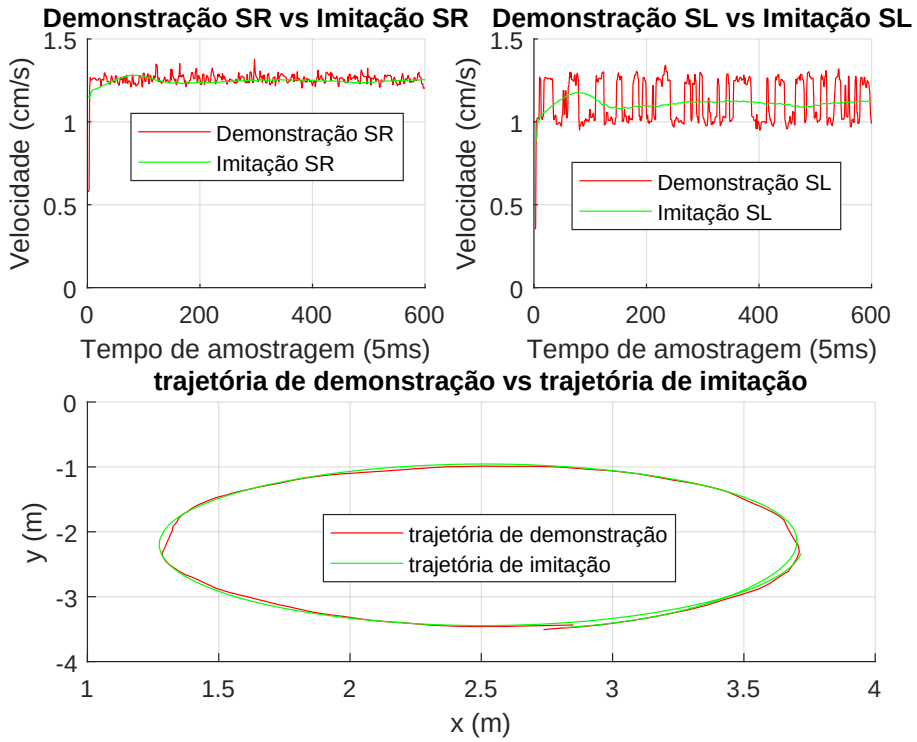


Figura 24 – Acima: velocidades direita e esquerda para o micro comportamento3: girar no sentido anti-horário. Abaixo: trajetória demonstrada (vermelho) vs trajetória imitada (verde). Detalhes podem ser vistos em https://youtu.be/FiA-vhrX1F4

A tabela 4 resume os erros de velocidade e posição dos três micro-comportamentos.

É importante destacar que o micro-comportamento 1 apresenta um erro de trajetória maior que os demais. No entanto, neste caso, os erros de velocidade na roda direita e na roda esquerda são menores do que os outros micro-comportamentos.

Tabela 4 – Erro quadrático médio (MSE) da velocidade da roda direita (RWS), velocidade da roda esquerda (LWS) e posição (X,Y) após treino com PSO.

	X	Y	RWS	LWS
Micro-Comportamento 1	3,15e-04m^2	1,0826m^2	6,36e-04(cm/s)^2	3,99e-04(cm/s)^2
Micro-Comportamento 2	0,1037m^2	0,1038m^2	1,80e-02(cm/s)^2	2,85e-02(cm/s)^2
Micro-Comportamento 3	0,3453m^2	0,3052m^2	1,85e-02(cm/s)^2	2,69e-03(cm/s)^2

3.5.4 Treinamento do *referee*

A figura 25 apresenta o cenário de teste usado para coletar o conjunto de dados para treinamento do *referee*. Este cenário combina os três micro-comportamentos em um macro-comportamento simples composto por um corredor duplo em forma de L.

Conforme explicado na seção anterior, o *referee* foi modelado por um único neurônio, contendo seis pesos e um bias, que devem ser ajustado para ensinar o robô a alternar entre os três micro-comportamentos aprendidos. Para isso, um padrão de saída esperado $\hat{P}$ foi criado manualmente, conforme mostrado na linha vermelha da figura 26. A linha azul representa a saída do *referee* após o processo de treinamento, alcançando um erro MSE de 0,0042.

Para testar o *referee* o mesmo cenário mostrado na figura 25 foi usado inicialmente. A linha verde representa a trajetória imitada. Embora sejam observadas algumas diferenças entre os caminhos demonstrados e os imitados, pode-se concluir que o robô conseguiu realizar as três habilidades aprendidas no momento esperado. Os erros de posição x e y foram de 0,6287m^2 e 0,5896m^2, respectivamente.

Figura 25 – Cenário de teste para treinamento do *referee*. O robô se move em um corredor duplo em forma de L Vermelho: trajetória demonstrada. Verde: trajetória imitada. Os detalhes do processo de imitação podem ser vistos em https://youtu.be/HFh8zHNB0SY

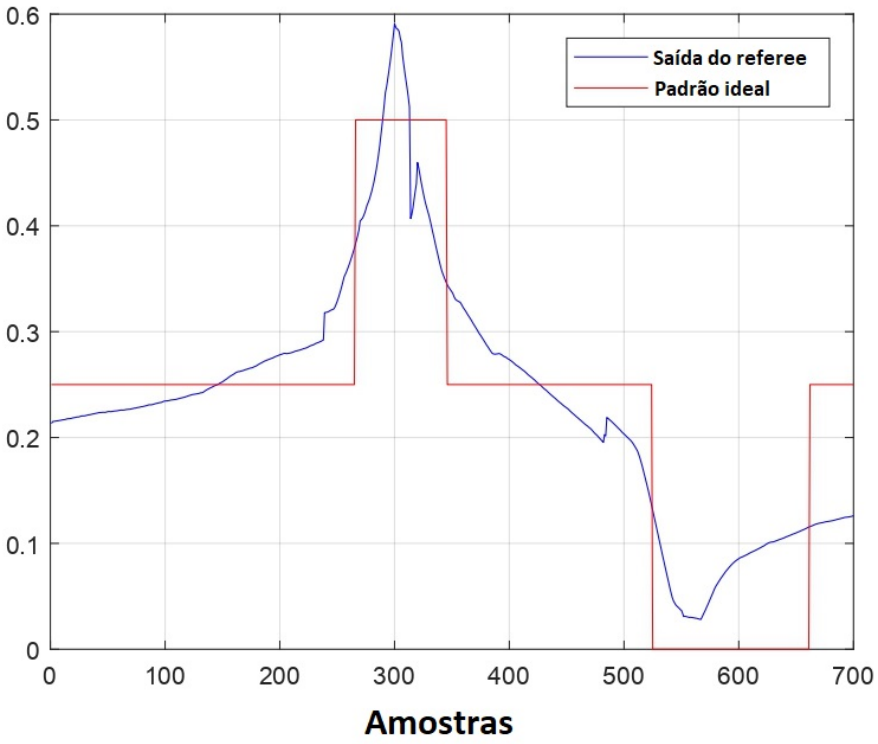


Figura 26 – Padrão para treinar o *referee*. Vermelho: padrão esperado vs Azul: saída do referee.

A próxima seção apresentará resultados para cenários desconhecidos. Para testar a robustez da solução, serão incluídos obstáculos e mudanças na orientação inicial.

3.5.5 Testes em cenários desconhecidos

Para testar o modelo LFD obtido após o treinamento com o algoritmo PSO, cinco cenários de teste foram usados para validar o comportamento do robô móvel em situações desconhecidas, conforme mostrado na figura 27. O robô móvel deve se movimentar no meio de corredores de diferentes tamanhos, com várias interseções ou evitando obstáculos.

Na figura 27a o robô movimenta-se num corredor desconhecido onde as paredes vão-se estreitando-se, neste cenário o robô inicialmente tenta achar o meio do corredor, seguidamente, quando chega na parte mais estreita do cenário, ele muda a orientação para girar em sentido horário, visando não colidir com a parede da parte esquerda. Por último o robô sai da parte estreita e tenta novamente alcançar o meio do corredor até que a simulação acaba. Cabe salientar que a posição inicial foi mudada de zero graus (como foi treinado no micro-comportamento 1) para cento e oitenta graus.

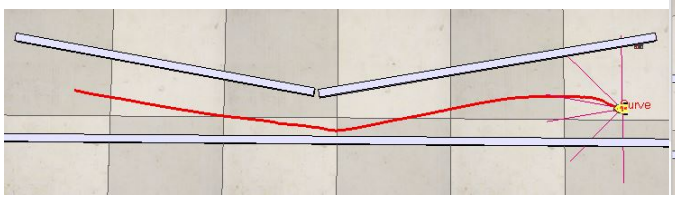
Na figura 27b o robô movimenta-se num corredor desconhecido onde as paredes estão fixas, mas são adicionados objetos no meio do corredor, neste cenário o robô inicialmente tenta achar o meio do corredor, seguidamente, quando chega na parte dos obstáculos, ele percebe que tem mais espaço na parte esquerda do corredor e muda a orientação para girar em sentido anti horário visando não colidir com o objeto que esta na sua frente. Após isto acontecer o robô tenta novamente achar o centro do corredor, percebendo novamente um obstaculo na sua frente, o qual faz atuar da forma que foi descrita anteriormente. Por último o robô sai da dos obstáculos e tenta novamente alcançar o meio do corredor até que a simulação acaba. Cabe salientar que a posição inicial foi mudada de zero graus (como foi treinado no micro-comportamento 1) para cento e oitenta graus.

Na figura 27c o robô movimenta-se num corredor desconhecido, alcançando inicialmente o meio do corredor (o qual faz rapidamente devido a que o robô incia quase no meio do corredor), seguidamente faz um giro em sentido horário com, seguidamente alcança o meio do corredor e posteriormente gira novamente em sentido horário até que a simulação finaliza.Cabe salientar que a posição inicial foi mudada de zero graus (como foi treinado no micro-comportamento 1) para cento e oitenta graus.

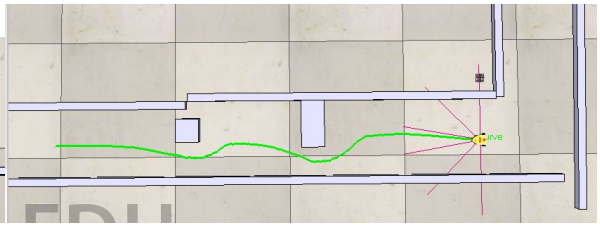
Na figura 27d robô movimenta-se inicialmente no cenário dublo em forma de L, apresentado no treinamento do *referee*, após sair deste cenário o robô esta em terreno desconhecido onde tem que girar em sentido anti horário, depois tenta ficar no meio do corredor, seguidamente gira em sentido horário, estabilizando novamente no meio do corredor e girando em sentido anti-horário novamente, por último, estabiliza no meio do corredor, até que a simulação finaliza. Este cenário é interessante devido a que tem varias mudanças de micro-comportamentos em trechos pequenos.

Na figura 27e o robô movimenta-se num corredor desconhecido onde a parede da esquerda é fixa e na parte direita do corredor tem buracos de diferentes tamanhos (alguns

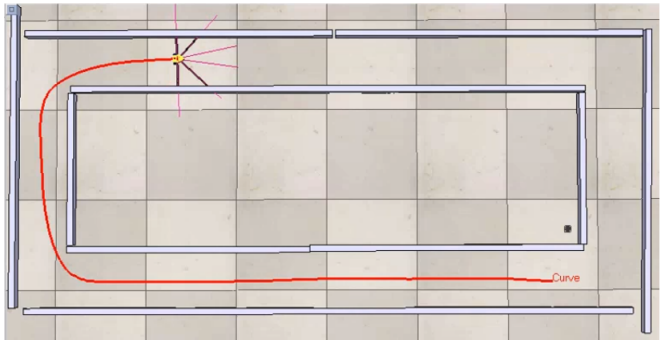
buracos são tão grandes como o robô mesmo), aqui o robô desvia a trajetória quando percebe uma grande distancias nos sensores da parte direita, mas como os buracos não são grande demais, o robô ao detetar o trecho seguinte consegue corrigir a trajetória e finalizar o cenário de forma satisfatória. Cabe salientar que a posição inicial foi mudada de zero graus (como foi treinado no micro-comportamento 1) para cento e oitenta graus.



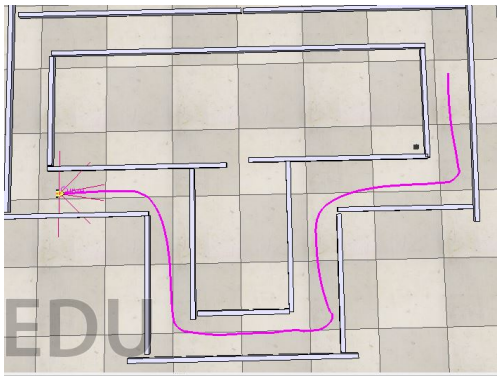
(a) Avança num corredor que vai estreitando-se, detalhes em https://youtu.be/rtZEvl5vMc4



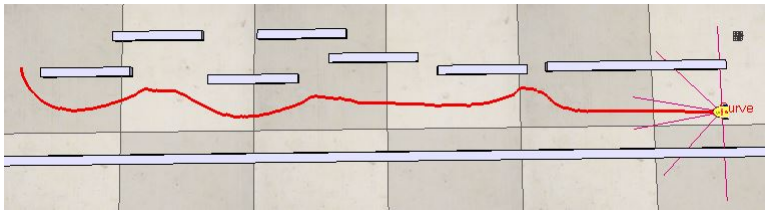
(b) Avança no meio de um corredor e evita obstáculos, detalhes em https://youtu.be/mqkncgy9Z_k



(c) Avança no meio de um corredor e gira à direita, detalhes em https://youtu.be/6tMBa65oGOQ



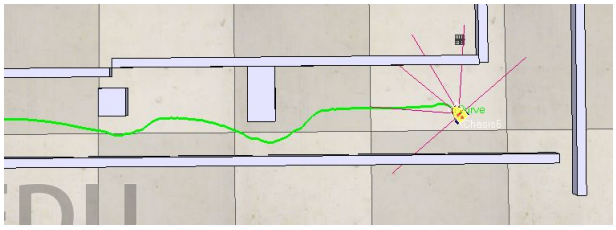
(d) Avança no meio de um corredor, gira à esquerda e gira à direita, detalhes em https://youtu.be/dvmFW4fuhW8



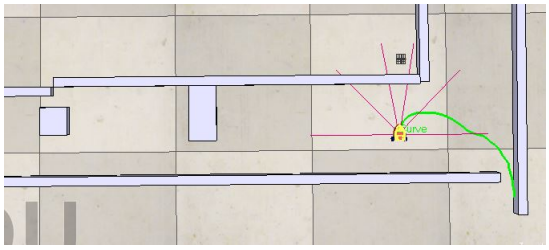
(e) Avança no meio de um corredor com buracos na parede direita, detalhes em https://youtu.be/LkwiSHQFXcQ

Figura 27 – Cenários de testes desconhecidos no software de V-rep

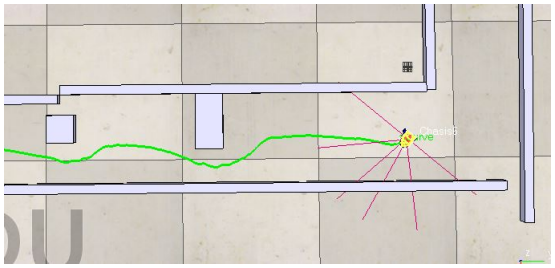
Para testar ainda mais a robustez do modelo LFD proposto, foi selecionado o segundo cenário de teste (vide figura 27b e foi alternada a orientação inicial do robô para -50, -90, +50 e +90 graus. Os resultados são apresentados na figura 28. Observou-se que, na maioria dos casos, o robô corrige a orientação inicial, girando para a direita ou para a esquerda sem colidir com as paredes, e depois se desloca pelo meio do corredor. No entanto, para a orientação inicial de −90 graus (vede figura 28b), o robô percebe mais espaço para o lado direito e então gira para a direita, colidindo com a parede externa.



