UNIVERSIDADE FEDERAL DE SERGIPE

PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

COMPARAÇÃO ENTRE CONTROLADORES *FUZZY* E NEURAL DESENVOLVIDOS VIA SIMULAÇÃO E TRANSFERIDOS PARA AMBIENTES REAIS NO ÂMBITO DA ROBÓTICA EVOLUTIVA

WESLLEY ALVES FARIAS

São Cristóvão-SE, Brasil

Julho de 2018

COMPARAÇÃO ENTRE CONTROLADORES *FUZZY* E NEURAL DESENVOLVIDOS VIA SIMULAÇÃO E TRANSFERIDOS PARA AMBIENTES REAIS NO ÂMBITO DA ROBÓTICA EVOLUTIVA

Weslley Alves Farias

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica – PROEE, da Universidade Federal de Sergipe, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica

Orientador: Eduardo Oliveira Freire
Co-orientador: Sidney Nascimento Givigi Júnior

São Cristóvão-SE, Brasil

Julho de 2018

COMPARAÇÃO ENTRE CONTROLADORES *FUZZY* E NEURAL DESENVOLVIDOS VIA SIMULAÇÃO E TRANSFERIDOS PARA AMBIENTES REAIS NO ÂMBITO DA ROBÓTICA EVOLUTIVA

Weslley Alves Farias

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica – PROEE, da Universidade Federal de Sergipe, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica

Examinada por:

Prof. Dr. Elyson Ádan Nunes Carvalho
PROEE/UFS

Prof. Dr. Lucas Molina
UFS

Prof. Dr. Guilherme Sousa Bastos
POSCOMP/UNIFEI

São Cristóvão-SE, Brasil
26 de Julho de 2018

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Farias, Weslley Alves
F224c Comparação entre controladores *fuzzy* e neural desenvolvidos via simulação e transferidos para ambientes reais no âmbito da robótica evolutiva / Weslley Alves Farias ; orientador Eduardo Oliveira Freire. - São Cristóvão, 2018.
112 f. : il.

Dissertação (mestrado em Engenharia Elétrica) – Universidade Federal de Sergipe, 2018.

1. Robótica evolutiva. 2. Algorítmos genéticos. 3. Redes neurais (Computação). 4. Lógica difusa. I. Freire, Eduardo Oliveira orient. II. Título.

CDU 621.3:007.52

Agradecimentos

Agradeço, primeiramente, a Deus pela proteção e saúde.

Ao meu pai Roque, a minha mãe Maria e a todos os meus irmãos pelo carinho e apoio emocional. A todos os familiares, avós, tios, tias, primos e primas que se fizeram presente sempre, pelo apoio e por torcerem por mim nesta caminhada.

Em especial, ao meu orientador e co-orientador, Eduardo e Sidney pelo tempo disponibilizado e dedicação para comigo durante este grande desafio pessoal.

Aos demais professores do GPRUFS, pela colaboração com sugestões e esclarecimento de dúvidas durante o desenvolvimento do trabalho.

Aos amigos que conheci na UFS e no GPRUFS pelo apoio, experiências compartilhadas e pelos momentos de descontração, meus sinceros agradecimentos.

Por fim, agradeço ao apoio financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES.

Resumo da Dissertação apresentada ao PROEE/UFS como parte dos requisitos necessários para a obtenção do grau de Mestre (Me.)

COMPARAÇÃO ENTRE CONTROLADORES *FUZZY* E NEURAL DESENVOLVIDOS VIA SIMULAÇÃO E TRANSFERIDOS PARA AMBIENTES REAIS NO ÂMBITO DA ROBÓTICA EVOLUTIVA

WESLLEY ALVES FARIAS

Julho/2018

Orientador: Eduardo Oliveira Freire

Co-orientador: Sidney Nascimento Givigi Júnior

Uma das grandes limitações da Robótica Evolutiva diz respeito à transferência de controladores evoluídos por simulação e transferidos ao ambiente real. Tal limitação deve-se, sobretudo, a simplificações de modelo e dificuldades na representação de características dinâmicas, tanto do robô quanto do ambiente, e isso resulta na queda de desempenho do controlador evoluído após a transferência, fenômeno denominado de *reality gap*. Muitas soluções vêm sendo propostas desde a década de 90, em virtude deste problema ser uma limitação para aplicações práticas e complexas da robótica evolutiva. Até o momento, a maioria dos trabalhos de pesquisa desenvolvidos utiliza estratégias de controle baseadas em redes neurais artificiais por permitirem que algoritmos possam ser evoluídos com menor influência do projetista. Por outro lado, a lógica *fuzzy* também pode ser usada para o desenvolvimento de controladores no âmbito da robótica evolutiva, pois também permite emular a inteligência humana. Portanto, nesta dissertação é investigado se sistemas de controle *fuzzy* são mais robustos que sistemas de controle neurais, ambos otimizados por um algoritmo genético em simulação e posteriormente transferidos para um robô real em ambiente físico na tarefa de navegação autônoma evitando obstáculos. Como resultado, obteve-se que nas condições analisadas, os controladores *fuzzy* apresentaram uma melhor transferência, com destaque para a suavidade da trajetória executada, e um desempenho equivalente, quando comparados com controladores neurais.

Palavras-chaves: robótica evolutiva, algoritmo genético, redes neurais (computação), lógica difusa.

Abstract of Dissertation presented to PROEE/UFS as a partial fulfillment of the requirements for the degree of Master

COMPARISON BETWEEN FUZZY AND NEURAL CONTROLLERS DEVELOPED BY SIMULATION AND TRANSFERRED TO REAL ENVIRONMENTS IN THE SCOPE OF EVOLUTIONARY ROBOTICS

WESLLEY ALVES FARIAS

July/2018

Advisor: Eduardo Oliveira Freire

Co-advisor: Sidney Nascimento Givigi Júnior

One of the greatest limitations of Evolutionary Robotics is when transfering controllers evolved by simulation to real environments. This limitation is mainly caused by model simplifications and difficulties to represent dynamic characteristics, whether from the robot or the environment. And this results in performance degradation of the evolved controller after the transfer, a phenomenon called reality gap. Because this problem is a limitation for practical and complex applications of evolutionary robotics, many solutions have been proposed since the 90s. Until now, most of the research use control strategies based on artificial neural networks because they allow algorithms to be evolved with less designer influence. On the other hand, fuzzy logic can also be used for the development of controllers in the field of evolutionary robotics because it also allows emulating human intelligence. Therefore, this dissertation investigates whether fuzzy control systems are more robust than neural control systems, both optimized by a genetic algorithm in simulation and later transferred to a real robot in physical environment in the task of autonomous navigation while avoiding obstacles. The results show that in the analyzed conditions, fuzzy controllers present better transfer characteristics, mainly considering the smoothness of the executed trajectory, and an equivalent performance, when compared with neural controllers.

Key-words: evolutionary robotics, genetic algorithm, neural network (computation), fuzzy logic.

Lista de ilustrações

Figura 1 – Geometria de uma função de estimação neural e *fuzzy* (modificada de Kong e Kosko (1990)). A abordagem neural (a) utiliza muitas amostras de pontos e a abordagem fuzzy (b) usa poucos conjuntos *fuzzy*. 12

Figura 2 – Referência global e local do robô diferencial (Fonte: Siegwart, Nourbakhsh e Scaramuzza (2011)). 17

Figura 3 – Fluxograma de um Algoritmo Genético aplicado à Robótica Evolutiva (DONCIEUX et al., 2015). 19

Figura 4 – Princípio de funcionamento do operador *Roulette Wheel* (SILVA, 2005). 20

Figura 5 – Pais selecionados para operação de *crossover* (ZUBEN, 2000). 21

Figura 6 – Indivíduos filhos resultantes da operação de *crossover* (ZUBEN, 2000). 21

Figura 7 – Modelo artificial de um neurônio biológico (HAYKIN, 2001). 22

Figura 8 – Variável linguística Temperatura (Fonte: adaptado de Gomide, Gudwin e Tanscheit (1995)). 25

Figura 9 – Arquitetura de um controlador *fuzzy* típico (Fonte: traduzido de Abraham e Nath (2000)). 26

Figura 10 – Pioneer P3-DX com Laser SICK LMS200. 28

Figura 11 – Simulador e Interface de controle para o robô móvel Pioneer 3DX do GPRUFS 30

Figura 12 – Vista dos conjuntos de entradas utilizados nos controladores. 32

Figura 13 – (a) Controlador neural e (b) Função de ativação dos neurônios (sigmoide). 35

Figura 14 – (a) Ambiente 1 simulado, e (b) Ambiente 1 real. 43

Figura 15 – Normalização da menor leitura dos sensores aplicada ao Ambiente 1. . 44

Figura 16 – Conjuntos *fuzzy* referentes às variáveis distância à direita, esquerda e frontal aplicados ao Ambiente 1. 46

Figura 17 – (a) Processo evolutivo neural, e (b) Processo evolutivo *fuzzy* com velocidade linear máxima das rodas de $30cm/s$. 46

Figura 18 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 2 com velocidade linear máxima das rodas de 30cm/s, ... 47

Figura 19 – (a) Processo evolutivo neural, e (b) Processo evolutivo *fuzzy* com velocidade linear máxima das rodas de $25cm/s$. 50

Figura 20 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 1 com velocidade linear máxima das rodas de 25cm/s, ... 51
Figura 21 – (a) Processo evolutivo neural, e (b) Processo evolutivo *fuzzy* com velocidade linear máxima das rodas de $20cm/s$. 53
Figura 22 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 1 com velocidade linear máxima das rodas de 20cm/s, ... 54
Figura 23 – (a) Processo evolutivo neural, e (b) Processo evolutivo *fuzzy* com velocidade linear máxima das rodas de $15cm/s$. 56
Figura 24 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 2 com velocidade linear máxima das rodas de 15cm/s. 57
Figura 25 – (a) Ambiente 2 simulado, e (b) ambiente 2 real. 59
Figura 26 – Normalização da menor leitura dos sensores aplicada ao Ambiente 2. . 60
Figura 27 – (a) Conjuntos *fuzzy* esqueda e direita, e (b) Conjuntos *fuzzy* frontal. . . 61
Figura 28 – Comparação do processo evolutivo neural e *fuzzy* do ambiente 2 com velocidade linear máxima das rodas de 30cm/s. 61
Figura 29 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 2 com velocidade linear máxima das rodas de 30cm/s. 62
Figura 30 – Comparação do processo evolutivo neural e *fuzzy* do ambiente 2 com velocidade linear máxima das rodas de 25cm/s. 64
Figura 31 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 2 com velocidade linear máxima das rodas de 25cm/s. 65
Figura 32 – Comparação do processo evolutivo neural e *fuzzy* do ambiente 2 com velocidade linear máxima das rodas de 20cm/s. 68
Figura 33 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 2 com velocidade linear máxima das rodas de 20cm/s. 68
Figura 34 – Comparação do processo evolutivo neural e *fuzzy* do ambiente 2 com velocidade linear máxima das rodas de 15cm/s. 71
Figura 35 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 2 com velocidade linear máxima das rodas de 15cm/s. 71
Figura 36 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 30cm/s. 75
Figura 37 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 25cm/s. 75

Figura 38 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 20cm/s. 76
Figura 39 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 15cm/s. 76
Figura 40 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 30cm/s. 77
Figura 41 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 25cm/s. 77
Figura 42 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 20cm/s. 78
Figura 43 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 15cm/s. 78

Figura 44 – (a) *Fitness* 1 no Ambiente 1 simulado, e (b) *Fitness* 1 no Ambiente 2 simulado. 92
Figura 45 – (a) *Fitness* 2 no Ambiente 1 simulado, e (b) *Fitness* 2 no Ambiente 2 simulado. 92
Figura 46 – (a) *Fitness* 3 no Ambiente 1 simulado, e (b) *Fitness* 3 no Ambiente 2 simulado. 93
Figura 47 – (a) *Fitenss* 4 no Ambiente 1 simulado, e (b) *Fitness* 4 no Ambiente 2 simulado. 94
Figura 48 – (a) *Fitness* 5 no Ambiente 1 simulado, e (b) *Fitness* 5 no Ambiente 2 simulado. 94

Figura 49 – Controlador Neural Estendido. 96
Figura 50 – Processo evolutivo neural do ambiente 2 com velocidade máxima de 30cm/s. 96
Figura 51 – Controlador neural simulado e real no ambiente 2 com velocidade linear máxima das rodas de 30cm/s. 97

Lista de tabelas

Tabela 1 – Conjunto de regras do controlador *fuzzy*. 38

Tabela 2 – Parâmetros do algoritmo genético aplicado ao ambiente 1 com velocidade linear máxima das rodas de 30cm/s. 47

Tabela 3 – Conjunto de regras do controlador *fuzzy* para o ambiente 1 com velocidade linear máxima das rodas de $30cm/s$. 48

Tabela 4 – Neural - melhor indivíduo ambiente 1 com velocidade linear máxima das rodas de 30cm/s. 49

Tabela 5 – *Fuzzy* - melhor indivíduo ambiente 1 com velocidade linear máxima das rodas de 30cm/s. 49

Tabela 6 – Análise estatística do experimento 1 com velocidade linear máxima das rodas de 30cm/s 50

Tabela 7 – Conjunto de regras do controlador *fuzzy* para o ambiente 1 com velocidade linear máxima das rodas de $25cm/s$. 52

Tabela 8 – Neural - melhor indivíduo ambiente 1 com velocidade linear máxima das rodas de 25cm/s. 52

Tabela 9 – *Fuzzy* - melhor indivíduo ambiente 1 com velocidade linear máxima das rodas de 25cm/s. 52

Tabela 10 – Análise estatística do experimento 1 com velocidade linear máxima das rodas de 25cm/s 53

Tabela 11 – Conjunto de regras do controlador *fuzzy* para o ambiente 1 com velocidade linear máxima das rodas de $20cm/s$. 54

Tabela 12 – Neural - melhor indivíduo ambiente 1 com velocidade linear máxima das rodas de 20cm/s. 55

Tabela 13 – *Fuzzy* - melhor indivíduo ambiente 1 com velocidade linear máxima das rodas de 20cm/s. 55

Tabela 14 – Análise estatística do experimento 1 com velocidade linear máxima das rodas de 20cm/s 56

Tabela 15 – Conjunto de regras do controlador *fuzzy* para o ambiente 1 com velocidade linear máxima das rodas de $15cm/s$. 58

Tabela 16 – Neural - melhor indivíduo do ambiente 1 com velocidade linear máxima das rodas de 15cm/s. 58

Tabela 17 – *Fuzzy* - melhor indivíduo do ambiente 1 com velocidade linear máxima das rodas de 15cm/s. 58

Tabela 18 – Análise estatística do experimento 1 com velocidade linear máxima das rodas de 15 cm/s 59

Tabela 19 – Parâmetros do algoritmo genético aplicado ao ambiente 2 com velocidade linear máxima das rodas de 30cm/s. 61
Tabela 20 – Conjunto de regras do controlador *fuzzy* para o ambiente 2 com velocidade linear máxima das rodas de $30cm/s$. 63
Tabela 21 – Neural - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $30cm/s$ 63
Tabela 22 – *Fuzzy* - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $30cm/s$ 63
Tabela 23 – Análise estatística do Ambiente 2 com velocidade linear máxima das rodas de 30cm/s 64
Tabela 24 – Conjunto de regras do controlador *fuzzy* para o ambiente 2 com velocidade linear máxima das rodas de $25cm/s$. 66
Tabela 25 – Neural - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $25cm/s$ 66
Tabela 26 – *Fuzzy* - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $25cm/s$ 67
Tabela 27 – Análise estatística do Ambiente 2 com velocidade linear máxima das rodas de 25cm/s 67
Tabela 28 – Conjunto de regras do controlador *fuzzy* para o ambiente 2 com velocidade linear máxima das rodas de $20cm/s$. 69
Tabela 29 – Neural - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $20cm/s$ 69
Tabela 30 – *Fuzzy* - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $20cm/s$ 70
Tabela 31 – Análise estatística do Ambiente 2 com velocidade linear máxima das rodas de 20cm/s 70
Tabela 32 – Conjunto de regras do controlador *fuzzy* para o ambiente 2 com velocidade linear máxima das rodas de $15cm/s$. 72
Tabela 33 – Neural - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $15cm/s$ 72
Tabela 34 – *Fuzzy* - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $15cm/s$ 72
Tabela 35 – Análise estatística do Ambiente 2 com velocidade linear máxima das rodas de 15cm/s 73
Tabela 36 – Comparação da trajetória extraída com a câmera e simulação com base no DTW. 74
Tabela 37 – Análise dos experimentos com base na variação de desempenho com a transferência. 79

Tabela 38 – Análise dos experimentos com base no comportamento evoluído e apresentado nos experimentos em realidade. 80

Tabela 39 – Parâmetros do algoritmo genético do controlador neural aplicado ao ambiente 2 com velocidade máxima de 30cm/s. 96

Tabela 40 – Neural - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $30cm/s$ 98

Sumário

Lista de ilustrações . . . . . **VII**

Lista de tabelas . . . . . **X**

1 Introdução **1**
1.1 Trabalhos relacionados 3
1.1.1 Características de modelos que possivelmente causam o *reality gap* . 4
1.1.2 Principais abordagens 5
1.1.2.1 Otimização unicamente em simulação 6
1.1.2.2 Otimização parte por simulação e parte em ambiente real 10
1.1.3 Abordagem *Fuzzy* 11
1.2 Objetivos 14
1.2.1 Objetivos gerais 14
1.2.2 Objetivos específicos 15
1.3 Estrutura do trabalho 15

2 Fundamentação Teórica **16**
2.1 Modelagem do Robô 16
2.2 Robótica Evolutiva 18
2.2.1 Algoritmos Genéticos 18
2.3 Redes Neurais Artificiais 22
2.3.1 Neurônio Artificial 22
2.3.2 Arquiteturas de redes neurais 24
2.3.3 Aprendizado 24
2.4 Lógica *Fuzzy* 24
2.4.1 Modelo *Fuzzy* Genético 27

3 Abordagem Proposta **28**
3.1 Tarefa 28
3.2 Robô 28
3.3 Simulador 29
3.3.1 Sensoriamento 31
3.3.2 Modelo Cinemático e Dinâmico 32
3.4 Estruturas de controle 34
3.4.1 Controlador Neural 34
3.4.2 Controlador *Fuzzy* 37

3.5 Algoritmo genético 38
3.6 Métricas de avaliação 41

4 Resultados 43
4.1 Experimento - Ambiente 1 43
4.1.1 Ambiente 1 com velocidade linear máxima das rodas de 30cm/s . . 46
4.1.2 Ambiente 1 com velocidade linear máxima das rodas de 25cm/s . . 50
4.1.3 Ambiente 1 com velocidade linear máxima das rodas de 20cm/s . . 53
4.1.4 Ambiente 1 com velocidade linear máxima das rodas de 15cm/s . . 55
4.2 Experimento - Ambiente 2 58
4.2.1 Ambiente 2 com velocidade linear máxima das rodas de 30cm/s . . 61
4.2.2 Ambiente 2 com velocidade linear máxima das rodas de 25cm/s . . 64
4.2.3 Ambiente 2 com velocidade linear máxima das rodas de 20cm/s . . 67
4.2.4 Ambiente 2 com velocidade linear máxima das rodas de 15cm/s . . 69
4.3 Análise com base em câmera 73
4.4 Análise comparativa dos resultados experimentais 74

5 Conclusões 82

Referências 85

Apêndices 90

APÊNDICE A Robustez quanto a variações na função de *fitness* 91

APÊNDICE B Controlador Neural Estendido 96

1 Introdução

Durante muito tempo os robôs foram tratados meramente como equipamentos cuja função era executar tarefas pré-programadas, em substituição aos humanos em serviços cotidianos, arriscados, repetitivos e em linhas de montagem, sem nenhum nível de inteligência incorporada. Mas com o passar dos anos, a necessidade de uma maior independência na execução de suas tarefas motivou muitas pesquisas com o objetivo de dotá-los com algum nível de inteligência para que pudessem ser capazes de se adaptar ao ambiente para tomar decisões em situações para as quais não foram programados.

Os robôs precisam de inteligência pela necessidade de interpretação de informações sensoriais, interação com seres humanos, recuperação de falhas, aprendizado, etc. Atualmente, a área de inteligência artificial possui várias ferramentas que são usadas para estudar esses problemas, como a representação do conhecimento, planejamento, busca, inferência, aprendizado e visão, por exemplo.

Além disso, o desenvolvimento de sistemas de controle para um robô, muitas vezes, exige conhecimentos por parte do projetista que não estão disponíveis durante o processo de desenvolvimento, seja pela complexidade do sistema, ou principalmente se o ambiente no qual o robô irá atuar for ruidoso, não estruturado ou desconhecido. Nesse contexto, é importante que ele possua algum grau de autonomia e seja dotado de mecanismos que permitam a sua auto adaptação para poder tomar decisões em situações adversas.

Para tornar o sistema de controle de um robô móvel mais adaptável, é necessário utilizar algumas técnicas de desenvolvimento que permitem que o sistema se modifique ao longo de sua execução. Neste contexto, a Robótica Evolutiva (RE) utiliza-se da Computação Evolutiva (CE) no desenvolvimento de sistemas adaptativos para robôs. Esta área se destina ao desenvolvimento de robôs autônomos com a mínima intervenção humana no processo, sendo necessário, basicamente, descrever o problema a ser solucionado e a otimização evolutiva se encarrega de encontrar a melhor solução (controlador) automaticamente dentro do espaço de soluções (PEREZ, 2009). Devido ao grande número de soluções possíveis, e que são investigadas aleatoriamente durante o processo de otimização, esta abordagem utiliza-se fundamentalmente de ambientes simulados para avaliar o desempenho de soluções. Embora a simulação reduza o tempo total de otimização (ou evolução) de controladores, a diferença entre simulação e realidade, fato referenciado na literatura como *reality gap* (JAKOBI et al., 1998; DULLERUD; PAGANINI, 2013), frequentemente ocorre quando controladores otimizados (ou evoluídos) em simuladores se mostram ineficientes ou apresentam uma queda expressiva em seu desempenho quando transferidos para o ambiente real (SILVA et al., 2016; KOOS; MOURET; DONCIEUX,

2013). Dessa forma, o problema do *reality gap* tem sido um grande empecilho para o progresso da robótica evolutiva. De acordo com Silva et al. (2016) esse problema é uma das questões abertas mais relevantes em robótica evolutiva, o que restringe sua utilização em aplicações práticas e complexas da robótica.

Uma das vantagens do processo de evolução simulada é a possibilidade de se fazer estudos preliminares do processo evolutivo, por exemplo, definir uma série de parâmetros iniciais, que mais tarde poderão ser aplicados ao sistema em ambiente real (PEREZ, 2009).

Entretanto, o desenvolvimento de um controlador para um sistema, seja em robótica ou qualquer outra área, na maioria das vezes é baseado no uso de modelos matemáticos que são uma ferramenta de mapeamento entre entrada e saída (ZHOU; DOYLE, 1998). Experimentos utilizando plataformas reais podem exigir investimento de tempo e recursos financeiros. Portanto, para que seja possível a implementação física real, o sistema de controle deve ser anteriormente projetado, desenvolvido e testado em um ambiente de simulação tão realístico quanto for possível (PESSIN; OSÓRIO; MUSSE, 2008).

Quando um sistema físico é modelado, alguns aspectos de fenômenos físicos são sempre aproximados de maneira que facilite convenientemente a sua manipulação e análise, até o ponto em que se acredita que tais simplificações não trarão prejuízos significativos (DULLERUD; PAGANINI, 2013).

O ramo da teoria de controle que lida com incertezas na representação do modelo da planta, como incertezas paramétricas, perturbações, dinâmicas não modeladas, ou outras incertezas presentes no sistema, é chamado de Controle Robusto. Controladores projetados utilizando-se controle robusto estão aptos a superarem pequenas diferenças entre a planta e o modelo idealizado utilizado para projeto.

Dullerud e Paganini (2013) levantaram duas observações relevantes sobre modelos para sistemas físicos complexos. A primeira é que modelos analíticos ou computacionais que descrevem sistemas físicos são difíceis ou impossíveis de caracterizar e de simular precisamente. A segunda é que um modelo, não importa o quão detalhado seja, nunca é uma representação completa do sistema real. A primeira observação significa que é necessário usar modelos simplificados de sistemas por razões de simplicidade no manuseio e o segundo aspecto retrata que modelos são ineficazes em algum grau.

Existem muitos motivos pelos quais os sistemas podem ser considerados incertos, Dullerud e Paganini (2013) destacam três deles:

- As condições iniciais de um sistema podem não ser especificadas precisamente ou não ser completamente conhecidas;
- Perturbações externas e comandos em sistemas são tipicamente não conhecidos *a priori*;

- A imprecisão do modelo do sistema é uma fonte de incerteza. Qualquer modelo dinâmico de um sistema negligencia algum fenômeno físico, e isto quer dizer que alguma abordagem de controle analítica baseada somente neste modelo irá negligenciar alguns regimes de operação.

Algumas técnicas de desenvolvimento de controladores robustos foram desenvolvidas com o passar dos anos, como as abordagens lineares multivariável (linearização por realimentação), baseada em passividade, saturação robusta e adaptação robusta (ABDALLAH et al., 1991). Também podem ser citados os controladores H^2 e H^∞, LQG (*Linear Quadratic Gaussian*), controle adaptativo e abordagens baseadas na teoria de jogos, função de Lyapunov, dentre outras (DULLERUD; PAGANINI, 2013). Contudo, a robustez fornecida por estas metodologias exigem um grande esforço analítico.

De acordo com Zhou e Doyle (1998), um bom modelo deve ser simples o suficiente para facilitar o desenvolvimento do controlador e ainda complexo o suficiente para dar confiança ao projetista na estratégia de controle desenvolvida e que esta possa trabalhar em planta real. Portanto, o modelo adotado para o desenvolvimento de uma estratégia de controle afeta diretamente seu desempenho em um sistema real. Isso não se restringe somente à robótica, mas a toda área de controle.

Consequentemente, quando uma estrutura de controle obtida a partir de modelo é transferida a uma planta de processo ou a um robô real, uma queda no desempenho deve ser esperada. Em RE, este problema é ainda mais pronunciado devido o projetista não desenvolver diretamente o sistema de controle, mas definindo uma função de custo que avalia o desempenho da estratégia de controle por meio de um algoritmo genético responsável por sintonizar o controlador. Diante deste fato, evidenciado na robótica evolutiva como *reality gap*, algumas abordagens têm sido propostas para contorná-lo. Na seção seguinte é realizado um levantamento das principais causas e abordagens propostas como solução ao *reality gap*.

1.1 Trabalhos relacionados

Desde o fim da década de 1990, alguns pesquisadores têm investigado o problema da transferência de sistemas de controle de robôs autônomos da simulação para a realidade. Diante disto, são apresentadas nesta seção possíveis características de modelos que causam o *reality gap*, as principais abordagens utilizadas como solução e também a investigação da aplicação da lógica *fuzzy* como uma possível ferramenta na minimização dos efeitos com a transferência.

Na sequência, são apresentados alguns termos específicos referentes aos temas robótica evolutiva, redes neurais artificiais e lógica *fuzzy*, caso o leitor não esteja famili-

arizado com esses temas sugere-se, primeiramente, a leitura do Capítulo 2, referente à fundamentação teórica.

1.1.1 Características de modelos que possivelmente causam o *reality gap*

É evidente que a simulação de sistemas robóticos é essencialmente necessária quando se trata de robôs caros, grandes ou frágeis. É uma ferramenta extremamente poderosa para agilizar o ciclo de desenvolvimento de sistemas de controle robóticos eliminando desperdício de recursos, tanto financeiros quanto computacionais. Entretanto, para que seja útil, o simulador deve incorporar características importantes do mundo físico para o problema em questão (PESSIN; OSÓRIO; MUSSE, 2008).

Segundo Koos, Mouret e Doncieux (2013), simuladores precisos podem se tornar mais lentos que experimentos na realidade, o que leva a um processo de otimização longo. Para obter modelos de simulação com baixo custo computacional, é necessário, às vezes, negligenciar alguns fenômenos físicos complexos, interações com o ambiente, morfologia e controlador, o que permite simplificar o simulador, tornando-o incerto, porém mais rápido.

Sensores e atuadores são sempre sujeitos a erros e interferências, consequentemente, o controle das ações de um robô deve sempre levar em conta estas imprecisões. Um sistema robusto deve permitir que, mesmo com sensores e atuadores imprecisos, o agente cumpra o seu objetivo (PESSIN; OSÓRIO; MUSSE, 2008). Contudo, existem influências que agem sobre os sistemas e que são desconhecidas, como distúrbios ou também comandos externos (DULLERUD; PAGANINI, 2013).

Além disso, muitos simuladores não incluem a modelagem dinâmica do robô, se limitando à modelagem cinemática que descreve o robô em função da velocidade e orientação das rodas (YAMAMOTO et al., 2003). O modelo dinâmico é responsável por descrever o robô em função das forças generalizadas aplicadas pelos atuadores, por exemplo, torque nas rodas. Porém, em algumas aplicações mais complexas, a dinâmica do robô pode não ser conhecida completamente e consequentemente, o modelo dinâmico simplificado do robô utilizado para a simulação pode ser ineficaz (KOOS; MOURET; DONCIEUX, 2013).

Quanto às colisões, Yamamoto et al. (2003) citam que em alguns casos elas são detectadas, porém não tratadas, em outros casos, os robôs têm sua forma geométrica simplificada para um círculo, e outro problema frequente é que o tempo de simulação não corresponde ao tempo real, o que dificulta o teste e o acoplamento com os controladores reais.

De acordo com Pessin, Osório e Musse (2008) a interação com o ambiente se dá através de ciclos de percepção-ação que consistem em três etapas fundamentais: (*i*) obtenção de informação através de sensores; (*ii*) processamento das informações para a seleção

da ação; e, (*iii*) execução da ação através do acionamento dos atuadores. Esse conjunto de operações, em uma análise superficial, pode ser simples, porém o controle robusto de sistemas robóticos tem complicações físicas, mecânicas, eletrônicas e computacionais que tornam a criação de uma estratégia de controle uma tarefa árdua e sujeita a erros (PESSIN; OSÓRIO; MUSSE, 2008).

Considerando que robôs móveis com rodas podem estar sujeitos a perturbações externas, como desníveis da superfície por onde o robô circula, escorregamento das rodas, colisão com obstáculos ou com outros robôs e incertezas paramétricas, o controle utilizado deve ser robusto o suficiente para a realização de uma tarefa pré-estabelecida (INOUE, 2007).

Mouret e Chatzilygeroudis (2017) realizaram uma análise sobre o problema do *reality gap* que aflige a robótica evolutiva com o objetivo de ressaltar a necessidade de simuladores voltados para a robótica evolutiva que permitam ultrapassar o *reality gap* nos mais diversos experimentos. Diante desta necessidade, são levantadas algumas características que estes simuladores devem contemplar, como a rapidez, pois a RE necessita avaliar muitas soluções que requerem tempo; estabilidade, a fim de garantir que não ocorra parada prematura por erros devido a singularidades como em robôs com juntas; precisão do simulador, que depende do tipo de estudo, ou seja, se o problema exige simulação física ou não; e por fim, simuladores que sejam desenvolvidos sobre plataformas abertas que não requeiram uma licença, que possam ser atualizados e rodar em sistemas operacionais atualizados e, até mesmo, na condição de o fabricante interromper a produção. Esta última característica é de fundamental importância para permitir a reprodução de trabalhos e difusão da RE. Neste sentido, os autores propuseram plataformas abertas com bibliotecas que englobam características físicas de robôs de tração diferencial e robôs com patas, a exemplo do Fastsim, RobDyn e Dart-based Simulator. Outras características que podem ser citadas são o custo do simulador, possibilidade de desenvolvimento colaborativo de projeto, além do desenvolvimento ativo da própria plataforma de simulação. Neste sentido pode-se destacar o V-REP e Gazebo, plataformas gratuitas, ROS (Robot Operating System) e Webots, plataforma de código aberto flexível com diversas bibliotecas integradas, dentre outros.

Na subseção seguinte, são apresentadas as principais abordagens que lidam com o *reality gap*. Por conseguinte, será possível verificar como as características inerentes aos simuladores e destacadas anteriormente são tratadas na robótica evolutiva.

1.1.2 Principais abordagens

As metodologias mais relevantes propostas a minimizar o problema do *reality gap* são apresentadas nesta subseção. São divididas em dois tipos de abordagens: a primeira em que o processo de otimização é realizado unicamente em simulação, e a segunda na qual

o processo de otimização é feito parte por simulação e parte em ambiente real.

Rieffel et al. (2017) ressalta que muitos dos desafios atuais do campo são mais tecnológicos do que científicos, porque para projetar máquinas que evoluam no mundo físico, primeiramente são necessárias máquinas que possam se auto-reproduzir ou, pelo menos, máquinas autônomas que podem, por sua vez, ajudar outras máquinas a evoluir. E até que os simuladores computacionalmente eficientes de alta fidelidade sejam desenvolvidos, o papel da incorporação física na Robótica Evolutiva permanece essencial.

Diante deste fato, pode ser verificado na literatura que alguns autores realizam todo o processo de otimização no robô físico em ambiente real, de modo a evitar o *reality gap*. Nesse sentido, pode ser citado o exemplo de Floreano e Mondada (1996), que apresentaram a evolução de um controlador baseado em rede neural para um robô móvel autônomo. Neste processo o robô é conectado a um computador que processa as leituras fornecidas pelo robô, roda o algoritmo genético e também é responsável por fornecer alimentação ao robô.

Porém, como já foi previamente mencionado, a realização do processo evolutivo em plataformas reais também é um desafio complexo, pois requer muito tempo e envolve problemas, como o risco de dano ao equipamento, por exemplo.

A seguir é apresentada as abordagens que utilizam otimização unicamente em simulação de modo a lidar com o *reality gap*.

1.1.2.1 Otimização unicamente em simulação

Jakobi, Husbands e Harvey (1995) propuseram uma das primeiras metodologias. Nela utilizou-se um simulador mais robusto, que incorpora uma taxa de ruído aos sensores e atuadores correspondente à incerteza do ambiente, e como resultado foi possível observar que as discrepâncias com a transferência foram bastante minimizadas. Porém, esta metodologia necessita, para cada tipo de robô, de um levantamento da taxa de ruído realístico correspondente.

Tremblay et al. (2018), de modo similar, apresenta um sistema para treinamento de uma rede neural para detecção de objetos usando imagens sintetizadas. Para lidar com a variabilidade dos dados do mundo real, o sistema baseia-se na técnica de randomização de domínios, na qual os parâmetros do simulador − como iluminação, posição e texturas de objetos, etc. − são randomizados de maneiras não realistas para forçar a rede neural a aprender as características essenciais do objeto de interesse. Foi explorada a importância desses parâmetros, mostrando que é possível produzir uma rede com bom desempenho usando apenas dados sintetizados.

Jakobi et al. (1998) propuseram uma metodologia denominada simulação mínima. Essa metodologia busca encontrar um simulador simples, capaz de evoluir controladores

baseados em um conjunto mínimo de características ambientais relevantes para evolução do comportamento requerido. Assim, numa simulação mínima, os vários aspectos da interação entre um robô e seu ambiente são divididos em aspectos de conjunto de base e aspectos de implementação, onde os primeiros têm contrapartida física no mundo real, ao passo que os demais não. Assim, em uma simulação mínima, os aspectos de implementação devem ser variados aleatoriamente de avaliação para a avaliação, tornando-os assim inúteis para a evolução e forçando a estratégia evolutiva a se concentrar nos aspectos do conjunto base. Além disso, um certo grau de variação é introduzido também nos aspectos do conjunto base de forma a representar o fato de que mesmo nos aspectos do conjunto base haverá discrepâncias entre a simulação e o mundo real. Portanto, essa metodologia apresenta como limitação, a dependência da experiência do projetista na identificação de tais características que podem ser de difícil reconhecimento de acordo com a complexidade da tarefa e a plataforma robótica a ser utilizada.

Boeing e Bräunl (2012) investigaram a transferência para robôs reais empregando o uso de múltiplos simuladores dinâmicos. Segundo eles, a transferência de um simulador para outro fornece uma fonte de ruído e imita a transferência para a realidade, devido à existência de vários fatores que influem na característica dinâmica de simulação. Avaliando-se um controlador em múltiplos simuladores, elimina-se a dependência de características particulares, minimizando o efeito na transferência ao mundo real. A metodologia é aplicada a um robô submarino na tarefa de seguir parede e seu controlador é um PID (proporcional integral derivativo) sintonizado via algoritmo genético. São avaliados dois simuladores, um com e outro sem dinâmica da água, e em relação ao resultado obtido, comparando-se ao desempenho no robô real, as discrepâncias são mínimas nos dois casos. Porém, este trabalho apresenta limitação quanto a escolha dos simuladores a serem utilizados, pois, em determinadas tarefas, tais simuladores podem apresentar restrições de codificação, necessitando de adequações, além de um levantamento de quais características dinâmicas o simulador proporcionará ao sistema evolutivo. Outro ponto é a obtenção das licenças de tais plataformas de simulação.

Cappelle, Bernatskiy e Bongard (2017) e Ligot e Birattari (2018) também defendem a ideia do uso de um outro simulador para testar o controlador e assim avaliar o *reality gap*, sem ter que transferir ao ambiente real diretamente. Tal técnica foi nomeada de pseudo-realidade. A ideia por trás dessa abordagem é que um controlador X pode se apresentar melhor no simulador que um controlador Y e com a transferência ao ambiente real seu desempenho ser inferior a este. Tal fato evidencia que o *reality gap* não é causado apenas pela aplicação de um modelo simplificado. Contudo, não são apresentadas as características de modelo que podem causar esse declínio no desempenho.

Scheper e Croon (2016) investigaram a abstração nas entradas dos sensores e ações dos motores como uma solução potencial ao *reality gap*. O termo abstração quer dizer que

o robô usa entradas pré-processadas e um controlador de baixo nível executa comandos de alto nível de um controlador evoluído. A malha de controle de baixo nível ajuda a rejeitar distúrbios devido a ruído ou diferenças entre a dinâmica de simulação e realidade. Com isso, investiga-se como a abstração afeta a habilidade do robô de explorar o ambiente para solucionar uma tarefa complexa e o que acontece com a relação sensor-atuador quando o robô não tem acesso às entradas do sensor e sua relação com os atuadores. Foi aplicada a abstração à tarefa de formar uma configuração triangular assimétrica com robôs aéreos homogêneos, sendo que as entradas para o controlador são as posições dos agentes no ambiente e as distâncias entre os agentes. Os resultados mostraram que o comportamento evoluído é efetivo tanto em simulação quanto em realidade, o que sugere que a abstração pode ser usada como uma ferramenta na evolução de comportamento robusto para o *reality gap*. Contudo, percebe-se que a necessidade de um pré-processamento decorrente de um sensoriamento de qualidade e os controladores de baixo nível sugerem a aplicação de plataformas robóticas mais elaboradas.

Milano, Carvalho e Nolfi (2017) analisaram a variação moderada de características ambientais para promover melhor evolução de agentes. Essa variação caracteriza um comportamento similar à mutação genética, pois impede a acomodação genética. Duas variáveis são consideradas na evolução, o número de características do ambiente e o número da geração que ela mudará. O processo evolutivo é dado no controle de um carrinho com dois pêndulos sobre ele. As características que sofrem variações são a inclinação e coeficiente de atrito do plano e os estados iniciais do carro e dos pêndulos (posição e velocidade do carro, posição e velocidade angular dos dois pêndulos).

Já Peng et al. (2017), afirmam que os comportamentos desenvolvidos por agentes em ambientes simulados são frequentemente específicos para características do simulador. Portanto, a metodologia empregada no trabalho visa transpor o *reality gap* pela aleatoriedade na dinâmica do simulador durante o treinamento. A abordagem é demonstrada sobre um braço manipulador na tarefa de colocar um objeto em um ponto específico e seus resultados são satisfatórios com a transferência para o ambiente físico. Os autores defendem a ideia que a aleatoriedade na dinâmica, mesmo usando um simulador simples, pode ser uma abordagem efetiva para o desenvolvimento de estratégias que podem ser transferidas diretamente para o mundo real devido a sua robustez às dinâmicas do sistema. O modelo dinâmico proposto do manipulador sofre alterações nos parâmetros de massa de cada elo, amortecimento de cada junta, massa, fricção, altura da mesa onde fica o objeto a ser manipulado, ganhos para controle de posição, tempo de cada ação (latência) e ruído de sensores.

Claes (2017) abordou a influência de parâmetros chave para gerar robustez ao aprendizado do controle de um robô quadrupede, sendo adicionado ruído à morfologia do robô durante o processo de otimização (evolução). Os parâmetros investigados foram

rigidez das molas nas patas, fricção das patas e massa total. Todos os parâmetros com ruído apresentam melhores respostas no nível de velocidades, principal comportamento avaliado. Com os testes no robô real, verificou-se que em alguns casos o robô executou velocidades similares e em outros teve dificuldades devido à fricção das patas (característica difícil de ser modelada) fazendo com que robô executasse uma trajetória curva. No trabalho são realçadas características que foram utilizadas para selecionar o simulador, tais como, facilidade no uso, execução rápida que permita uma simulação mais rápida que o processamento em ambiente real, exatidão (ou seja, contemple características físicas do robô) e visualização da simulação.

Nos últimos anos, o termo *reality gap* vem sendo utilizado em outras áreas que também visam desenvolver controladores autônomos. Neste sentido, Ruud, Samuelsen e Glette (2016) propuseram a aplicação de aprendizado por reforço juntamente com algoritmo genético para evoluir parâmetros do controlador de um robô de quatro patas. Como resultado verificou-se que o aprendizado evolutivo melhora o desempenho do controlador se comparado ao processo somente com algoritmo genético e, por fim, verifica-se que a aplicação de estratégias de aprendizado após a transferência minimizam ainda mais o *reality gap* apresentado.

Além disso, as técnicas empregadas em RE estão sendo abordadas quando se trata em aprendizado por reforço, como em Tobin et al. (2017), que utiliza o domínio aleatório em simulação para transpor o *reality gap* na identificação de objetos em uma imagem RGB, criada através de uma câmera simulada de baixa fidelidade, para auxílio na identificação de objetos por meio do treinamento de uma rede neural por *deep learning*. Essa abordagem defende a hipótese que a variabilidade de simulação implica que modelos treinados em simulação irão generalizar o mundo real sem a necessidade de treinamento adicional.

Hanna (2017) propõe a fundamentação de uma ponte para o *reality gap* através do desenvolvimento de um método que permita uma melhor avaliação do aprendizado em simulação antes de ser executado em ambiente real. Para isso, utiliza aprendizado por reforço. Através de uma função de transformação que prevê dinâmicas do robô físico, é gerada uma ação para o simulador correspondente a uma ação realizada em ambiente real. Tal função pode ser treinada com métodos clássicos aplicados a redes neurais, como o *backpropagation*. Além disso, com o intuito de minimizar o número de dados, é proposto o uso da técnica *bootstrapping* para reduzir a quantidade de dados necessários para atingir um bom desempenho.

As abordagens citadas visam solucionar o *reality gap* sem ter contato direto com o ambiente real durante a otimização. Por outro lado, algumas pesquisas visam melhorar esse processo expondo o robô ao ambiente real e, consequentemente, utilizar o desempenho verificado com a transferência como uma realimentação para melhor sintonia do controlador, bem como de parâmetros do simulador. A seguir são apresentadas algumas

abordagens nesse sentido.

1.1.2.2 Otimização parte por simulação e parte em ambiente real

Miglino, Lund e Nolfi (1995) também trabalharam com a incorporação de ruído ao simulador como solução, e além disso, verificou-se ainda que se uma queda expressiva no desempenho é apresentada com a transferência após o processo de otimização via simulação, resultados melhores podem ser obtidos pela continuidade do processo evolutivo em ambiente real para poucas gerações.

Zagal, Solar e Vallejos (2004) apresentaram uma metodologia que se utiliza da evolução por simulação com algumas transferências para o robô em ambiente físico durante o processo evolutivo. A partir das transferências para o ambiente real, é medido o desempenho do controlador (*fitness*) e utilizado este valor para sintonizar os parâmetros do próprio simulador, a fim de minimizar os efeitos com as transferências seguintes. O processo é dividido em três módulos, um para o ambiente simulado, outro para o ambiente real e um terceiro para a sintonia dos parâmetros do simulador. No primeiro e último módulos são utilizados algoritmos genéticos e no segundo, aprendizado por reforço.

Brule (2008) também aplica uma metodologia similar, chamada de intercalação. Em seus experimentos são testados métodos de evolução que intercalam o processo evolutivo em simulação e no robô real. Três experimentos são estabelecidos, um em que o processo inicia por simulação (grande parte das gerações são executadas nesta etapa) e em seguida é transferido para o ambiente real, dando continuidade à evolução. No segundo experimento, esse processo é repetido duas vezes e no terceiro, três vezes, sempre mantendo o mesmo número de gerações no total. Como resultado, foi verificado que no primeiro experimento o desempenho foi inferior aos demais, e entre os dois seguintes, as discrepâncias notadas foram mínimas e ambos apresentaram melhores resultados com a transferência ao fim do processo evolutivo.

Ainda na mesma linha, Mouret, Koos e Doncieux (2013), propuseram uma abordagem denominada Transferibilidade. Esta abordagem consta de uma formulação baseada na *fitness* (função que mede o desempenho do controlador) e na transferibilidade, estimada pela disparidade entre simulação e realidade através da análise comportamental do robô. É criado um modelo baseado em interpolação para estimar essa disparidade e com isso se tem uma estimativa de como os controladores não transferidos viriam a se comportar no ambiente real. A transferência também é regida por um critério de disparidade que restringe o número de transferências a serem realizadas durante o processo evolutivo. Através dele é possível avaliar o comportamento dos controladores simulados em relação aos que já foram transferidos e os que atingirem um determinado limiar tornam-se aptos a serem portados ao robô real para continuação do processo evolutivo até que sejam atingidos critérios estabelecidos.

Como visto, muitos dos trabalhos citados vêm sendo desenvolvidos total ou parcialmente em simuladores. Porém, a falta de consistência entre simulação e realidade faz com que surja a necessidade de adequar a implementação de muitas características do mundo real ao simulador, como ruídos e incertezas dos sensores e não linearidades dos atuadores (COUCEIRO; VARGAS; ROCHA, 2014). Estas abordagens, de uma maneira ou de outra, tentam incorporar ao modelo de simulação características dinâmicas do sistema, principalmente, baseadas na aleatoriedade de características específicas, sejam elas do robô ou do ambiente. Contudo, nenhuma das metodologias apresentadas traz uma solução fechada e em sua maioria são utilizadas estratégias de controle baseadas em redes neurais artificiais (NELSON; BARLOW; DOITSIDIS, 2009) integradas a algoritmos genéticos para sua otimização com a função de incorporar inteligência e adaptabilidade aos robôs.

Nolfi et al. (1994) ressaltam o uso de redes neurais pela sua facilidade de explorar várias formas de aprendizado, auxiliando na escolha de um processo evolutivo rápido e pela sua robustez a ruído nas interações robô-ambiente. Os autores também acreditam que os componentes primitivos (pesos sinápticos e nós) manipulados pelo processo evolutivo devem estar no mais baixo nível possível para evitar escolhas indesejáveis pelo projetista.

A Lógica *Fuzzy* é uma outra estratégia de controle que possibilita emular a inteligência humana por intermédio da sua descrição baseada na experiência do projetista. Essa inteligência artificial também é encontrada nas redes neurais artificiais, porém, esta possibilita o seu aprendizado por meio de dados fornecidos pelo ambiente e não mediante o conhecimento de um especialista. A seguir são destacadas as características e potencialidades da lógica *Fuzzy* para sua aplicação em robótica evolutiva.

1.1.3 Abordagem *Fuzzy*

Os sistemas *fuzzy* consideram o modo como a falta de exatidão e a incerteza são descritas facilmente por termos linguísticos (como pouco, muito, baixo, alto, lento, rápido, etc.) e, fazendo isso, tornam-se suficientemente poderosos para manipular de maneira conveniente o conhecimento *a priori* sobre sistemas dinâmicos (GOMIDE; GUDWIN; TANSCHEIT, 1995). Vale ressaltar que nenhum trabalho envolvendo a lógica *fuzzy* foi encontrado na literatura no âmbito da robótica evolutiva com o intuito de averiguar seu comportamento diante do *reality gap*, sendo esse o ponto de contribuição do presente trabalho.

A complexidade e falta de informação *a priori* sobre um sistema é um fator preponderante na escolha de uma estratégia de controle para ser usada em robótica evolutiva (KONG; KOSKO, 1990). A rede neural pode ser indicada para sistemas complexos ou tarefas nas quais o projetista tenha pouca ou nenhuma informação sobre o sistema. E mesmo com a falta de conhecimento, um controlador *fuzzy* pode ser adotado, permitindo

que um algoritmo genético possa sintonizar muitos dos seus parâmetros (número e estruturas das funções de pertinência de entrada e saída, método de defuzzificação, estruturas das regras e método de inferência), por outro lado, todo conhecimento que for incorporado ao controlador *fuzzy* pode direcioná-lo mais facilmente à solução do problema.

Tanto o controlador *fuzzy* quanto o controlador neural possuem resistência a ruído. Porém, a rede neural funciona como uma função que a partir de uma entrada pontual gera uma saída também pontual, podendo ser interpretada como uma função com domínio e imagem específicas, como pode ser visto na Figura 1(a). Portanto, se uma incerteza for atribuída à entrada, a depender da superfície de controle (se varia suavemente ou não) sua saída pode variar suavemente ou de forma muito brusca, levando a saída para um ponto distante do que realmente seria. Já no controlador *fuzzy* as entradas são representadas por termos linguísticos que representam conjuntos, isso possibilita que a entrada com incerteza dentro de uma região gerará uma saída em sua região respectiva, resistindo mais a ruído na entrada do sistema, vide Figura 1(b).

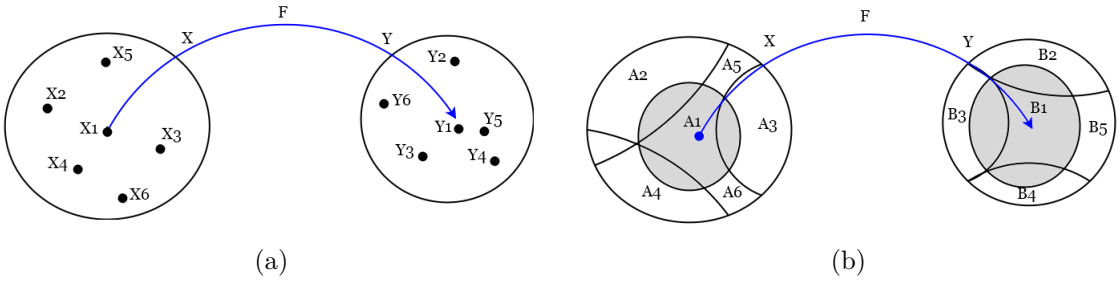


Figura 1 – Geometria de uma função de estimação neural e *fuzzy* (modificada de Kong e Kosko (1990)). A abordagem neural (a) utiliza muitas amostras de pontos e a abordagem fuzzy (b) usa poucos conjuntos *fuzzy*.

Outra característica relevante dos controladores *fuzzy* é a fácil interpretação do mesmo, devido ao uso da linguagem natural por termos linguísticos e seu processamento ser executado serialmente, seguindo os passos: fuzzificação, inferência e deffuzificação. Já o controlador neural, executa um processo paralelo, o que dificulta sua interpretação, além de impossibilitar a inserção de conhecimento *a priori* sobre o sistema. Também não há uma regra para indicar o número de neurônios, camadas e neurônios por camada em seu projeto.

Sendo conhecido o sistema a ser controlado, um controlador *fuzzy* pode ser indicado. O controlador *fuzzy* é simples de ser implementado quando se possui conhecimento estruturado do sistema que se deseja controlar. Contudo, depende muito do projetista, para definir os conjuntos *fuzzy*, as funções de pertinência, seus universos de discurso e o conjunto de regras a ser utilizado. Os dois controladores são, acima de tudo, algoritmos determinísticos (KONG; KOSKO, 1990).

Em seguida, serão apresentados alguns trabalhos que apontam a possibilidade

dessa estratégia de controle proporcionar maior robustez às incertezas do modelo de simulação.

Kong e Kosko (1990) desenvolveram um trabalho que visou comparar um controlador baseado em rede neural com outro baseado em lógica *fuzzy* fora do escopo da robótica evolutiva. Essas duas estratégias foram implementadas para realizar o controle de navegação de um robô com tração diferencial, em que as saídas dos controladores definem o ângulo de direção das rodas para que o robô entre de ré em uma doca. Foram analisadas as trajetórias e a robustez de ambos, e como resultado, verificou-se que o controlador *fuzzy* apresentou uma resposta mais suave que a rede neural e seu tempo de treinamento foi menor.

Segundo Cordón et al. (2004), durante a década de 1990 houve um aumento no interesse por sistemas com aprendizado e capacidade adaptativa, nos quais o processo genético aprende ou sintoniza diferentes componentes de um sistema *fuzzy* baseado em regras, o qual apresenta uma relação entre entradas e saídas descritas por regras do tipo se-então.

Dentro desse escopo, Matellán, Fernández e Molina (1998) desenvolveram um trabalho utilizando *fuzzy* genético, oriundo da união do controlador *fuzzy* com um algoritmo genético, de modo que a evolução foi responsável por gerar as regras do controlador para um robô autônomo. Essas regras foram obtidas a partir de variáveis linguísticas pré-definidas de acordo com o conhecimento humano. O processo foi desenvolvido em ambiente real. De acordo com os autores, os motivos que os levaram a utilizar a lógica *fuzzy* foram a possibilidade de escolher a estrutura das regras e as variáveis linguísticas de modo a expressar o comportamento desejado, sendo possível avaliar com facilidade os resultados obtidos com a evolução, e além disso, os controladores *fuzzy* são bastante flexíveis e a teoria de conjuntos *fuzzy* é bem definida e tem apresentado bons resultados em muitos sistemas autônomos.

Halal e Dumitrache (2007) apresentaram um controlador *fuzzy* com otimização baseada também em algoritmo genético, cuja função era sintonizar as funções de pertinência. Esse sistema genético possibilitou que o robô aprendesse a se locomover a um ponto de destino evitando choque com obstáculos.

Já Kim, Park e Harashima (2001) propuseram um controlador *fuzzy* auto organizado, assim denominado por eles, para otimizar as funções de pertinência das entradas e das saídas e também para determinar a base de regras. O processo foi dividido em três etapas, cada uma delas regida por um algoritmo evolutivo. Na primeira etapa, as funções de pertinência de saída são geradas, então, no segundo estágio, a base de regras é variada a fim de incrementar o desempenho já alcançado. Esses dois sistemas competem para selecionar os melhores indivíduos. Por fim, a sintonia das funções de entrada é realizada. Esse trabalho traz como contribuição a possibilidade de sintonizar automaticamente tanto as

funções de pertinência do controlador quanto a base de regras. Esta metodologia foi aplicada, por simulação, a um robô móvel de tração diferencial na execução de uma trajetória circular e outra linear e verificou-se seu comportamento em relação ao *setpoint* usando-se a odometria do robô, o que apresentou resultados satisfatórios.

Hoffmann e Pfister (1996), utilizaram um algoritmo genético para sintonizar as regras de um controlador *fuzzy* e um outro aplicado para co-evoluir um conjunto de situações de treinamento para avaliar o desempenho do controlador. Após a otimização em simulação, a estratégia de controle foi transferida para o ambiente real. A simulação empregou um modelo simplificado dos sensores reais, baseado em uma suposição ideal sobre a característica de sinais ultrassônicos. Quando transferido para o robô real, pequenas diferenças em relação à simulação foram observadas. Além disso, foi possível verificar um comportamento adequado mesmo tendo sido treinado com um modelo de sonar simplificado. A fim de verificar a generalização do controlador, o mesmo foi posto em mapas diferentes dos utilizados durante o processo evolutivo, e verificou-se que quando a geometria era similar à geometria dos mapas utilizados na evolução, a tarefa era executada com sucesso, enquanto que falhas ocorriam quando os mapas apresentavam diferenças substanciais.

Já em Hoffmann (2001) foram desenvolvidos dois controladores *fuzzy* genéticos: um com a estratégia de evolução do fator de escala entrada-saída e as funções de pertinência e outro que aprende as regras. O primeiro é aplicado ao controle de um pêndulo sobre um carro por simulação, e o segundo, à tarefa de navegar evitando obstáculos. A tarefa de evitar obstáculos é embarcada em robô real e é executada com êxito pelo robô, porém não são discutidas as discrepâncias de comportamento com a transferência da simulação para o ambiente físico.

A seguir são delineados os objetivos do trabalho e, posteriormente, como está organizada a estrutura do mesmo.

1.2 Objetivos

1.2.1 Objetivos gerais

O objetivo deste trabalho é verificar se a RE pode ser usada em conjunto com sistemas de controle *fuzzy* no desenvolvimento de robôs autônomos para lidar com o *reality gap*. Além disso, pretende-se estudar a robustez de controlador *fuzzy* comparado a um controlador neural quando ambos são derivados através de um algoritmo genético apenas em simulação antes de serem transferidos para o ambiente real.

1.2.2 Objetivos específicos

Para atingir os objetivos deste trabalho, são propostos os seguintes objetivos específicos:

- Desenvolver um controlador *fuzzy* e um controlador neural com parâmetros ajustados usando algoritmos genéticos via simulação.
- Transferir os controladores a um robô real para realização de experimentos;
- Utilizar métricas de desempenho de arquiteturas de navegação de robôs móveis para comparar os resultados obtidos por simulação com os resultados dos experimentos reais.

1.3 Estrutura do trabalho

A sequência desta dissertação está estruturada da seguinte maneira:

Capítulo 2 - apresenta uma fundamentação teórica acerca da modelagem de robôs, robótica evolutiva (com atenção especial aos algoritmos genéticos), redes neurais artificiais e lógica *fuzzy*;

Capítulo 3 - apresenta uma descrição da metodologia planejada para atingir os objetivos propostos com o projeto, abordando tarefa, robô e simulador adotados, as estruturas de controle desenvolvidas, o algoritmo genético utilizado e as métricas de avaliação dos sistemas perante a transferência para o ambiente real;

Capítulo 4 - apresentam-se a descrição dos experimentos realizados, resultados e discussão dos mesmos;

Capítulo 5 - são discutidas as conclusões e recomendações de trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo é apresentada a fundamentação teórica necessária para o desenvolvimento da proposta do trabalho. Inicialmente é realizada uma explanação a respeito da modelagem de robôs, seguida por um aprofundamento referente a Robótica Evolutiva, e por fim, são introduzidos os controladores baseados em redes neurais artificiais e em lógica *fuzzy*.

2.1 Modelagem do Robô

Grande parte dos simuladores utilizados no desenvolvimento de controladores incluem somente o modelo cinemático do robô, deixando de lado a modelagem dinâmica.

Cinemática é o estudo mais básico de como se comporta um sistema robótico. Relaciona-se ao comportamento mecânico do sistema, estabelecendo restrições na movimentação dos robôs. Em robótica móvel, é preciso compreender o comportamento do robô, tanto no que diz respeito ao desenvolvimento apropriado do robô móvel para tarefas, quanto para criar a estratégia de controle, sendo que a modelagem cinemática é que dá o suporte a este desenvolvimento. Por outro lado, a modelagem dinâmica do robô impõe restrições adicionais ao espaço de trabalho e trajetória devido à incorporação de massa e força ao modelo (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

A partir dos estudos realizados com a revisão bibliográfica, é evidente a importância da modelagem dinâmica dos atuadores, características paramétricas e sensoriamento na modelagem de robôs. Um robô móvel, geralmente, é modelado como um corpo rígido com rodas, operando no plano horizontal. O robô com tração diferencial possui duas rodas, cada uma com raio r (vide Figura 2), sendo um ponto P centrado entre as duas rodas considerado o centro de massa do robô e ponto de referência local. Para especificar a posição do robô no plano é estabelecida uma relação entre a referência global e a referência local do robô, como pode ser visto na Figura 2. Os eixos X_I e Y_I definem arbitrariamente a base inercial do plano de referência global. A base X_R, Y_R define os eixos relativos a P no chassi do robô e refere-se às coordenadas locais. A posição P na referência global é especificada por coordenadas x e y, e a diferença angular entre a referência global e local é dada por θ. Assim, é possível descrever a posição do robô como um vetor com três elementos, $\xi_I = [x, y, \theta]$, em que o I indica referência global.

Porém, os raios das rodas de robôs reais podem apresentar pequenas diferenças, o centro de massa pode estar um pouco deslocado e essas características unidas a outras incertezas não consideradas no simulador (sejam paramétricas ou relacionadas as simpli-

ficações realizadas) podem influenciar no desempenho do controlador com a transferência para ambientes reais.

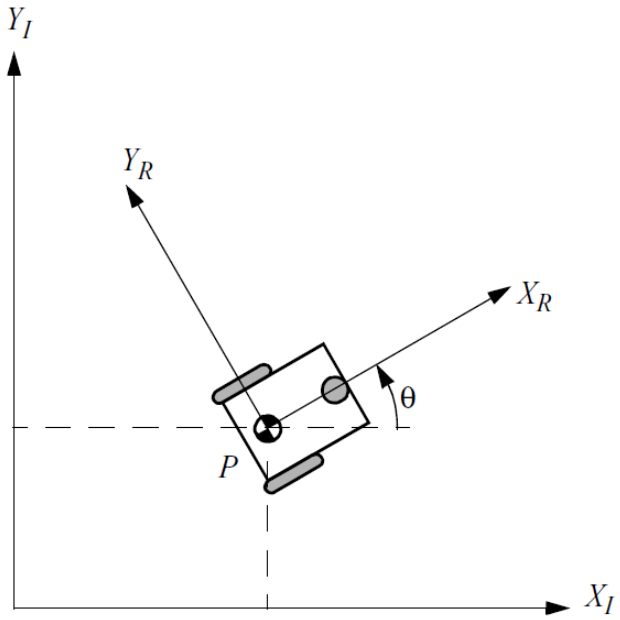


Figura 2 – Referência global e local do robô diferencial (Fonte: Siegwart, Nourbakhsh e Scaramuzza (2011)).