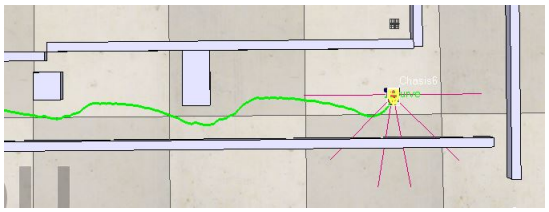
(a) Mudança posição inicial de -50 graus, detalhes em https://youtu.be/0DExhZlXoBI



(b) Mudança posição inicial de -90 graus, detalhes em https://youtu.be/4TVCcQVD8Ec



(c) Mudança posição inicial de 50 graus, detalhes em https://youtu.be/cCT7goqG9x0



(d) Mudança posição inicial de 90 graus, detalhes em https://youtu.be/e2AnH3F_oxI

Figura 28 – Cenários de testes desconhecidos no software de V-rep mudando posição inicial.

3.6 Construção mecânica do robô *Maria*

O projeto CAD do robô *Maria* foi implementado a partir de manufatura aditiva usando uma impressora Prusa I3 com um bico de impressão de 0,4 mm e ácido poli-lático (PLA) de 1,75 mm. O tempo de impressão foi de aproximadamente doze horas, alcançando um peso aproximado de 200 gramas. O peso total do robô, incluindo as baterias, foi de 547 gramas.

O robô construído pode ser observado na figura 29. O custo total do chassis é de aproximadamente 80,00 reais.

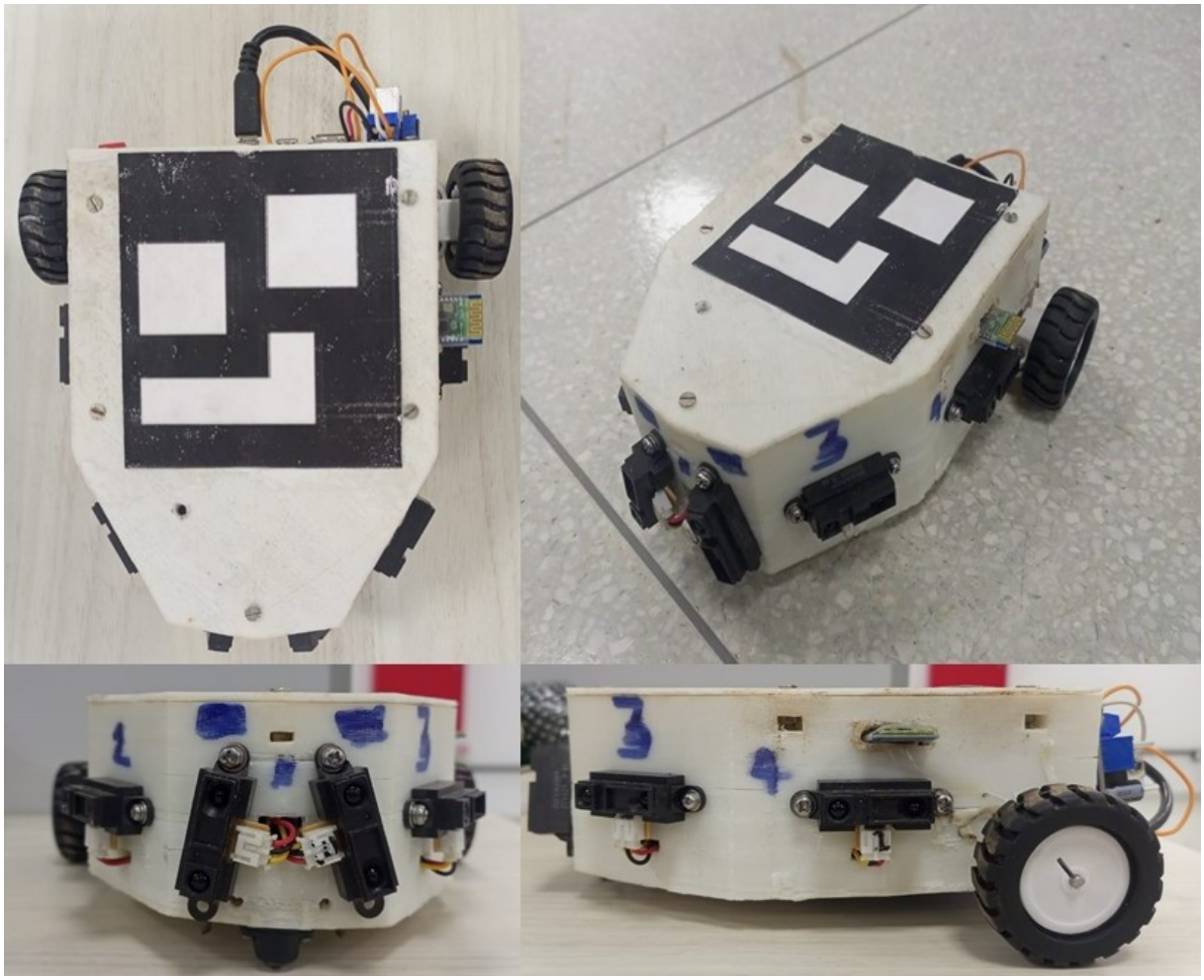


Figura 29 – Robô *Maria* construído com PLA utilizando manufatura aditiva em impressão 3D.

Na tabela 5 são listados cada um dos componentes eletrônicos com seu valor em reais (R) para a construção do robô *Maria*. Foi usada a cotação do dólar e preços no mês de julho de 2022. Com esta lista de componentes é possível padronizar o robô *Maria*.

Tabela 5 – Componentes eletrônicos utilizados para fabricar o robô *Maria*

Item	Carateristica	Unidades	Valor Unitario	Valor Total
1	sensores Sharp GP2Y0A21YK0F	4	66,40	265,6
2	Motores pololu de 6V, 220 RPM,e relação de150:1	2	161	322
3	Roda, encoder e suporte pololu	2	59,15	118,3
4	FPGA Minized	1	1.283	1.283
5	Arduino UNO	1	110	110
6	Driver LM298N	1	26	26
7	Bluetooth Arduino	1	33	33
8	4 Baterias 4.2V REF 18650 com carregador	1	55	55
9	Regulador de tensão DC-DC LM317	1	17	17
10	Roda Caster	1	5	5
11	Parte Mecânica	1	80	80
			Total	2.314,9

3.6.1 Criação da interface de controle

Conforme mencionado anteriormente, a interface de controle do usuário foi criada utilizando o site do MIT aPP Inventor (criado pelo Instituto Tecnológico de Massachusetts). Este site é totalmente gratuito e permite criar aplicações móveis de forma fácil e rápida.

A interface gráfica da aplicação do site do MIT pode ser observada na figura 30 na parte esquerda, já no telefone pode ser observada no centro da figura, e por último a programação da aplicação móvel feita em blocos é observada à direita.

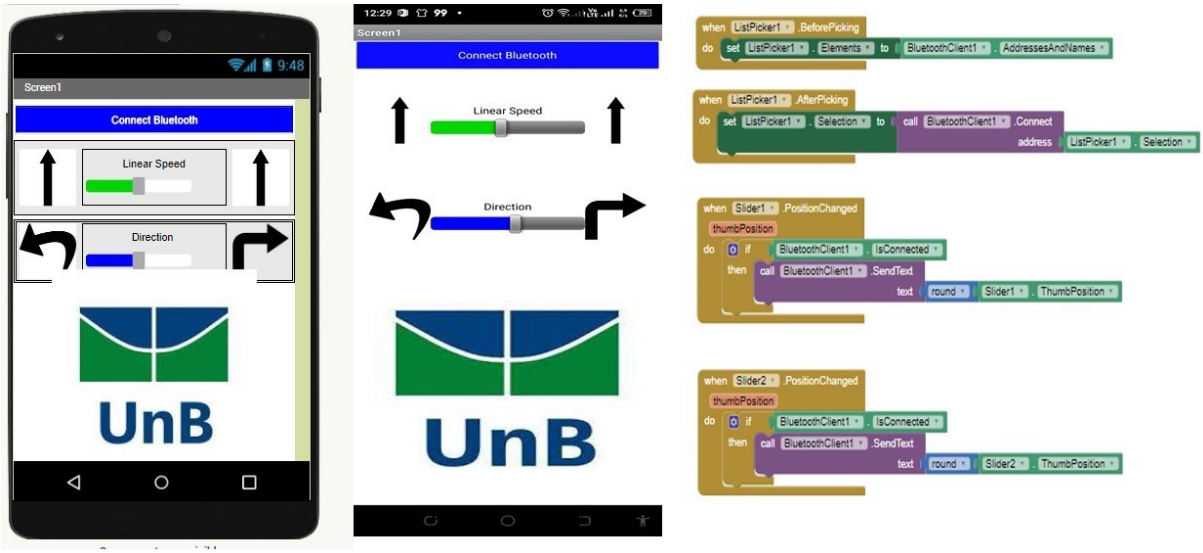


Figura 30 – Aplicação móvel para o controle do robô *Maria*.

4 Co-projeto *hardware-software* da metodologia LFD

A metodologia LFD foi implementada utilizando redes neurais artificiais treinadas pelo algoritmo PSO. Esta implementação foi feita utilizando o kit de desenvolvimento SoC FPGA Minized, o qual conta com um chip Zynq da Xilinx dividido em um sistema processador (PS) ARM Cortex A9 a 666 Mhz e em uma lógica programável (PL) com um FPGA Artix7. O co-projeto HW-SW (*hardware-software*) pode ser observado na figura 31. No PL foram implementadas a rede neural, a aquisição dos encoders, os controladores de velocidade Proporcional-Integral (PI) e os módulos de PWM (*Pulse Width Modulation*). No PS, foram implementados o algoritmo bioinspirado, e rotinas para a comunicação com o módulo bluetooth e o Arduino.

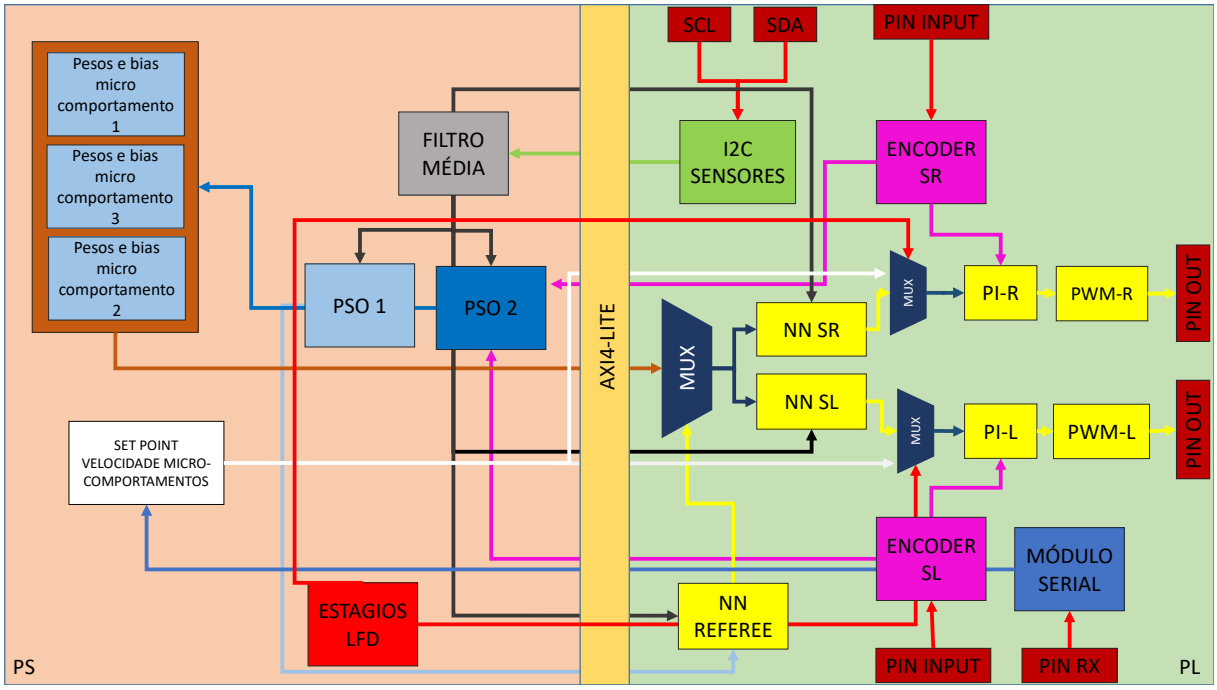


Figura 31 – Metodologia LFD implementada no Soc FPGA Minized.

4.1 Projeto em software

O desenvolvimento do código principal foi feito em linguagem C, na qual o algoritmo que implementa o LFD foi implementado conforme o diagrama de fluxo da figura 32. Inicialmente o programa faz a inicialização de variáveis. Seguidamente é realizado sequencialmente o ensinamento dos micro-comportamentos, o treinamento e a imitação dos mesmos. Após a imitação de cada micro-comportamento o usuário pode escolher entre o

retreinamento ou o ensinamento de outro micro-comportamento. Uma vez que a imitação de todos os micro-comportamento desejados forem aprovados, o algoritmo passa para o estágio de treinamento do *referee*. Concluído o treinamento do *referee*, o robô imita o macro-comportamento desejado em um cenário de teste e, se não for aprovado, pode iniciar um novo treinamento do *referee*. Após a imitação do *referee* ser aprovada, o robô está pronto para ser validado em ambientes mais complexos que incluam os micro-comportamentos ensinados.

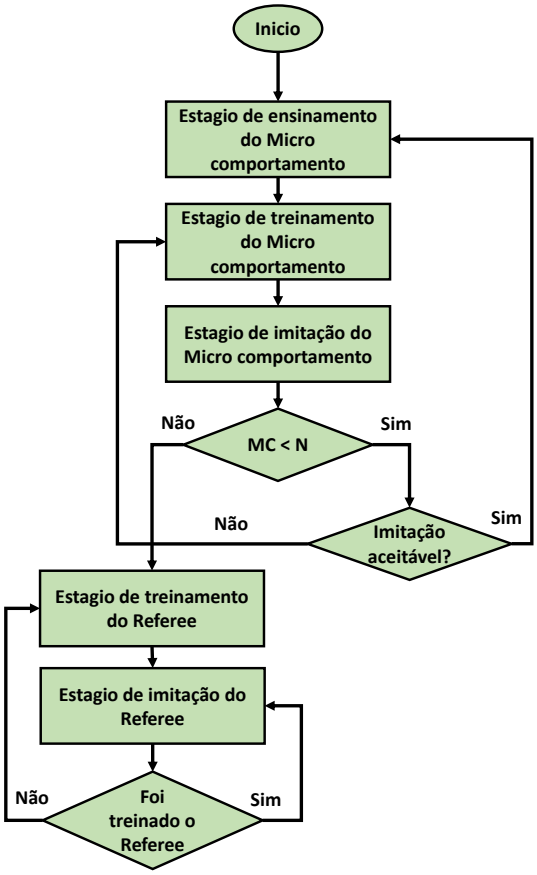


Figura 32 – Diagrama de fluxo da metodologia LFD implementada em C.