Assim, diante do que foi apresentado, alguns trabalhos recentes vêm tratando da investigação dos efeitos paramétricos do modelo no desenvolvimento de estratégias de controle, a exemplo de Milano, Carvalho e Nolfi (2017), Peng et al. (2017) e Claes (2017). A qualidade do modelo depende do quanto a sua saída se parece com a saída real. Como um simples modelo fixado não pode responder do mesmo modo que a planta real, necessita-se, pelo menos, de um conjunto de modelos. Portanto, não se deve usar somente um modelo parametrizado, mas um conjunto de modelos que permita que a dinâmica da planta seja melhor representada (ZHOU; DOYLE, 1998). Porém, essa abordagem não será empregada no presente trabalho, pois, um levantamento de parâmetros relevantes exigiria um estudo aprofundado quanto a tarefa a ser adotada, robô a ser utilizado e, possivelmente, levantamento do modelo do sensor e do atuador, para que assim fosse possível adotar critérios que permitissem uma escolha plausível dos parâmetros a serem variados, sejam relacionados ao ambiente ou ao próprio robô. No presente trabalho, foi adotado um simulador com parametrização fixa e que contempla o atraso cinemático e uma dinâmica não linear para o robô. Essa dinâmica modela a resposta dos atuadores com aceleração constante. Tais características o caracteriza como um modelo puramente determinístico. Esta escolha foi feita devido o intuito do trabalho consistir em averiguar como controladores *fuzzy* e neural evoluídos por simulação se comportam quando transferidos para experimentos reais. A próxima seção é dedicada a apresentar os conceitos básicos sobre robótica evolutiva, necessários para o desenvolvimento deste trabalho.

2.2 Robótica Evolutiva

A Robótica Evolutiva é um campo de pesquisa que usa a técnica da Computação Evolutiva para solucionar tarefas em robótica sem uma interação explícita de um humano, ou seja, objetiva desenvolver métodos automáticos para criar controladores para robôs autônomos inteligentes, de maneira que não requeiram diretamente uma programação realizada por um indivíduo. Esta abordagem, requer somente a definição do problema a ser resolvido e a otimização evolutiva determina o comportamento requerido para solucioná-lo (SCHEPER; CROON, 2016).

A Computação Evolutiva é um ramo de pesquisa emergente da inteligência artificial que propõe um novo paradigma para solução de problemas inspirado na evolução natural das espécies. Os algoritmos baseados nesse paradigma possuem características como auto organização e comportamento adaptativo, no qual uma população de indivíduos se reproduz e compete pela sobrevivência, em que os indivíduos mais adaptados sobrevivem e transferem suas características às gerações seguintes.

Dentre os algoritmos evolutivos existentes, destacam-se os algoritmos genéticos, introduzidos por Holland (1975). De acordo com Silva (2005), os algoritmos genéticos são mais difundidos e estudados devido à sua flexibilidade relativa, à simplicidade de implementação e eficácia em realizar busca global em ambientes adversos. Portanto, esta técnica será tratada em detalhes na subseção seguinte.

2.2.1 Algoritmos Genéticos

O algoritmo genético (AG) é um método de resolver problemas de otimização com e sem restrições. Emprega uma terminologia originada da teoria da evolução natural e da genética em que uma população de indivíduos (soluções) representados por cromossomos, os quais contêm a codificação (genótipo) de soluções para o problema em questão (fenótipo), é gerada aleatoriamente. O cromossomo é usualmente implementado na forma de uma lista de atributos ou vetor, onde cada atributo é conhecido como gene e representa uma determinada característica. Os possíveis valores que um determinado gene pode assumir são denominados alelos (ZUBEN, 2000).

O processo evolutivo é dirigido por uma função objetivo. Essa função associa a cada indivíduo um valor de aptidão (*fitness*) referente ao seu desempenho na execução de uma dada tarefa, seja em ambiente real ou simulado (PEREZ, 2009). A *fitness* é utilizada para selecionar os indivíduos que depois se reproduzem (através dos operadores genéticos que possibilitam a seleção e variação genética) para criar uma nova geração até que se atinja o número total de gerações estipuladas ou o problema seja solucionado de acordo com algum critério de projeto. Caso contrário, os operadores genéticos são aplicados e as novas soluções são novamente avaliadas em ambiente real ou simulado. Vale ressaltar

que em RE as avaliações das soluções são realizadas em uma plataforma robótica, real ou simulada, já na computação evolutiva não é necessário esse teste, a solução é avaliada diretamente. A Figura 3 exemplifica todo esse procedimento.

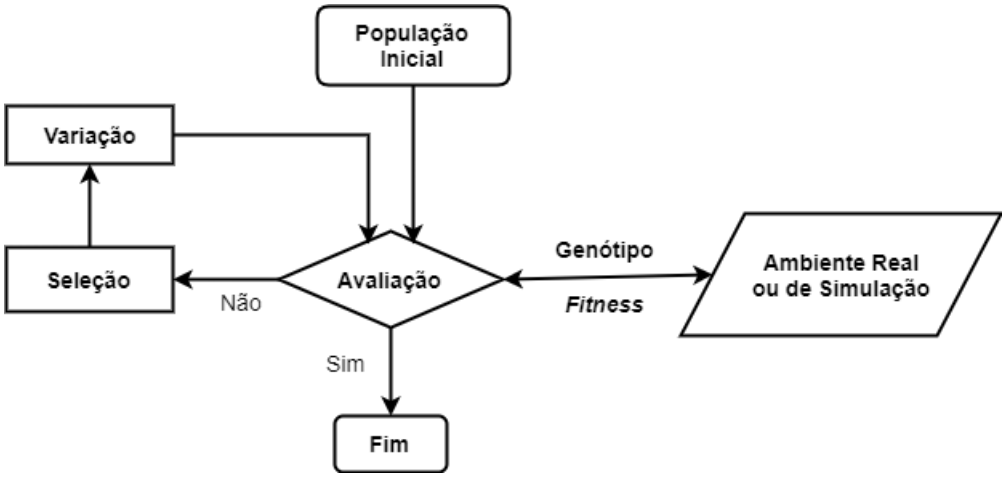


Figura 3 – Fluxograma de um Algoritmo Genético aplicado à Robótica Evolutiva (DONCIEUX et al., 2015).

As populações geralmente são codificadas em arranjos binários de tamanho fixo. Entretanto, em diversas aplicações práticas a utilização de codificação binária leva a um desempenho insatisfatório. Em problemas de otimização numérica com parâmetros reais, algoritmos genéticos com representação inteira ou em ponto flutuante frequentemente apresentam desempenho superior à codificação binária (ZUBEN, 2000). O método mais comum utilizado na criação da população é a inicialização aleatória dos indivíduos, porém, se algum conhecimento inicial a respeito do problema estiver disponível, pode ser utilizado na inicialização da população.

Os algoritmos genéticos empregam operadores genéticos para criar uma variabilidade genética e estimular a busca pelo espaço de soluções. Os operadores frequentemente utilizados são o *crossover* ou recombinação, reprodução (por métodos de seleção) e mutação.

Reprodução: também chamada de reprodução assexuada, seleciona um indivíduo conforme sua aptidão e copia esse indivíduo para a nova geração sem modificá-lo. Um desses operadores é o *roulette wheel*, o qual atribui a cada indivíduo de uma população uma probabilidade de passar para a próxima geração proporcional ao seu *fitness* em relação ao somatório dos valores de *fitness* de todos os indivíduos da população atual, $s_n = \frac{f(i)}{\sum_{i=1}^{n} f(i)}$, em que s_n representa a probabilidade de passagem do n-ésimo indivíduo, $f(n)$ representa a *fitness* do n-ésimo indivíduo e n é o número de indivíduos da população. Sendo assim, a seleção de indivíduos por *roulette wheel* pode fazer com que o melhor indivíduo da população seja replicado várias vezes ou até mesmo perdido, ou seja, não é garantida sua passagem para a próxima geração. É ilustrado na Figura 4 o princípio desse operador.

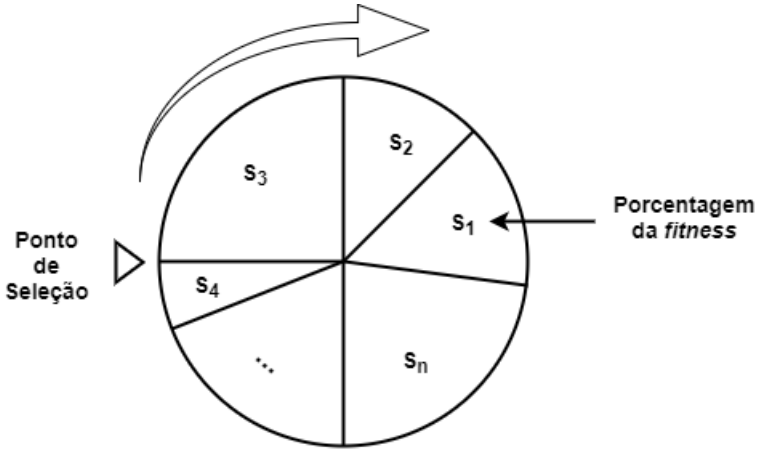


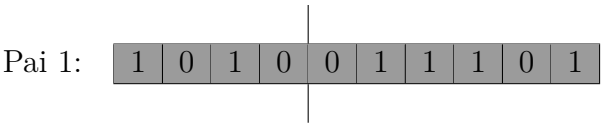
Figura 4 – Princípio de funcionamento do operador *Roulette Wheel* (SILVA, 2005).

Pode ser citado também a seleção baseada em *ranking*. Esta estratégia utiliza as posições dos indivíduos quando ordenados de acordo com a *fitness* para determinar a probabilidade de seleção. Podem ser usados mapeamentos lineares ou não lineares para determinar a probabilidade de seleção. Uma variação deste mecanismo é simplesmente passar os N melhores indivíduos para a próxima geração, estratégia essa conhecida como seleção elitista ou elitismo, cujo objetivo é simplesmente manter sempre uma parcela dos melhores indivíduos da geração atual na geração seguinte.

Outros possíveis mecanismos são baseados na diversidade, em que são selecionados os indivíduos mais diversos da população, seleção bi-classista, em que são selecionados os $P\%$ melhores indivíduos e os $(100 - P)\%$ piores indivíduos, e a seleção aleatória, em que são selecionados aleatoriamente N indivíduos da população por dois mecanismos, um salvacionista, em que seleciona-se o melhor e os outros aleatoriamente e a não salvacionista, em que seleciona-se todos os indivíduos aleatoriamente.

***Crossover* ou recombinação:** este operador cria novos indivíduos através da combinação de dois ou mais indivíduos. A ideia intuitiva por trás desse operador é a troca de informação entre diferentes soluções candidatas. No algoritmo genético clássico é atribuída uma probabilidade de *crossover* fixa aos indivíduos da população.

O operador *crossover* mais comumente empregado é o *crossover* de um ponto. Para a aplicação deste operador, são selecionados dois indivíduos (pais) e a partir de seus cromossomos são gerados dois indivíduos (filhos). Para gerar os filhos, seleciona-se um mesmo ponto de corte aleatoriamente nos cromossomos dos pais, e os segmentos de cromossomos criados a partir do ponto de corte são trocados, esse procedimento é exemplificado a seguir com o auxílio das Figuras 5 e 6.



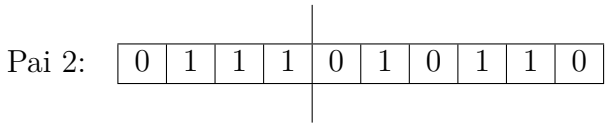


Figura 5 – Pais selecionados para operação de *crossover* (ZUBEN, 2000).

Após o *crossover* ser aplicado no ponto de corte entre os genes 4 e 5 em destaque, tem-se os seguintes indivíduos filhos.

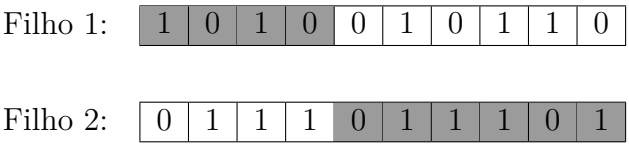


Figura 6 – Indivíduos filhos resultantes da operação de *crossover* (ZUBEN, 2000).

Uma extensão simples do *crossover* de um ponto é o de dois pontos, onde dois pontos de corte são escolhidos e o material genético é trocado entre eles. Outro tipo de *crossover* muito comum é o *crossover* uniforme, em que para cada gene do primeiro filho é decidido (com alguma probabilidade fixa) qual pai vai contribuir com seu valor para aquela posição.

No entanto, não há nenhum operador de *crossover* que claramente apresente um desempenho superior aos demais. Pode-se concluir que cada operador pode ser eficiente em determinada classe de problemas e ineficiente em outras (ZUBEN, 2000).

Mutação: é um operador que modifica aleatoriamente um ou mais genes de um cromossomo. Esta alteração é importante, pois acaba por criar novos valores de características que não existiam ou apareciam em pequena quantidade na população em análise. O operador de mutação é necessário para a introdução e manutenção da diversidade genética da população. A mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de solução, possivelmente, não será zero.

Considerando uma codificação binária, o operador de mutação padrão simplesmente troca o valor de um gene em um cromossomo de acordo com uma probabilidade (ou taxa de mutação). Desse modo, se um gene selecionado para mutação tem um valor '1', o seu valor passará a ser '0' após a aplicação da mutação, e vice-versa. Já para codificação do cromossomo com valores reais, o operador de mutação pode ser aplicado alterando o valor do gene ou somando um valor aleatório ao valor do gene atual, ambos com distribuição similar à da criação da população inicial, como descrito em Montana e Davis (1989).

Nesta seção foram apresentados os aspectos necessários para implementação de um algoritmo genético a ser aplicado em robótica evolutiva ressaltando suas principais

características e operadores. Nas duas próximas seções são apresentadas as estratégias de controle neural e *fuzzy*, respectivamente. A partir da sua descrição serão apontados os parâmetros de cada controlador que o algoritmo genético será encarregado de otimizar.

2.3 Redes Neurais Artificiais

Como visto anteriormente, em RE, as estruturas de controle baseadas em redes neurais artificiais são bastante difundidas por simular a inteligência humana e possibilitar ao robô tomar decisões em situações inesperadas. Portanto, será realizado um aprofundamento sobre o tema, pois será realizado o seu desenvolvimento com o propósito de comparação quanto a robustez com o controlador *fuzzy*.

De acordo com Haykin (2001), uma rede neural é um processador constituído de unidades de processamento simples, os neurônios, que tem a propensão natural para armazenar conhecimento experimental e torná-lo disponível. Assemelha-se ao cérebro humano nos seguintes aspectos: o conhecimento é adquirido pela rede a partir do ambiente através de um processo de aprendizagem e as forças de conexões entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.

2.3.1 Neurônio Artificial

O diagrama da Figura 7 apresenta o modelo de um neurônio artificial, que forma a base para o projeto de redes neurais. Nele é possível identificar quatro elementos básicos:

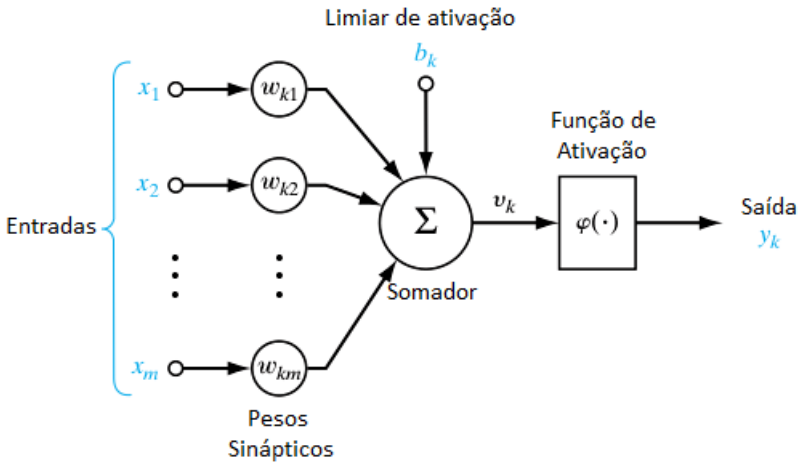


Figura 7 – Modelo artificial de um neurônio biológico (HAYKIN, 2001).

1. Um conjunto de sinapses ou elos de conexão, caracterizado por pesos ou forças próprias w_{km}. Especificamente, um sinal x_m na entrada da sinapse m conectada ao neurônio k é multiplicado pelo peso sináptico w_{km}. É importante notar a maneira como são escritos os índices do peso sináptico w_{km}. O primeiro índice se refere ao

neurônio em questão e o segundo se refere ao terminal de entrada da sinapse a qual o peso se refere. Ao contrário da sinapse do cérebro, o peso sináptico de um neurônio artificial pode estar em um intervalo que inclui valores negativos ou positivos;

2. Um somador para os sinais de entrada, ponderados pelas respectivas sinapses do neurônio;

3. Uma função de ativação $\varphi(.)$ para restringir a amplitude da saída do neurônio. A função de ativação é também referida como função restritiva, já que limita o intervalo permissível de amplitude do sinal de saída a um valor finito (no intervalo normalizado de $[0, 1]$ ou $[-1, 1]$). A função de ativação pode ser de três tipos básicos, a função limiar (ou Heaviside), função linear por partes e a função sigmoide, que é a mais utilizada (HAYKIN, 2001).

4. O neurônio também inclui um limiar de ativação (*bias*) aplicado externamente, representado por b_k. Possui o efeito de aumentar ou diminuir a entrada líquida da função de ativação, dependendo se ele é positivo ou negativo, respectivamente.

Em termos matemáticos pode-se descrever um neurônio k com m entradas (x_1, x_2, ..., x_m) como um produto escalar entre o vetor de entradas x e o vetor de pesos sinápticos correspondentes a cada uma delas w_k (com elementos w_{k1}, w_{k2}, ..., w_{km}) e posteriormente somar o resultado ao *bias* correspondente (b_k):

$$\upsilon_k = w_{k1}x_1 + w_{k2}x_2 + \cdots + w_{km}x_m + b_k \tag{2.1}$$

isso pode ser descrito matricialmente por

$$\upsilon_k = \mathbf{w_k^T x} + b_k \tag{2.2}$$

em que para um simples neurônio w_k é apenas um vetor de pesos e b_k uma constante. Portanto, a saída do neurônio (y_k) pode ser descrita como

$$y_k = \varphi\left(\upsilon_k\right) = \varphi\left(\mathbf{w_k^T x} + b_k\right) \tag{2.3}$$

em que $\varphi(.)$ representa a função de ativação.

Já para uma camada de neurônios, composta por n neurônios e m entradas, $\mathbf{W}$ torna se uma matriz na qual cada linha representa os pesos sinápticos do n-ésimo neurônio da camada, sendo representada da seguinte forma

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{bmatrix} \tag{2.4}$$

Neste caso o *bias* deixa de ser um escalar e passa a ser um vetor de tamanho n, e a Equação 2.3 pode ser usada normalmente, sendo agora $\varphi(.)$ uma função de ativação vetorial e y_k refere-se a saída da camada atual e entrada para as camadas subsequentes, caso haja.

2.3.2 Arquiteturas de redes neurais

A arquitetura da rede define a forma com a qual seus nerônios estão dispostos e o direcionamento das conexões sinápticas. De acordo com Haykin (2001), é possível identificar três classes de arquiteturas de rede fundamentalmente diferentes, *feedforward* de camada simples, *feedforward* de múltiplas camadas e recorrentes. Porém, existem outras topologias de redes, a exemplo das estruturas reticuladas, na qual se enquadra a rede de Kohonen. A maneira pela qual os neurônios de uma rede neural estão estruturados está intimamente ligada ao algoritmo de aprendizado utilizado para treiná-la.

2.3.3 Aprendizado

O aprendizado é um processo pelo qual os parâmetros livres de uma rede neural (pesos sinápticos, níveis do limiar de ativação e topologia) são adaptados através de um processo de estimulação pelo ambiente no qual a rede está inserida. O tipo de aprendizagem é determinado pela maneira que as modificações dos parâmetros ocorrem.

Neste trabalho, os pesos sinápticos de uma rede neural fixa são otimizados por um algoritmo genético, que é uma abordagem comumente empregada em robótica evolutiva.

2.4 Lógica *Fuzzy*

De acordo com Gomide, Gudwin e Tanscheit (1995), a Lógica *Fuzzy* (ou Nebulosa) é a lógica que suporta os modos de raciocínio que são aproximados ao invés de exatos. A modelagem e controle *fuzzy* de sistemas são técnicas para o tratamento de informações qualitativas de uma forma rigorosa. Derivada do conceito de conjuntos *fuzzy* de Zadeh (1965), a lógica *fuzzy* constitui a base para o desenvolvimento de métodos e algoritmos de modelagem e controle de processos, permitindo a redução da complexidade na implantação do projeto, tornando-se a solução para problemas de controle até então intratáveis por técnicas clássicas (GOMIDE; GUDWIN; TANSCHEIT, 1995).

Para melhor entendimento do funcionamento e implementação de um sistema *fuzzy*, serão introduzidos alguns conceitos fundamentais.

a) Variáveis Linguísticas

Uma variável linguística ou *fuzzy* é uma variável cujos valores são nomes de conjuntos *fuzzy*. Por exemplo, a velocidade angular de uma roda pode ser uma variável linguística assumindo valores baixa, média, alta, etc. Estes valores são descritos por intermédio de conjuntos *fuzzy* e são usualmente definidos por funções de pertinências (vide Figura 8). O formato de função de pertinência mais comumente aplicado é o triangular, embora curvas trapezoidais e gaussianas também sejam empregadas.

b) Conjuntos *fuzzy* ou funções de pertinência

Um conjunto *fuzzy* é um par ordenado (X, f), em que X é um vetor espacial (usualmente em $\mathbb{R}$) e f é um conjunto de funções de pertinência que mapeia X em um intervalo $[0, 1]$ de uma linha real $\mathbb{R}$, isto é, $f : X \rightarrow [0, 1]$.

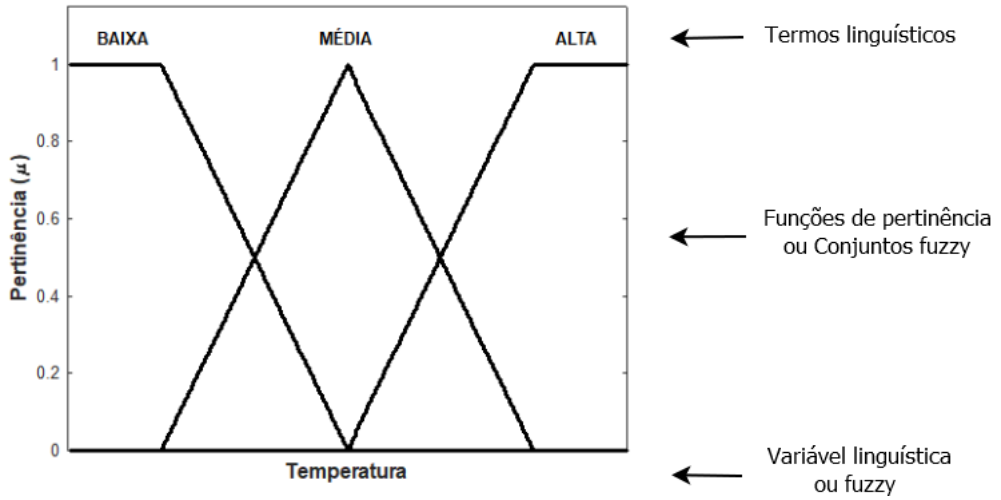


Figura 8 – Variável linguística Temperatura (Fonte: adaptado de Gomide, Gudwin e Tanscheit (1995)).

Embora estratégias de controle inteligente possam ser implementadas por outros meios, a exemplo, principalmente, das redes neurais artificiais, as implementações por lógica *fuzzy* frequentemente se tornam mais eficientes devido aos seguintes aspectos, de acordo com Simoes e Shaw (2007): estratégias de controle *fuzzy* nascem da experiência prática em vez de modelos matemáticos, portanto, uma implementação linguística é muito mais rápida e efetiva do ponto de vista de desenvolvimento e, em relação a soluções convencionais de controle, a lógica *fuzzy* é frequentemente mais eficiente, ao se comparar a codificação e o tempo computacional de execução (SIMOES; SHAW, 2007).

O controlador *fuzzy*, tipicamente, consiste de um estágio de entrada, nomeado de fuzzificação, um estágio de conhecimento base, um estágio de processamento, também chamado de inferência, e um estágio de saída, conhecido como defuzzificação (vide Figura 9):

1. O estágio de entrada recebe os valores atuais das variáveis de entrada e mapeia sensores e outras entradas para funções de pertinência (ou conjuntos *fuzzy*) apropriadas;

2. O conhecimento base contém informações sobre o domínio das variáveis e os conjuntos *fuzzy* associados com os termos linguísticos. A estrutura linguística das regras também é armazenada nesse estágio;

3. O estágio de processamento ou decisão lógica invoca as regras apropriadas e gera os resultados para cada uma com a ajuda dos valores das entradas e do conhecimento base, e depois combina os resultados obtidos com cada uma das regras;

4. O estágio de saída converte o resultado dessa combinação em um valor de saída numérico para a variável controlada. Os métodos mais utilizados na *defuzzificação* são centro da área ou centroide, centro do máximo e média do máximo (SIMOES; SHAW, 2007).

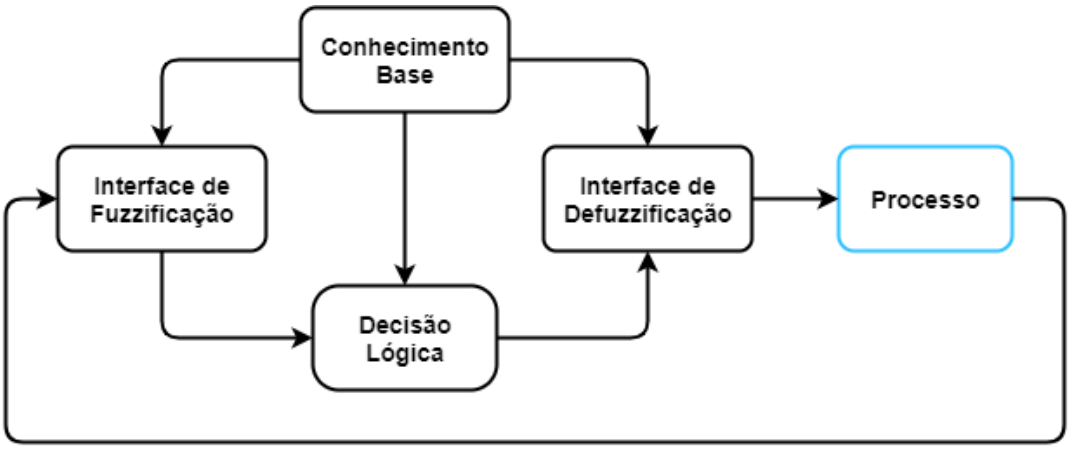


Figura 9 – Arquitetura de um controlador *fuzzy* típico (Fonte: traduzido de Abraham e Nath (2000)).

Este trabalho foca somente em sistemas *fuzzy* baseados em regras. Dependendo da estrutura particular da proposição do consequente das regras, dois principais modelos podem ser distinguidos: o Modelo *Fuzzy* linguístico, também conhecido como Modelo Mamdani, no qual tanto o antecedente como o consequente são proposições *fuzzy* (MAMDANI; ASSILIAN, 1975), e o Modelo *Fuzzy* Takagi-Sugeno, no qual o consequente é uma função explícita das variáveis antecedentes (TAKAGI; SUGENO, 1985). Esse segundo modelo baseado em regras foi introduzido para a aproximação de uma larga classe de sistemas não lineares.

Quando o consequente do Modelo *Fuzzy* Takagi-Sugeno são funções constantes, obtém-se o modelo *singleton*, que é um caso particular. Partindo do Modelo Mandani também pode-se obter esse modelo, sendo vista as proposições *fuzzy* consequentes como

valores constantes (*singletons*). Mais informações sobre este caso particular podem ser obtidas em Babuska (1998).

2.4.1 Modelo *Fuzzy* Genético

O sistema *fuzzy* típico não possui capacidade de adaptação. A obtenção do conhecimento base para a implementação de um controlador *fuzzy* é frequentemente baseada em uma abordagem tediosa e ineficaz de tentativa e erro (HOFFMANN, 2001). Porém, a computação evolutiva possibilita o desenvolvimento de sistemas adaptativos, inteligentes e robustos, os quais imitam a habilidade humana em lidar com incerteza e conhecimento incompleto. Portanto, para contornar as dificuldades na sintonia manual de um controlador *fuzzy*, o controlador pode ser integrado a algoritmos evolutivos para incorporar a capacidade de adaptabilidade, sendo essa integração nomeada modelo *fuzzy* genético.

De acordo com Hoffmann e Pfister (1996), existem duas abordagens principais no desenvolvimento de um controlador *fuzzy* genético. A primeira, visa sintonizar os parâmetros do conhecimento base (domínio das variáveis *fuzzy* e funções de pertinência), e a segunda abordagem, adapta a base de regras (parâmetros de agregação de antecedentes e semântica das regras).

No presente trabalho foi aplicada a primeira abordagem e desenvolvido um controlador *fuzzy* genético baseado no modelo Takagi-Sugeno com consequentes *singletons* (constantes), sendo essas constantes sintonizadas por um algoritmo genético.

3 Abordagem Proposta

Neste capítulo é apresentada a abordagem proposta a fim de cumprir com os objetivos do trabalho. A seguir, são detalhados a tarefa adotada nos experimentos, o robô e simulador utilizados, bem como a descrição das estratégias de controle implementadas, algoritmo genético e métricas de avaliação de desempenho perante a transferência do ambiente simulado para o ambiente real.

3.1 Tarefa

A tarefa adotada neste trabalho foi a navegação autônoma evitando obstáculos, pois de acordo com Nelson, Barlow e Doitsidis (2009), essa é a tarefa mais investigada em robótica evolutiva. Os autores também citam outras tarefas que podem ser encontradas na literatura, tais como, aprender a caminhar para robôs com patas, a busca por fonte de luz que é similar a navegação com destino, a busca de um objeto no ambiente e também a competição predador-presa.

A tarefa adotada facilita a reprodução do ambiente simulado (mapa) em ambiente real e também possibilita utilizar o robô de tração diferencial Pioneer 3-DX, disponível no laboratório de robótica da instituição de ensino (vide Figura 10).

Figura 10 – Pioneer P3-DX com Laser SICK LMS200.

3.2 Robô

O Pioneer 3-DX é um robô móvel de tração diferencial muito utilizado em pesquisas, em virtude da sua versatilidade, confiabilidade e durabilidade. Possui como finalidade

principal sua aplicação em pesquisas envolvendo atividades como mapeamento, localização, monitoramento, visão, cooperação entre múltiplos robôs, reconhecimento e navegação autônoma, sendo esta última tarefa o foco deste trabalho.

Esse robô possui duas rodas tracionadas de 19 cm de diâmetro e uma roda livre (tipo castor). Cada uma das rodas tracionadas contém um *encoder* de 500 *ticks* de precisão e o robô pode atingir uma velocidade de $1,4\ m/s$. Ao redor do seu corpo apresentam-se distribuídos 16 sensores ultrassônicos capazes de detectar objetos a uma distância de $7\ m$. O sistema de alimentação é composto por 3 baterias de $12\ V$, dando-lhe uma autonomia de 8 a 10 horas, aproximadamente. Conta com controladores PID (proporcional integral derivativo) de baixo nível atuando em cada um dos motores conectados às rodas, o que possibilita uma melhor ação do robô, aproximando a sua resposta à de um simulador. A especificação completa do Pioneer 3-DX pode ser consultada no site do fabricante Mobile Robots[1].

Devido a ineficiência do sensoriamento baseado em ondas sonoras, principalmente devido à forte influência do material (superfície do obstáculo) e da própria abertura dos sensores que geram uma incerteza sobre a real distância medida, foi adotado o sensor laser SICK LMS-200 que é capaz de varrer uma área de 180° graus com uma resolução angular de $0,5°$ e alcance de distância que atinge 8 m com precisão de 15 mm (vide Figura 10).