Durante o processo de ensinamento, o robô é controlado através do aplicativo do telefone, que envia para o *SoC FPGA* o *setpoint* de velocidade de cada roda. O controlador PI, embarcado em hardware, produz uma ação controle que posteriormente é convertida em um sinal PWM.

O controlador de velocidade PI foi sintonizado usando o lugar geométrico das raízes e um modelo da função de transferência do motor CC, obtido experimentalmente através do toolbox *System Identification* do Matlab. O valor de PWM foi usado como entrada (começando

em 1 e indo até 100 com passos de 1) e como saída a velocidade da roda medida em RPM. Nesta sintonização os valores de ganhos da roda direita foram de $kp = 0{,}16634$ e $ki = 4{,}5934$, e os valores de ganho para a roda esquerda foram de $kp = 0{,}12035$ e $ki = 3{,}5323$, com um tempo de amostragem de $62{,}5ms$. Após o processo de demonstração de cada micro-comportamento, o processador armazena em memória o conjunto de dados de entrada e saída. Os dados dos sensores de distância infravermelhos são filtrados usando um filtro de media móvel com janela de cinco amostras.

Após os ensaios inicias, percebeu-se que a memória on chip do ARM de 256 KBytes é insuficiente para armazenar todos os conjuntos de dados necessários. Visando contornar tal problema, foram desativados os dois sensores infravermelhos frontais e, portanto, o número de entradas da rede diminuiu de seis para quatro entradas. Com essa configuração foi possível armazenar 150 amostras (cada uma com quatro entradas e duas saídas) de cada micro-comportamento.

No estagio de treinamento os pesos e bias da rede SLP (vide figura 16) são ajustados pelo algoritmo PSO embarcado no processador ARM, o qual acelera em hardware a avaliação da função custo, comunicando-se com a mesma por meio do protocolo *on chip AXI4-Lite.*

No estagio de imitação, os controladores de velocidade PI recebem o *setpoint* direto da rede neural. Para isso, um multiplexador controlado pelo código principal no ARM, seleciona qual valor de *setpoint* deve ser direcionado ao PI (vide figura 31, no estagio de ensinamento utiliza o *setpoint* do dispositivo *bluetooth* e no estagio de imitação utiliza o *setpoint* das redes neurais).

As configurações dos algoritmos PSO implementados no ARM para treinamento dos micro-comportamentos e do *referee* são apresentadas na tabela 6. Note-se que as configurações mudaram com respeito a simulação, isto foi devido a que em simulação (que foi uma simulação cinemática e não dinâmica) muitos parâmetros não foram levados em consideração (como atrito das rodas com o chão, o peso do robô e o V-rep utilizou um controlador PID). mas aqueles parâmetros de simulação foram o ponto de partida na implementação.

Tabela 6 – Parâmetros do PSO2 para treinamento do controlador SLP em C e do PSO1 para o treinamento do *referee*

Parâmetros	**PSO2(SLP)**	**PSO1(*referee*)**
Número de partículas S	30	30
dimensões N	10	5
Fator de inercia w	[0,9 - 0,3]	[0,9 - 0,3]
Coeficiente cognitivo c_1	2,05	2,05
Coeficiente social c_2	2,05	2,05
Limite máximo X_{max}	2	10
Limite mínimo X_{min}	-2	-10
Numero de iterações Max_{iter}	300	700

4.2 Projeto em *hardware*

Os blocos implementados no PL são a rede neural SLP adaptativa (vide figura 17), o controlador PI, o módulo de estimação de velocidade e o módulo de PWM. Os blocos de comunicação *AXI-I2C* e comunicação *AXI-UART* foram aproveitados do repositório de IPs da Xilinx.

4.2.1 Arquitetura do neurônio

A figura 33 apresenta a estrutura interna do neurônio utilizado para implementar a rede neural SLP adaptativa. Observe-se a implementação do neurônio em ponto flutuante usando um multiplicador e dois somadores de 27 bits com 8 bits de exponente e 18 bits de mantissa.

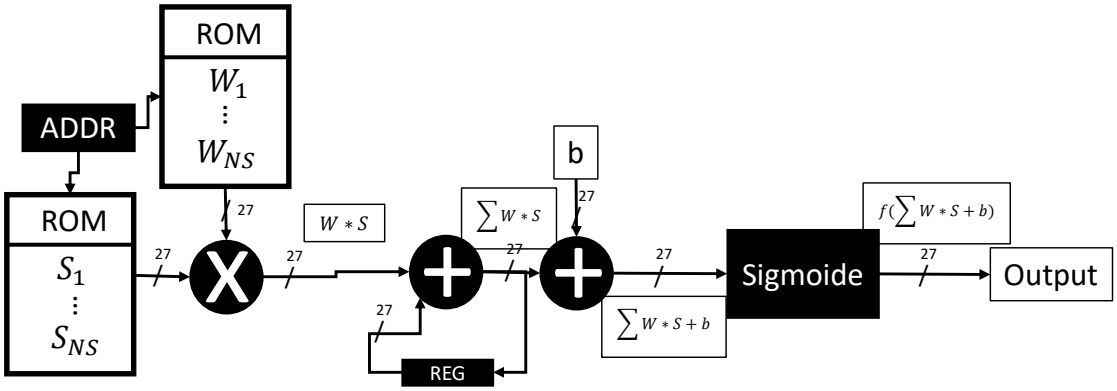


Figura 33 – Diagrama de blocos do neurônio utilizado para implementar a SLP adaptativa. *S* indica os sensores utilizados , *NS* o número total de sensores utilizados, *W* indica os pesos do neurônico, *ADDR* é o controlador das memorias ROM e *b* indica o bias.

4.2.2 Arquitetura da função Sigmoide

A figura 34 apresenta a estrutura interna da função Sigmoide implementada mediante interpolação lineal. Observa-se a implementação da equação da reta em ponto flutuante usando um multiplicador e um somador de 27 bits com 8 bits de exponente e 18 bits de mantissa. Os valores da inclinação e do intercepto são armazenados em LUTs e são direcionados à arquitetura da equação da reta através de dois multiplexadores, controlados por um comparador.

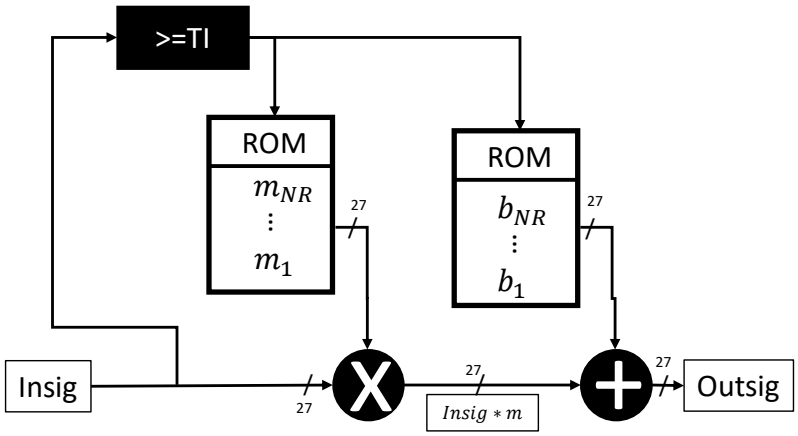


Figura 34 – Diagrama do bloco da função Sigmoide utilizando interpolação linear. NR indica o número de retas usadas na interpolação, m e b indicam a inclinação e o intercepto da reta, por último Tl é o trecho da interpolação da função Sigmoide, neste caso 100 trechos de linha foram utilizados para implementar a função Sigmoide.

É importante destacar que o código VHDL da função Sigmoide é gerado automaticamente desde uma ferramenta desenvolvida em Matlab, na qual o usuário pode escolher os seguintes parâmetros: a) tamanho da mantissa e do expoente do palavra; b) o número de retas em cada região da Sigmoide (foram usadas três regiões, a saber, inferior esquerda, centro, superior direita); c) valor máximo e mínimo de avaliação da função Sigmoide. Com esse conjunto de parâmetros, o usuário pode usar a precisão numérica como parâmetro de projeto do circuito digital da Sigmoide.

4.2.3 Arquitetura do controlador de velocidade PI

A figura 35 apresenta a estrutura interna do bloco de controle de velocidade PI, o qual utiliza três somadores e três multiplicadores em ponto flutuante em um pipeline de cinco estágios. As constantes de KP, KI e Δt são direcionadas desde o PS da Zynq por meio da interface *AXI4-Lite*.

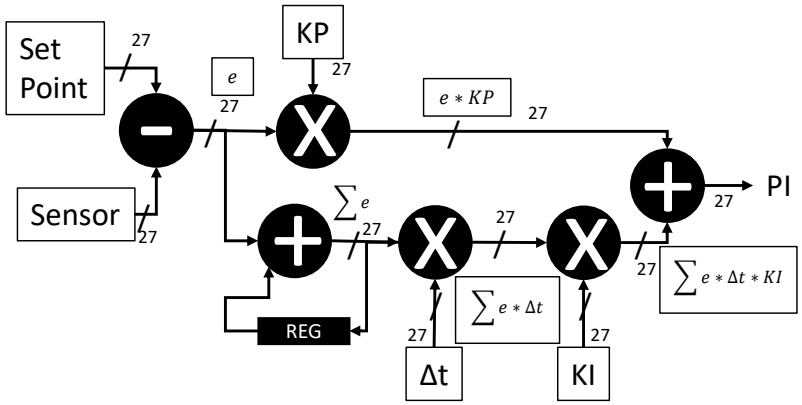


Figura 35 – Diagrama de blocos do controlador PI.

4.2.4 Implementação do bloco de estimação de velocidade

No intuito de realizar a adquisição das velocidades das rodas o enconder da família Pololu *enc01a* foi utilizado. Este encoder foi colocado em cada uma das rodas e o código de adquisição das velocidades foi programado em VHDL e embarcado no kit de desenvolvimento Minized, a velocidade é adquirida cada 0.5 segundos em $\frac{tics}{seg}$ (mudança de 0 para 1) e seguidamente é convertida para RPM esta conversão é apresentada na equação 4.1, onde NT é o número de tics adquiridos no espaço de tempo mencionado anteriormente

$$vel_{RPM} = \frac{NT * tics}{seg} * \frac{5 * RPM * seg}{tics} \tag{4.1}$$

4.3 Protocolo experimental

Para realizar a avaliação da metodologia LFD proposta foram desenhados seis cenários distintos, implementados com os três micro-comportamentos ensinados, conforme mostrado na tabela 7. Os cenários do protocolo experimental podem ser observados na tabela 8. A parte cinza é a configuração de espaço livre onde o robô pode se movimentar.

Tabela 7 – Cenários de treinamento dos micro-comportamentos para a metodologia LFD.

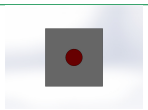
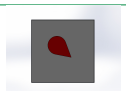
Num	Cenário	Gráfica	Observações
1	Micro-comportamento 1: Avançar para frente		- O robô tenta ir recto num corredor de 62 cm de comprimento -150 amostras foram adquiridas
2	Micro-comportamento 2: Girar no sentido horário		-Gira no sentido horário num cilindro de 70 cm de diâmetro -150 amostras foram adquiridas -Não esta fechado o quadro de treinamento
3	Micro-comportamento 3: Girar no sentido anti-horário		-Gira no sentido anti-horário num cilindro de 70 cm de diâmetro -150 amostras foram adquiridas -Não esta fechado o quadro de treinamento

Tabela 8 – Protocolo experimental para o robô *Maria*.

Num	Cenário	Gráfica	Observações
1	U Recto		-Testa-se o Micro-comportamento 1 ensinado -A posição inicial muda -A orientação inicial muda -Se faz dois testes (32 tentativas) estatísticos, entrando para girar em sentido anti-horário e o outro em sentido horário
2	U Curvo		-Testa-se dois micro-comportamentos (1 e 2 ou 1 e 3) -A posição inicial muda -A orientação inicial muda -Se faz dois testes (32 tentativas) estatísticos, entrando para girar em sentido horário e outro para girar em sentido anti-horário
3	L Curvo		-Testa-se os três micro-comportamentos ensinados -A posição inicial muda -A orientação inicial muda -Se faz um estatístico, gira em sentido horário, seguidamente avança para frente e depois gira em sentido anti-horário
4	Corredor Mudando abertura		-Testa-se os três Micro-comportamento ensinados -A posição inicial muda -A orientação inicial muda -Se faz um estatístico, e pode ser observado o robô em cenários desconhecidos
5	Cíclico com figura desconhecida		-Testa-se os três micro-comportamentos ensinados -A posição inicial muda -A orientação inicial muda -Se faz um estatístico, e pode ser observado o robô em cenários desconhecidos
6	U Semi Curvo		-Testa-se dois (aproximadamente) Micro-comportamentos ensinados -A posição inicial muda -A orientação inicial muda -Se faz dois testes (32 tentativas) estatísticos, entrando para girar em sentido anti-horário e o outro em sentido horário direita

Com a ideia de realizar um analises de trajetórias, o robô *Maria* precisa ser detectado

para estimar a posição (X e Y), esta estimação é feita mediante visão computacional, neste sentido um marcador (vide figura 36) ArUco(PHAM et al., 2021) foi colocado na parte superior do robô. Nos cenários uma câmera (fixa e sempre com a mesma câmera) colocada a uns 3 metros de distancia do chão permite fazer a adquisição das imagens para realizar a estimação de posição. A câmara encaminha as imagens para Matlab e posteriormente um algoritmo implementado em *Python* permite adquirir a posição do robô colocando no centro do marcador um ponto vermelho e nas bordas do marcador linhas verdes.

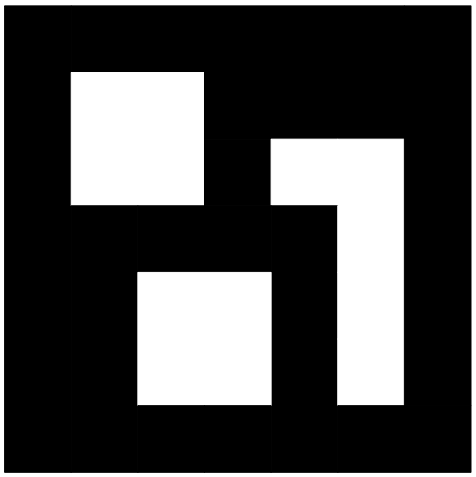


Figura 36 – Marcador ArUco colocado na parte superior do robô *Maria*

5 Análise de resultados

Este capítulo apresenta os resultados obtidos tanto na implementação física do co-projeto *hardware-software* proposto no capítulo anterior, como da execução do protocolo experimental. Inicialmente são apresentados dados de consumo de recursos, estimação da dissipação de potencia, reporte de Timing e de comparação de desempenho se comparado com uma implementação em software. Posteriormente, se apresenta uma validação experimental e estatística do comportamento do robô móvel nos diversos cenários propostos.

5.1 Consumo de recursos

O consumo de recursos em termos de LUTs (do inglês *LookUp Table*), FFs (do inglês *Flip Flops*), DSPs (do inglês digital signal processor) do controlador de velocidade PI é apresentado na tabela11. O consumo total foi de 1.304 LUTs, 504 FFs, e 3 blocos DSPs, representando 9,06%, 1,75% e 4,55% do total dos recursos, respectivamente.