3.3 Simulador

No projeto foi adotado um simulador desenvolvido e disponibilizado pelo Grupo de Pesquisa em Robótica da Universidade Federal de Sergipe (GPR-UFS) baseado no modelo cinemático de robôs diferenciais que contempla o atraso cinemático e uma dinâmica não linear para o robô. Essa dinâmica modela a resposta dos atuadores com aceleração constante. O simulador foi desenvolvido na plataforma MATLAB, que é bastante difundida internacionalmente para a pesquisa acadêmica, o que possibilita uma fácil reprodução do trabalho, extensão para incorporar outras características dinâmicas e com isso permitir a difusão de conhecimento para robótica evolutiva, já que simuladores direcionados à área são escassos Mouret e Chatzilygeroudis (2017). A escolha deste simulador foi movida pela facilidade no uso e na manipulação de parâmetros, visualização da simulação, possibilidade de desenvolvimento dos controladores na mesma plataforma e interface direta com o robô real.

A Figura 11 apresenta a interface do simulador utilizado no projeto. A interface de comunicação com o robô físico e com o laser é realizada via comunicação serial, um *link* para cada dispositivo. O simulador permite quatro modos de navegação, simulada ou real

[1] Especificação completa do Pioneer P3-DX pode ser vista em: http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx

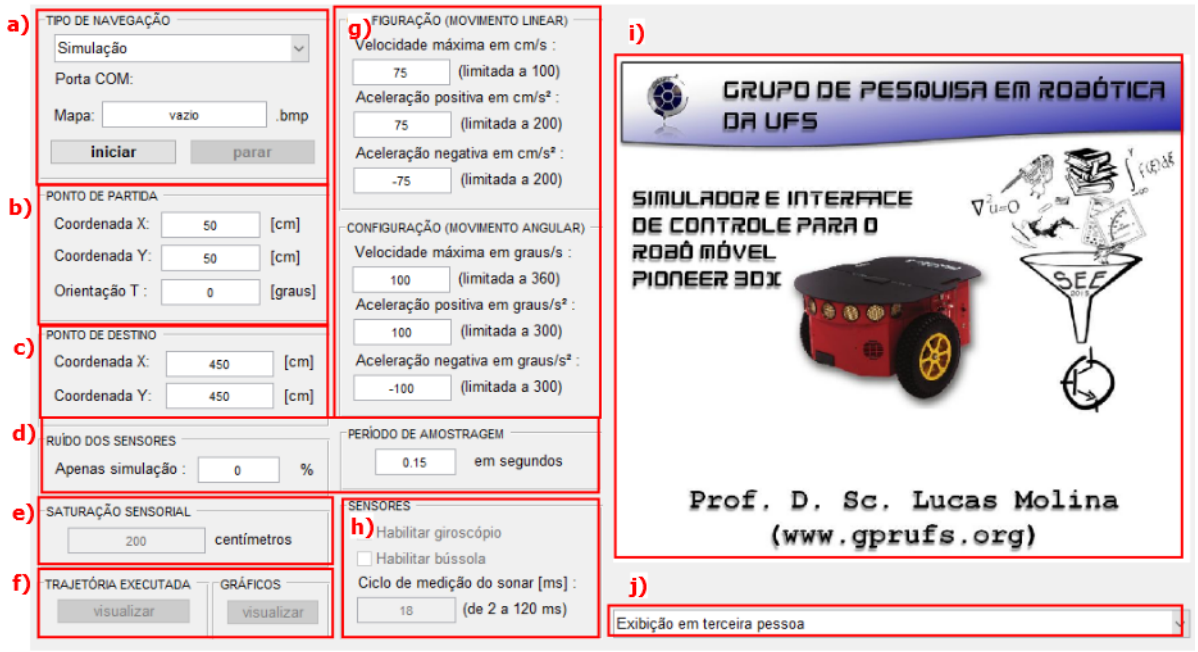


Figura 11 – Simulador e Interface de controle para o robô móvel Pioneer 3DX do GPRUFS

sendo que em ambas a navegação pode ser autônoma (ou seja, o controlador recebe dados dos sensores e gera as ações de controle para o robô) ou manual em que o usuário por meio de um *joystick* comanda o robô. E esta definição pode ser configurada no retângulo ressaltado com a Figura 11(a). Neste mesmo retângulo também é possível, em caso da simulação, selecionar o mapa (imagem) a ser utilizada no experimento.

Continuando a parametrização do simulador, o mesmo permite que o usuário defina limites de velocidade linear e angular do robô a ser executada pelo robô, obedecendo aos limites fornecidos pelo próprio fabricante, esta configuração é realizada na Figura 11(g). Neste caso os valores limites adotados foram 100 cm/s e 300°$/s$. Além desses parâmetros, a plataforma também permite controle da resposta dinâmica do mesmo através dos limites de aceleração linear e angular dos controladores de baixo nível (PID), ou seja, são atribuídos limites positivos e negativos que implicam em quão rápido são as mudanças de *setpoint* dos controladores de baixo nível. Estes parâmetros também são sintonizados na Figura 11(g).

Outras características que podem ser definidas no simulador são, posição de partida (configurada na Figura 11(b)), posição de destino quando aplicada ao experimento (configurada na Figura 11(c)), inserção de ruído nas leituras dos sensores e período de amostragem (ambos configurados na Figura 11(d)), limite de saturação sensorial (configurado na Figura 11(e)) e, por fim, a ativação de outros tipos de sensores, como giroscópio e bússola (configurado na Figura 11(h)).

A interface também permite acompanhar a navegação em tempo real, com duas visões, em primeira e terceira pessoa, selecionado na Figura 11(j) antes do início da nave-

gação e exibido na Figura 11(i). Ao fim da execução da tarefa são armazenadas variáveis para análises futuras do comportamento do robô, como posições e orientações durante o trajeto, leitura dos sensores e comandos de velocidade linear e angular enviados e executados. Os gráficos de trajetória e velocidades podem ser obtidos na Figura 11(f).

Mouret e Chatzilygeroudis (2017) indicam como características essenciais de simuladores para robótica evolutiva a rapidez, estabilidade, precisão e utilização de plataformas abertas. O simulador utilizado neste projeto contempla rapidez, pois permite maior velocidade no processo evolutivo se comprado ao desenvolvimento em um robô real, o cálculo de *fitness* do robô é realizado ao fim do experimento, reduzindo a perda de tempo durante o experimento. O código utilizado para simulação é aberto, de modo que o projetista possa identificar erros de programação e melhor sintonizar parâmetros do processo evolutivo. Como auxílio, o simulador também permite a exibição através do gráfico da trajetória em duas dimensões do experimento em tempo real e, ao fim do experimento, gera gráficos de velocidade angular e linear com base na odometria. Além disso, dados de leitura dos sensores são armazenados durante todo o experimento. Todas essas características possibilitam verificar a estabilidade do sistema. Em relação à precisão, o modelo segue as caraterísticas dinâmicas explicitadas no manual do fabricante e, além disso, possibilita o incremento de características desejadas com o projeto, bem como inserção de variação paramétrica. Por fim, tanto o controlador quanto o algoritmo genético são implementados na mesma plataforma em que foi desenvolvido o simulador. E o simulador pode ser disponibilizado a outros pesquisadores.

3.3.1 Sensoriamento

O modelo considera um sensoriamento com alcance de $3m$. As leituras de interesse são as que correspondem aos ângulos dos 8 sensores ultrassônicos frontais do Pioneer 3-DX ($0°, 40°, 60°, 80°, 100°, 120°, 140°$ e $180°$). Esses sensores são agrupados em três conjuntos, reduzindo a 3 entradas para os controladores que serão apresentados posteriormente. A Figura 12 apresenta esses agrupamentos, em que os sensores $0-2$ pertencem ao conjunto da direita, $3-4$ ao conjunto frontal e os demais, $5-7$, ao conjunto da esquerda. O nível de atuação de cada um dos conjuntos é dado pela mínima leitura dos sensores do conjunto.

O tempo de amostragem utilizado no projeto foi de $0,6s$, em decorrência do laser LMS200 exigir $0,5s$ para realizar uma varredura e transmitir os dados via comunicação serial com taxa de transmissão (*baud rate*) de 9600 $bits/s$. O tempo restante é necessário para geração das ações de controle e envio para o robô.

O sensoriamento recebe como parâmetro o alcance do sensor, levando-se em consideração o limite especificado no manual. No caso é utilizado o laser com ângulos específicos, apresentados anteriormente. A leitura é dada por uma varredura em cada pixel da imagem na direção de cada sensor. O mapa utilizado em simulação é representado por

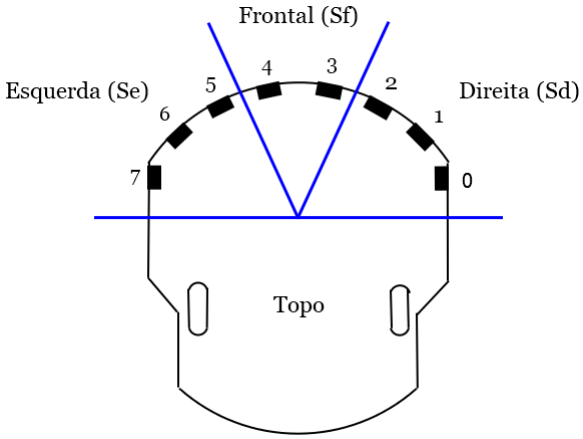


Figura 12 – Vista dos conjuntos de entradas utilizados nos controladores.

uma imagem preto e branca em que cada pixel representa uma área de 1 cm x 1 cm em ambiente real. Os obstáculos são identificados com pixel na cor preta e as regiões livres por pixel na cor branca. A leitura fornecida pelo sensor detecta o obstáculo por meio de averiguação de cada célula do mapa na direção do sensor até que se atinja o alcance do sensor ou detecte um obstáculo, ou seja, pixel preto.

3.3.2 Modelo Cinemático e Dinâmico

As leituras realizas são processadas pelos controladores e geram as ações de controle, velocidade linear e angular, respectivamente. Antes destes dados serem aplicados ao modelo cinemático do robô, é checado se essas ações estão dentro dos limites estipulados, caso ultrapassem são truncadas nos limites que o robô pode receber, 100 cm/s e 300°$/s$, respectivamente.

O simulador apresenta um atraso cinemático inerente ao robô que é de aproximadamente 25 milissegundos. De acordo com o tempo de amostragem definido pelo usuário, o simulador realiza essa compensação. A compensação é realizada criando-se um vetor de comandos de tamanho $atraso/t_{amos}$, em que o t_{amos} corresponde ao tempo de amostragem. Isso é gerado caso o tempo de amostragem definido seja menor que o atraso cinemático, isso é possível em aplicações utilizando os sensores de ultrassom. Caso contrário, ou seja, se o tempo de amostragem for igual ou superior, ele cobrirá todo o atraso, e o vetor terá tamanho 1, o que indica que receberá diretamente o comando atual gerado. Essa abordagem foi aplicada devido o uso do laser. Quando se tem um vetor de atraso de dimensão maior que 1, os comandos de velocidade são recebidos sempre em sua última posição do vetor e são executados os comandos armazenado na primeira posição (v e ω) de modo a cumprir com o atraso dinâmico do robô real.

O modelo cinemático é descrito com base na matriz de rotação que relaciona o

sistema de coordenadas do robô e o sistema de coordenadas global, dada por

$$R(\theta) = \begin{bmatrix} cos(\theta) & sen(\theta) & 0 \\ -sen(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

sendo o modelo dependente da velocidade linear e angular

$$vel = \begin{bmatrix} \upsilon \\ 0 \\ \omega \end{bmatrix} \tag{3.2}$$

A simulação dinâmica do Pioneer 3-DX é dada em função das acelerações linear e angular impostas na mudança de *setpoints*. Os limites estipulados e mantidos fixos no presente trabalho foram $\pm 75\ cm/s^2$ e $\pm 100°/s^2$. Sendo que os limites máximos permitidos pelo robôs são $\pm 200 cm/s^2$ e $\pm 300°/s^2$. Portanto, a atualização de velocidade do robô é expressa por

$$\upsilon(t) = \upsilon(t-1) + a_\upsilon t_{amos} \tag{3.3}$$

$$\omega(t) = \omega(t-1) + a_\omega t_{amos} \tag{3.4}$$

sendo υ a velocidade linear atual efetivamente comandada (ou real), ω a velocidade angular atual efetivamente comandada (ou real), ambas em função da aceleração linear e angular, a_υ e a_ω respectivamente. O sinal da aceleração é dado por verificação se o sinal de velocidade anterior é o mesmo que o novo comando gerado pelo controlador para saber se o robô deve continuar acelerando ou passará a desacelerar e vice-versa.

A não linearidade é integrada ao sistema por meio de duas condições:

(*i*) se o sinal da aceleração for negativo e o sinal de velocidade estimado pelas Equações 3.3 e 3.4 for diferente do comando anterior, então $\upsilon(t)$ e $\omega(t)$ serão zero;

$$\upsilon(t) = 0 \tag{3.5}$$

$$\omega(t) = 0 \tag{3.6}$$

(*ii*) se a velocidade atual, calculada com as Equações 3.3 e 3.4, menos a velocidade anterior (em $t-1$) for menor que a aceleração vezes o tempo de amostragem, a velocidade atual receberá o novo comando diretamente do controlador do robô, em resumo, se $\upsilon(t) - \upsilon(t-1) < a_\upsilon t_{amos}$ e $\omega(t) - \omega(t-1) < a_\omega t_{amos}$, então

$$\upsilon(t) = \upsilon \tag{3.7}$$

$$\omega(t) = \omega. \tag{3.8}$$

Posteriormente, são realizadas as operações de conversão entre o sistema de coordenada do robô e o sistema de coordenada global, bem como atualização da posição do robô levando-se em consideração o atraso cinemático e dinâmica não linear. As expressões são apresentadas a seguir.

$$\upsilon_I(t) = R^{-1}\upsilon(t) \tag{3.9}$$

$$Pos = Pos + \upsilon_I(t)t_{amos} \tag{3.10}$$

sendo $\upsilon_I(t)$ a velocidade no sistema de coordenadas do ambiente ou inercial e Pos a posição do robô.

3.4 Estruturas de controle

A seguir são apresentados os controladores neural e *fuzzy* desenvolvidos e que utilizam um algoritmo genético em sua sintonia. O algoritmo genético é responsável por sintonizar os pesos sinápticos da rede neural e gerar os valores dos *singletons* de saída (ou seja, dos consequentes das regras do modelo Takagi-Sugeno) do controlador *fuzzy*.

3.4.1 Controlador Neural

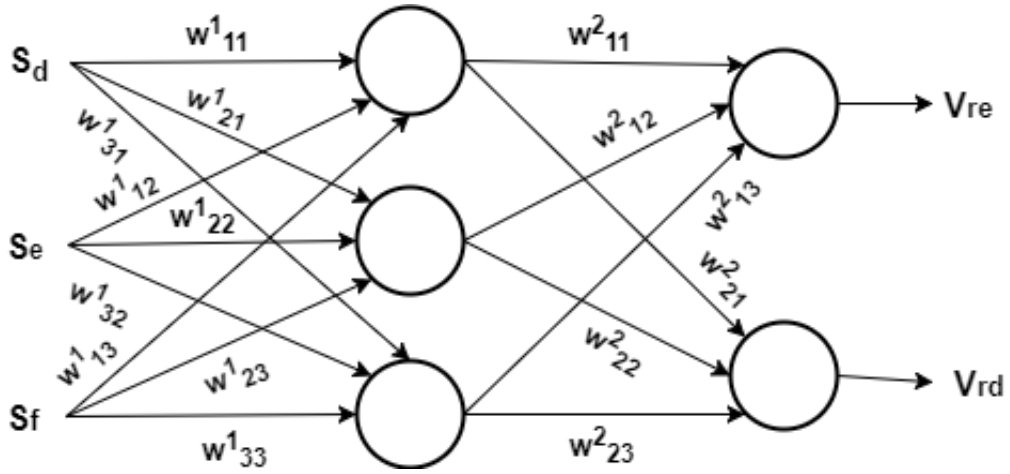
A estrutura de controle neural utilizada é fixa, com configuração apresentada na Figura 13(a). A rede é composta por 3 camadas, sendo a primeira referente às entradas, a segunda é uma camada escondida composta por 3 neurônios e por fim a camada de saída estruturada com dois neurônios, sem valor de *bias* atribuído aos neurônios. Portanto, o controlador é composto por três entradas (distância à esquerda, frontal e à direita) e duas saídas que comandam a velocidade de cada uma das rodas, totalizando 15 pesos sinápticos que correspondem ao tamanho do cromossomo do algoritmo genético.

As saídas dos neurônios variam entre $[0, 1]$, pois as funções de ativação de cada um deles é uma função sigmoide, expressa por

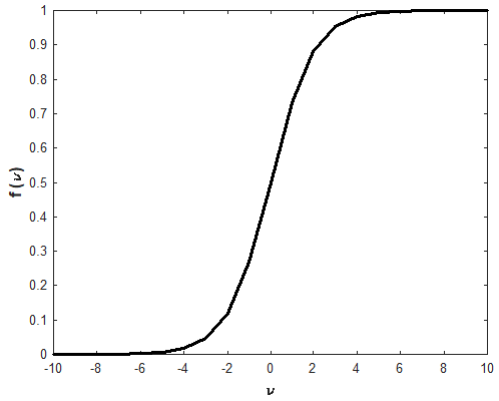
$$f(\upsilon) = \frac{1}{1 + e^{-\upsilon}}, \tag{3.11}$$

como apresentada na Figura 13(b).

A fim de limitar as ações de controle e também devido tanto ao simulador quanto ao robô receberem comandos de velocidade linear e velocidade angular para atuar, foi utilizado o modelo cinemático para escrever as equações de velocidade linear e angular



(a)



(b)

Figura 13 – (a) Controlador neural e (b) Função de ativação dos neurônios (sigmoide).

do robô em função das velocidades das rodas direita e esquerda (baseado em Floreano e Mondada (1996)). Como a saída do controlador variam entre $[0, 1]$, elas foram multiplicadas por uma constante que as pondera a uma velocidade linear máxima permitida a cada roda do robô, como apresentadas a seguir:

$$\upsilon = \frac{\upsilon_{re} + \upsilon_{rd}}{2}\upsilon_{max}, \tag{3.12}$$

$$\omega = \frac{\upsilon_{re} - \upsilon_{rd}}{2l}\upsilon_{max}, \tag{3.13}$$

sendo, υ_{max} a velocidade linear máxima de cada roda, e l a metade da distância entre as duas rodas tracionadas do robô (equivalente ao raio do robô). Percebe-se que tanto a velocidade linear quanto a angular são dependente da velocidade linear máxima estipulada a cada uma das rodas. Esta conversão de velocidades, apresentadas nas Equações 3.12 e 3.13, também é aplicada às saídas do controlador *fuzzy* que será discutido na subseção seguinte.

De acordo com o modelo apresentado com as Equações 2.3 e 2.4, cada camada da rede pode ser representada por uma matriz de pesos, descritas da seguinte maneira:

i) Segunda camada:

$$\mathbf{W_1} = \begin{bmatrix} w_{11}^1 & w_{12}^1 & w_{13}^1 \\ w_{21}^1 & w_{22}^1 & w_{23}^1 \\ w_{31}^1 & w_{32}^1 & w_{33}^1 \end{bmatrix} \tag{3.14}$$

$$\mathbf{x_1} = \begin{bmatrix} s_d \\ s_e \\ s_f \end{bmatrix} \tag{3.15}$$

$$\mathbf{b_1} = \begin{bmatrix} b_1^1 \\ b_2^1 \\ b_3^1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.16}$$

$$\mathbf{c_1} = f\left(\mathbf{W_1 x_1} + \mathbf{b_1}\right) \tag{3.17}$$

ii) Terceira camada:

$$\mathbf{W_2} = \begin{bmatrix} w_{11}^2 & w_{12}^2 & w_{13}^2 \\ w_{21}^2 & w_{22}^2 & w_{23}^2 \end{bmatrix} \tag{3.18}$$

$$\mathbf{x_2} = \mathbf{c_1} = \begin{bmatrix} c_{11} \\ c_{12} \\ c_{13} \end{bmatrix} \tag{3.19}$$

$$\mathbf{b_2} = \begin{bmatrix} b_1^2 \\ b_2^2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.20}$$

$$\mathbf{c_3} = f\left(\mathbf{W_2 x_2} + \mathbf{b_2}\right) = \begin{bmatrix} \upsilon_{re} \\ \upsilon_{rd} \end{bmatrix} \tag{3.21}$$

O algoritmo genético é responsável por otimizar os pesos (conexões sinápticas da rede neural) como mostrados em 3.14 e 3.18, isto é, o cromossomo contém 15 genes correspondentes aos termos das respectivas matrizes $\mathbf{W_1}$ e $\mathbf{W_2}$.

3.4.2 Controlador *Fuzzy*

O controlador *fuzzy* apresenta estrutura similar à rede neural, como apresentado anteriormente: 3 entradas (distância à esquerda, frontal e à direita) e duas saídas referentes às velocidades das rodas direita e esquerda. Cada variável linguística de entrada apresenta dois termos linguísticos associados, pequeno (P) e grande (G) definidos por funções de pertinência triangular e trapezoidal, respectivamente.

A estrutura adotada para o controlador *fuzzy* também é fixa e baseada no modelo de regras Takagi e Sugeno (1985) com o consequente constante, também conhecido como modelo *singleton*.

O controlador constitui-se de oito regras que relacionam todas as combinações possíveis das três entradas (vide Tabela 1). Portanto, cada regra é representada da seguinte maneira:

$$\begin{aligned} R_k \;=\; & Se\ S_d \text{ é } F_j^k,\ S_f \text{ é } F_j^k,\ e\ S_e \text{ é } F_j^k \\ & então\ v_{re}^k \;=\; c_1^k\ e\ v_{rd}^k \;=\; c_2^k, \end{aligned} \tag{3.22}$$

em que $k = 1, ..., 8$, $j = 1, 2$, S_i são as variáveis *fuzzy* (S_d = sensor direita, S_f = sensor frontal e S_e = sensor da esquerda), F_j^k são as funções de pertinência (F_1^k = pequeno e F_2^k = grande) referentes às variáveis *fuzzy*, v_{re} e v_{rd} são as variáveis de saída, e c_1^k e c_2^k são os consequentes das k regras que são otimizados pelo algoritmo genético e compõem seu cromossomo.

De acordo com a estrutura adotada, o cromossomo possui o dobro do número de regras, totalizando 16 genes, sendo metade deles correspondes aos *singletons* de saída para o comando de velocidade da roda direita e os demais para o comando de velocidade da roda esquerda. Note que o tamanho dos cromossomos dos controladores neural e *fuzzy* são similares, portanto, geram espaços de busca de dimensões semelhantes para o processo evolutivo.

A depender do experimento, ou seja, do ambiente em que o robô será testado, a abertura das funções de pertinência necessitam de um ajuste para tornar o processo de fuzzificação mais efetivo. As funções de pertinência foram definidas por experimentação, que é uma abordagem típica em controle *fuzzy* baseado em conhecimento (BABUSKA, 1998) e serão apresentadas juntamente com a descrição dos experimentos.

O universo de discurso das entradas adotado foi de $3m$. Como metodologia de inferência para cada regra foi aplicada a norma T, ou produto, que modela o operador mínimo, o qual pode ser analisado pela Tabela 1 que apresenta todas as combinações de regras.

Tabela 1 – Conjunto de regras do controlador *fuzzy*.

Regra	**Semântica**
1	Se S_d é P, S_e é P e S_f é P, então $v_{re} = c_1^1$ e $v_{rd} = c_2^1$
2	Se S_d é P, S_e é P e S_f é G, então $v_{re} = c_1^2$ e $v_{rd} = c_2^2$
3	Se S_d é P, S_e é G e S_f é P, então $v_{re} = c_1^3$ e $v_{rd} = c_2^3$
4	Se S_d é P, S_e é G e S_f é G, então $v_{re} = c_1^4$ e $v_{rd} = c_2^4$
5	Se S_d é G, S_e é P e S_f é P, então $v_{re} = c_1^5$ e $v_{rd} = c_2^5$
6	Se S_d é G, S_e é P e S_f é G, então $v_{re} = c_1^6$ e $v_{rd} = c_2^6$
7	Se S_d é G, S_e é G e S_f é P, então $v_{re} = c_1^7$ e $v_{rd} = c_2^7$
8	Se S_d é G, S_e é G e S_f é G, então $v_{re} = c_1^8$ e $v_{rd} = c_2^8$

O método de *defuzzificação* adotado foi o centro da média ou centroide, que para o modelo *singleton* se resume a uma média ponderada:

$$v_{re} = \frac{\sum_{i=1}^{k} \beta_i c_1^i}{\sum_{i=1}^{l} \beta_i}, \tag{3.23}$$

$$v_{rd} = \frac{\sum_{i=1}^{k} \beta_i c_2^i}{\sum_{i=1}^{l} \beta_i}, \tag{3.24}$$

em que v_{re} e v_{rd} representam as saídas, k é o número de regras ($k = 1, ..., 8$), β_i a pertinência de cada regra resultante da inferência (mínimo entre os graus de pertinência de cada variável ou interseção) e c_i são os valores *singletons* que compõem o cromossomo do algoritmo genético. Note que cada regra contribui para a deffuzificação com seu grau de pertinência resultante.

O universo de discurso das saídas de ambos os controladores, v_{re} e $v_{rd} \in [0, 1]$, é convertido em velocidade linear e angular por meio das Equações 3.12 e 3.13.

3.5 Algoritmo genético

Neste trabalho, o robô inicia sem saber qual tarefa deve executar e que as estruturas dos controladores desenvolvidos são fixas. Deste modo, o algoritmo genético é responsável por evoluir parâmetros específicos do sistema de controle a partir de informações providas pelo sensoriamento para que o robô desempenhe a tarefa da melhor maneira possível, sendo seu desempenho avaliado por meio de uma função de *fitness*. Os dados fornecidos ao algoritmo genético são a estrutura de cada controlador, tamanho do cromossomo, o simulador a ser utilizado, tempo de simulação de cada indivíduo, tempo de amostragem, tamanho da população, número de gerações, taxa dos operadores genéticos envolvidos e o ambiente de simulação (ou mapa), mas não é informada a relação entre essas informações. Esta tarefa é de responsabilidade do processo evolutivo, que deve garantir, através da sintonia do controlador, que o robô navegue sem se chocar com obstáculos.

O algoritmo genético inicializa sua população de indivíduos com uma variável aleatória uniforme que varia entre $[-0,5, 0,5]$ para o controlador neural, baseado em Floreano e Mondada (1996) e $[0, 1]$ para o controlador *fuzzy*. Essas variáveis aleatórias são responsáveis por inicializar os cromossomos que caracterizam os parâmetros de cada um dos controladores. Em seguida, toda a população é avaliada em ambiente simulado e seu desempenho é estimado por meio da função de *fitness* adotada. A definição da função de *fitness* é de fundamental importância no processo evolutivo, pois a mesma é responsável por avaliar a solução do problema e deve caracterizar da melhor maneira possível a solução que se deseja. Como a *fitness* depende do experimento, será apresentada durante a descrição do mesmo, posteriormente.

Os operadores genéticos utilizados para gerar a população da geração seguinte são: *crossover* com *roulette wheel*, elitismo e mutação. Para o *crossover*, os indivíduos são selecionados para reprodução usando o método *roulette wheel* de acordo com a sua *fintess* na geração. A seleção por *roulette wheel* relaciona o valor de *fitness* do indivíduo com a sua probabilidade de ser selecionado: quanto maior a *fitness*, maior a probabilidade. Após a seleção, os indivíduos são pareados aleatoriamente e o operador *crossover* de um ponto é aplicado, ou seja, os indivíduos selecionados compartilham entre si seu código genético. De modo a manter os melhores indivíduos na população seguinte, o elitismo é aplicado. O Elitismo é responsável por garantir a passagem direta para a geração seguinte dos N melhores indivíduos, usualmente expresso em termos de porcentagem da população total (no presente trabalho, 10%).

Finalmente, cada um dos indivíduos obtidos do *crossover* é exposto à mutação. Vale ressaltar que os indivíduos que compõem a elite da nova população não são expostos à mutação. A mutação é aplicada a cada gene com uma baixa probabilidade de ocorrência, que decide se o gene irá ou não ser mutado. Se sim, o gene é modificado pela adição de um número selecionado de um predeterminado *range*, no presente trabalho $[-0,5, 0,5]$ (baseado em Montana e Davis (1989)). No controlador *fuzzy*, a mutação é restrita a adicionar, no máximo, cerca de 40% da variável de inicialização sobre o valor do gene, sendo o valor do gene sempre mantido entre $[0, 1]$. Isto é devido ao cromossomo *fuzzy* representar os pesos dos consequentes que devem ser positivos e mantido dentro deste *range*. Esta taxa foi escolhida empiricamente para que a adição não causasse uma saturação prematura dos valores *singletons*. O critério de parada do presente algoritmo genético é apenas o número de gerações. O Algoritmo 1 apresenta uma descrição de todo procedimento descrito anteriormente.

A fim de comparar o desempenho dos controladores perante a transferência do ambiente simulado para o ambiente real, foram adotadas algumas métricas relacionadas diretamente com a tarefa que o robô deve executar, neste caso, a navegação autônoma.

Algorithm 1 - Algoritmo Genético

ENTRADA: estrutura do controlador que determina o tamanho do cromossomo, simulador a ser utilizado, mapa do ambiente e *fitness* associada a tarefa.
INICAILIZAÇÃO:
t_s, tempo de simulação de cada indivíduo
t_a, tempo de amostragem
t_p, tamanho da população
n_g, número de gerações
t_{crom}, tamanho do cromossomo (número de genes)
t_c, taxa de *crossover*
t_m, taxa de mutação
t_e, taxa de elitismo
$p_a \leftarrow$ Inicialização da população com soluções candidatas aleatórias (com cada gene $\in [-0,5,\ 0,5]$)
for i $\leftarrow 1$ **até** n_g **faça**
    for j $\leftarrow 1$ **até** t_p **faça**
        Geração da posição inicial aleatória no mapa do indivíduo j - $P_i(j)$
        fitness(j) $\leftarrow$ simulação($P_i(j)$, mapa, $p_a(j)$)
    end for
    Nova população $np(x)$, sendo $x = 1, ..., t_p$
    for k $\leftarrow 1$ **até** t_p **faça**
        $np(k) \leftarrow$ *roulette-whell*(*fitness*(:))
    end for
    pareamento aleatório de np
    Operação de *crossover*
    for l $\leftarrow 1$ **até** $t_p - (t_e t_p - 1)$ **faça**
        $[f_1, f_2] \leftarrow$ *crossover*$(np(l), np(l+1))$ % o *crossover* retorna dois indivíduos filhos (f_1 e f_2) após o *crossover* de um ponto
        $np(l) \leftarrow f_1$
        $np(l+1) \leftarrow f_2$
    end for
    Operação de mutação
    for m $\leftarrow 1$ **até** $t_p - (t_e t_p)$ **faça**
        $np(m) \leftarrow$ mutação$(np(m), t_m)$ % a mutação retorna o indivíduo com cada gene varrido com probabilidade de mutação t_m
    end for
    Operação de elitismo
    $ind \leftarrow$ elitismo($t_p,\ t_e$, *fitness*(:)) % o elitismo retorna os N indivíduos da elite
    for n $\leftarrow 1$ **até** $t_e t_p$ **faça**
        $np(t_p - (t_e t_p) + n) \leftarrow$ ind(n)
    end for
    $p_a \leftarrow np$
end for
SAÍDA: população final de soluções p_a

3.6 Métricas de avaliação

As métricas de avaliação adotadas foram obtidas de outros trabalhos e objetivam caracterizar da melhor maneira possível a tarefa de navegação.