Tabela 9 – Consumo de recursos do controlador PI implementado em VHDL

Controlador PI	LUTs %	DSP %	Flip Flops %
Somador 1	333 2,31%	0 0%	51 0,18%
Somador 2	351 2,44%	0 0%	51 0,18%
Somador 3	132 0,92%	0 0%	51 0,18%
Multiplicador 1	98 0,68%	1 1,52%	30 0,1%
Multiplicador 2	110 0,76%	1 1,52%	30 0,1%
Multiplicador 3	196 1,36%	1 1,52%	30 0,1%

O consumo de recursos do neurônio e seus componentes internos é apresentado na tabela 10. O consumo total foi de 2307 LUTs, 1668 FFs e 2 blocos DSPs, os quais representam, respectivamente 16,02%, 5,79%, e 3,03% dos recursos disponíveis no chip selecionado.

Tabela 10 – Consumo de recursos do neurônio implementado em VHDL

Neurônio	Slice LUTs Slice LUTs%	DSP DSP%	Flip Flops Flip Flops%
Somador 1	334 2,32%	0 0%	51 0,18%
Somador 2	1.232 8,56	0 0%	51 0,18%
Multiplicador	229 1,59%	1 1,52%	30 0,1%
Sigmoide	334 2,32%	1 1,54%	82 0,28%

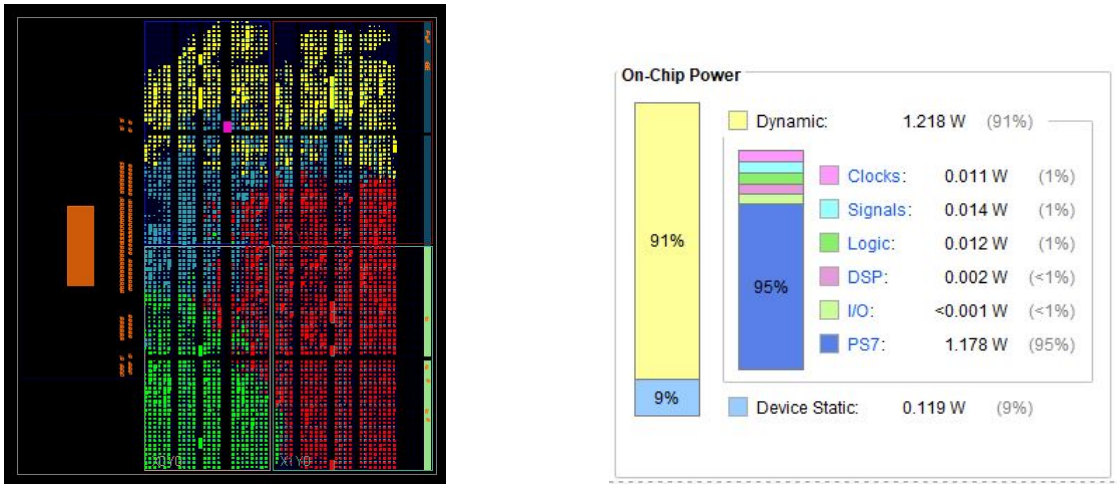
A tabela 11 apresenta um resumo do consumo total de recursos no PL do co-projeto *hardware-software* integrado.

Tabela 11 – Consumo de recursos dos módulos implementados no PL da Minized.

Modulo	LUTs %	DSP %	Flip Flops %
PI-R	1.304 9,06%	3 4,55%	504 1,75%
PI-L	1.301 9,03%	3 4,55%	504 1,75%
REFEREE	2.526 17,54 %	2 3,03 %	1.146 3,98%
PWM-R	99 0,69%	0 0%	207 0,72%
PWM-L	99 0,69 %	0 0%	207 0,72%
NN SR	2.235 15,52 %	2 3,03%	1.668 5,79%
NN SL	2.307 16,02 %	2 3,03%	1.668 5,79%
ENCODER SL	142 0,99%	0 0%	279 0,97%
ENCODER SR	142 0,99 %	0 0%	279 0,97%
SENSORES I2C	372 2,58%	0 0%	357 1,24%
MÓDULO SERIAL	89 0,62%	0 0%	108 0,38%
OTHERS	3.046 21,12 %	0 0%	2.066 7,99%
TOTAL	**13.662** **94,88%**	**12** **18,18%**	**8.993** **31,23%**

Na figura 37a é apresentado o layout dos circuitos propostos mapeados na Zynq 7007S do kit Minized. A cor amarela representa os módulos de velocidade (PI-R,PI-L,PWM-L,PWM-R, ENCODER SL, ENCODER SR), a cor vermelha representa os neurônios NN SR e NN SL, por último em cor verde se tem o neurônio do *referee*.

A dissipação de potencia total estimada foi de 1,336W, dos quais a potência estática dissipada foi de 0,119W e a potência dinâmica de 1,218W com uma confiança de estimação baixa (*LOW*). A figura38b apresenta o gráfico de consumo de energia obtido pelo Vivado.

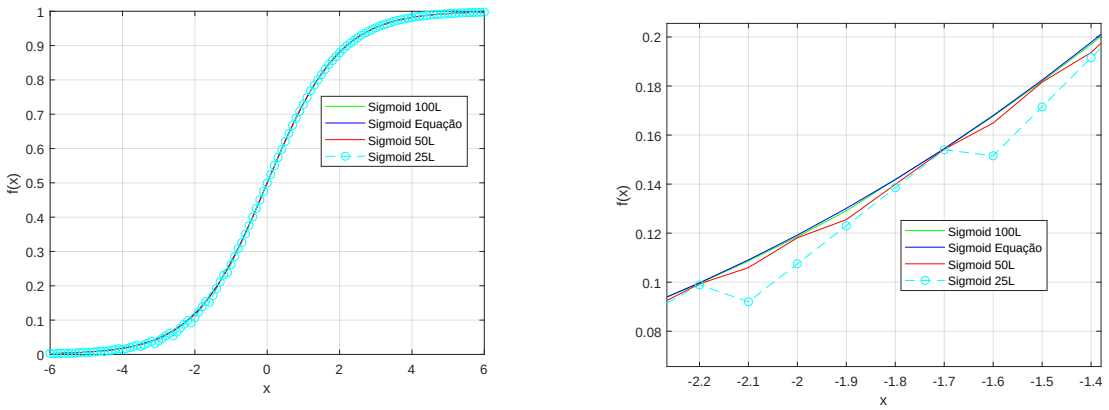


(a) Consumo de recursos (b) Consumo de energia

Figura 37 – Consumo de recursos e energia da metodologia LFD na FPGA Minized.A cor amarela representa os módulos de velocidade (PI-R,PI-L,PWM-L,PWM-R, ENCODER SL, ENCODER SR), a cor vermelha representa os neurônios NN SR e NN SL e a cor verderepresenta o neurônio do *referee*

5.2 Função Sigmoide

Conforme descrito na seção 4.2 foi usada uma função de ativação sigmoidal aproximada por interpolação linear para cada um dos neurônios. Nessa abordagem, mais retas apresentam uma maior precisão alcançada. A figura 38 apresenta três aproximações da função sigmoide com 25, 50, e 100 retas. Neste trabalho foi escolhida a aproximação com cem retas, alcançando um erro de $8{,}5205 * 10^{-8}$ se comparado com a função matemática implementada em Matlab (64 bits).



(a) Sigmoide (b) Sigmoide Zoom

Figura 38 – Função Sigmoide implementada no software de Matlab.

O erro médio quadrático (MSE) da função Sigmoide interpolada em comparação com a equação teórica em Matlab é apresentada na tabela 12. Na tabela também se apresenta o MSE da implementação em *hardware* da função Sigmoide usando cem retas e uma

representação aritmética em ponto flutuante de 27 bits. a função que descreve o MSE pode ser observado em 5.1

$$MSE = \frac{\sum(y - \hat{y})^2}{nm} \tag{5.1}$$

onde y é o valor desejado, $\hat{y}$ é o valor obtido e nm é o número de amostras utilizadas

Tabela 12 – MSE da função Sigmoide

Comparação	MSE
100L Matlab vs Equação Matlab	$8{,}5205 * 10^{-8}$
50L Matlab vs Equação Matlab	$1{,}0267 * 10^{-6}$
25L Matlab vs Equação Matlab	$1{,}6754 * 10^{-5}$
100L Matlab vs 100L VHDL	$1{,}4543 * 10^{-7}$

5.3 Tempo de execução da metodologia LFD

Os circuitos propostos para o co-projeto *hardware-software* foram efetivamente mapeados no PL usando uma frequência de *clock* de 50MHz. O relatório de *Timing*, demonstrou uma folga do tempo de *Hold* de 0,016ns, e uma folga do tempo *Setup* de 0,352ns. O relatório de *Timing* é apresentado na figura 39

Figura 39 – Relatorio de *Timing* no Vivado

A tabela 13 apresenta os tempos de execução dos algoritmos PSO para treinamento dos micro-comportamento e do *referee*.

Tabela 13 – Tempo de execução do treinamento da rede SLP adaptativa.

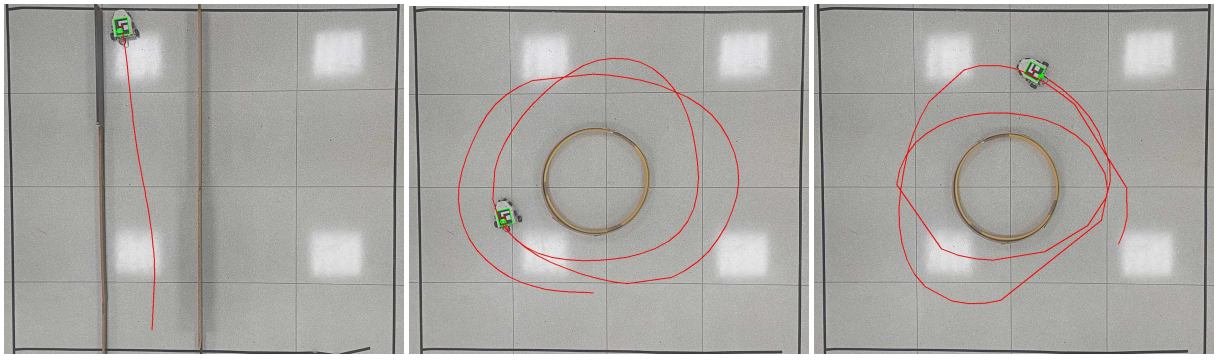
Função	Tempo de Execução Co-projeto HW-SW	Tempo de Execução em ARM	Fator de aceleração
PSO do micro comportamento	9,1344s	11,8817s	1,3001
PSO do Referee	2,630s	6,8754s	2,61

Observou-se que a aquisição de cada sensor requer 4,09 ms, portanto, o *loop* de controle do robô com quatro sensores esta limitado a um tempo de amostragem de pelo menos $4 \times 4{,}09 = 16{,}36ms$. Por outro lado, embora o algoritmo PSO para treinamento dos micro-comportamentos pode ser realizado de forma online, o seu tempo de execução de aproximadamente 9s impede a aplicação para sistemas de dinâmica rápida, este tempo pode ser diminuindo mediante a redução de número de partículas e número de iterações, más pode que os resultados de sintonização sejam não desejáveis. No capítulo 6 se apresentam propostas para reduzir o tempo de execução.

5.4 Testes dos micro-comportamentos

Os cenários reais para implementar a metodologia LFD proposta podem ser observados na figura 40. Aqui tentou-se replicar os mesmos cenários utilizados na simulação. Pode-se observar um vídeo do ensinamento de cada comportamento acessando aos links presentes na legenda das figuras. Note-se que o ensinamento dos micro-comportamentos não foi perfeito devido à dificuldade de controlar o robô desde o aplicativo do telefone e ao tempo de estabilização do controlador de velocidade PI.

O cenário do micro-comportamento 1 foi construído utilizando madeira de diferentes cores e tamanhos, os cenários para os micro-comportamentos 2 e 3 foram construídos utilizando papel kraft para fazer o objeto cilíndrico. Ressalta-se que os micro-Comportamentos 2 e 3 não estão fechados (não tem paredes externas no cenário).



(a) Micro-comportamento 1: Avançar para frente

(b) Micro-comportamento 2:Girar no sentido horário

(c) Micro-comportamento 3:Girar no sentido anti-horário

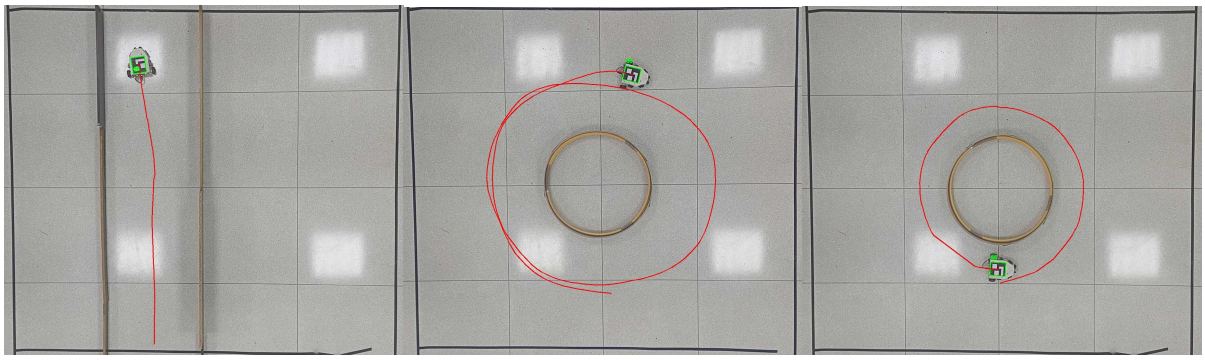
Figura 40 – Ensinamento dos três micro-comportamentos. Maiores detalhes em: (a) https://youtu.be/tn056LnRWm8; (b) https://youtu.be/$5_BW8vQeUlw$; e (c) https://youtu.be/b4yFFMxhV5Y

Os pesos achados para cada comportamento são apresentados na tabela 14, aqui para cada micro-comportamento (MC) foram achados oito pesos (pelos quatro sensores) e dois bias uma para cada roda.

Tabela 14 – Pesos (W) e bias (B) para o controle das rodas direita (R) e esquerda(L)

Cenario	BR	BL	W1R	W2R	W3R	W4R	W1L	W2L	W3L	W4L
MC 1	0,7638	0,7802	0,1659	-0,2066	-1,7049	0,4433	0,4188	-0,3576	0,4908	0,4412
MC 2	-0,3890	-0,6967	-0,4635	-1,9869	-0,4920	-0,0806	-1,4249	0,2379	-0,9382	0,1291
MC 3	0,5779	-1,1991	0,6136	0,9814	-1,2788	0,3824	0,1633	0,0027	-1,9845	0,9939

A imitação de cada micro-comportamento pode ser observada na figura 41. Destaca-se que a rede neural além de aproximar o micro-comportamento desejado, filtra o ruído dos dados de entrada, permitindo abstrair o comportamento ensinado pelo usuário. Cada micro-comportamento ensinado foi imitado 16 vezes, e os resultados estatísticos podem ser observados na tabela 17. Na tabela, um sucesso é entendido como a realização de um comportamento sem colisão com os obstáculos internos ou externos.