De Freire et al. (2010) foi retirada a métrica de velocidade linear média (v_m) que caracteriza quão veloz o robô navegou e também permite avaliar se o mesmo ficou parado ou não. Foi incluída também a velocidade angular média (ω_m) que caracteriza a capacidade de manobra do indivíduo. Deste trabalho, também foi obtida a métrica de suavidade da trajetória (S), cujo valor é estimado pela média da diferença em módulo da orientação atual e anterior do robô durante o experimento. De Wong et al. (2002) foi retirada a métrica de distância total percorrida (d_s) e, atrelada a esta, foi adotada também a métrica de maior distância entre dois pontos da trajetória ($d_{\Delta s}$), possibilitando verificar se o robô explorou o ambiente ou ficou rodando em uma pequena região.

Por sua vez, Muñoz, Valencia e Londono (2007) apresentaram um conjunto de métricas de segurança para navegação. Elas levam em conta a distância entre o robô e o obstáculo durante o trajeto. A primeira delas, denominada métrica de segurança 1 (MS1), estipula a média da distância entre o veículo e os obstáculos durante toda tarefa, medida por todos os sensores. A segunda, métrica de segurança 2 (MS2), calcula a média da distância mínima, dada pela menor medida dentre os sensores para os obstáculos, durante o experimento. Também há a distância mínima (d_{min}) que nada mais é que a mínima distância entre qualquer sensor e um obstáculo durante toda a trajetória.

Por fim, foi usada a métrica de sucesso, assim denominada, responsável por informar se o robô cumpriu a tarefa sem bater ou se a mesma foi interrompida devido a colisão com algum obstáculo.

A seguir, serão apresentadas as descrições matemáticas de cada uma das métricas adotadas no presente trabalho para avaliação dos indivíduos após a execução do experimento, seja simulado ou real.

Para o cálculo das métricas são necessários alguns dados fornecidos pela odometria e sensoriamento do robô e que são armazenados a cada iteração (n) realizada durante o experimento real ou simulado. Esses dados são os vetores de velocidade linear (v), velocidade angular (ω), orientação do robô (θ), posição x, posição y, em que v_n, ω_n, θ_n, x_n, y_n representam o n-ésimo elemento, respectivamente, e também são necessárias as leituras de todos os sensores, s_{ni}, que se refere ao i-ésimo sensor no n-ésimo instante de tempo. Portanto, $i = 1,\ ...,\ n_s$ e $n = 1,\ ...,\ n_a$, sendo n_s o número de sensores e n_a o número total de (amostras ou) ações realizadas pelo robô. Para o cálculo da métrica de sucesso utiliza-se um *flag* denominado 'colidiu' que a partir do seu nível lógico é possível identificar se o agente bateu (colidiu = '1') ou não (colidiu = '0'). Assim, tem-se:

Velocidade linear média:

$$v_m = \frac{\sum_{n=1}^{n_a} v_n}{n_a} \tag{3.25}$$

Velocidade angular média:

$$\omega_m = \frac{\sum_{n=1}^{n_a} |\omega_n|}{n_a} \tag{3.26}$$

Sucesso:

$$sucesso = \begin{cases} 0, & colidiu = 1 \\ 1, & colidiu = 0 \end{cases} \tag{3.27}$$

Distância total percorrida:

$$d_s = \sum_{n=1}^{n_a-1} \sqrt{(x_{n+1} - x_n)^2 + (y_{n+1} - y_n)^2} \tag{3.28}$$

Maior distância entre dois pontos:

$$f(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \quad i = j = 1,\ ..., n_a \tag{3.29}$$

$$d_{\Delta s} = \max_{i,j}(f(i,j)) \tag{3.30}$$

Média das leituras dos sensores:

$$MS1 = \frac{\sum_{n=1}^{n_a} \sum_{i=1}^{n_s} s_{ni}}{n_s n_a} \tag{3.31}$$

Média das medidas mínimas dos sensores

$$m_n = \min_i(s_{ni}), \quad n = 1,\ ..., n_a \ e \ i = 1,\ ..., n_s \tag{3.32}$$

$$MS2 = \frac{\sum_{n=1}^{n_a} m_n}{n_a} \tag{3.33}$$

Distância mínima:

$$d_{min} = \min_n(m_n) \tag{3.34}$$

Suavidade:

$$\Delta\theta_n = \theta_{n+1} - \theta_n, \quad n = 1,\ ...,\ n_a - 1 \tag{3.35}$$

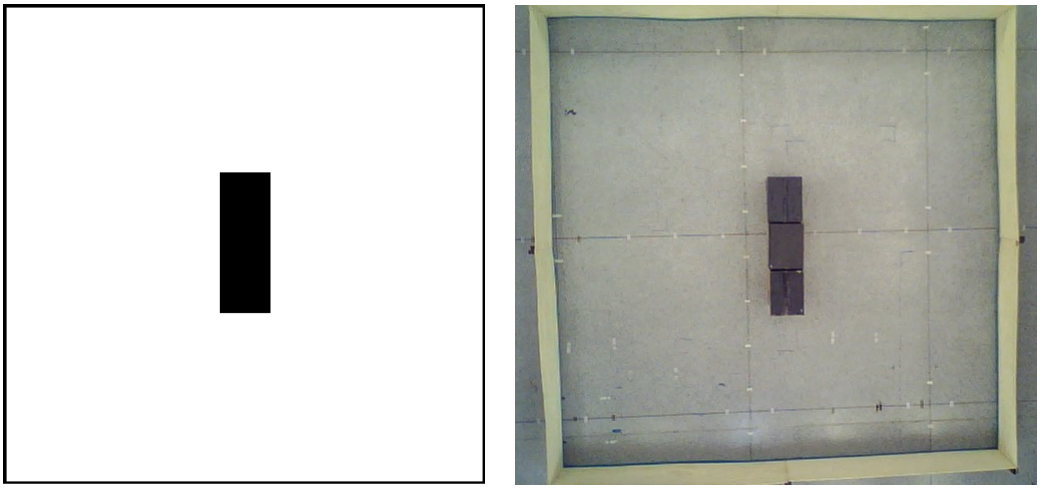
$$S = \frac{\sum_{n=1}^{n_a-1} |\Delta\theta_n|}{n_a - 1} \tag{3.36}$$

4 Resultados

Neste capítulo são apresentados e discutidos os resultados do trabalho. Para isso, foram definidos dois ambientes de simulação de fácil reprodução em ambiente físico e que permitem explorar características dinâmicas diferentes para os controladores desenvolvidos. São apresentados os resultados da simulação e experimentos reais, bem como, a avaliação de cada controlador com a transferência da simulação para o ambiente físico sob quatro níveis de velocidade. A discussão será realiz ada com base nas métricas apresentadas na seção 3.6, em dados estatísticos e em uma análise comparativa considerando as variações percentuais de desempenho com a transferência dos controladores para o ambiente físico, bem como a comparação quanto ao desempenho dos controladores no que diz respeito às características melhor evoluídas.

4.1 Experimento - Ambiente 1

Como mencionado anteriormente, a tarefa adotada foi a navegação autônoma. O primeiro ambiente de teste corresponde a uma área quadrada de dimensão $4\ m \times 4\ m$ com um obstáculo retangular de $1,2\ m \times 0,3\ m$ localizado no centro. Este ambiente se caracteriza como uma simplificação do ambiente utilizado por Floreano e Mondada (1996). A Figura 14(a) apresenta o ambiente utilizado em simulação, e na Figura 14(b) tem-se a sua constituição em ambiente real, no qual o obstáculo ao centro foi confeccionado com caixas de papelão e a parede externa constituída por folhas de compensado. Toda a estrutura física possui uma altura de $0,5\ m$, aproximadamente.



(a) (b)

Figura 14 – (a) Ambiente 1 simulado, e (b) Ambiente 1 real.

A função de *fitness* adotada neste experimento foi baseada no trabalho de Floreano e Mondada (1996), com algumas modificações requeridas pelo robô aqui utilizado. Estruturas similares também podem ser encontradas em Matellán, Fernández e Molina (1998) e Nelson, Barlow e Doitsidis (2009). Essa função é maximizada quando o robô evita obstáculos, se move com alta velocidade e anda em linha reta. Consequentemente, a função de *fitness* aplicada, $F(\upsilon_n, \omega_n, d_n)$, é função da velocidade linear (υ_n), angular (ω_n) e distância aos obstáculos (d_n), dados estes fornecidos pela odometria e sensoriamento do robô a cada instante de tempo n durante o experimento, seja simulado ou real. A função de *fitness* é dada por:

$$F(\upsilon_n, \omega_n, d_n) = \frac{\sum_{n=1}^{n_a} \left[K_\upsilon \left(\frac{\upsilon_n}{\upsilon_{max}} \right) - K_\omega \left(\frac{|\omega_n|}{\omega_{max}} \right) - K_d \; d_n \right]}{t_{lim}/t_s} \tag{4.1}$$

$$d_n(x, \mu, \sigma) = \begin{cases} 1, \; x \; \leq \; 10 \\ e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}, \; 10 \; < \; x \; \leq \; 300 \end{cases} \tag{4.2}$$

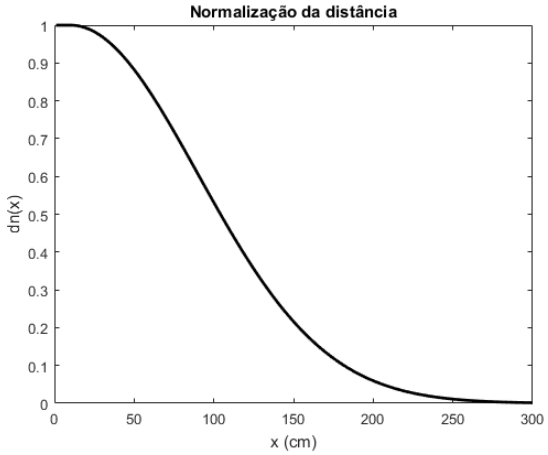


Figura 15 – Normalização da menor leitura dos sensores aplicada ao Ambiente 1.

A primeira parcela, relacionada a υ_n, é responsável por forçar o robô a se mover, penalizando-o se ficar parado a fim de evitar colisão. O segundo termo (relacionado a ω_n) penaliza velocidade angular, o que restringe o robô de ficar girando em torno do próprio eixo ou em uma região pequena do ambiente. O terceiro e último termo, relacionado a d_n, informa o maior nível de ativação entre os sensores, ou seja, quão próximo se está de um obstáculo, penalizando mais distâncias menores.

Especificamente, υ_n é a n-ésima amostra da velocidade linear, ω_n se refere a n-ésima amostra da velocidade angular e $d_n \in [0, 1]$ a n-ésima amostra da saída de 4.2 (função de Gauss com média diferente de zero) que normaliza a menor distância dentre os sensores para um obstáculo. Essa função recebe como parâmetros a menor leitura, $x \in [0, 300]$, fornecida pelos três conjuntos que formam as entradas dos controladores,

a defasagem μ e o fator de amortecimento σ, resultando na normalização expressa pela Figura 15. A proposta de aplicação desta função é punir mais as distâncias menores. Os parâmetros adotados para os experimentos no Ambiente 1 são apresentados na Tabela 2. υ_{max} e ω_{max} são os limites máximos de velocidade linear e angular, respectivamente, utilizados para normalizar estas componentes. A cada ação de controle tomada pelo robô é gerada uma parcela da *fitness* que é calculada e acumulada. Se o robô bate antes que cumpra o tempo de simulação desejado (t_{lim} - tempo de simulação de cada indivíduo da população), o desempenho do robô é afetado, por ter realizado um número de ações de controle ou número de amostras (n_a) inferior ao que deveria, t_{lim}/t_s, sendo t_s o tempo de amostragem. Esta quantidade acumulada, ao fim do experimento, é dividida pelo número total de ações que deveriam ser realizadas pelo robô.

Os valores adotados nos processos evolutivos foram, υ_{max} = 30cm/s, 25cm/s, 20cm/s e 15cm/s, ω_{max} = 0,79rad/s, 0,66rad/s, 0,53rad/s e 0,39rad/s, que correspondem respectivamente, a 45,26°/s, 37,82°/s, 30,37°/s e 22,35°/s, $t_{lim} = 300s$ e $t_s = 0,6s$. As constantes K_υ, K_ω e K_d foram obtidas através de uma busca exaustiva, ou seja, valores entre 0 e 1 foram variados com passo $0,1$ para cada uma das constantes e, posteriormente, testadas as combinações na evolução de controladores. Essa busca exaustiva foi executada sobre ambos os controladores com nível de velocidade linear das rodas de $30cm/s$, tanto no Ambiente 1 quanto no Ambiente 2 e, a partir dos resultados, foram selecionadas as contantes a serem aplicadas ao processo evolutivo de ambos os controladores. Os valores adotados para os experimentos levaram em conta a frequência com que cada constante gerava bons resultados no processo evolutivo para ambos os controladores. A Tabela 2 apresenta os valores adotados para as constantes da *fitness* no primeiro experimento, bem como a parametrização dos operadores do algoritmo genético. Todas as combinações testadas com a busca exaustiva foram catalogadas para análises futuras.

Os níveis de velocidade linear adotados nos experimentos visam permitir a manifestação de efeitos dinâmicos distintos no comportamento do robô, como derrapagem das rodas, aceleração e desaceleração, seja brusca ou suave, de maneira que evidencie o *reality gap* perante as simplificações do modelo de simulação adotado em diferentes graus. Nas subseções seguintes serão analisados tais experimentos.

Neste primeiro ambiente, caracterizado por um amplo espaço entre o obstáculo ao centro e as paredes externas, foram utilizados os conjuntos *fuzzy* apresentados na Figura 16 para caracterizar as variáveis *fuzzy* distância à direita, esquerda e frontal do controlador *fuzzy*. Portanto, todas as variáveis são caracterizadas por dois termos linguísticos, um referente a distâncias pequenas, P, e caracterizado por uma função de pertinência triangular com uma abertura de $0 - 150$cm, e o termo G, referente a distâncias grandes, configurado por uma função de pertinência trapezoidal excursionando entre $50 - 300$cm.

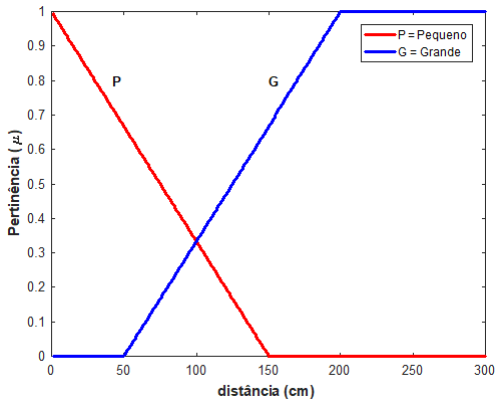
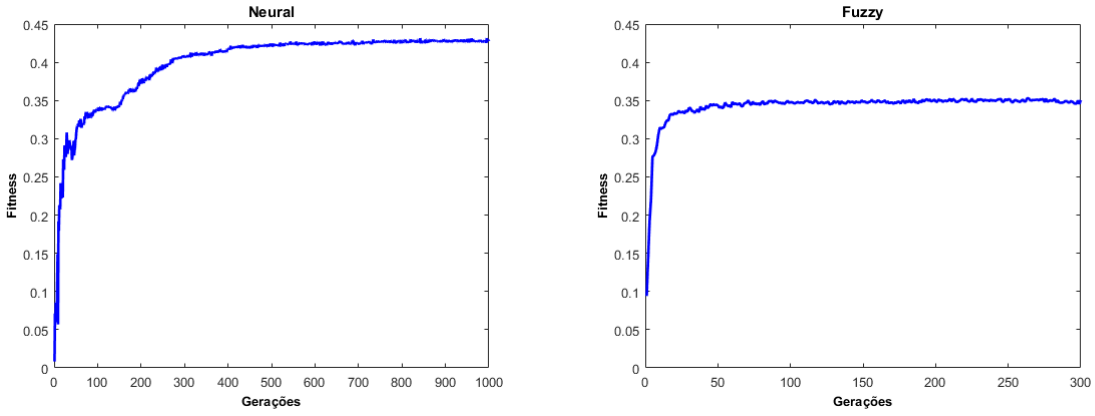


Figura 16 – Conjuntos *fuzzy* referentes às variáveis distância à direita, esquerda e frontal aplicados ao Ambiente 1.

4.1.1 Ambiente 1 com velocidade linear máxima das rodas de 30cm/s

A Figura 17 apresenta a evolução da *fitness* para os controladores neural (Figura 17(a)) e *fuzzy* (Figura 17(b)) e os parâmetros utilizados no processo evolutivo são apresentados na Tabela 2. Esses mesmos parâmetros foram mantidos para os demais níveis de velocidades que serão analisados posteriormente. No controlador *fuzzy* foi realizado um processo evolutivo com 300 gerações e para o controlador neural 1000 gerações. É possível destacar que o controlador *fuzzy* estabiliza com 50 gerações enquanto o neural necessita de, aproximadamente, 500 gerações. Além disso, o controlador neural atinge um nível de *fitness* superior ao *fuzzy* ao fim do processo evolutivo.



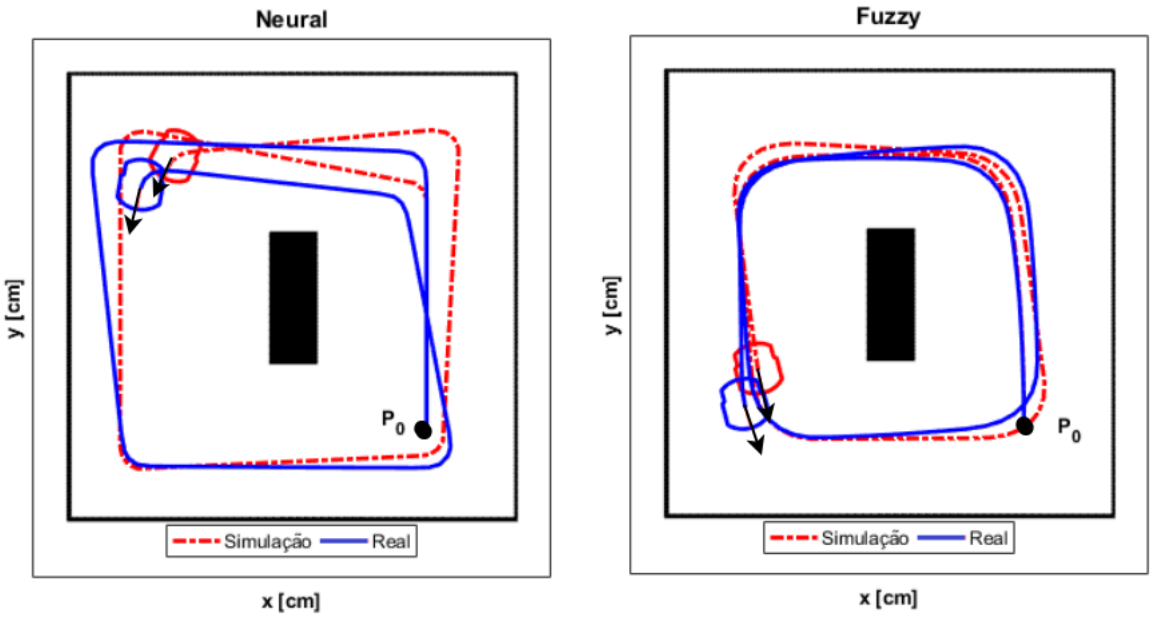
(a) (b)

Figura 17 – (a) Processo evolutivo neural, e (b) Processo evolutivo *fuzzy* com velocidade linear máxima das rodas de $30cm/s$.

O melhor indivíduo da última geração do processo evolutivo (neural e *fuzzy*) foi transferido para o ambiente real e 10 experimentos foram realizados sob condições iniciais similares, isto é, pose inicial P_0(3,2m; 0,8m; 90°) com sequência de experimentos iniciada com 100% da bateria do robô carregada. Durante experimentos práticos e simulados

Tabela 2 – Parâmetros do algoritmo genético aplicado ao ambiente 1 com velocidade linear máxima das rodas de 30cm/s.

Parâmetros	**Valor**
Tamanho da população	100
Número de gerações	1000 (Neural) / 300 (*Fuzzy*)
Comprimento do cromossomo	15 (Neural) / 16 (*Fuzzy*)
Taxa de *crossover*	0,2
Taxa de mutação	0,3
Taxa de elitismo	0,1
Range de mutação	±0,5 (Neural) / ±0,4 (Fuzzy)
d_n	$\mu = 10,\ \sigma = 80$
K_v	0.9
K_ω	0,5
K_d	0,4
Posição inicial	aleatória



(a) (b)

Figura 18 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 2 com velocidade linear máxima das rodas de 30cm/s, .

realizados, foi perceptível que a pose inicial do robô influi em seu comportamento, podendo ou não cumprir a tarefa. Para evitar o risco de não cumprimento da tarefa, foi adotada como pose inicial um ponto no sentido da trajetória evoluída pelo robô.

A Figura 18(a) apresenta o resultado do controlador neural em simulação e em ambiente real do primeiro experimento dos 10 realizados. Os pesos obtidos para o controlador neural são apresentados em (4.3) e (4.4) e sua trajetória se apresenta em sentido anti-horário. A Figura 18(b) apresenta o resultado do controlador *fuzzy* em simulação e em ambiente real (também para o primeiro experimento físico) e sua trajetória evoluída também se configura em sentido anti-horário. Os pesos dos consequentes das regras são apresentados na Tabela 3. Visivelmente algumas regras não foram bem exploradas, con-

tudo, a defuzzificação depende de todas as regras com seu respectivo grau de pertinência resultante.

$$\mathbf{W_1} = \begin{bmatrix} -3,2322 & 1,6110 & -0,9330 \\ 15,5423 & -14,3584 & -0,2075 \\ -5,9638 & 10,3544 & -3,4371 \end{bmatrix} \tag{4.3}$$

$$\mathbf{W_2} = \begin{bmatrix} 3,4502 & 2,8570 & 10,6560 \\ 3,7624 & -29,0520 & 8,8811 \end{bmatrix} \tag{4.4}$$

Tabela 3 – Conjunto de regras do controlador *fuzzy* para o ambiente 1 com velocidade linear máxima das rodas de $30cm/s$.

Regra	**Estrutura das regras com consequentes otimizados**
1	Se S_d é P, S_e é P e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,0030$
2	Se S_d é P, S_e é P e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 1,0000$
3	Se S_d é P, S_e é G e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,0560$
4	Se S_d é P, S_e é G e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 0,7003$
5	Se S_d é G, S_e é P e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,0350$
6	Se S_d é G, S_e é P e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 1,0000$
7	Se S_d é G, S_e é G e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,0247$
8	Se S_d é G, S_e é G e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 1,0000$

As Tabelas 4 e 5 apresentam a comparação entre simulação e os experimentos reais. Nesta comparação foi considerada a média das métricas obtidas com os sucessos dos 10 experimentos em ambiente físico, desvio padrão das amostras e diferença percentual entre a média e o único experimento simulado, dado que o simulador é determinístico, ou seja, nenhuma aleatoriedade é atribuída ao modelo. Pode ser visto com a transferência que o desempenho dos controladores melhoraram em virtude de executarem trajetória mais distante dos obstáculos. Neste experimento, o controlador *fuzzy* apresenta menores variações com a transferência para todas as métricas. Por outro lado, é notório no controlador neural uma dispersão maior entre a trajetória simulada e real se comparada com o controlador *fuzzy*. Em relação às métricas resultantes também pode ser notado que os experimentos são bem similares um ao outro.

Com a finalidade de comparar os experimentos, foi realizado o teste estatístico T-student para verificar se os experimentos são diferentes ou iguais.

O teste T-student tem diversas variações de aplicação, mas sempre há a limitação do mesmo ser usado na comparação de duas, e somente duas, médias e as variações dizem respeito às hipóteses que são testadas. Os testes de hipótese constituem uma forma de inferência estatística que permitem rejeitar ou não uma hipótese. O teste T deriva da distribuição normal. Ele fornece um valor-p que é uma quantificação da probabilidade de se errar ao rejeitar uma hipótese. Se o valor-p é menor que o nível de significância dado ao teste, conclui-se que o correto é rejeitar a hipótese de nulidade. A rejeição da hipótese de

Tabela 4 – Neural - melhor indivíduo ambiente 1 com velocidade linear máxima das rodas de 30cm/s.

Métricas - Controlador Neural	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,4097	0,4230 ± 0,0076	-3,2699
Velocidade linear média (cm/s)	26,5365	26,2636 ± 0,1635	1,0284
Velocidade angular média (rad/s)	0,1490	0,1648 ± 0,0087	-10,5918
Sucesso	1	10	-
Distância percorrida (cm)	1592,1907	1568,1348 ± 11,6438	1,5109
Maior distância entre dois pontos (cm)	409,6957	378,4858 ± 19,3685	7,6178
MS1 (cm)	145,5408	151,3240 ± 2,3821	-3,9736
MS2 (cm)	41,7911	58,4384 ± 4,0100	-39,8346
Min (cm)	25,4171	16,0700 ± 10,5049	36,7748
Suavidade (°)	11,6365	12,3267 ± 0,3659	-5,9310

Tabela 5 – *Fuzzy* - melhor indivíduo ambiente 1 com velocidade linear máxima das rodas de 30cm/s.

Métricas - Controlador *Fuzzy*	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,3616	0,3649 ± 0,0020	-0,9020
Velocidade linear média (cm/s)	26,3356	26,2724 ± 0,0741	0,2400
Velocidade angular média (rad/s)	0,1613	0,1591 ± 0,0034	1,3776
Sucesso	1	10	-
Distância percorrida (cm)	1580,1345	1572,0703 ± 3,9733	0,5104
Maior distância entre dois pontos (cm)	343,5990	334,7366 ± 3,9471	2,5793
MS1 (cm)	152,9207	152,5078 ± 0,4387	0,2700
MS2 (cm)	56,1428	58,8527 ± 1,5666	-4,8268
Min (cm)	41,3214	44,3200 ± 2,6199	-7,2568
Suavidade (°)	12,5937	13,2456 ± 2,2514	-5,1762

nulidade induz à aceitação de uma hipótese alternativa que tem sentido inverso à hipótese de nulidade. Neste trabalho, este método foi aplicado para verificar se há igualdade entre os controladores *fuzzy* e neural perante a transferência para o ambiente real. Para aplicação do teste é necessário definir uma taxa tolerável de erro, a qual é chamada de nível de significância. O valor de significância adotado foi de 5%, isso quer dizer que o teste verifica se os controladores são iguais com erro tolerável de até 5%.

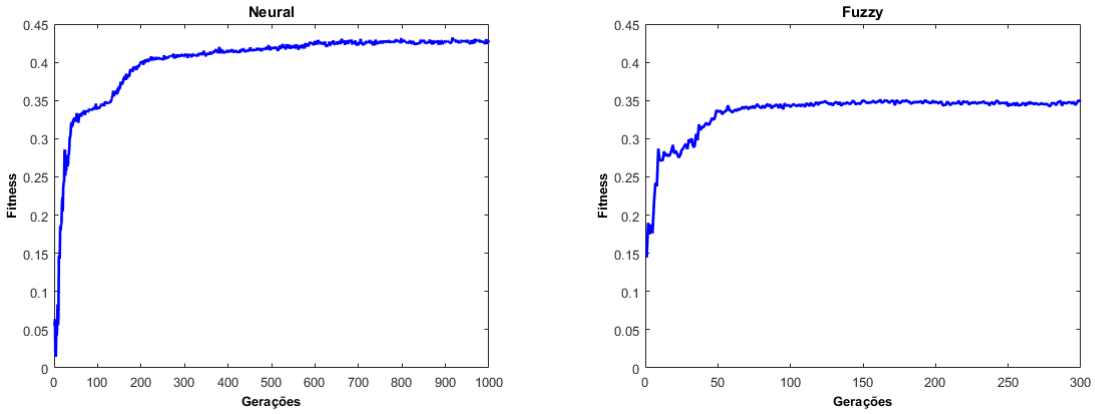
Para aplicação do teste é necessário saber inicialmente se as amostras de dados experimentais (práticos) apresentam uma normalidade. Por consequência, foi aplicado o teste Shapiro-Wilk para checar a normalidade de cada uma das métricas dos experimentos com os controladores em ambiente real. A Tabela 6 apresenta os resultados da análise estatística para este experimento. O resultado do teste Shapiro-Wilk para checar a normalidade de cada uma das métricas aponta que todas as métricas para ambos controladores caracterizam-se por uma distribuição normal (P>0,05). Com o teste T observa-se, na mesma tabela, que 5 das 9 métricas indicam que os experimentos são diferentes um do outro (P<0,05) e 4 delas que os experimentos são iguais (P>0,05). Conclui-se, portanto, que prevalece a diferença entre os controladores.

Tabela 6 – Análise estatística do experimento 1 com velocidade linear máxima das rodas de 30cm/s

	Fuzzy	**Neural**	***Fuzzy*/Neural**
Métricas	**Shapiro-Wilk (P)**		**Test T Student (P)**
Fitness	0,9112	0,9573	$2{,}4565\ 10^{-10}$
Velocidade linear média	0,9272	0,7982	0,4396
Velocidade angular média	0,9406	0,8187	0,0373
Sucesso	-	-	-
Distância percorrida	0,9280	0,8758	0,1667
Maior distância entre dois pontos	0,9555	0,9221	$1{,}8100\ 10^{-5}$
MS1	0,9277	0,9621	0,0766
MS2	0,9365	0,8656	0,3830
Min	0,9100	0,9517	$4{,}4857\ 10^{-6}$
Suavidade	0,4280	0,6792	0,1173

4.1.2 Ambiente 1 com velocidade linear máxima das rodas de 25cm/s

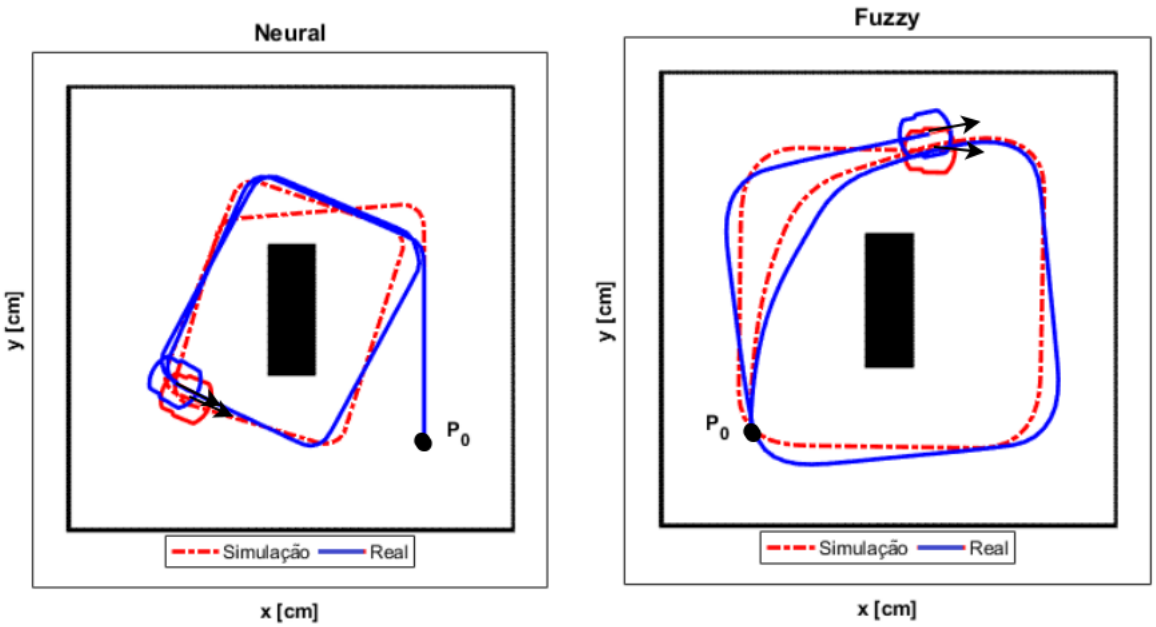
A Figura 19 apresenta a evolução da *fitness* para os controladores neural (Figura 19(a)) e *fuzzy* (Figura 19(b)). Os parâmetros utilizados no processo evolutivo e aplicados a ambos os controladores são os mesmos apresentados na Tabela 2.