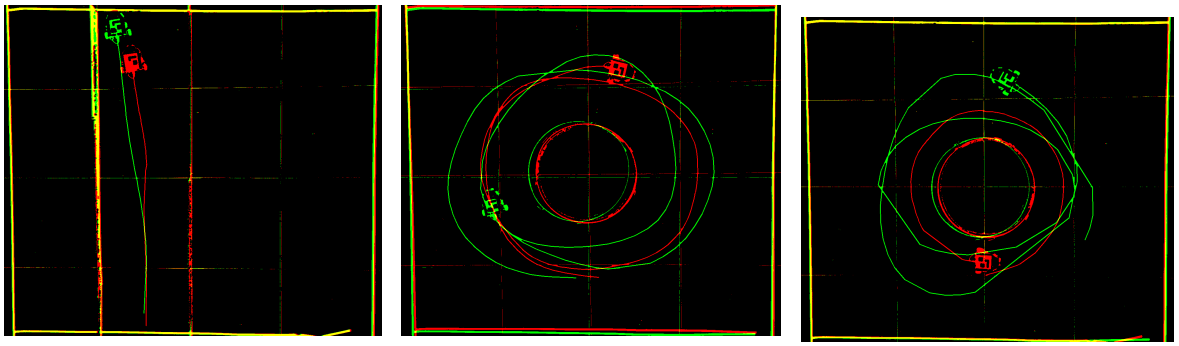
(a) Micro-comportamento 1: Avançar para frente

(b) Micro-comportamento 2:Girar no sentido horário

(c) Micro-comportamento 3:Girar no sentido anti-horário

Figura 41 – Imitação dos três micro-comportamentos, para mais informação de (a) ver https://youtu.be/bPwJmUNV3Wk, de (b) ver https://youtu.be/1HgkeuzbUME e de (c) ver https://youtu.be/Vxq26MT9qpc

Por último para ter uma melhor observação das trajetórias ensinadas e imitadas, a figura 42 apresenta a superposição das trajetórias para cada micro-comportamento. No intuito de realizar uma comparação numérica, a tabela 15 apresenta as áreas calculadas baseadas nas imagens dos micro-comportamentos de giro no sentido horário e giro no sentido anti-horário, estas áreas podem ser observadas na figura 43. Já para o micro-comportamento de avançar para frente, foi calculada a distancia acumulada do robô com respeito a parede esquerda, como apresentado na figura 44.

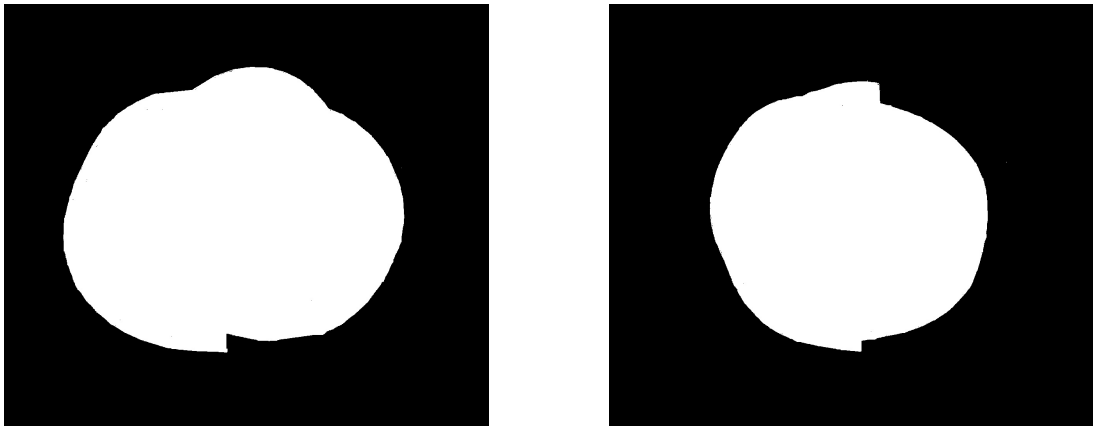


(a) Micro-comportamento 1: Avançar para frente

(b) Micro-comportamento 2: Girar no sentido horário

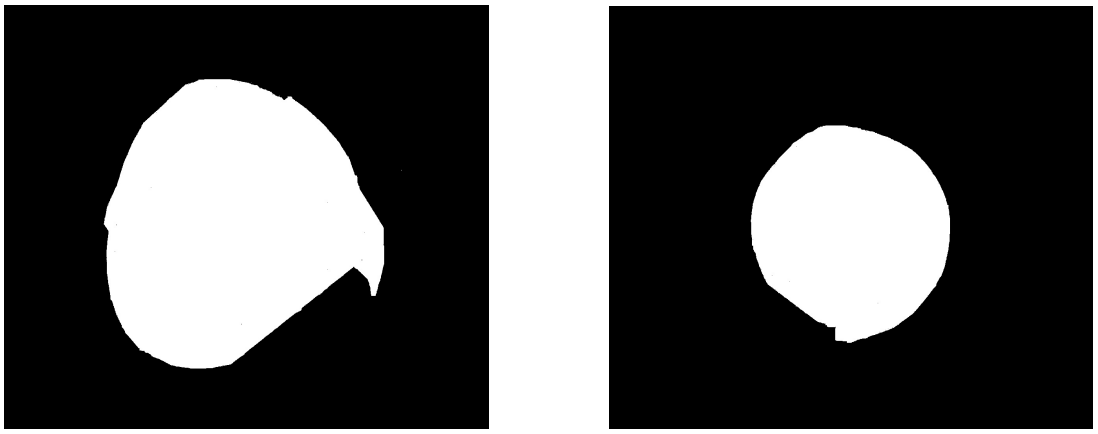
(c) Micro-comportamento 3: Girar no sentido anti-horário

Figura 42 – Trajetórias ensinadas e imitadas dos três micro-comportamentos. Verde: trajetória ensinada. Vermelho: trajetória imitada.



(a) Área micro-comportamento 2 ensinamento

(b) Área micro-comportamento 2 imitação



(c) Área micro-comportamento 3 ensinamento

(d) Área micro-comportamento 3 imitação

Figura 43 – Áreas ensinadas e imitadas dos micro-comportamentos 2 e 3.

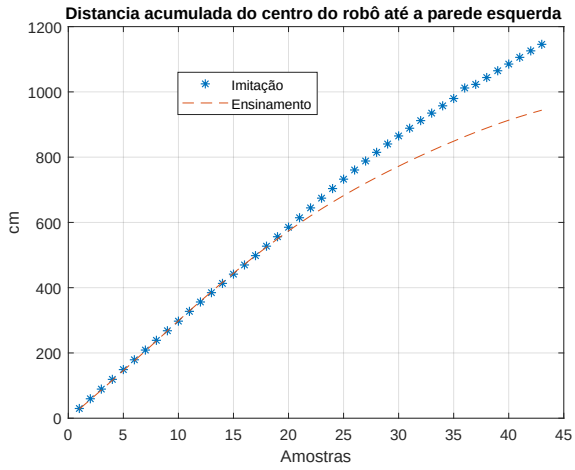


Figura 44 – Distancia acumulada da trajetória percorrida pelo robô com respeito a parede esquerda do micro-comportamento 1.

Na figura 44 pode-se observar de uma maneira gráfica como a imitação do micro-comportamento de avançar para frente fica mais centrada no corredor do que a trajetória ensinada. Dessa forma, a distancia acumulada na trajetória imitada é mais linear do que na ensinada, demonstrando a capacidade da rede SLP adaptativa em aproximar este micro-comportamento.

A tabela 15 apresenta o cálculo estimado das áreas calculadas utilizando o número de pixeis que estão dentro das trajetórias ensinadas e imitadas. Utilizou-se como referencia os quadrados cerâmicos do chão que têm uma área $63cm \times 63cm$, portanto, cada pixel da imagem tem um área de $0{,}042cm^2$. Dessa forma, nos cenários cíclicos o erro de imitação pode ser estimado subtraindo as áreas das trajetórias.

Tabela 15 – Calculo do erro dos micro-comportamentos.

Cenário	Área ou distancia acumulada Treinamento	Área ou distancia acumulada Imitação	Erro
Micro-comportamento 1	$943cm$	$1{,}145cm$	$202cm$
Micro-comportamento 2	$20.448cm^2$	$15.618cm^2$	$4.830cm^2$
Micro-comportamento 3	$15.733cm^2$	$8.860{,}6cm^2$	$6.872{,}4cm^2$

5.5 Testes com *referee*

Nesta seção são apresentados os resultados de desempenho do robô *Maria* em cenários desconhecidos do protocolo experimental. Primeiramente são apresentados resultados em cenários nos quais foram conduzidos testes estatísticos. Posteriormente, são apresentados resultados em cenários fora do protocolo experimental, para os quais não se coletaram

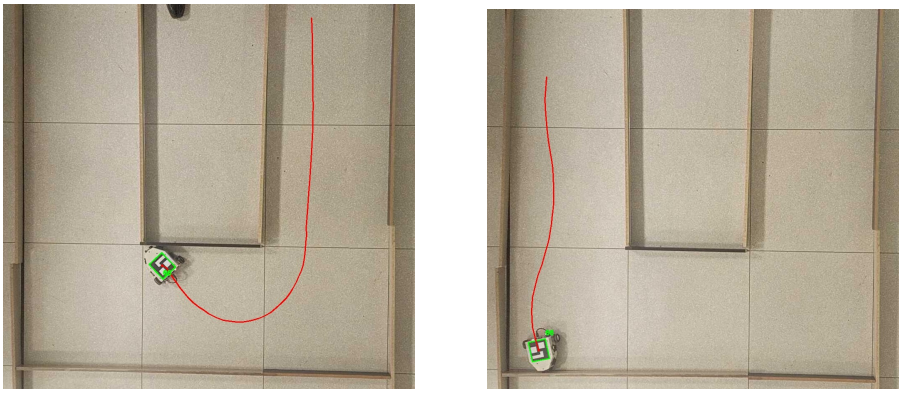
resultados estatísticos. Na tabela 16 são apresentados os valores dos pesos e bias achados do PSO para o neurônio do *referee*

Tabela 16 – Pesos e bias do neurônio do *referee*

	B	W1	W2	W3	W4
referee	0,5394	-2,3809	0,5057	-6,3658	-4,3377

5.5.1 Testes nos cenários do protocolo experimental

Na figura 45 são apresentados os resultados no cenário 1 constituído de três trechos do micro-comportamento de avançar para frente, separados com paredes com ângulos retos.



(a) Avançar para frente e girar em sentido horário

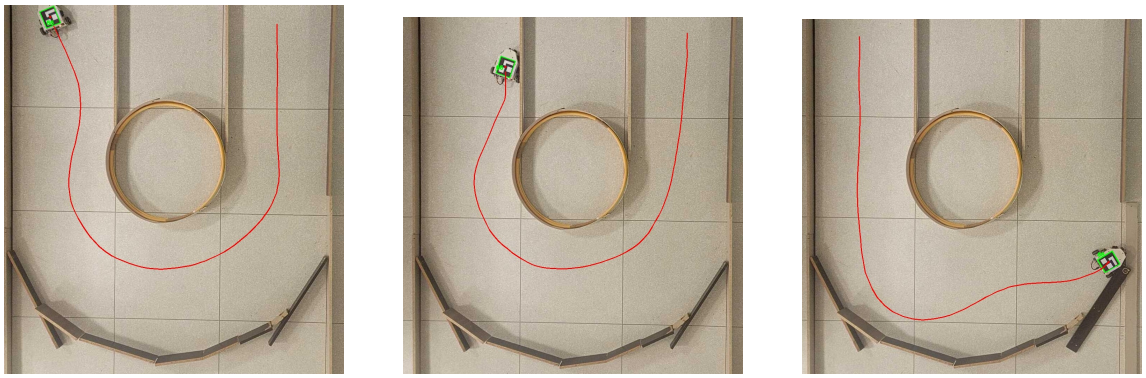
(b) Avançar para frente e girar em sentido anti-horário

Figura 45 – Robô *Maria* no cenário estatístico 1, para mais informação de (a) ver https://youtu.be/NXtfXTQJsxk, e de (b) ver https://youtu.be/QrdgqmFOhCw

Os resultados no cenário 1, apresentam que o robô colidiu com as paredes internas ou externas, sem conseguir completar a trajetória esperada. Em particular este cenário é um desafio para a rede SLP adaptativa, pois o cenário com uma U fechada com ângulos retos, dificultam coletar amostras suficientes para o *referee* detectar a mudança do Micro-comportamento de girar em sentido horário para o micro-comportamento de avançar para frente (vide 45a). Adicionalmente, a falta dos sensores frontais favorecem o aumento de velocidade em alguns trechos do percurso, de forma que o robô tem pouco tempo para realizar possíveis correções de trajetória (vide 45b).

O segundo cenário é constituído de dois micro-comportamentos, a saber, avançar para frente e girar no sentido horário ou anti-horário. Os resultados podem ser observados na figura 46. Novamente, a falta dos sensores frontais foi crítica para regular a velocidade do robô, de forma que tivesse maior tempo para realizar correções de trajetória quando o *referee* realiza a mudança de micro-comportamentos. Por outro lado, após analisar os vídeos pode-se concluir que existe uma tendencia do robô ir para a direita, possivelmente porque

os pesos da rede do lado direito tem maior contribuição do que os pesos do lado esquerdo do robô.



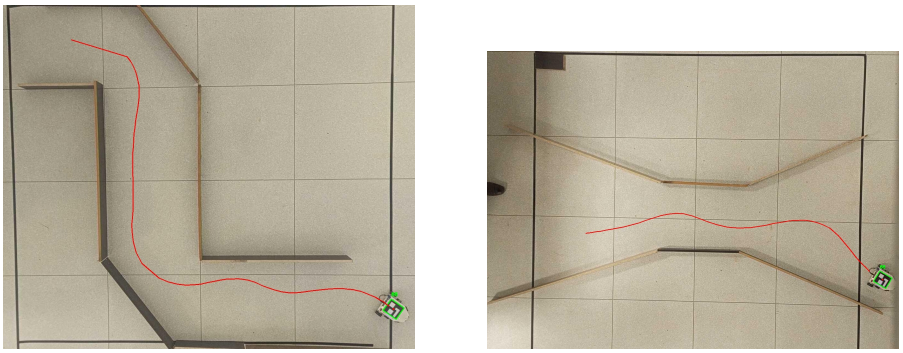
(a) Avançar para frente e girar em sentido horário (Sucesso)

(b) Avançar para frente e girar em sentido horário (Falha)

(c) Avançar para frente e girar em sentido anti-horário

Figura 46 – Robô *Maria* no cenário estatístico 2, para mais informação de (a) ver https://youtu.be/X8PWSvFTh5w, de (b) ver https://youtu.be/xySUXmsFdV0 e de (c) ver https://youtu.be/oIOtPFCnFiY

No cenário três foi testada uma composição dos três micro-comportamentos ensinados, tendo sido observado mais falhas do que acertos, provavelmente pela falta dos sensores frontais (que favorecem velocidades altas) e pela tendência de girar à direita. No cenário quatro foi testado apenas um micro-comportamento, porém com mudança na largura do corredor. Neste cenário observou-se um comportamento satisfatório.