(a) (b)

Figura 19 – (a) Processo evolutivo neural, e (b) Processo evolutivo *fuzzy* com velocidade linear máxima das rodas de $25cm/s$.

Pode ser visualizado na Figura 19 que o controlador *fuzzy* estabiliza primeiro, com 100 gerações, enquanto que o controlador neural requer 750 gerações para estabilizar. Além disso, o controlador neural atinge um nível de *fitness* superior ao *fuzzy* ao fim do processo evolutivo. O melhor indivíduo da última geração de cada processo evolutivo (neural e *fuzzy*) foi transferido para o ambiente real e 10 experimentos foram realizados sob as mesmas condições iniciais apresentadas no experimento anterior, exceto pela pose inicial, em que para o controlador neural foi P_0(3,2m; 0,8m; 90°) devido sua trajetória ser em sentido anti-horário e, para o controlador *fuzzy* que evoluiu o comportamento de andar no sentido horário a pose inicial adotada foi P_0(0,8m; 0,8m; 90°). Percebe-se que as posições adotadas são simétricas, proporcionada pela geometria do ambiente.



(a) (b)

Figura 20 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 1 com velocidade linear máxima das rodas de 25cm/s, .

A Figura 20(a) apresenta o resultado do controlador neural em simulação e em ambiente real do primeiro experimento dos 10 realizados. Os pesos obtidos para o controlador neural são apresentados em (4.5) e (4.6). A Figura 20(b) apresenta o resultado do controlador *fuzzy* em simulação e em ambiente real (também para o primeiro experimento físico). Os pesos dos consequentes das regras são apresentados na Tabela 7. As regras 4, 5 e 8 da tabela, aparentemente não foram bem sintonizadas com a evolução. Como o robô não se aproxima muito de obstáculos, o comportamento evoluído resulta em ações com valores *singletons* próximos à unidade. Tal fato também pode ser retratado pela sua trajetória, na qual o robô muda de direção em posições específicas do ambiente e de maneira acentuada.

$$\mathbf{W_1} = \begin{bmatrix} 0,3405 & -9,7569 & 9,0223 \\ 7,2246 & -14,1838 & 4,8576 \\ -3,0912 & 14,6586 & -9,5078 \end{bmatrix} \tag{4.5}$$

$$\mathbf{W_2} = \begin{bmatrix} -0,5515 & 5,3685 & 9,3899 \\ -14,6378 & -16,1459 & 16,6380 \end{bmatrix} \tag{4.6}$$

As Tabelas 8 e 9 apresentam a comparação entre simulação e os experimentos reais. Pode ser visto que com a transferência a rede neural apresentou uma menor variação para as métricas de velocidade linear média, velocidade angular média, distância percorrida, maior distância entre dois pontos, MS2 e suavidade, enquanto o controlador *fuzzy* mostrou

Tabela 7 – Conjunto de regras do controlador *fuzzy* para o ambiente 1 com velocidade linear máxima das rodas de $25cm/s$.

Regra	Estrutura das regras com consequentes otimizados
1	Se S_d é P, S_e é P e S_f é P, então $v_{re} = 0,0342$ e $v_{rd} = 1,0000$
2	Se S_d é P, S_e é P e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 1,0000$
3	Se S_d é P, S_e é G e S_f é P, então $v_{re} = 0,0443$ e $v_{rd} = 1,0000$
4	Se S_d é P, S_e é G e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 1,0000$
5	Se S_d é G, S_e é P e S_f é P, então $v_{re} = 0,1255$ e $v_{rd} = 1,0000$
6	Se S_d é G, S_e é P e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 0,9812$
7	Se S_d é G, S_e é G e S_f é P, então $v_{re} = 0,0099$ e $v_{rd} = 1,0000$
8	Se S_d é G, S_e é G e S_f é G, então $v_{re} = 0,1819$ e $v_{rd} = 1,0000$

menor variação para a *fitness*, MS1 e distância mínima. Além disso, o controlador *fuzzy* apresenta uma melhor suavidade que o neural, já que suas mudanças de direção são mais suaves. Além disso, o 8° experimento neural terminou em uma colisão do robô. Tal choque é resultante de efeitos dinâmicos do sistema, já que em suas mudanças de trajetória bruscas é necessário que uma das rodas pare enquanto a outra gira para mudar de direção.

Tabela 8 – Neural - melhor indivíduo ambiente 1 com velocidade linear máxima das rodas de 25cm/s.

Métricas - Controlador Neural	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,3905	0,3838 ± 0,0045	1,7136
Velocidade linear média (cm/s)	21,0585	21,0074 ± 0,0923	0,2425
Velocidade angular média (rad/s)	0,1769	0,1755 ± 0,0039	0,7847
Sucesso	1	9	-
Distância percorrida (cm)	1263,5084	1254,2038 ± 5,0893	0,7364
Maior distância entre dois pontos (cm)	282,8729	283,4795 ± 4,5416	-0,2145
MS1 (cm)	155,6289	153,6058 ± 0,6382	1,3000
MS2 (cm)	54,4376	54,4478 ± 2,2790	-0,0187
Min (cm)	8,1571	16,8000 ± 6,6002	-105,9555
Suavidade (°)	12,4086	12,3610 ± 0,1237	0,3833

Tabela 9 – *Fuzzy* - melhor indivíduo ambiente 1 com velocidade linear máxima das rodas de 25cm/s.

Métricas - Controlador *Fuzzy*	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,3561	0,3576± 0,0076	-0,4117
Velocidade linear média (cm/s)	21,9840	21,8250 ± 0,1852	0,7233
Velocidade angular média (rad/s)	0,1314	0,1342 ± 0,0092	-2,1486
Sucesso	1	10	-
Distância percorrida (cm)	1319,0380	1306,9569 ± 10,7160	0,9159
Maior distância entre dois pontos (cm)	348,8189	342,5255 ± 3,1383	1,8042
MS1 (cm)	152,4268	152,4436 ± 0,7229	-0,0110
MS2 (cm)	52,0358	56,1211 ± 2,8308	-7,8509
Min (cm)	38,4170	30,3900 ± 2,5584	20,8944
Suavidade (°)	7,9672	8,1255 ± 0,3296	-1,9868

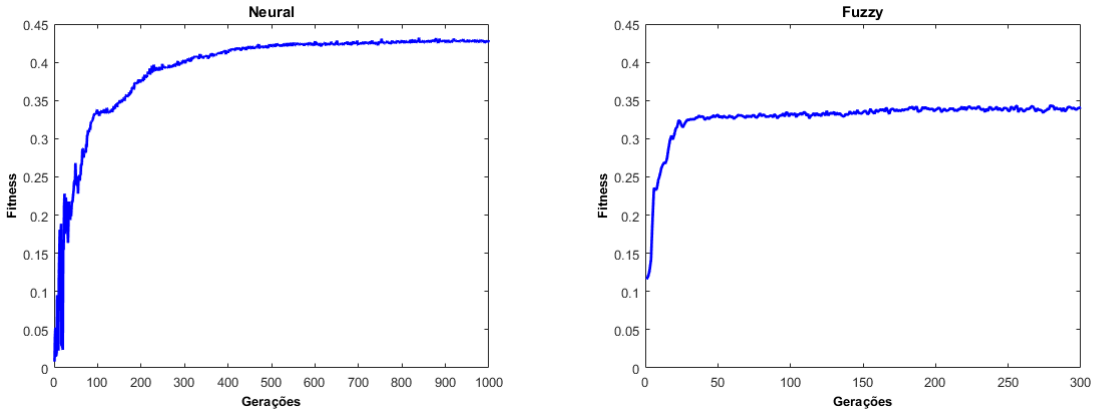
A Tabela 10 apresenta os resultados da análise estatística para este experimento. O resultado do teste Shapiro-Wilk para checar a normalidade de cada uma das métricas aponta que todas as métricas para ambos controladores caracterizam-se por uma distribuição normal (P>0,05). Com o teste T observa-se, na mesma tabela, que todas as métricas indicam que os experimentos são diferentes um do outro (P<0,05).

Tabela 10 – Análise estatística do experimento 1 com velocidade linear máxima das rodas de 25cm/s

	Fuzzy	**Neural**	***Fuzzy*/Neural**
Métricas	**Shapiro-Wilk (P)**		**Test T Student (P)**
Fitness	0,7377	0,9575	$4{,}9359\ 10^{-7}$
Velocidade linear média	0,8553	0,8819	$4{,}0668\ 10^{-8}$
Velocidade angular média	0,8027	0,9327	$6{,}1799\ 10^{-8}$
Sucesso	-	-	-
Distância percorrida	0,8873	0,9462	$2{,}5453\ 10^{-8}$
Maior distância entre dois pontos	0,9546	0,9594	$9{,}0325\ 10^{-15}$
MS1	0,9563	0,9179	0,0019
MS2	0,9321	0,9331	0,0494
Min	0,9096	0,975	$7{,}2150\ 10^{-5}$
Suavidade	0,7807	0,9657	$3{,}9670\ 10^{-12}$

4.1.3 Ambiente 1 com velocidade linear máxima das rodas de 20cm/s

A Figura 21 apresenta a evolução da *fitness* para os controladores neural (Figura 21(a)) e *fuzzy* (Figura 21(b)). Os parâmetros utilizados no processo evolutivo e aplicados a ambos os controladores são os mesmos apresentados na Tabela 2.

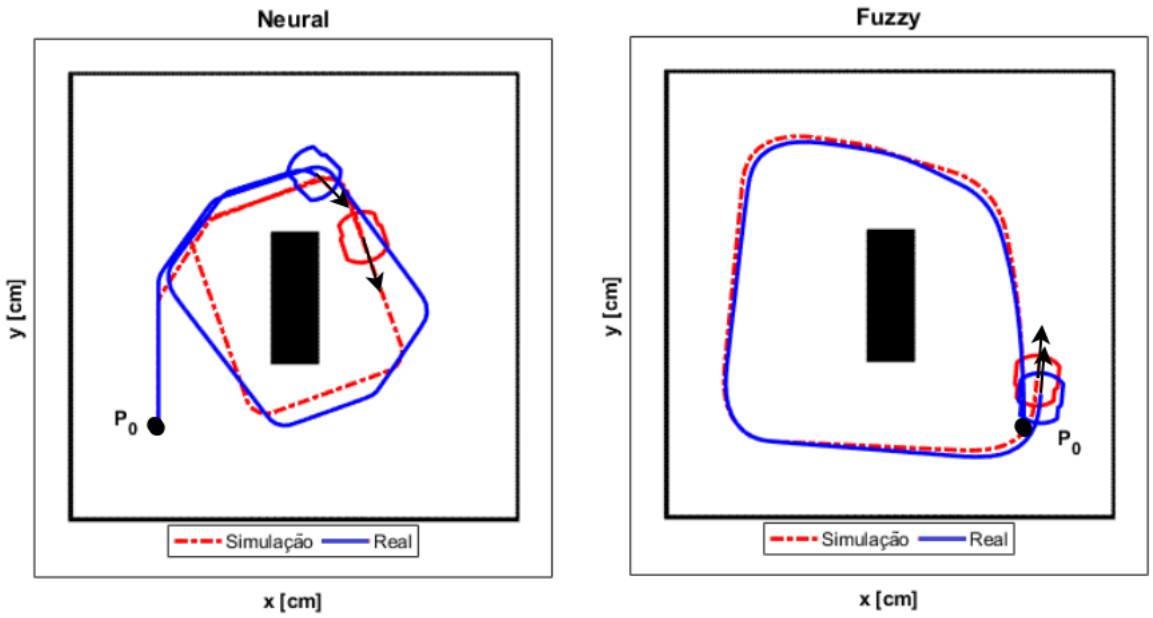


(a) (b)

Figura 21 – (a) Processo evolutivo neural, e (b) Processo evolutivo *fuzzy* com velocidade linear máxima das rodas de $20cm/s$.

Pode ser visualizado na Figura 21(b) que o controlador *fuzzy* estabiliza primeiro, com 175 gerações, enquanto que o controlador neural requer 700 gerações para estabilizar. Além disso, novamente o controlador neural atinge um nível de *fitness* superior ao *fuzzy* ao fim do processo evolutivo. O melhor indivíduo da última geração de cada processo evolutivo (neural e *fuzzy*) foi transferido para o ambiente real e 10 experimentos foram realizados. O controlador neural teve como pose inicial P_0(0,8m; 0,8m; 90°) devido sua trajetória ser em sentido horário. Já o controlador *fuzzy* evoluiu o comportamento de andar no sentido anti-horário, portanto, a pose inicial adotada foi P_0(3,2m; 0,8m; 90°).

A Figura 22(a) apresenta o resultado do controlador neural em simulação e em ambiente real do primeiro experimento dos 10 realizados. Os pesos obtidos para o contro-



(a) (b)

Figura 22 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 1 com velocidade linear máxima das rodas de 20cm/s, .

lador neural são apresentados em (4.7) e (4.8). Já a Figura 22(b) apresenta o resultado do controlador *fuzzy* em simulação e em ambiente real (também para o primeiro experimento físico). Os pesos dos consequentes das regras são apresentados na Tabela 11. Nela pode ser verificado que as regras 3 e 8 da tabela, aparentemente não foram bem sintonizadas com a evolução.

$$\mathbf{W_1} = \begin{bmatrix} -14,2487 & 16,1138 & -3,0320 \\ 14,4224 & -13,6885 & -0,9934 \\ -4,0268 & -4,3913 & 2,0011 \end{bmatrix} \tag{4.7}$$

$$\mathbf{W_2} = \begin{bmatrix} 17,0037 & -21,3935 & -4,8013 \\ 13,5175 & 11,0307 & 5,5582 \end{bmatrix} \tag{4.8}$$

Tabela 11 – Conjunto de regras do controlador *fuzzy* para o ambiente 1 com velocidade linear máxima das rodas de $20cm/s$.

Regra	Estrutura das regras com consequentes otimizados
1	Se S_d é P, S_e é P e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,1290$
2	Se S_d é P, S_e é P e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 1,0000$
3	Se S_d é P, S_e é G e S_f é P, então $v_{re} = 0,9933$ e $v_{rd} = 0,0705$
4	Se S_d é P, S_e é G e S_f é G, então $v_{re} = 0,0002$ e $v_{rd} = 0,5371$
5	Se S_d é G, S_e é P e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,0054$
6	Se S_d é G, S_e é P e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 1,0000$
7	Se S_d é G, S_e é G e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,0135$
8	Se S_d é G, S_e é G e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 0,0795$

As Tabelas 12 e 13 apresentam a comparação entre simulação e os experimentos reais. Pode ser visto que com a transferência a rede neural apresentou menor variação para as métricas de *fitness*, velocidade linear e angular e distância percorrida. O *fuzzy* mostrou menor queda para a maior distância entre dois pontos, MS1, MS2, distância mínima e suavidade. Neste caso, o controlador *fuzzy* apresenta uma melhor suavidade que o neural, além de se manter mais afastado dos obstáculos. Por outro lado, é notório no controlador neural uma dispersão maior entre a trajetória simulada e real que no controlador *fuzzy*.

Tabela 12 – Neural - melhor indivíduo ambiente 1 com velocidade linear máxima das rodas de 20cm/s.

Métricas - Controlador Neural	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,3755	0,3762 ± 0,00335	-0,1891
Velocidade linear média (cm/s)	16,7177	16,7355 ± 0,0711	-0,1062
Velocidade angular média (rad/s)	0,1517	0,1502 ± 0,0016	0,9776
Sucesso	1	10	-
Distância percorrida (cm)	1003,0605	1001,7311 ± 5,7610	0,1325
Maior distância entre dois pontos (cm)	274,2867	277,1196 ± 1,2191	-1,0328
MS1 (cm)	154,9866	155,1283 ± 1,1885	-0,0914
MS2 (cm)	44,5504	53,6638 ± 1,9470	-20,4564
Min (cm)	11,3135	19,4200 ± 4,0775	-71,6533
Suavidade (°)	8,4531	8,3139 ± 0,0745	1,6473

Tabela 13 – *Fuzzy* - melhor indivíduo ambiente 1 com velocidade linear máxima das rodas de 20cm/s.

Métricas - Controlador *Fuzzy*	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,3157	0,3077 ± 0,0045	2,5442
Velocidade linear média (cm/s)	16,6580	16,3151 ± 0,1280	2,0588
Velocidade angular média (rad/s)	0,1056	0,1043 ± 0,0013	1,1884
Sucesso	1	10	-
Distância percorrida (cm)	999,4821	976,9112 ± 7,5211	2,2583
Maior distância entre dois pontos (cm)	345,5463	344,8321 ± 2,2956	0,2067
MS1 (cm)	150,6870	150,7557 ± 0,8843	-0.0456
MS2 (cm)	51,5895	54,5027 ± 0,6655	-5,6469
Min (cm)	31,3543	30,7300 ± 1,8209	1,9911
Suavidade (°)	7,0971	7,0878 ± 0,0431	0,1316

A Tabela 14 apresenta os resultados da análise estatística para este experimento. O resultado do teste Shapiro-Wilk para checar a normalidade de cada uma das métricas aponta que todas as métricas para ambos controladores caracterizam-se por uma distribuição normal (P>0,05). Com o teste T observa-se, na mesma tabela, que 8 das 9 métricas indicam que os experimentos são diferentes um do outro (P<0,05) e 1 delas, que os experimentos são iguais (P>0,05). Conclui-se, portanto, que prevalece a diferença entre os controladores.

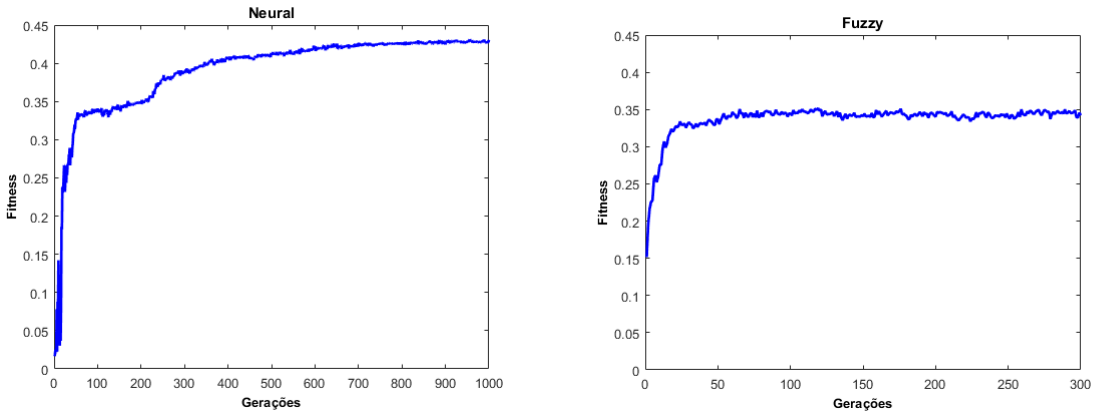
4.1.4 Ambiente 1 com velocidade linear máxima das rodas de 15cm/s

A Figura 23 apresenta a evolução da *fitness* para os controladores neural (Figura 23(a)) e *fuzzy* (Figura 23(b)) com limitação de velocidade linear máxima às ações de

Tabela 14 – Análise estatística do experimento 1 com velocidade linear máxima das rodas de 20cm/s

	Fuzzy	**Neural**	***Fuzzy*/Neural**
Métricas	**Shapiro-Wilk (P)**		**Test T Student (P)**
Fitness	0,8787	0,9175	$2{,}6721\ 10^{-18}$
Velocidade linear média	0,9432	0,9086	$1{,}5215\ 10^{-7}$
Velocidade angular média	0,8600	0,9880	$1{,}0832\ 10^{22}$
Sucesso	-	-	-
Distância percorrida	0,9201	0,7240	$1{,}1302\ 10^{-7}$
Maior distância entre dois pontos	0,9637	0,8446	$1{,}6427\ 10^{-20}$
MS1	0,9074	0,9158	$2{,}1052\ 10^{-8}$
MS2	0,7990	0,8652	0,1119
Min	0,9527	0,9307	$1{,}8584\ 10^{-6}$
Suavidade	0,9196	0,9391	$7{,}4909\ 10^{-17}$

controle de 15cm/s. Os parâmetros utilizados no processo evolutivo e aplicados a ambos os controladores são os mesmos apresentados na Tabela 2.

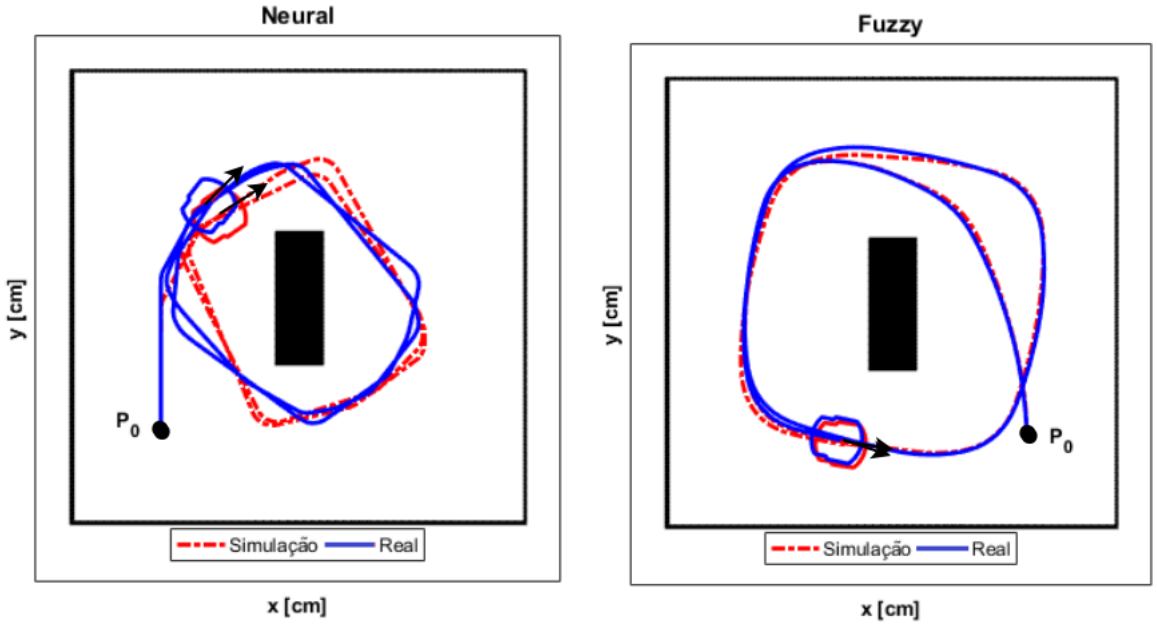


(a) (b)

Figura 23 – (a) Processo evolutivo neural, e (b) Processo evolutivo *fuzzy* com velocidade linear máxima das rodas de $15cm/s$.

Ainda na Figura 23, pode ser verificado que ambos os controladores passam a ter mais dificuldade para estabilizar, contudo, o controlador *fuzzy* mantém a característica de estabilizar primeiro, com apenas 75 gerações ele atinge uma região de estabilidade. Em contrapartida, no caso do controlador neural, o gráfico dá indícios de que a *fitness* poderia subir mais. Mantendo as mesmas características dos experimentos anteriores, em que o processo evolutivo se deu até esse limite de gerações, o melhor indivíduo da última geração de cada processo evolutivo (neural e *fuzzy*) foi transferido para o ambiente real e 10 experimentos foram realizados. Como o controlador neural evoluiu o comportamento de andar em sentido horário, sua pose inicial adotada foi P_0(0,8m; 0,8m; 90°), já o controlador *fuzzy* evoluiu o comportamento de andar no sentido anti-horário, portanto, sua pose inicial foi P_0(3,2m; 0,8m; 90°).

A Figura 24(a) apresenta o resultado do controlador neural em simulação e em ambiente real do primeiro experimento dos 10 realizados. Os pesos obtidos para o contro-



(a) (b)

Figura 24 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 2 com velocidade linear máxima das rodas de 15cm/s.

lador neural são apresentados em (4.9) e (4.10). Já a Figura 24(b) apresenta o resultado do controlador *fuzzy* em simulação e em ambiente real (para o primeiro experimento físico). O pesos dos consequentes das regras são apresentados na Tabela 15. A análise das regras permite concluir que os consequentes otimizados estão de acordo com a trajetória executada, com exceção das regras 1, 3 e 4, que ressalta que elas sozinhas não permitiriam a execução correta do trajeto, mas em conjuntos com as demais geram o resultado satisfatório obtido.

$$\mathbf{W_1} = \begin{bmatrix} 18,5861 & -18,5099 & 1,0101 \\ -4,8608 & 8,1909 & 1,0030 \\ -7,5069 & 5,6152 & 5,2790 \end{bmatrix} \tag{4.9}$$

$$\mathbf{W_2} = \begin{bmatrix} -25,6774 & 5,6190 & 8,3732 \\ 13,5973 & 10,0844 & 4,0320 \end{bmatrix} \tag{4.10}$$

As Tabelas 16 e 17 apresentam uma comparação entre simulação e experimentos reais dos respectivos controladores. Com a transferência, o controlador neural apresentou menor queda para as métricas de velocidade linear e angular, distância percorrida, MS1, MS2 e distância mínima, enquanto o *fuzzy* apresentou menor variação para a *fitness*, maior distância entre dois pontos e suavidade. Em uma avaliação superficial das métricas obtidas, pode-se notar que os experimentos possuem desempenhos similares com o resultado da evolução, isto é, as métricas são bem similares.

Tabela 15 – Conjunto de regras do controlador *fuzzy* para o ambiente 1 com velocidade linear máxima das rodas de $15cm/s$.

Regra	Estrutura da regra com consequente otimizado
1	Se S_d é P, S_e é P e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,0291$
2	Se S_d é P, S_e é P e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 1,0000$
3	Se S_d é P, S_e é G e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,0626$
4	Se S_d é P, S_e é G e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 0,8421$
5	Se S_d é G, S_e é P e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,0591$
6	Se S_d é G, S_e é P e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 0,7664$
7	Se S_d é G, S_e é G e S_f é P, então $v_{re} = 0,8999$ e $v_{rd} = 0,0104$
8	Se S_d é G, S_e é G e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 1,0000$

Tabela 16 – Neural - melhor indivíduo do ambiente 1 com velocidade linear máxima das rodas de 15cm/s.

Métricas - Controlador Neural	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,3808	0,3635 ± 0,0048	4,5257
Velocidade linear média (cm/s)	12,7085	12,6787 ± 0,0365	0,2343
Velocidade angular média (rad/s)	0,1127	0,1133 ± 0,0010	-0,5297
Sucesso	1	10	-
Distância percorrida (cm)	1525,0221	1525,2001 ± 3,5161	-0,0117
Maior distância entre dois pontos (cm)	280,8266	266,9526 ± 3,0871	4,9404
MS1 (cm)	154,2593	153,5947 ± 0,4113	0,4309
MS2 (cm)	48,2564	50,5974 ± 0,7206	-4,8512
Min (cm)	11,2945	11,7700 ± 2,5764	-4,2100
Suavidade (°)	7,2033	7,1848 ± 0,0368	0,2561

Tabela 17 – *Fuzzy* - melhor indivíduo do ambiente 1 com velocidade linear máxima das rodas de 15cm/s.

Métricas - Controlador *Fuzzy*	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,3447	0,3502 ± 0,0014	-1,5755
Velocidade linear média (cm/s)	13,0994	13,0017 ± 0,0154	0,7456
Velocidade angular média (rad/s)	0,0904	0,0885 ± 0,0005	2,1248
Sucesso	1	10	-
Distância percorrida (cm)	1571,9298	1563,4404 ± 1,2634	0,5401
Maior distância entre dois pontos (cm)	316,6743	320,9526 ± 1,3584	-1,3510
MS1 (cm)	150,6401	152,4408 ±0,3446	-1,1954
MS2 (cm)	58,6001	62,0461 ± 0,3957	-5,8806
Min (cm)	25,3297	27,9400 ± 2,5971	-10,3053
Suavidade (°)	6,6351	6,6296 ± 0,0145	0,0829

A Tabela 18 apresenta os resultados da análise estatística para este experimento. O resultado do teste Shapiro-Wilk para checar a normalidade de cada uma das métricas aponta que todas as métricas para ambos controladores caracterizam-se por uma distribuição normal (P>0,05). Com o teste T observa-se, na mesma tabela, que todas as métricas indicam que os experimentos são diferentes um do outro (P<0,05).

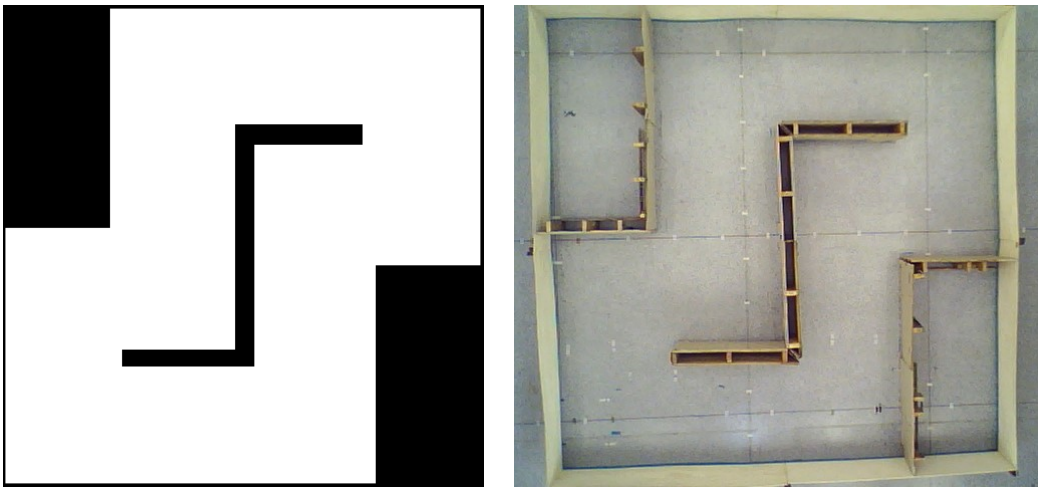
4.2 Experimento - Ambiente 2

Com o objetivo de verificar o comportamento dos controladores em outro ambiente com maior complexidade, que exigisse que o robô realizasse curvas para ambos os lados

Tabela 18 – Análise estatística do experimento 1 com velocidade linear máxima das rodas de 15 cm/s

	Fuzzy	**Neural**	***Fuzzy*/Neural**
Métricas	**Shapiro-Wilk (P)**		**Test T Student (P)**
Fitness	0.9318	0.7508	$3{,}6896\ 10^{-6}$
Velocidade linear média	0.9750	0.9200	$3{,}5355\ 10^{-12}$
Velocidade angular média	0.8008	0.9270	$1{,}2211\ 10^{-18}$
Sucesso	-	-	-
Distância percorrida	0.8501	0.9103	$1{,}4589\ 10^{-12}$
Maior distância entre dois pontos	0.9377	0.8737	$1{,}1563\ 10^{-15}$
MS1	0.8953	0.8835	$1{,}5457\ 10^{-6}$
MS2	0.9223	0.8214	$1{,}0203\ 10^{-16}$
Min	0.9411	0.9115	$2{,}0870\ 10^{-11}$
Suavidade	0.9252	0.9586	$5{,}4745\ 10^{-15}$

e tivesse que acelerar e desacelerar, foi proposto este novo experimento realizado com os mesmos níveis de velocidade avaliados no Ambiente 1. A estrutura deste novo ambiente é dada por um labirinto em forma de 'S' configurado dentro da mesma área de $4m \times 4m$ com corredores de $1m$ de largura. A Figura 25(a) apresenta o mapa utilizado em simulação e sua respectiva constituição em ambiente real é ilustrada na Figura 25(b). Neste caso, os obstáculos foram confeccionados com paletes forrados com papel e a parede externa é a mesma utilizada no ambiente anterior. Toda a estrutura mantém a altura de $0,5m$.