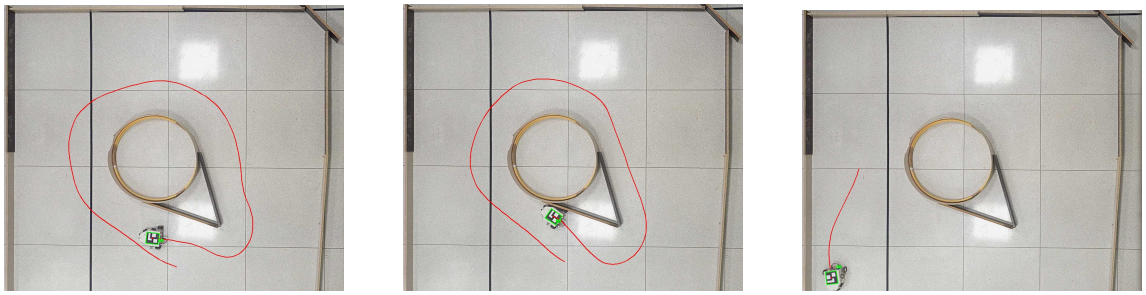
(a) Cenário estatístico 3

(b) Cenário estatístico 4

Figura 47 – Robô *Maria* no cenário estatístico 3 e 4, para mais informação de (a) ver https://youtu.be/y0urUrrdnHI, e de (b) ver https://youtu.be/006aJa$_D$yA0

O cenário cinco é constituído de um cenário fechado (este cenário tem paredes externas formando um quadro. É desconhecido para o robô) com obstáculo desconhecido no meio do cenário. O robô testou uma parte conhecida (a parte circular do objeto no meio do corredor), o restante do cenário é desconhecido para *Maria*, aqui conseguiu-o observar que quando girava de forma horária, o robô na parte reta, tenta-se desestabilizar o qual gera que algumas vezes o robô colida com o objeto por culpa deste trecho, mas a grande maioria das vezes o robô consegue estabilizar e desta forma afastar-se do objeto estranho. Quando foi

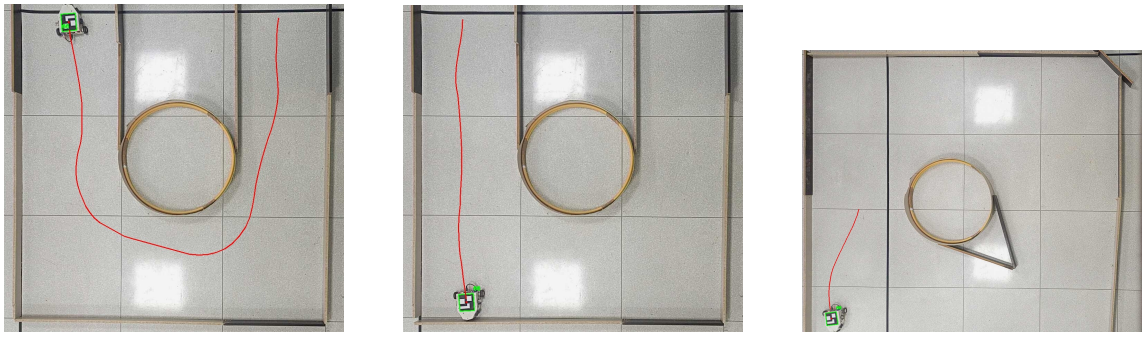
testado o giro anti-horário o robô não conseguiu-o fazer uma ótima trajetória, simplesmente ele desconheceu totalmente o cenário colidindo com as paredes que fechavam o cenário.



(a) Giro no sentido horário (Sucesso) (b) Giro no sentido horário (Falha) (c) Giro no sentido anti-horário

Figura 48 – Robô *Maria* no cenário estatístico 5, para mais informação de (a) ver https://youtu.be/7fbCDhtUiUE, de (b) ver https://youtu.be/IK297kOQNFA e de (c) ver https://youtu.be/ZD9fBjfSOhs

No ultimo cenário do protocolo de teste, foi testado um cenário intermédio do primeiro e o segundo do protocolo de teste. Neste cenário fora colocado um circulo para ao momento de girar (retirando as curvas retas do cenário), mas mantendo reto a parte de fora do cenário. Neste cenário o robô consegue na maioria das vezes (girando no sentido horário) realizar a trajetória desejada de forma satisfatória, com isto pode-se dizer que o robô tem sérios problemas ao momento de se encarar com curvas retas. Quando foi testado para girar em sentido anti-horário o robô não conseguiu nem uma vez fazer a trajetória desejada, neste sentido em trabalhos futuros tentara-se melhorar os giros nesta direção para testar em outros cenários.



(a) Giro no sentido horário (Sucesso) (b) Giro no sentido horário (Falha) (c) Giro no sentido anti-horário

Figura 49 – Robô *Maria* no cenário estatístico 6, para mais informação de (a) ver https://youtu.be/6GuWRIiXQ2E, de (b) ver https://youtu.be/Pnq$_N$*UIrGpk* e de (c) ver https://youtu.be/HFix2h-KHbs

Os resultados dos testes estatísticos dos cenários implementados podem ser observados na figura 17, nesta tabela os primeiros três cenários foram as imitações dos três micro-comportamentos ensinados, tendo o maior porcentagem de acertos em comparação aos outros seis cenários de testes estatísticos.

Tabela 17 – Resultados estatísticos dos cenários

Cenário	Sucessos	Erros	Total Exp
Micro-comportamento 1	13 81,25%	3 18,75%	16
Micro-comportamento 2	16 100%	0 0%	16
Micro-comportamento 3	12 75%	4 25%	16
U Reto Giro horário	0 0%	16 100%	16
U Reto Giro anti-horário	0 0%	16 100%	16
U Curvo Giro horário	15 93,75%	1 6,25%	16
U Curvo Giro anti-horário	2 12,5%	14 87,5%	16
Corredor Mudando Abertura	12 75%	4 25%	16
L Curvo	2 12,5%	14 87,5%	16
Cíclico com figura desconhecida	8 50%	8 50%	16
U semi curvo Giro horário	9 56,25%	7 43,75%	16
U semi curvo Giro anti-horário	0 0%	16 100%	16

5.5.2 Outros cenários testados

Foram testados outros cenários desconhecidos para o robô *Maria* os quais são apresentados a seguir, aqui só foi implementado um experimento por cada cenário. A ideia de cada cenário era testar a robustez da metodologia LFD em outros cenários desconhecidos. Os cenários podem conter obstáculos ou simplesmente formas que não foram ensinados nos micro-comportamentos, estos cenário foram construídos utilizando madeira de diferentes cores e tamanhos e o objeto cilíndrico utilizado no cenário de treinamento.

A figura 50a apresenta o cenário quatro do protocolo de testes com dois obstáculos na entrada da parte estreita. Nestes cenário o robô consegue evitar os dois obstáculos ficando no meio deles e após isto girar a direita para sair do corredor.

A figura 50b apresenta o cenário quatro do protocolo de testes com três obstáculos, adicionando um obstaculo mais ao cenário anterior. Na entrada da parte estreita foram colocados dois obstáculos e na saída foi colocado o terceiro. Note-se que neste cenário o robô até a parte estreita faz o mesmo que no cenário passado, já quando encontra o terceiro obstaculo faz o giro em sentido horário mas rápido tentando evadir o obstaculo terceiro e

seguidamente o robô acaba o recorrido saindo do corredor.

A figura 50c apresenta um cenário que vai-se estreitante até chegar numa largura fixa, aqui o robô tenta inicialmente estar perto da parede direita, seguidamente quando entra no corredor com largura menor, o robô oscila ate sair do corredor.

A figura 50d é o cenário inverso do anterior onde o robô começa num corredor estreito ate chegar num corredor que vai ficando mais grande, neste cenário o robô inicialmente fica no meio do corredor até perceber que o corredor esta-se ampliando, aqui o robô começa a se aproximar à parede do lado direito até sair do corredor.

A figura 50e este é um cenário composto de quatro corredores de diferentes tamanhos que tem uma intercepção comum no centro, aqui, o robô oscila inicialmente no corredor e quando chega na intercepção ele faz a escolha de girar no sentido horário, apos isto, o robô fica no meio, até sair do cenário.

A figura 50f este é um cenário composto de dois corredores de diferentes tamanhos que tem uma intercepção comum no centro, aqui, o robô tenta-se manter no meio do corredor e quando percebe o espaço na esquerda (o segundo corredor) ele escolhe continuar avançando para frente e acaba o cenário saindo do corredor por onde começo.

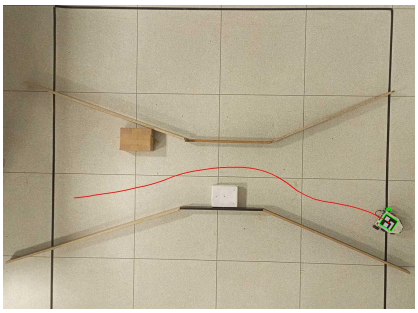
A figura 50g este é um cenário composto de quatro corredores de diferentes tamanhos que tem uma intercepção comum no centro, aqui, o robô oscila inicialmente no corredor e quando chega na intercepção ele faz a escolha de girar no sentido horário, apos isto, o robô fica no meio, até sair do cenário.

A figura 50h este é um cenário composto de quatro corredores de diferentes tamanhos que tem uma intercepção comum no centro, aqui, o robô oscila inicialmente no corredor e quando chega na intercepção ele faz a escolha de girar no sentido horário, apos isto, o robô fica no meio, até sair do cenário.

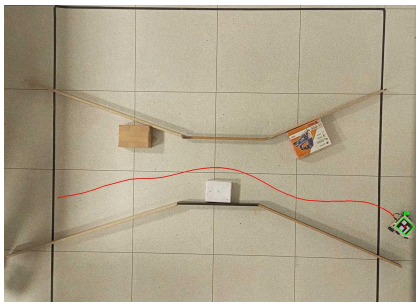
A figura 51a apresenta um cenário onde o robô inicia num corredor e seguidamente tem um giro em sentido anti-horário, este cenário foi proposto devido a que nos cenários de protocolo experimental girar em sentido anti-horário foi tudo um desafio para o robô, mas neste cenário o robô consegue inicialmente ficar no meio do corredor e depois girar em sentido horário realizando satisfatoriamente o cenário.

A figura 51b apresenta um cenário anterior onde foram adicionados obstáculos no inicio da mudança de direção do corredor e ao final do trajeto. Neste cenário o robô consegue evitar os obstáculos e sair de forma satisfatório do cenário. Nestos dois cenários pode-se dizer que embora nos cenários do protocolo experimental não funciona-se de forma ótima o giro em sentido anti-horário, o *referee* (em alguns casos) consegue fazer a mudança para girar em forma anti-horária de forma correta.

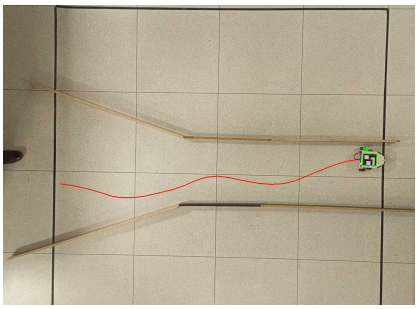
A figura 51c apresenta um cenário cíclico onde no meio foi colocado um objeto



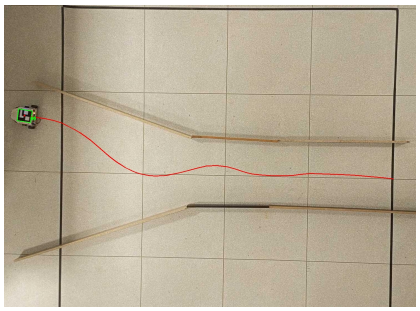
(a) Cenário desconhecido 7



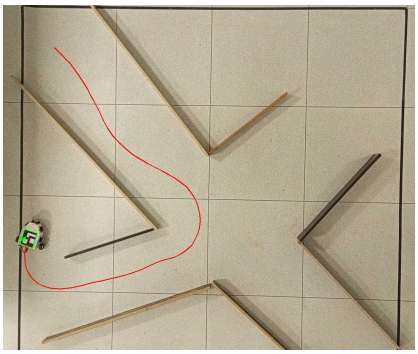
(b) Cenário desconhecido 8



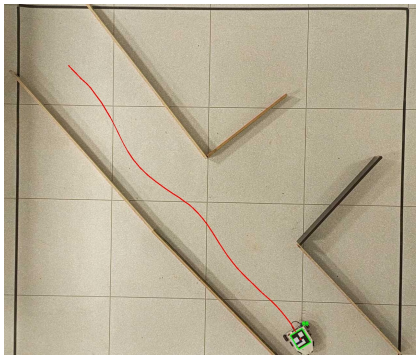
(c) Cenário desconhecido 9



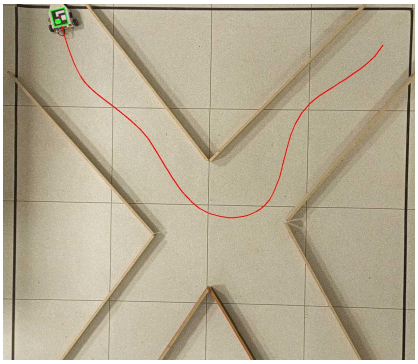
(d) Cenário desconhecido 10



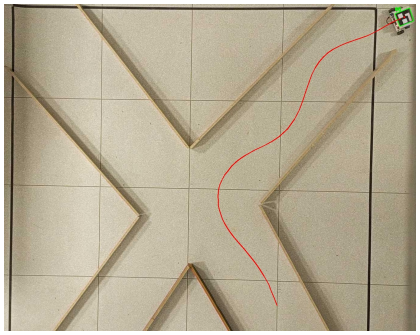
(e) Cenário desconhecido 11



(f) Cenário desconhecido 12



(g) Cenário desconhecido 13



(h) Cenário desconhecido 14

Figura 50 – Robô *Maria* nos cenários desconhecidos 7,...,14, para mais informação de (a) ver https://youtu.be/RWAxVqbU4BE, de (b) ver https://youtu.be/sfiI7bCmPa4, de (c) ver https://youtu.be/zD9dw-pg6a0, de (d) ver https://youtu.be/Ttb2CjNOo9I, de (e) ver https://youtu.be/N7SPnoIpxlM, de (f) ver https://youtu.be/PvfZgqPEQi4, de (g) ver https://youtu.be/0mi8I8JRghM e de (h) ver https://youtu.be/1U$_4UxjFU60$

desconhecido para o robô, aqui foi fechado o cenário e colocou-se o robô a girar em sentido horário conseguindo realizar a trajetória desejada, embora com um pouco de oscilação no

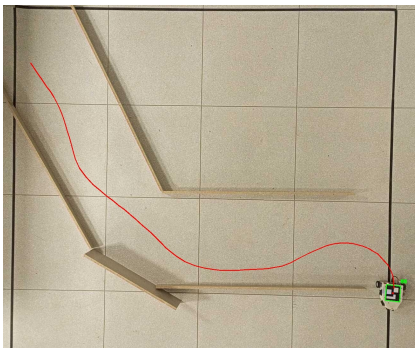
contorno do objeto.

A figura 51d apresenta um cenário cíclico similar ao cenário anterior (mesmo objeto no meio do cenário), o cenário foi fechado, só que, de forma não estruturada (a madeira for colocada de forma aleatória no cenário), gerando mais desafio para o robô, já que neste cenário teve trechos onde o espaço é reduzido e outros onde tem mais espaço de ação. O robô consegue fazer o trajeto de forma satisfatória.

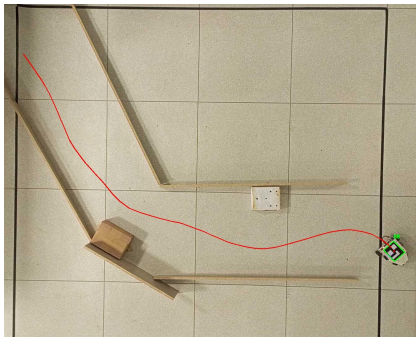
A figura 51e apresenta um cenário cíclico similar ao cenário anterior (mesmo objeto no meio do cenário e das paredes externas), aqui foram adicionados objetos no cenário, gerando mais desafio para o robô, já que ele precisa evitar os obstáculos presentes no caminho. O robô consegue evitar os obstáculos e realizar a trajetória sem nem um contratempo.