(a) (b)

Figura 25 – (a) Ambiente 2 simulado, e (b) ambiente 2 real.

Para este novo ambiente foi proposta uma nova função de *fitness*, em virtude das dificuldades que a *fitness* anterior apresentou na evolução de controladores para este novo cenário. A *fitness* anterior permitia somente que o robô evoluísse comportamentos que faziam com que o mesmo se deslocasse em sentido horário ou anti-horário, ou seja, o robô basicamente só conseguia realizar curva em um sentido ao fim do processo evolutivo. Outro ponto, é que a mesma não contemplava a exploração do ambiente, característica relevante em um ambiente tipo labirinto. Portanto, este novo cenário foi proposto para forçar o robô a fazer curvas para direita e esquerda e, consequentemente, explorar características

dinâmicas e ambientais que o Ambiente 1 não proporcionou. Em 4.11 é apresentada a nova função de *fitness* caracterizada por dois componentes igualmente ponderados empiricamente, o primeiro componente é idêntico à *fitness* anterior e o segundo contempla o produto entre a distância total percorrida (d_t) e a maior distância entre dois pontos distintos da trajetória ($d_{\Delta s}$) executada, características estas importantes para o ambiente em análise, pois o primeiro termo do produto reforça a componente que faz com que o robô tenha que andar e o segundo avalia se o ambiente foi explorado. Cada componente desta *fitness* foi ponderado empiricamente com peso $0,5$. Para o cálculo desse último componente é necessário passar à *fitness* o vetor de posições do robô, $Pos = [x, y]$, durante o experimento, além das dimensões do mapa para normalizar a componente $d_{\Delta s}$ em relação à diagonal do mapa (ou da imagem). A normalização da distância percorrida é dada em função do produto entre a velocidade linear máxima e o tempo de simulação ou experimento. Esta *fitness* traz como limitação a dependência das dimensões do ambiente.

$$F(\upsilon_n, \omega_n, d_n, Pos, mapa) = \frac{1}{2}\left(\frac{\sum_{n=1}^{n_a}\left[K_\upsilon\left(\frac{\upsilon_n}{\upsilon_{max}}\right) - K_\omega\left(\frac{|\omega_n|}{\omega_{max}}\right) - K_d d_n\right]}{t_{lim}/t_s} + d_t\ d_{\Delta s}\right) \tag{4.11}$$

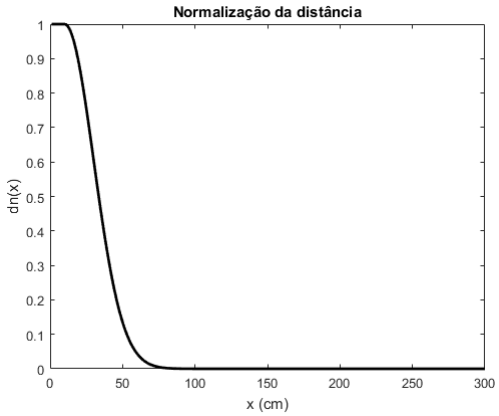
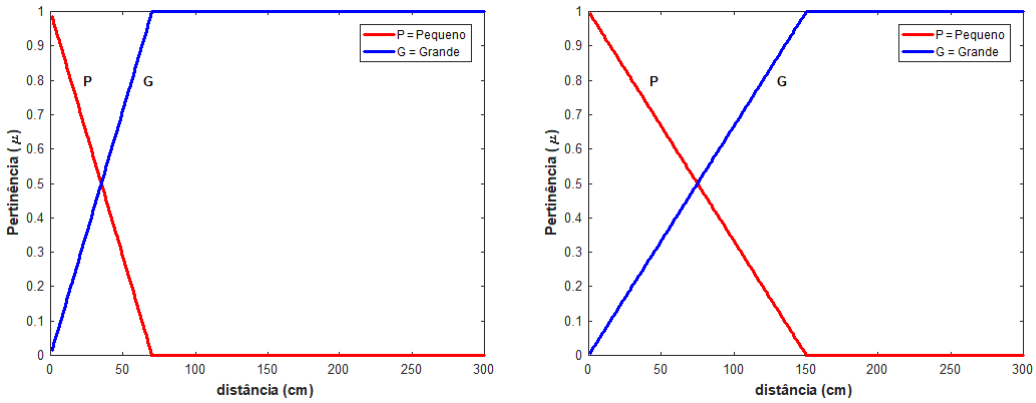


Figura 26 – Normalização da menor leitura dos sensores aplicada ao Ambiente 2.

No Ambiente 2, d_n (Equação 4.2) recebe os parâmetros de defasagem e fator de amortecimento apresentados na Tabela 19, e sua representação gráfica é apresentada na Figura 26. Como pode ser visto, o robô só passa a ser punido quando está a uma distância menor que $75cm$ do obstáculo.

Neste cenário, caracterizado por ser estreito, foram utilizados os conjuntos *fuzzy* apresentados na Figura 27(a) para caracterizar as variáveis *fuzzy* distância à direita e esquerda e os conjuntos apresentados na Figura 27(b), similares aos conjuntos do Ambiente 1, para caracterizar a variável *fuzzy* distância frontal. Esta alteração foi proposta para diminuir a sensibilidade do sensoriamento lateral devido o ambiente ser mais estreito que o anterior e assim não ficar restrito a uma pequena região de atuação dos conjuntos *fuzzy*.



(a) (b)

Figura 27 – (a) Conjuntos *fuzzy* esqueda e direita, e (b) Conjuntos *fuzzy* frontal.

4.2.1 Ambiente 2 com velocidade linear máxima das rodas de 30cm/s

A Figura 28 apresenta a evolução da *fitness* para os controladores neural e *fuzzy* com limitação de velocidade linear máxima de 30cm/s para o Ambiente 2. Os parâmetros utilizados no processo evolutivo neste e nos experimentos subsequentes são apresentados na Tabela 19. Pode ser verificado na figura que o controlador *fuzzy*, neste cenário, mantém a característica de estabilizar primeiro, por volta de 150 gerações, enquanto o controlador neural requer 900 gerações. Percebe-se também que o controlador neural atinge um valor de *fitness* superior ao fim do processo evolutivo, este fato pode ser evidenciado pelas métricas obtidas pelo controlador neural, em que sua velocidade linear média e a sua distância percorrida foram maiores que as obtidas pelo controlador *fuzzy* (como pode ser visto nas Tabelas 21 e 22), sendo estas métricas componentes da função de *fitness* adotada.

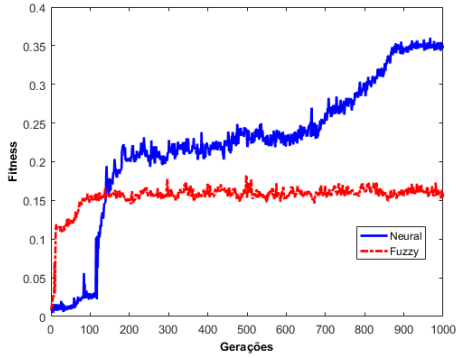


Figura 28 – Comparação do processo evolutivo neural e *fuzzy* do ambiente 2 com velocidade linear máxima das rodas de 30cm/s.

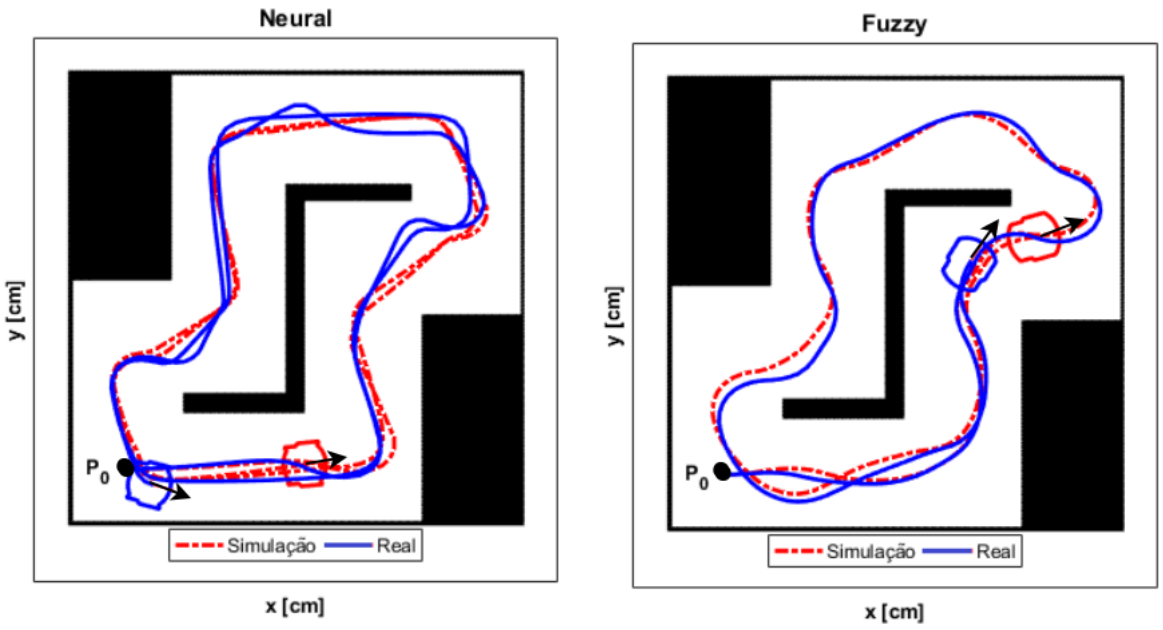
Tabela 19 – Parâmetros do algoritmo genético aplicado ao ambiente 2 com velocidade linear máxima das rodas de 30cm/s.

Parâmetros	**Valor**
Tamanho da população	100
Número de gerações	1000
Comprimento do cromossomo	15 (Neural) / 16 (*Fuzzy*)
Taxa de *crossover*	0,2
Taxa de mutação	0,3
Taxa de elitismo	0,1
Range de mutação	±0,5 (Neural) / ±0,4 (Fuzzy)
d_n	$\mu = 10,\ \sigma = 20$
K_v	0,2
K_ω	0,35
K_d	0,6
Posição inicial	aleatória

Posteriormente, o melhor indivíduo da última geração de cada processo evolutivo (neural e *fuzzy*) foi transferido para o ambiente real e 10 experimentos foram realizados, com a sequência iniciada com 100% da bateria do robô carregada e pose inicial P_0(0,5m;

0,5m; 0°) para ambos os controladores, em conformidade com os comportamentos apresentados ao fim do processo evolutivo, ou seja, o sentido de movimento.

Na Figura 29(a) tem-se o resultado do controlador neural em simulação e em ambiente real para o primeiro experimento dos 10 realizados. Os pesos obtidos para o controlador neural são apresentados em (4.12) e (4.13). Já a Figura 29(b) apresenta o resultado do controlador *fuzzy* em simulação e em ambiente real (também para o primeiro experimento físico). Os consequentes otimizados das regras são apresentados na Tabela 20. A regra de número 2 contribui para parada do robô, pois seus consequentes são nulos, já regra de número 8 não foi bem explorada no processo evolutivo, pois a mesma conduz o robô para a esquerda, quando deveria conduzi-lo à frente. As trajetórias executadas pelos controladores são bem similares em ambiente simulado e real, porém o controlador neural apresenta curvas mais acentuadas.



(a) (b)

Figura 29 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 2 com velocidade linear máxima das rodas de 30cm/s.

$$\mathbf{W_1} = \begin{bmatrix} 1,9108 & -11,1944 & 4,2621 \\ 11,9771 & -6,1326 & -13,2374 \\ -10,0757 & 0,6559 & 3,9605 \end{bmatrix} \tag{4.12}$$

$$\mathbf{W_2} = \begin{bmatrix} 2,0767 & -14,4825 & 5,2928 \\ -9,8762 & 2,2785 & 4,2006 \end{bmatrix} \tag{4.13}$$

As Tabelas 21 e 22 demonstram a comparação entre simulação e experimentos reais. Com a transferência, o controlador neural apresentou menor variação para as métricas

Tabela 20 – Conjunto de regras do controlador *fuzzy* para o ambiente 2 com velocidade linear máxima das rodas de $30cm/s$.

Regra	Estrutura das regras com consequentes otimizado
1	Se S_d é P, S_e é P e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,0576$
2	Se S_d é P, S_e é P e S_f é G, então $v_{re} = 0,0000$ e $v_{rd} = 0,0000$
3	Se S_d é P, S_e é G e S_f é P, então $v_{re} = 0,0036$ e $v_{rd} = 1,0000$
4	Se S_d é P, S_e é G e S_f é G, então $v_{re} = 0,0004$ e $v_{rd} = 0,8459$
5	Se S_d é G, S_e é P e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,0000$
6	Se S_d é G, S_e é P e S_f é G, então $v_{re} = 0,7361$ e $v_{rd} = 0,0109$
7	Se S_d é G, S_e é G e S_f é P, então $v_{re} = 0,5416$ e $v_{rd} = 0,0010$
8	Se S_d é G, S_e é G e S_f é G, então $v_{re} = 0,6065$ e $v_{rd} = 1,0000$

de maior distância entre dois pontos, MS1 e distância mínima. O controlador *fuzzy* apresentou menor variação para a *fitness*, velocidade linear e angular, distância percorrida, MS2 e suavidade. Também é possível destacar que o controlador fuzzy apresentou melhor suavidade e que em ambos os controladores não ocorreram choques com os obstáculos nos 10 experimentos realizados .

Tabela 21 – Neural - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $30cm/s$

Métricas - Controlador Neural	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,2485	0,2384 ± 0,0040	4,0423
Velocidade linear média (cm/s)	20,3998	19,4304 ± 0,2534	4,7520
Velocidade angular média (rad/s)	0,2470	0,2651 ± 0,0079	-7,3049
Sucesso	1	10	-
Distância percorrida (cm)	2447,9807	2324,1373 ± 31,6727	5,0590
Maior distância entre dois pontos (cm)	413,0248	417,5731 ± 3,1467	-1,1012
MS1 (cm)	94,5505	91,9990 ± 0,7955	2,6985
MS2 (cm)	29,7741	31,2099 ± 0,4173	-4,8223
Min (cm)	6,3585	13,0000 ± 2,9181	-104,4507
Suavidade (°)	11,9933	21,1433 ± 2,4435	-76,2923

Tabela 22 – *Fuzzy* - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $30cm/s$

Métricas - Controlador *Fuzzy*	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,1609	0,1555 ± 0,0013	3,3516
Velocidade linear média (cm/s)	12,9778	12,7416 ± 0,0416	1,8204
Velocidade angular média (rad/s)	0,1879	0,1777 ± 0,0036	5,4343
Sucesso	1	10	-
Distância percorrida (cm)	1557,3415	1533,5512 ± 2,9269	1,5276
Maior distância entre dois pontos (cm)	411,4764	405,4743 ± 2,2289	1,4587
MS1 (cm)	95,2638	92,2779 ± 0,4797	3,1344
MS2 (cm)	29,9676	29,4817 ± 0,1884	1,6214
Min (cm)	3,8392	8,5000 ± 2,2667	-121,4003
Suavidade (°)	8,1636	8,3417 ± 1,4388	-2,1813

A Tabela 23 apresenta os resultados da análise estatística para este experimento. O resultado do teste Shapiro-Wilk para checar a normalidade de cada uma das métricas aponta que todas as métricas para ambos controladores caracterizam-se por uma distribuição normal (P>0,05). Com o teste T observa-se, na mesma tabela, que 8 das métricas indicam que os experimentos são diferentes um do outro (P<0,05) e uma delas, que os

experimentos são iguais (P>0,05). Conclui-se, portanto, que prevalece a diferença entre os controladores.

Tabela 23 – Análise estatística do Ambiente 2 com velocidade linear máxima das rodas de 30cm/s

	Fuzzy	**Neural**	***Fuzzy*/Neural**
Métricas	**Shapiro-Wilk (P)**		**Test T Student (P)**
Fitness	0,9244	0,9168	$7{,}1501\ 10^{-14}$
Velocidade linear média	0,9681	0,9321	$1{,}4519\ 10^{-14}$
Velocidade angular média	0,9764	0,9322	$4{,}9658\ 10^{-14}$
Sucesso	-	-	-
Distância percorrida	0,9711	0,8712	$2{,}2105\ 10^{-14}$
Maior distância entre dois pontos	0,8685	0,9535	$1{,}5319\ 10^{-8}$
MS1	0,9667	0,8857	0,1788
MS2	0,9264	0,9428	$1{,}1121\ 10^{-8}$
Min	0,8191	0,9164	0,0006
Suavidade	0,5712	0,8288	$1{,}9473\ 10^{-10}$

4.2.2 Ambiente 2 com velocidade linear máxima das rodas de 25cm/s

A Figura 30 apresenta a evolução da *fitness* para os controladores neural e *fuzzy* com limitação de velocidade linear máxima de 25cm/s no Ambiente 2. Os parâmetros utilizados nos processos evolutivos de ambos os controladores são os mesmos apresentados na Tabela 19. Pode ser verificado na Figura 30 que o controlador *fuzzy* atinge uma região de estabilidade após 600 gerações enquanto que o controlador neural requer 900 gerações, além disso, este atinge um valor de *fitness* superior ao fim do processo evolutivo. Tal fato pode ser evidenciado pelo controlador neural evoluir uma velocidade linear média superior e uma velocidade angular inferior comparada ao controlador *fuzzy* e com isso, percorrer um trajeto maior (vide Tabelas 25 e 26). Tais métricas são levadas em conta na estimação do desempenho pela *fitness*.

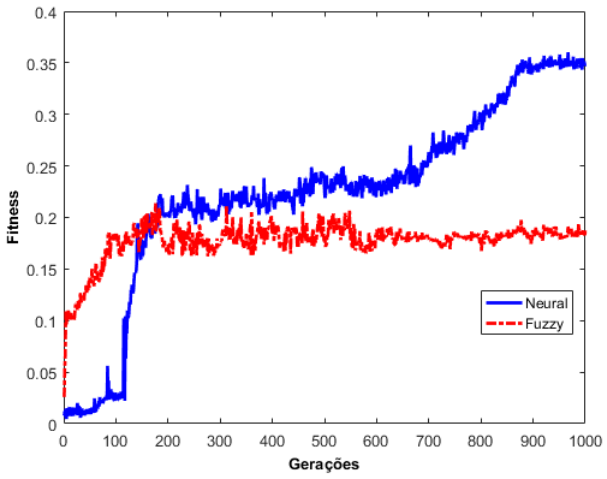
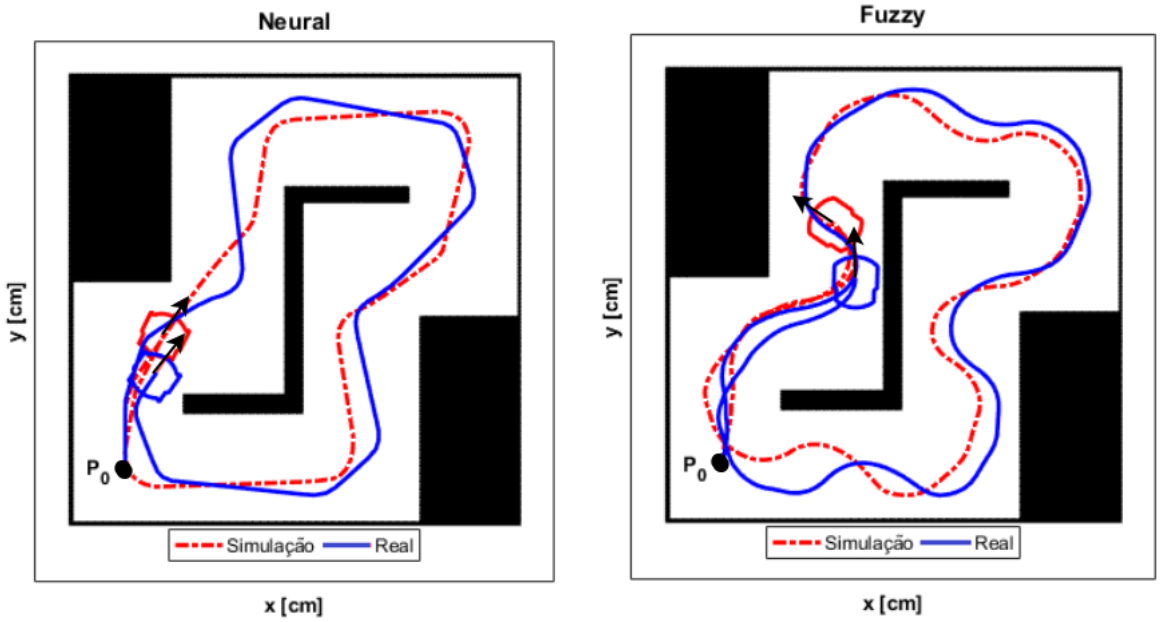


Figura 30 – Comparação do processo evolutivo neural e *fuzzy* do ambiente 2 com velocidade linear máxima das rodas de 25cm/s.

Posteriormente, o melhor indivíduo da última geração de cada processo evolutivo (neural e *fuzzy*) foi transferido para o ambiente real e 10 experimentos foram realizados com a mesma condição inicial aplicada ao experimento anterior, exceto pela pose inicial, que sofreu alteração na orientação do robô em decorrência do trajeto evoluído, portanto, P_0(0,5m; 0,5m; 90°).

Na Figura 31(a) tem-se o resultado do controlador neural em simulação e em ambiente real para o primeiro experimento dos 10 realizados. Os pesos obtidos para o controlador neural são apresentados em (4.14) e (4.15) e sua trajetória se apresenta em sentido horário, bem como o controlador *fuzzy*. Os consequentes otimizados das regras são apresentados na Tabela 24. As regras 2, 7 e 8 não são bem exploradas durante a evolução, na segunda regra o robô é induzido a ir para a direita, sendo que o ideal seria seguir em frente, na sétima regra o robô é induzido a seguir em frente, quando deveria girar em qualquer sentido para desviar do obstáculo em frente, e na última regra o robô deveria seguir em frente, porém esta regra gera consequentes nulos que induz o robô a parar. Vale destacar que as ações de controle são tomadas levando-se em consideração as pertinências resultantes de todas as regras, o que permite a execução com sucesso da tarefa mesmo quando algumas das regras não são bem evoluídas.



(a) (b)

Figura 31 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 2 com velocidade linear máxima das rodas de 25cm/s.

$$\mathbf{W_1} = \begin{bmatrix} 0,7856 & 6,9993 & -4,7305 \\ -12,1827 & 2,6356 & 8,5001 \\ 4,1506 & -9,0332 & 2,1358 \end{bmatrix} \tag{4.14}$$

$$\mathbf{W_2} = \begin{bmatrix} 2,0860 & 10,2408 & -11,5331 \\ 10,4717 & -4,7882 & -6,2074 \end{bmatrix} \tag{4.15}$$

Tabela 24 – Conjunto de regras do controlador *fuzzy* para o ambiente 2 com velocidade linear máxima das rodas de $25cm/s$.

Regra	Estrutura das regras com consequentes otimizado
1	Se S_d é P, S_e é P e S_f é P, então $v_{re} = 0,8559$ e $v_{rd} = 0,0000$
2	Se S_d é P, S_e é P e S_f é G, então $v_{re} = 0,8148$ e $v_{rd} = 0,0000$
3	Se S_d é P, S_e é G e S_f é P, então $v_{re} = 0,0016$ e $v_{rd} = 1,0000$
4	Se S_d é P, S_e é G e S_f é G, então $v_{re} = 0,1146$ e $v_{rd} = 1,000$
5	Se S_d é G, S_e é P e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,0000$
6	Se S_d é G, S_e é P e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 0,0028$
7	Se S_d é G, S_e é G e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,9679$
8	Se S_d é G, S_e é G e S_f é G, então $v_{re} = 0,0000$ e $v_{rd} = 0,0000$

As Tabelas 25 e 26 demonstram os resultados obtidos com a transferência da simulação para a realidade. Com a transferência para o ambiente físico o controlador neural obteve sucesso nos 10 experimentos realizados, enquanto o controlador *fuzzy* gerou choques em 3 ocasiões, nas quais o robô toca no obstáculo lateralmente. O controlador neural apresentou variação menor com a transferência para as métricas de velocidade linear e maior distância entre dois pontos, enquanto o controlador *fuzzy* apresentou menor variação para a *fitness*, velocidade angular, distância percorrida, MS1, MS2, distância mínima e suavidade. Pode ser observado que os controladores apresentam desempenhos similares perante as métricas relacionadas às distâncias aos obstáculos (métricas de segurança). Além disso, neste experimento o controlador neural foi mais suave que o *fuzzy*, já que realiza trechos do trajeto em linha reta.

Tabela 25 – Neural - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $25cm/s$

Métricas - Controlador Neural	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,3489	0,3236 ± 0,0022	7,2536
Velocidade linear média (cm/s)	20,8683	20,0864 ± 0,1516	3,7468
Velocidade angular média (rad/s)	0,1774	0,2063 ± 0,0039	-16,3270
Sucesso	1	10	-
Distância percorrida (cm)	1252,0958	1196,2609 ± 8,6419	4,4593
Maior distância entre dois pontos (cm)	429,4783	420,0105 ± 4,5646	2,2045
MS1 (cm)	98,5393	92,0591 ± 0,7901	6,5762
MS2 (cm)	26,1790	29,8029 ± 0,7013	-13,8428
Min (cm)	1,8711	9,5300 ± 2,9014	-409,3261
Suavidade (°)	9,4142	10,4424 ± 0,1199	-10,9217

Através do teste estatístico Shapiro-Wilk obteve-se que ambos os experimentos apresentam normalidade em cada métrica em análise (P>0,05) e, para o teste T, obteve-se que todas as métricas indicam que os experimentos são diferentes (P<0,05).

Tabela 26 – *Fuzzy* - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $25cm/s$

Métricas - Controlador *Fuzzy*	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,1837	0,1838 ± 0,0013	-0.0249
Velocidade linear média (cm/s)	12,5245	12,0402 ± 0,1155	3,8672
Velocidade angular média (rad/s)	0,2295	0,2222 ± 0.0082	3,1665
Sucesso	1	7	-
Distância percorrida (cm)	1502,9352	1451,3817 ± 15,0070	3,4302
Maior distância entre dois pontos (cm)	408,9547	425,0121 ± 3,8080	-3,9264
MS1 (cm)	93,8928	93,9671 ± 1,0598	-0,0792
MS2 (cm)	27,6377	27,8137 ± 0,3344	-0,6369
Min (cm)	2,4907	2,1429 ± 1,0438	13,9657
Suavidade (°)	13,0006	12,2508 ± 0,2613	5,7673

Tabela 27 – Análise estatística do Ambiente 2 com velocidade linear máxima das rodas de 25cm/s

	Fuzzy	**Neural**	***Fuzzy*/Neural**
Métricas	**Shapiro-Wilk (P)**		**Test T Student (P)**
Fitness	0,9820	0,9502	$3{,}5401\ 10^{-26}$
Velocidade linear média	0,9752	0,9704	$2{,}6208\ 10^{-24}$
Velocidade angular média	0,9576	0,9189	0,0007
Sucesso	-	-	-
Distância percorrida	0,9134	0,9399	$8{,}4543\ 10^{-12}$
Maior distância entre dois pontos	0,9676	0,9412	0,0139
MS1	0,8491	0,9556	0,0010
MS2	0,9475	0,9654	$9{,}2763\ 10^{-7}$
Min	0,8783	0,9536	$4{,}1561\ 10^{-6}$
Suavidade	0,9292	0,8915	$6{,}9769\ 10^{-8}$

4.2.3 Ambiente 2 com velocidade linear máxima das rodas de 20cm/s

A Figura 32 apresenta a evolução da *fitness* para os controladores neural e *fuzzy* com limitação de velocidade linear máxima de 20cm/s no Ambiente 2. Os parâmetros utilizados nos processos evolutivos dos controladores são os mesmos apresentados na Tabela 19. Pode ser verificado que o controlador *fuzzy* atinge uma região de estabilidade rapidamente, e após 400 gerações os efeitos de oscilação minimizam, enquanto que o controlador neural requer 900 gerações para estabilizar sua evolução. Percebe-se também que o controlador neural atinge um valor de *fitness* superior ao *fuzzy* ao fim do processo evolutivo, este fato pode ser evidenciado pelo controlador neural evoluir para uma velocidade linear média maior e, consequentemente, percorrer uma distância maior que o controlador *fuzzy* (vide Tabelas 29 e 30).

Posteriormente, o melhor indivíduo da última geração de cada processo evolutivo (neural e *fuzzy*) foi transferido para o ambiente real e 10 experimentos foram realizados, com a sequência iniciada com 100% da bateria do robô carregada e pose inicial P_0(0,5m; 0,5m; 0°).