A figura 51f apresenta um cenário cíclico similar ao cenário anterior, aqui foram adicionados dois objetos no cenário (para ter um total de quatro obstáculos no cenário, dois perto das paredes externas e dois perto do objeto no meio), gerando mais desafio para o robô, já que ele precisa evitar os obstáculos presentes no caminho. Nota-se que o objeto desconhecido foi retirado para colocar o objeto utilizado nos cenários de treinamento. O robô consegue evitar os obstáculos de forma quase perfeita (colide de forma sutil com o objeto branco) e realizar a trajetória desejada.

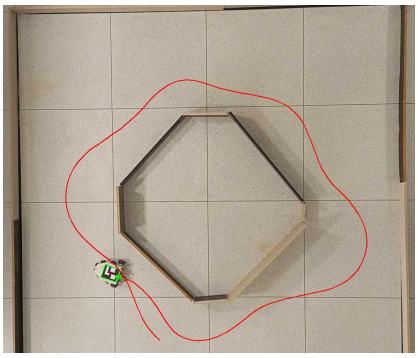
A figura 51g apresenta um cenário cíclico com as paredes externas colocadas da mesma forma que o cenário passado, só que com um objeto conhecido (embora o conjunto de objeto com paredes externas seja um cenário desconhecido para o robô). O robô consegue fazer a trajetória de forma ótima, com uma trajetória semelhante ao do Micro-comportamento 2.



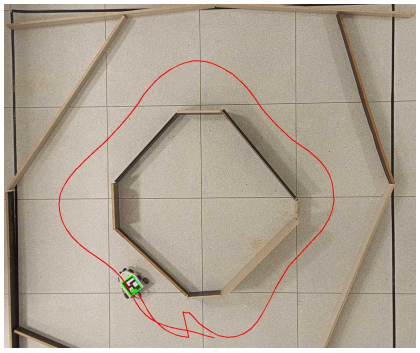
(a) Cenário desconhecido 15



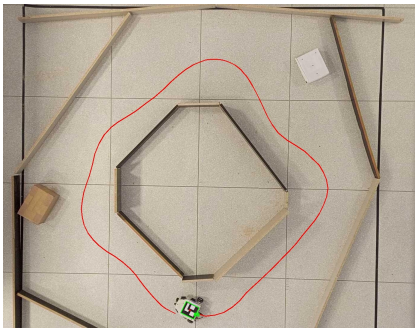
(b) Cenário desconhecido 16



(c) Cenário desconhecido 17

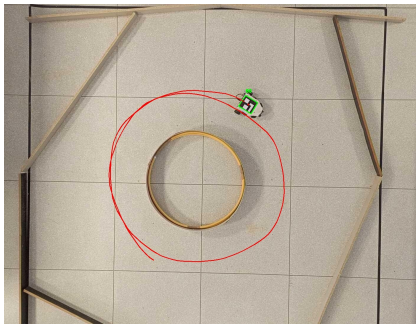


(d) Cenário desconhecido 18



(e) Cenário desconhecido 19

(f) Cenário desconhecido 20



(g) Cenário desconhecido 21

Figura 51 – Robô *Maria* nos cenário desconhecido 15,...,21, para mais informação de (a) ver https://youtu.be/MTG980CLHsA, de (b) ver https://youtu.be/6WTCpkgk5OY, de (c) ver https://youtu.be/xNnkIJNSCdE, de (d) ver https://youtu.be/JSTn6QuL45A, de (e) ver https://youtu.be/OiBsSVGaLDk, de (f) ver https://youtu.be/53PSdg1mZlE e de (g) ver https://youtu.be/-zKuhXHorjc

6 Conclusão

Nesta pesquisa foi desenvolvida uma plataforma robótica diferencial utilizada para realizar aprendizagem por demonstração mediante a utilização de *hardware* reconfigurável. Foi realizado o projeto mecânico do robô *Maria* o qual tem uma tração diferencial, e possui quatro sensores infravermelhos e dois motores de CC com encoders. As dimensões do robô foram de 120mm de largura, 160 mm de comprimento e 70 mm de altura, com peso total de 547 gramas.

A implementação da metodóloga LFD foi feita mediante a utilização de redes neurais artificiais e algoritmos bioinspirados criando uma rede neural SLP adaptativa, a qual permite mudar os pesos de conexão para alternar entre diferentes tipos de micro-comportamentos. A rede SLP adaptativa e o algoritmo de treinamento PSO foram implementados em um SoC FPGA por meio da técnica de co-projeto *Hardware-Software*, permitindo ganhar em flexibilidade e em tempo de processamento. O fator de aceleração obtido foi de 1,3 e 2,61 vezes para o treinamento da rede SLP que imita os micro-comportamentos e do neurônio do *referee*, respectivamente.

Os primeiros testes da metodologia proposta foram feitos em um ambiente de simulação, usando três micro-comportamentos. O MSE das rodas direita e esquerda para o micro-comportamento 1 foi de $6{,}36e^{-04}(cm/s)^2$ e de $3{,}99e^{-04}(cm/s)^2$, respectivamente. Para o micro-comportamento 2 foi de $1{,}80e^{-02}(cm/s)^2$ e $2{,}85e^{-02}(cm/s)^2$, respectivamente. Por fim, para o micro-comportamento 3 foi de $1{,}85e^{-02}(cm/s)^2$ e $2{,}69e^{-03}(cm/s)^2$, respectivamente. No caso do treinamento do *referee* o MSE com respeito ao padrão esperado foi de 0,0042.

Os cenários desconhecidos testados a nível de simulação foram resolvidos de forma satisfatória mediante a utilização da metodologia proposta onde o *referee* consegui realizar a mudança dos comportamentos do robô. Estas simulações permitiram validar e testar a topologia da rede neural SLP adaptativa para LFD, a qual constitui uma contribuição original no estado da arte.

Na implementação física teve um consumo de 13.362 (94,88%) LUTs, 12(18,18%) DSP e 8.993(31,23%) Flip-Flops, o tempo de execução do algoritmo PSO para o treinamento de cada micro-comportamento foi de $9{,}1344s$ e do *referee* foi de $2{,}630s$, A dissipação de potencia total estimada foi de $1{,}218W$ onde o 95% foi do ARM.

A implementação da metodologia proposta em cenários reais deu como resultado um erro de áreas para os micro-comportamentos 2 e 3 de $4.830cm^2$ e $6.872{,}4cm^2$, respectivamente e para o micro-comportamento 1 a distancia acumulada foi de $202cm$, cabe salientar que esta metodologia de avaliação para a metodologia LFD aplicada em robótica móvel, não tem sido

utilizada no estado da arte, sendo este o primeiro trabalho em implementar este calculo do erro nas trajetórias.Além disto 16 experimentos foram conduzidos no intuito de ter caculos estatísticos de cada micro-comportamento, aqui os resultados para o micro-comportamento 1 foram de 13(81,25%) sucessos e 3 (18,75%) erros, para o micro-comportamento 2 foi de 16(100%) sucessos e 0 (0%) erros, e para o micro-comportamento 3 foi de 12(75%) sucessos e 4 (25%), esta forma estatística para avaliar os comportamentos ensinados na metodologia LFD na foi descrita no estado da arte sendo esta uma contribuição original.

Seguidamente o teste da metodologia em cenários desconhecidos para o robô foi de um sucesso moderado, isto devido a que o robô em vários cenários não consegui girar em sentido anti-horário, tendo um 100% de erro nos cenários de U Semi-Curvo girando em sentido anti-horário e U Reto girando em sentido anti-horário. Mas o robô consegue resolver vários tipos de cenários desconhecidos sendo no melhor dos casos o cenário U Curvo girando em sentido horário onde teve 15(93,75%) sucesso e 1(6,25%) erro.

Esta pesquisa é a única no estado da arte que faz a implementação da metodologia LFD utilizando micro-comportamentos num Soc FPGA e cria um robô para fazer testes físicos, além de utilizar critérios de áreas, distancias acumuladas e dados estatísticos para avaliar os comportamentos aprendidos na metodologia proposta. o trabalho de (MUÑOZ et al., 2014) combina a implementação do controle em FPGA junto com a visualização do controle em simulação no *software* de Eye-Sim, neste trabalho o tempo de treinamento para a rede neural de controle utilizando o algoritmo do PSO por oposição em Hardware foi de 5,3s, sendo mais rápido que o treinamento da SLP que foi de 9,1344s. Neste trabalho também foram implementados dois neurônios para realizar o controle das velocidades do robô, diferenciando-se deste trabalho que o autor utiliza são as velocidades lineares e angulares do robô. Além disso o treinamento neste trabalho é feito online.

No trabalho de (BORGES, 2020) foi montado um robô móvel diferencial o qual utilizava Arduino para realizar o controle, a estrutura mecânica foi comprada. Aqui o autor utiliza redes neurais com o algoritmo de retropropagação do erro para treinamento no Matlab (treinamento *offline*) o qual é lento, se comparado com uma implementação em Soc FPGA. Aqui o autor utiliza 6 neurônios no estagio de simulação e 5 no estagio de implementação, os quais são 4 neurônios a mais no caso da simulação e 3 a mais na implementação comparado com este trabalho, o qual gera um maior consumo de recursos. O autor, não apresenta tempos de treinamento das redes neurais nem testes em cenários desconhecidos ao nível de implementação física.

Em (TAN, 2015) o autor apresenta um algoritmo evolutivo para ensinar 3 comportamentos mediante um algoritmo genético iterativo, neste *paper* somente são testados os comportamentos ensinados, e não são apresentados critérios de erros para cada um dos cenários. A implementação foi realizada num robô comercial como foi o robô E-Puck e não apresenta tempos de implementação de cada iteração.

Por último o trabalho de (VUKOVIĆ; MITIĆ; MILJKOVIĆ, 2015) apresenta a metodologia LFD implementada num robô Khepera II com um processador Motorola 68331 com uma frequência de 25 MHhz, desenvolvendo uma solução somente em *software*. O mencionado artigo apresentou a nível de implementação dois comportamentos diferentes nos quais faz o ensinamento três vezes. No entanto o *paper* não explora outros cenários e este não apresenta tempos de execução para realizar uma comparação numérica.

6.1 Trabalhos futuros

Para trabalhos futuros a ideia é melhorar o robô mediante a utilização de 6 sensores infravermelhos, seguidamente, utilizar atrasos e/ou a posição do robô como entradas da rede neural artificial. Isto permitirá tomar decisões em função dos estados passados e das medições atuais.

Com relação à implementação em SoC FPGA, pretende-se reduzir o consumo de recursos da FPGA mediante a utilização em aritmética de ponto fixo do controlador PI, assim como reduzir a dissipação de potencia mediante a utilização de um processador Microblaze.

Complementarmente, no intuito de fazer aplicações reais na industria, planeja-se criar uma célula de manufactura a escala, onde o robô possa transportar elementos de um ponto A para um ponto B, realizando percursos com paradas e com variações de carga. Por último, pretende-se escalar a solução para uso em robótica multiagente, onde tarefas como andar em formação, arrastras objetos, levantar objetos devem ser realizadas de forma colaborativa.

Referências

ABE, S.; GEE, A. Global convergence of the Hopfield neural network with nonzero diagonal elements. **IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing**, v. 42, n. 1, p. 39–45, 1995. DOI: 10.1109/82.363543. Citado na p. 21.

AMBHORE, S. A Comprehensive Study on Robot Learning from Demonstration. In: 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA). 2020. P. 291–299. DOI: 10.1109/ICIMIA48430.2020.9074946. Citado na p. 19.

ARGALL, B.; BROWNING, B.; VELOSO, M. Learning by demonstration with critique from a human teacher. In: 2007 2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI). 2007. P. 57–64. DOI: 10.1145/1228716.1228725. Citado na p. 20.

ARGALL, B. D.; CHERNOVA, S.; VELOSO, M.; BROWNING, B. A survey of robot learning from demonstration. **Robotics and Autonomous Systems**, v. 57, n. 5, p. 469–483, 2009. ISSN 0921-8890. DOI: https://doi.org/10.1016/j.robot.2008.10.024. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0921889008001772>. Citado na p. 19.

BERANEK, R.; AHMADI, M. A Learning Behavior Based Controller for Maintaining Balance in Robotic Locomotion. **Journal of Intelligent Robotic Systems**, v. 82, p. 189–205, 2016. ISSN 1573-0409. DOI: https://doi.org/10.1007/s10846-015-0254-7. Citado na p. 14.

BHANU, P.; BHUSHAN, C.; SUJATHA, K.; VENMATHI, M.; NALINI, A. Load frequency controller with PI controller considering non-linearities and boiler dynamics. In: INTERNATIONAL Conference on Sustainable Energy and Intelligent Systems (SEISCON 2011). 2011. P. 290–294. DOI: 10.1049/cp.2011.0376. Citado na p. 28.

BILLARD, A.; CALINON, S.; DILLMANN, R.; SCHAAL, S. Robot Programming by Demonstration. In: **Springer Handbook of Robotics**. Edição: Bruno Siciliano e Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. P. 1371–1394. ISBN 978-3-540-30301-5. DOI: 10.1007/978-3-540-30301-5_60. Disponível em: <https://doi.org/10.1007/978-3-540-30301-5_60>. Citado na p. 19.

BISHOP, C. M. **Pattern Recognition and Machine Learning (Information Science and Statistics)**. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0387310738. Citado na p. 22.

BORGES, J. **Redes neurais artificiais aplicadas em aprendizagem de trajetória em robótica móvel**. Dez. 2020. Diss. (Mestrado) – Universidade de Brasília, Brasília. An optional note. Citado nas pp. 30, 79.

CARLUCHO, I.; DE PAULA, M.; ACOSTA, G. G. An adaptive deep reinforcement learning approach for MIMO PID control of mobile robots. **ISA Transactions**, v. 102, p. 280–294, 2020. ISSN 0019-0578. DOI: https://doi.org/10.1016/j.isatra.2020.02.017. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0019057820300781>. Citado na p. 14.

CHEN, J.; ZELINSKY, A. Programing by Demonstration: Coping with Suboptimal Teaching Actions. **The International Journal of Robotics Research**, v. 22, n. 5, p. 299–319, 2003. DOI: 10.1177/0278364903022005002. eprint: https://doi.org/10.1177/0278364903022005002. Disponível em: <https://doi.org/10.1177/0278364903022005002>. Citado na p. 19.

CHEN, S.; BAI, W. Performance analysis of typical linear augmented observers for a class of MIMO systems with nonlinear uncertainty. **ISA Transactions**, v. 128, p. 316–327, 2022. ISSN 0019-0578. DOI: https://doi.org/10.1016/j.isatra.2021.11.023. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0019057821005930>. Citado na p. 14.

CHOI, Y. K.; LEE, S.-Y. Subthreshold MOS implementation of neural networks with on-chip error backpropagation learning. In: PROCEEDINGS of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan). 1993. v. 1, 849–852 vol.1. DOI: 10.1109/IJCNN.1993.714046. Citado na p. 23.

DANIEL, M.; JAVIER, A.; DEL, S. J.; SALVADOR, G.; AMIR, H.; HERRERA, F. Comprehensive Taxonomies of Nature- and Bio-inspired Optimization: Inspiration Versus Algorithmic Behavior, Critical Analysis Recommendations. **Cognitive Computation**, v. 12, p. 897–939, 2020. ISSN 1866-9964. DOI: 10.1007/s12559-020-09730-8. Disponível em: <https://doi.org/10.1007/s12559-020-09730-8>. Citado na p. 23.

DAUTENHAHN, K.; NEHANIV, C. L. (Ed.). **Imitation in Animals and Artifacts**. Cambridge, MA, USA: MIT Press, 2002. ISBN 0262042037. Citado na p. 19.

DOKEROGLU, T.; SEVINC, E.; KUCUKYILMAZ, T.; COSAR, A. A survey on new generation metaheuristic algorithms. **Computers Industrial Engineering**, v. 137, p. 106040, 2019. ISSN 0360-8352. DOI: https://doi.org/10.1016/j.cie.2019.106040. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0360835219304991>. Citado na p. 23.

DORIGO, M.; DI CARO, G. Ant colony optimization: a new meta-heuristic. In: PROCEEDINGS of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). 1999. v. 2, 1470–1477 vol. 2. DOI: 10.1109/CEC.1999.782657. Citado na p. 23.

EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. 1995. P. 39–43. DOI: 10.1109/MHS.1995.494215. Citado na p. 24.

FRIEDRICH, H.; DILLMANN, R. **Robot Programming Based On A Single Demonstration And User Intentions**. 1995. Citado na p. 19.

GAMAL, O.; CAI, X.; ROTH, H. Learning from Fuzzy System Demonstration: Autonomous Navigation of Mobile Robot in Static Indoor Environment using Multimodal Deep Learning. In: 2020 24th International Conference on System Theory, Control and Computing (ICSTCC). 2020. P. 218–225. DOI: 10.1109/ICSTCC50638.2020.9259786. Citado na p. 30.

GARCIA, D.; GORROSTIETA, E.; VARGAS SOTO, E.; RODRIGUEZ RIVERO, C.; DIAZ DELGADO, G. Learning from Demonstration with Gaussian Process Approach for an Omni-directional Mobile Robot. **IEEE Latin America Transactions**, v. 16, n. 4, p. 1250–1255, 2018. DOI: 10.1109/TLA.2018.8362164. Citado na p. 30.

GROLLMAN, D. H.; JENKINS, O. C. Dogged Learning for Robots. In: PROCEEDINGS 2007 IEEE International Conference on Robotics and Automation. 2007. P. 2483–2488. DOI: 10.1109/ROBOT.2007.363692. Citado na p. 20.

HACENE, N.; MENDIL, B. Fuzzy Behavior-based Control of Three Wheeled Omnidirectional Mobile Robot. **International Journal of Automation and Computing**, v. 16, p. 163–185, 2019. ISSN 1751-8520. DOI: https://doi.org/10.1007/s11633-018-1135-x. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494619301826>. Citado na p. 14.

HAUCK, S.; DEHON, A. **Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. ISBN 9780080556017. Citado na p. 26.

HAYKIN, S.; PRINCIPE, J. C.; SEJNOWSKI, T. J.; MCWHIRTER, J. Modeling Large Dynamical Systems with Dynamical Consistent Neural Networks. In: NEW Directions in Statistical Signal Processing: From Systems to Brains. 2007. P. 203–241. Citado na p. 21.

HOPFIELD, J. J. Neurons with Graded Response Have Collective Computational Properties like Those of Two-State Neurons. In: ARTIFICIAL Neural Networks: Theoretical Concepts. Washington, DC, USA: IEEE Computer Society Press, 1988. P. 82–86. ISBN 0818608552. Citado na p. 21.

IJSPEERT, A.; NAKANISHI, J.; SCHAAL, S. Movement imitation with nonlinear dynamical systems in humanoid robots. In: PROCEEDINGS 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292). 2002. v. 2, 1398–1403 vol.2. DOI: 10.1109/ROBOT.2002.1014739. Citado na p. 19.

JIANG, L.; WANG, W.; CHEN, Y.; JIA, Y. Personalize Vison-based Human Following for Mobile Robots by Learning from Human-Driven Demonstrations. In: 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN). 2018. P. 726–731. DOI: 10.1109/ROMAN.2018.8525791. Citado na p. 30.

KAISER, M.; DILLMANN, R. Building elementary robot skills from human demonstration. In: PROCEEDINGS of IEEE International Conference on Robotics and Automation. 1996. v. 3, 2700–2705 vol.3. DOI: 10.1109/ROBOT.1996.506570. Citado na p. 19.

KARABOGA, D.; BASTURK, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. **Journal of Global Optimization**, v. 39, p. 459–471, 2007. ISSN 1573-2916. DOI: 10.1007/s10898-007-9149-x. Disponível em: <https://doi.org/10.1007/s10898-007-9149-x>. Citado na p. 23.

KASHIF, H.; NAJIB, M. S. M.; SHI, C.; YUHUI, S. Metaheuristic research: a comprehensive survey. **Artificial Intelligence Review**, v. 52, p. 2191–2233, 2019. ISSN 1573-7462. DOI: 10.1007/s10462-017-9605-z. Disponível em: <https://doi.org/10.1007/s10462-017-9605-z>. Citado na p. 23.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: PROCEEDINGS of ICNN'95 - International Conference on Neural Networks. 1995. v. 4, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968. Citado nas pp. 23, 24.

KOENEMANN, J.; BURGET, F.; BENNEWITZ, M. Real-time imitation of human whole-body motions by humanoids. In: 2014 IEEE International Conference on Robotics and Automation (ICRA). 2014. P. 2806–2812. DOI: 10.1109/ICRA.2014.6907261. Citado na p. 20.

KULIĆ, D.; OTT, C.; LEE, D.; ISHIKAWA, J.; NAKAMURA, Y. Incremental learning of full body motion primitives and their sequencing through human motion observation. **The International Journal of Robotics Research**, v. 31, n. 3, p. 330–345, 2012. DOI: 10.1177/0278364911426178. eprint: https://doi.org/10.1177/0278364911426178. Disponível em: <https://doi.org/10.1177/0278364911426178>. Citado na p. 20.

KUNJITTIPONG, N.; KONGKANJANA, K.; KHWAN-ON, S. Comparison of Fuzzy Controller and PI Controller for a High Step-Up SingleSwitch Boost Converter. In: 2020 3rd International Conference on Power and Energy Applications (ICPEA). 2020. P. 94–98. DOI: 10.1109/ICPEA49807.2020.9280118. Citado na p. 28.

LIU, J.; LU, H.; LUO, Y.; YANG, S. Spiking neural network-based multi-task autonomous learning for mobile robots. **Engineering Applications of Artificial Intelligence**, v. 104, p. 104362, 2021. ISSN 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2021.104362. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197621002104>. Citado na p. 30.

LIU, K.; WAN, Q.; LI, Y. A Deep Reinforcement Learning Algorithm with Expert Demonstrations and Supervised Loss and its application in Autonomous Driving. In: 2018 37th Chinese Control Conference (CCC). 2018. P. 2944–2949. DOI: 10.23919/ChiCC.2018.8482790. Citado na p. 20.

MÁNTARAS, R. L. de; PLAZA, E. Case-Based Reasoning: An Overview. **AI Commun.**, IOS Press, NLD, v. 10, n. 1, p. 21–29, jan. 1997. ISSN 0921-7126. Citado na p. 19.

MASSELOS, K.; VOROS, N. S. Introduction to Reconfigurable Hardware. In: **System Level Design of Reconfigurable Systems-on-Chip**. Edição: Nikolaos S. Voros e Konstantinos Masselos. Boston, MA: Springer US, 2005. P. 15–26. ISBN 978-0-387-26104-1. DOI: 10.1007/0-387-26104-4_1. Disponível em: <https://doi.org/10.1007/0-387-26104-4_1>. Citado na p. 25.

MOHANTA, J.; KESHARI, A. A knowledge based fuzzy-probabilistic roadmap method for mobile robot navigation. **Applied Soft Computing**, v. 79, p. 391–409, 2019. ISSN 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2019.03.055. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494619301826>. Citado na p. 14.

MUÑOZ, D. M.; LLANOS, C. H.; S. COELHO, L. dos; AYALA-RINCÓN, M. Hardware opposition-based PSO applied to mobile robot controllers. **Engineering Applications of Artificial Intelligence**, v. 28, p. 64–77, 2014. ISSN 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2013.12.003. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197613002376>. Citado nas pp. 24, 30, 79.

OGATA, K. **Modern Control Engineering**. 4th. USA: Prentice Hall PTR, 2001. ISBN 0130609072. Citado nas pp. 15, 28.

PARDOWITZ, M.; KNOOP, S.; DILLMANN, R.; ZOLLNER, R. D. Incremental Learning of Tasks From User Demonstrations, Past Experiences, and Vocal Comments. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 37, n. 2, p. 322–332, 2007. DOI: 10.1109/TSMCB.2006.886951. Citado na p. 20.

PETROV, P.; GEORGIEVA, V. Adaptive Velocity Control for a Differential Drive Mobile Robot. In: 2018 20th International Symposium on Electrical Apparatus and Technologies (SIELA). 2018. P. 1–4. DOI: 10.1109/SIELA.2018.8447091. Citado na p. 14.

PHAM, L. H.; TRAN, D. N.-N.; RHIE, C. H.; JEON, J. W. A Mobile Vision-based System for Gap and Flush Measuring between Planar Surfaces using ArUco Markers. In: 2021 International Conference on Electronics, Information, and Communication (ICEIC). 2021. P. 1–4. DOI: 10.1109/ICEIC51217.2021.9369720. Citado na p. 61.

PIDD, M. **Systems Modelling: Theory and Practice**. Hoboken, NJ, USA: John Wiley Sons, Inc., 2004. ISBN 0470867310. Citado na p. 15.

PROFILLIDIS, V.; BOTZORIS, G. Chapter 8 - Artificial Intelligence—Neural Network Methods. In: PROFILLIDIS, V.; BOTZORIS, G. (Ed.). **Modeling of Transport Demand**. Elsevier, 2019. P. 353–382. ISBN 978-0-12-811513-8. DOI: https://doi.org/10.1016/B978-0-12-811513-8.00008-X. Disponível em: <https://www.sciencedirect.com/science/article/pii/B978012811513800008X>. Citado na p. 21.

QIAN, K.; XU, X.; LIU, H.; BAI, J.; LUO, S. Environment-adaptive learning from demonstration for proactive assistance in human–robot collaborative tasks. **Robotics and Autonomous Systems**, v. 151, p. 104046, 2022. ISSN 0921-8890. DOI: https://doi.org/10.1016/j.robot.2022.104046. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0921889022000185>. Citado na p. 14.

SAKR, M.; LI, Z. J.; VAN DER LOOS, H. F. M.; KULIĆ, D.; CROFT, E. A. Quantifying Demonstration Quality for Robot Learning and Generalization. **IEEE Robotics and Automation Letters**, v. 7, n. 4, p. 9659–9666, 2022. DOI: 10.1109/LRA.2022.3191950. Citado na p. 14.

SANT, A. V.; RAJAGOPAL, K. R.; SHETH, N. K. Permanent Magnet Synchronous Motor Drive Using Hybrid PI Speed Controller With Inherent and Noninherent Switching Functions. **IEEE Transactions on Magnetics**, v. 47, n. 10, p. 4088–4091, 2011. DOI: 10.1109/TMAG.2011.2159831. Citado na p. 28.

SCHÖLKOPF, B.; PLATT, J.; HOFMANN, T. An Application of Reinforcement Learning to Aerobatic Helicopter Flight. In: ADVANCES in Neural Information Processing Systems 19: Proceedings of the 2006 Conference. 2007. P. 1–8. Citado na p. 19.

SHAIK, M. E.; ISLAM, M. M.; HOSSAIN, Q. S. A review on neural network techniques for the prediction of road traffic accident severity. **Asian Transport Studies**, v. 7, p. 100040, 2021. ISSN 2185-5560. DOI: https://doi.org/10.1016/j.eastsj.2021.100040. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2185556021000080>. Citado na p. 21.

SHARMA, R.; GOEL, N.; CHACKO, S. Performance Investigation of Fractional PI Controller in Shunt Active Filter for a Three Phase Four Wire System. In: 2018 4th International

Conference on Computing Communication and Automation (ICCCA). 2018. P. 1–7. DOI: 10.1109/CCAA.2018.8777576. Citado na p. 28.

SMART, W.; PACK KAELBLING, L. Effective reinforcement learning for mobile robots. In: PROCEEDINGS 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292). 2002. v. 4, 3404–3410 vol.4. DOI: 10.1109/ROBOT.2002.1014237. Citado na p. 19.

SMYS; RANGANATHAN. Robot assisted sensing, control and manufacture in automobile industry. **Journal of IoT in Social, Mobile, Analytics, and Cloud**, v. 1, n. 3, p. 180–187, 2019. ISSN 2582-1369. DOI: 10.36548/jismac.2019.3.005. Disponível em: <https://irojournals.com/iroismac/article/view/1/3/5>. Citado na p. 19.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. Cambridge, MA, USA: A Bradford Book, 2018. ISBN 0262039249. Citado na p. 20.

TAN, H. Applying an Extension of Estimation of Distribution Algorithm (EDA) for Mobile Robots to Learn Motion Patterns from Demonstration. In: 2015 IEEE International Conference on Systems, Man, and Cybernetics. 2015. P. 2829–2834. DOI: 10.1109/SMC.2015.493. Citado nas pp. 30, 79.

VUKOVIĆ, N.; MITIĆ, M.; MILJKOVIĆ, Z. Trajectory learning and reproduction for differential drive mobile robots based on GMM/HMM and dynamic time warping using learning from demonstration framework. **Engineering Applications of Artificial Intelligence**, v. 45, p. 388–404, 2015. ISSN 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2015.07.002. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197615001499>. Citado nas pp. 30, 80.

WANG, H.; LI, Z.; XIONG, H.; NIAN, X. Robust H attitude tracking control of a quadrotor UAV on SO(3) via variation-based linearization and interval matrix approach. **ISA Transactions**, v. 87, p. 10–16, 2019. ISSN 0019-0578. DOI: https://doi.org/10.1016/j.isatra.2018.11.015. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0019057818304518>. Citado na p. 14.

WANG, Y.; HU, Y.; ZAATARI, S. E.; LI, W.; ZHOU, Y. Optimised Learning from Demonstrations for Collaborative Robots. **Robotics and Computer-Integrated Manufacturing**, v. 71, p. 102169, 2021. ISSN 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2021.102169. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0736584521000533>. Citado na p. 14.

WEI, Y.; FANG, S.; LIN, W.; ZHANG, J.; LUO, D. Acquiring Robot Navigation Skill with Knowledge Learned from Demonstration. In: 2021 IEEE International Conference on Development and Learning (ICDL). 2021. P. 1–6. DOI: 10.1109/ICDL49984.2021.9515639. Citado na p. 30.

WEISE, T. **Global Optimization Algorithms - Theory and Application**. Second: Self-Published, 2009. Online as e-book. Online available at http://www.it-weise.de/. Disponível em: <http://www.it-weise.de/>. Citado na p. 23.

XILINX. Zynq-7000 SoC Data Sheet: Overview Zynq-7000 SoC First Generation Architecture Processing System (PS) Application Processor Unit (APU) I / O Peripherals and Interfaces Programmable I / O Blocks. v. 90, p. 1–25, 2018. Citado nas pp. 26, 27.

XU, S.; OU, Y.; DUAN, J.; WU, X.; FENG, W.; LIU, M. Robot trajectory tracking control using learning from demonstration method. **Neurocomputing**, v. 338, p. 249–261, 2019. ISSN 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.01.052. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925231219300785>. Citado na p. 14.

ZHANG, W.; JING, Y. Active Queue Management Algorithm Based on RBF Neural Network Controller. In: 2020 Chinese Control And Decision Conference (CCDC). 2020. P. 2289–2293. DOI: 10.1109/CCDC49329.2020.9164092. Citado na p. 14.