Na Figura 33(a) tem-se o resultado do controlador neural em simulação e em ambiente real para o primeiro experimento dos 10 realizados. Os pesos obtidos para o controlador neural são apresentados em (4.16) e (4.17) e sua trajetória se apresenta em

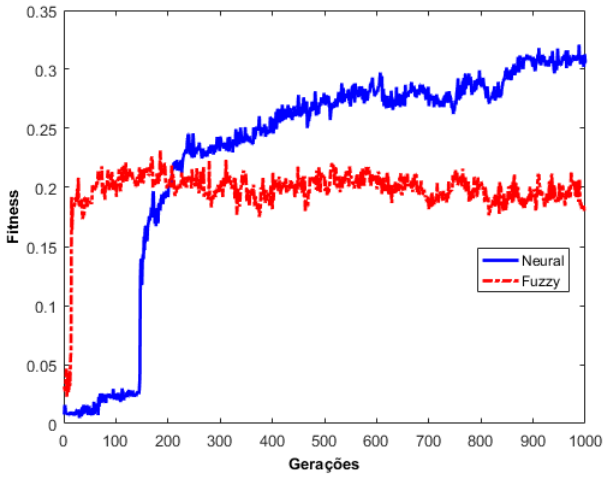
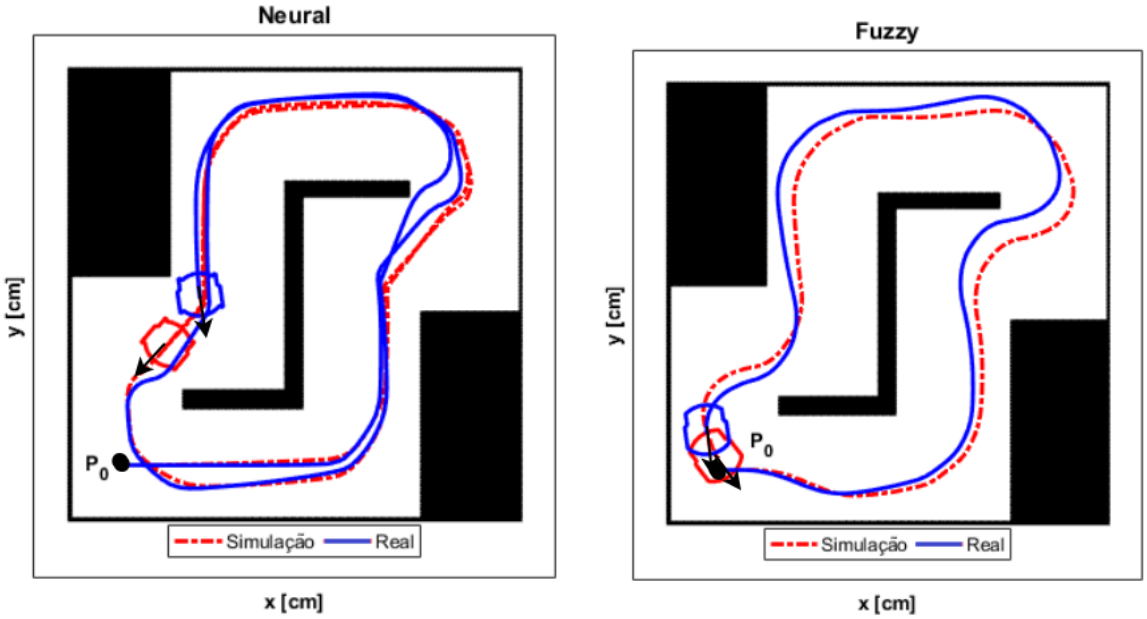


Figura 32 – Comparação do processo evolutivo neural e *fuzzy* do ambiente 2 com velocidade linear máxima das rodas de 20cm/s.

sentido anti-horário, bem como o controlador *fuzzy*. Já a Figura 33(b) apresenta o resultado do controlador *fuzzy* em simulação e em ambiente real (também para o primeiro experimento físico). Percebe-se com as trajetórias executadas que ambos os controladores apresentam suavidade similar e o *fuzzy* apresenta dispersão maior entre as trajetórias simulada e real. Os consequentes otimizados das regras são apresentados na Tabela 28. Dentre as regras, a de número 2 é a única que não foi bem evoluída, neste caso o robô gira para a esquerda quando deveria seguir em frente.



(a) (b)

Figura 33 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 2 com velocidade linear máxima das rodas de 20cm/s.

$$\mathbf{W_1} = \begin{bmatrix} -13,2780 & 10,0998 & 3,4836 \\ -3,7749 & -6,1269 & 7,5409 \\ 9,1199 & -3,3299 & -13,8219 \end{bmatrix} \tag{4.16}$$

$$\mathbf{W_2} = \begin{bmatrix} 10,3825 & -1,5588 & -11,4544 \\ 4,3063 & -7,3757 & 6,9275 \end{bmatrix} \tag{4.17}$$

Tabela 28 – Conjunto de regras do controlador *fuzzy* para o ambiente 2 com velocidade linear máxima das rodas de $20cm/s$.

Regra	Estrutura das regras com consequentes otimizado
1	Se S_d é P, S_e é P e S_f é P, então $v_{re} = 0,0000$ e $v_{rd} = 0,0127$
2	Se S_d é P, S_e é P e S_f é G, então $v_{re} = 0,0003$ e $v_{rd} = 1,0000$
3	Se S_d é P, S_e é G e S_f é P, então $v_{re} = 0,0008$ e $v_{rd} = 1,0000$
4	Se S_d é P, S_e é G e S_f é G, então $v_{re} = 0,0000$ e $v_{rd} = 0,4987$
5	Se S_d é G, S_e é P e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,0021$
6	Se S_d é G, S_e é P e S_f é G, então $v_{re} = 0,7870$ e $v_{rd} = 0,4594$
7	Se S_d é G, S_e é G e S_f é P, então $v_{re} = 0,0049$ e $v_{rd} = 0,0003$
8	Se S_d é G, S_e é G e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 0,9849$

As Tabelas 29 e 30 demonstram os resultados obtidos com a transferência da simulação para a realidade. Com a transferência para o ambiente físico, os controladores obtiveram sucesso nos 10 experimentos realizados. O controlador neural apresentou menor variação com a transferência apenas para a métrica de suavidade, enquanto o controlador *fuzzy* apresentou menor variação nas demais. Vale destacar que ambos os controladores apresentaram trajetórias similares no que concerne à suavidade.

Tabela 29 – Neural - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $20cm/s$

Métricas - Controlador Neural	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,3108	0,3026 ± 0,0020	2,6532
Velocidade linear média (cm/s)	16,7561	16,4155 ± 0,0267	2,0329
Velocidade angular média (rad/s)	0,1531	0,1611 ± 0,0020	-5,2077
Sucesso	1	10	-
Distância percorrida (cm)	2010,7373	1973,1314 ± 4,0413	1,8703
Maior distância entre dois pontos (cm)	416,3019	413,2135 ± 2,0901	0,74186
MS1 (cm)	93,6647	89,7813 ± 0,3567	4,1461
MS2 (cm)	26,4748	27,4533 ± 0,3537	-3,6958
Min (cm)	9,2915	5,2500 ± 1,9778	43,4967
Suavidade (°)	8,8325	9,0047 ± 0,0575	-1,9500

Através do teste estatístico Shapiro-Wilk obteve-se que ambos os experimentos apresentam normalidade em cada métrica em análise (P>0,05) e, para o teste T, obteve-se que todas as métricas indicam que os experimentos são diferentes (P<0,05).

4.2.4 Ambiente 2 com velocidade linear máxima das rodas de 15cm/s

A Figura 34 apresenta a evolução da *fitness* para os controladores neural e *fuzzy* com limitação de velocidade linear máxima de 15cm/s no Ambiente 2. Os parâmetros uti-

Tabela 30 – *Fuzzy* - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $20cm/s$

Métricas - Controlador*Fuzzy*	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,1815	0,1785 ± 0,0005	1,6411
Velocidade linear média (cm/s)	9,4945	9,3266 ± 0,0256	1,7681
Velocidade angular média (rad/s)	0,1061	0,1018 ± 0,0016	4,0971
Sucesso	1	10	-
Distância percorrida (cm)	1139,3389	1121,6296 ± 2,5961	1,5544
Maior distância entre dois pontos (cm)	422,5813	424,3287 ± 1,1949	-0,4135
MS1 (cm)	89,7502	86,9328 ± 0,3211	3,1392
MS2 (cm)	23,1447	22,8423 ± 0,2789	1,3068
Min (cm)	3,9565	3,4400 ± 0,4671	13,0545
Suavidade (°)	8,8968	7,0040 ± 1,9007	21,2756

Tabela 31 – Análise estatística do Ambiente 2 com velocidade linear máxima das rodas de 20cm/s

	Fuzzy	**Neural**	***Fuzzy*/Neural**
Métricas	**Shapiro-Wilk (P)**		**Test T Student (P)**
Fitness	0,9059	0,9315	$2{,}2644\ 10^{-19}$
Velocidade linear média	0,9713	0,8959	$1{,}4985\ 10^{-40}$
Velocidade angular média	0,9864	0,9472	$4{,}0194\ 10^{-23}$
Sucesso	-	-	-
Distância percorrida	0,8486	0,9433	$3{,}9499\ 10^{-34}$
Maior distância entre dois pontos	0,9127	0,9406	$3{,}6376\ 10^{-10}$
MS1	0,9413	0,9526	$1{,}4381\ 10^{-13}$
MS2	0,9400	0,9194	$5{,}1183\ 10^{-17}$
Min	0,9477	0,8518	0,0091
Suavidade	0,6702	0,9587	0,0044

lizados nos processos evolutivos de ambos os controladores são os mesmos apresentados na Tabela 19. Pode ser verificado que o controlador *fuzzy* atinge sua região de estabilidade rapidamente, contudo apresenta algumas oscilações que são minimizadas após 500 gerações. Já o controlador neural apresenta uma evolução lenta e suave sendo que ao fim das 1000 gerações o mesmo indica que pode gerar indivíduos ainda melhores. Percebe-se também que o controlador neural novamente atinge um valor de *fitness* superior ao fim das 1000 gerações, justificado pelos mesmos critérios dos experimentos anteriores, velocidade linear, velocidade angular e distância percorrida maiores, vide Tabelas 33 e 34.

Posteriormente, o melhor indivíduo da última geração de cada processo evolutivo (neural e *fuzzy*) foi transferido para o ambiente real e 10 experimentos foram realizados, com a sequência iniciada com 100% da bateria do robô carregada e pose inicial P_0(0,5m; 0,5m; 0°).

Na Figura 35(a) tem-se o resultado do controlador neural em simulação e em ambiente real para o primeiro experimento dos 10 realizados. Os pesos obtidos para o controlador neural são apresentados em (4.18) e (4.19) e sua trajetória se apresenta em sentido anti-horário, bem como o controlador *fuzzy*. Já a Figura 35(b) apresenta o resultado do controlador *fuzzy* em simulação e em ambiente real (também para o primeiro experimento físico). Os consequentes otimizados das regras são apresentados na Tabela

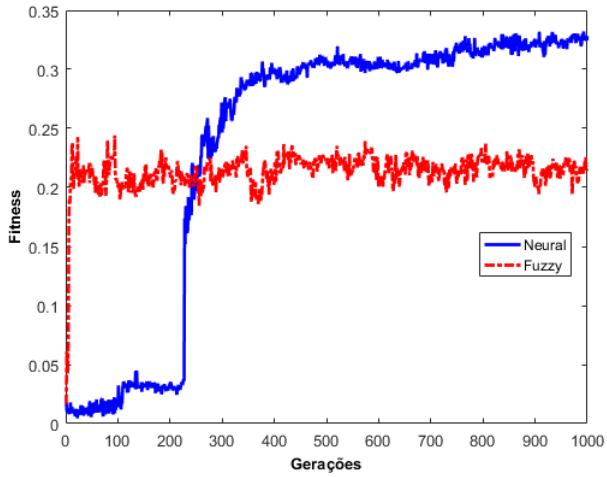
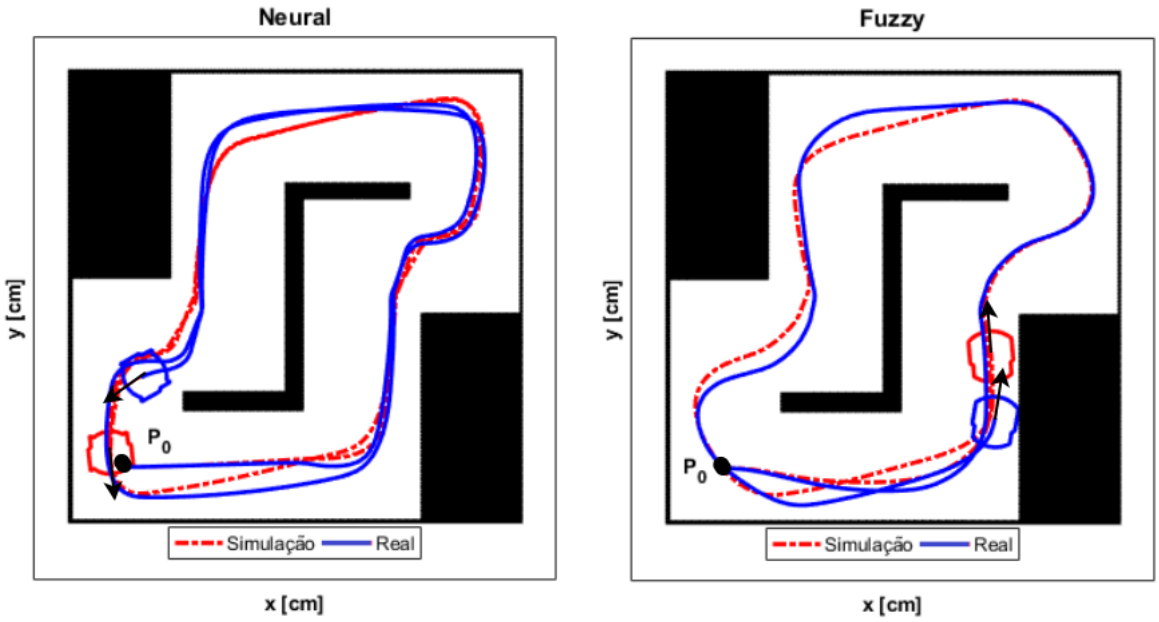


Figura 34 – Comparação do processo evolutivo neural e *fuzzy* do ambiente 2 com velocidade linear máxima das rodas de 15cm/s.

32. A regra 8 não foi bem evoluída para este experimento. Isoladamente, esta regra induz o robô a girar para a esquerda, quando deveria seguir em frente. Já a regra 7 é responsável pela parada do robô quando deveria girar para um dos lados.



(a) (b)

Figura 35 – (a) Controlador neural simulado e real, (b) Controlador *fuzzy* simulado e real no ambiente 2 com velocidade linear máxima das rodas de 15cm/s.

$$\mathbf{W_1} = \begin{bmatrix} -2,6947 & 14,4476 & -1,4381 \\ 5,2866 & -6,6643 & 1,5500 \\ -12,1183 & 0,2381 & 20,5133 \end{bmatrix} \tag{4.18}$$

$$\mathbf{W_2} = \begin{bmatrix} -5,2991 & -17,7487 & 18,3550 \\ 8,4605 & -4,6261 & -5,3333 \end{bmatrix} \tag{4.19}$$

Tabela 32 – Conjunto de regras do controlador *fuzzy* para o ambiente 2 com velocidade linear máxima das rodas de $15cm/s$.

Regra	Estrutura das regras com consequentes otimizado
1	Se S_d é P, S_e é P e S_f é P, então $v_{re} = 0,3637$ e $v_{rd} = 0,0013$
2	Se S_d é P, S_e é P e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 0,0001$
3	Se S_d é P, S_e é G e S_f é P, então $v_{re} = 0,0001$ e $v_{rd} = 1,0000$
4	Se S_d é P, S_e é G e S_f é G, então $v_{re} = 0,0000$ e $v_{rd} = 1,0000$
5	Se S_d é G, S_e é P e S_f é P, então $v_{re} = 1,0000$ e $v_{rd} = 0,0000$
6	Se S_d é G, S_e é P e S_f é G, então $v_{re} = 1,0000$ e $v_{rd} = 0,9321$
7	Se S_d é G, S_e é G e S_f é P, então $v_{re} = 0,0000$ e $v_{rd} = 0,0000$
8	Se S_d é G, S_e é G e S_f é G, então $v_{re} = 0,3780$ e $v_{rd} = 0,9854$

As Tabelas 33 e 34 demonstram os resultados obtidos com a transferência da simulação para a realidade. Com a transferência, o controlador neural apresentou uma menor variação com a transferência para as métricas de *fitness*, velocidade linear e angular, distância percorrida, MS1 e MS2. O controlador *fuzzy* apresenta menor variação para a maior distância entre dois pontos, distância mínima e suavidade. Os controladores apresentaram bom desempenho, sem choques com obstáculos nos 10 experimentos em ambiente físico.

Tabela 33 – Neural - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $15cm/s$

Métricas - Controlador Neural	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,3329	0,3261 ± 0,0008	2,0397
Velocidade linear média (cm/s)	12,2834	12,1567 ± 0,0207	1,0315
Velocidade angular média (rad/s)	0,1291	0,1296 ± 0,0012	-0,4424
Sucesso	1	10	-
Distância percorrida (cm)	2211,0050	2195,7854 ± 3,3768	0,6884
Maior distância entre dois pontos (cm)	455,4060	449,6890 ± 0,7792	1,2554
MS1 (cm)	92,2868	89,9660 ± 0,3508	2,5148
MS2 (cm)	26,2725	25,5054 ± 0,4370	2,9198
Min (cm)	3,7304	4,2300 ± 0,7704	-13,3927
Suavidade (°)	6,6903	9,0763 ± 1,5650	-35,6637

Tabela 34 – *Fuzzy* - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $15cm/s$

Métricas - Controlador *Fuzzy*	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,2150	0,2065 ± 0,0013	3,9547
Velocidade linear média (cm/s)	8,1314	7,9747 ± 0,0233	1,9269
Velocidade angular média (rad/s)	0,0794	0,0777 ± 0,0006	2,1678
Sucesso	1	10	-
Distância percorrida (cm)	1463,6574	1442,2372 ± 3,8649	1,4635
Maior distância entre dois pontos (cm)	427,1318	429,5676 ± 2,4046	-0,5703
MS1 (cm)	90,3551	87,0400 ± 0,3378	3,6690
MS2 (cm)	24,3931	23,5183 ± 0,1987	3,5864
Min (cm)	4,0863	3,9300 ± 0,8394	3,8250
Suavidade (°)	3,8932	3,8651 ± 0,0178	0,7220

Através do teste estatístico Shapiro-Wilk obteve-se que ambos os experimentos apresentam normalidade em cada métrica em análise (P>0,05) e, para o teste T, obteve-se que 8 das métricas indicam que os experimentos são diferentes (P<0,05) e uma delas que são iguais (P>0,05). Conclui-se, portanto, que prevalece a diferença entre os controladores.

Tabela 35 – Análise estatística do Ambiente 2 com velocidade linear máxima das rodas de 15cm/s

	Fuzzy	**Neural**	***Fuzzy*/Neural**
Métricas	**Shapiro-Wilk (P)**		**Test T Student (P)**
Fitness	0,8742	0,9050	$1{,}0487\ 10^{-28}$
Velocidade linear média	0,9202	0,9408	$9{,}0532\ 10^{-38}$
Velocidade angular média	0,9630	0,9131	$8{,}6757\ 10^{-23}$
Sucesso	-	-	-
Distância percorrida	0,8874	0,9608	$1{,}8280\ 10^{-38}$
Maior distância entre dois pontos	0,8345	0,9175	$2{,}2378\ 10^{-11}$
MS1	0,8780	0,9463	$1{,}1669\ 10^{-13}$
MS2	0,9088	0,9012	$3{,}6549\ 10^{-9}$
Min	0,8600	0,9195	0,2080
Suavidade	0,9418	0,8280	$1{,}1626\ 10^{-6}$

4.3 Análise com base em câmera

Os resultados apresentados anteriormente foram baseados na odometria do robô. Devido às incertezas inerentes a tal abordagem, como acúmulo de erros proporcional à distância trafegada, que podem ser classificados em erros sistemáticos e não sistemáticos. São exemplos de erros sistemáticos: pequenas diferenças no diâmetro das rodas, desalinhamento das rodas, resolução finita do *encorder*, taxa de amostragem de leitura do *encorder* finita, etc. Por outro lado, são exemplos de erros não sistemáticos: a irregularidade do terreno, que pode causar derrapagem das rodas por pouco atrito com o piso, acelerações bruscas, manobras rápidas e choques com obstáculos.

Diante desses fatos que prejudicam a localização do robô, um aspecto fundamental em se tratando de navegação autônoma, foram propostos novos experimentos que visaram averiguar se a consideração somente da odometria causou impactos significativos, dado que as métricas de velocidade linear, angular, distância percorrida, maior distância entre dois pontos e suavidade ficam comprometidas pelos erros de odometria. Para estes novos experimentos foi utilizada uma câmera fixada no teto do laboratório com o objetivo de filmar o trajeto percorrido pelo robô e, posteriormente, permitir a estimação da localização do robô sem os erros referentes à odometria.

Para avaliar o comportamento dos controladores perante a transferência e comparálos foi utilizado o algoritmo *Dynamic Time Warping* (DTW) (BERNDT; CLIFFORD, 1994). Esta técnica visa encontrar o melhor alinhamento entre elementos de duas sequências com amostragens distintas, neste caso, da trajetória obtida com a câmera (que possui um número de amostras superior) e a simulação. Esse algoritmo estipula uma estima-

tiva da menor distância entre as trajetórias. O algoritmo cria uma matriz com todas as possíveis combinações de dois em dois elementos fornecidos pelas duas séries fornecendo como resultado a distância total entre as duas sequências com base na soma do menor caminho construído por essa matriz. A distância total foi ponderada pelo tamanho do caminho resultante. Portanto, o DTW será utilizado como uma métrica complementar na comparação das trajetórias, sendo que a trajetória que apresentar menor distância para a trajetória simulada indica qual foi o controlador melhor avaliado nesse quesito. A Tabela 36 apresenta uma comparação entre as trajetórias extraídas com a câmera e as simulações com base no DTW. Como resultado pode ser averiguado que o controlador *fuzzy* apresentou trajetos mais próximos às simulações que o controlador neural em praticamente todos os cenários, exceto para o ambiente 2 com velocidade linear máxima de 15 cm/s.

As Figuras 36, 37, 38 e 39 apresentam as trajetórias comparativas para o ambiente 1 de ambos os controladores e as Figuras 40, 41, 42 e 43 trazem os resultados para o ambiente 2. Analisando as figuras é possível verificar que o controlador neural apresenta maior diferença entre trajeto real e simulado que o controlador *fuzzy*. Além disso, estes resultados não destoam significativamente das trajetórias obtidas com base na odometria.

Posteriormente é apresentada uma avaliação comparativa dos resultados obtidos de modo a facilitar a avaliação dos experimentos realizados no trabalho.

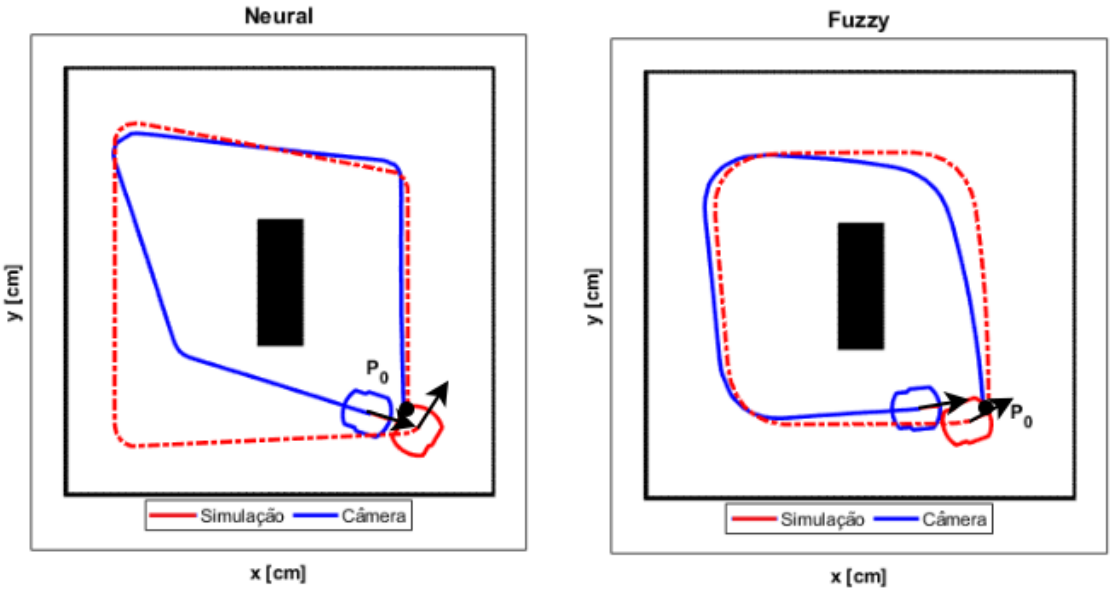
Tabela 36 – Comparação da trajetória extraída com a câmera e simulação com base no DTW.

	Ambiente 1				**Ambiente 2**			
Controlador	**30cm/s**	**25cm/s**	**20cm/s**	**15cm/s**	**30cm/s**	**25cm/s**	**20cm/s**	**15cm/s**
***Fuzzy* (cm)**	11,5673	8,6751	4,5892	8,4808	5,8995	4,8985	5,2025	8,9743
Neural (cm)	21,4717	17,8647	9,8859	10,7037	10,9111	12,9457	12,3709	6,0809

4.4 Análise comparativa dos resultados experimentais

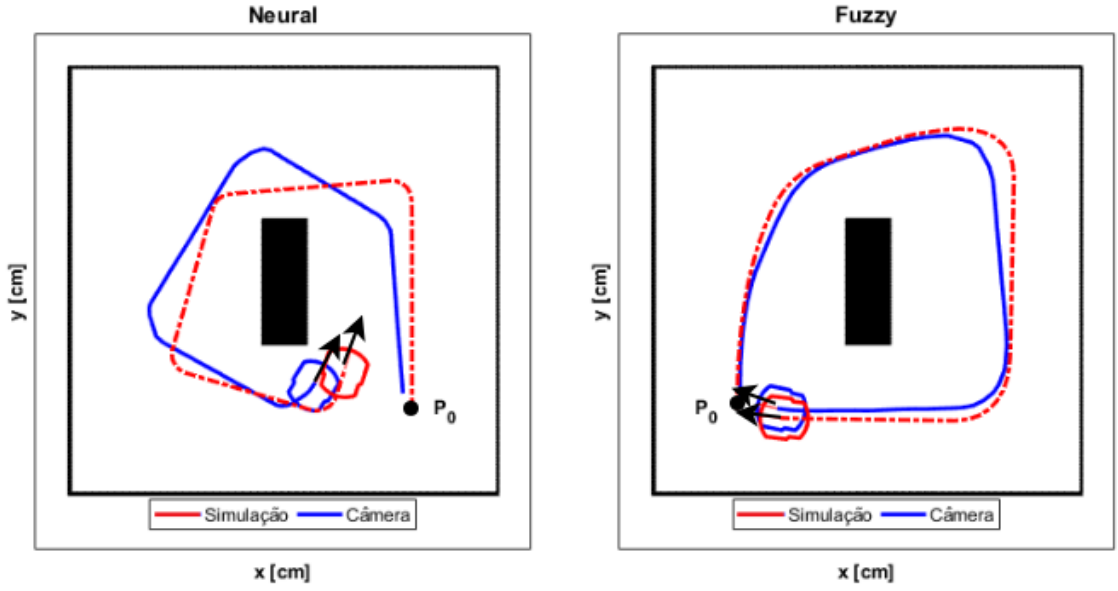
Para organizar e concatenar as informações dos experimentos realizados, será construída uma análise comparativa considerando, primeiramente, as variações percentuais de desempenho com a transferência dos controladores para o ambiente físico e, em complemento, será realizada uma comparação quanto ao desempenho dos controladores no que diz respeito às características evoluídas e apresentadas através das métricas.

Na avaliação das variações percentuais apresentadas nos experimentos, o foco está nas variações das métricas com a transferência dos controladores do ambiente simulado para o ambiente real. Neste sentido, será levado em consideração o módulo da variação, dado que determinada métrica pode apresentar melhor ou pior desempenho com a transferência.



(a) (b)

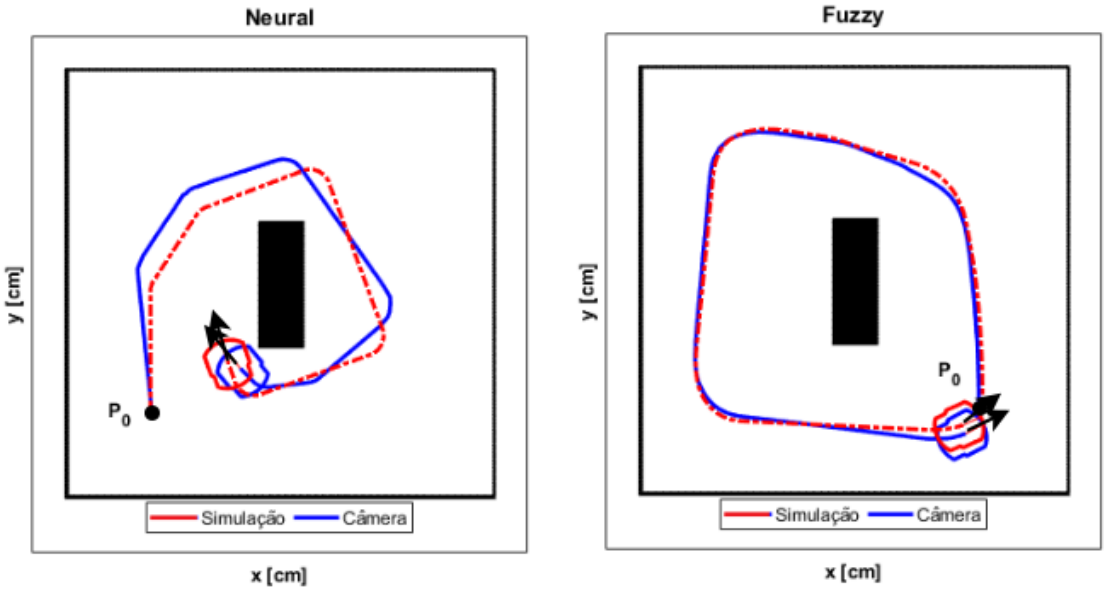
Figura 36 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 30cm/s.



(a) (b)

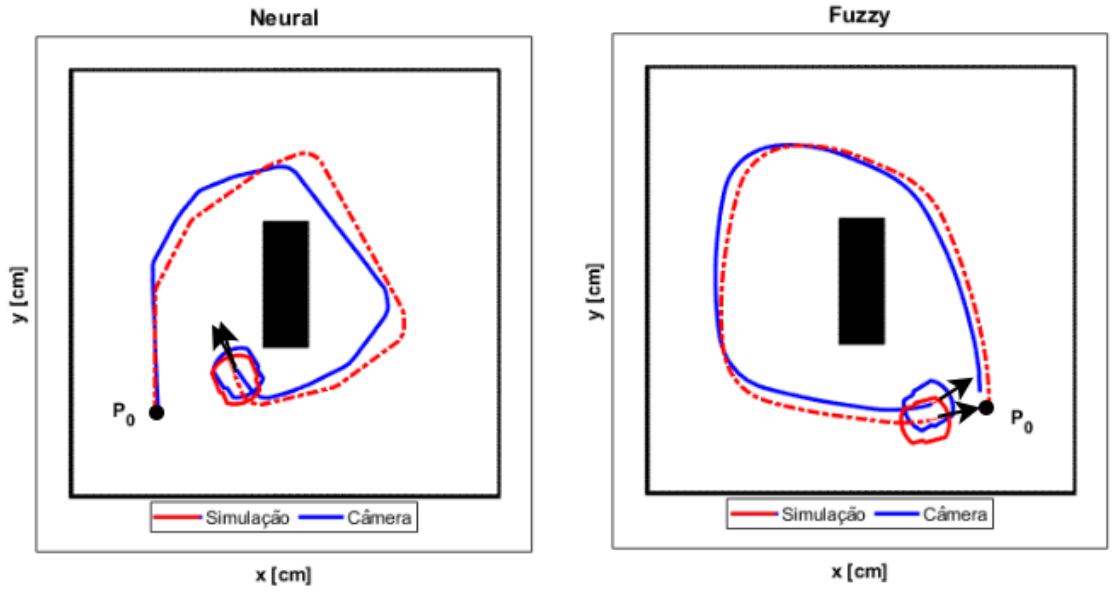
Figura 37 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 25cm/s.

A Tabela 37 apresenta esta comparação. O preenchimento é dado pelas letras F e N, que respectivamente referem-se ao controlador *fuzzy* e neural. Nos casos em que o desempenho seja similar é aplicada a representação N/F. Para cada métrica relacionada a um experimento será indicado qual controlador apresentou menor variação com a transferência. Deste modo, podem ser extraídas da tabela as seguintes conclusões. O controlador



(a) (b)

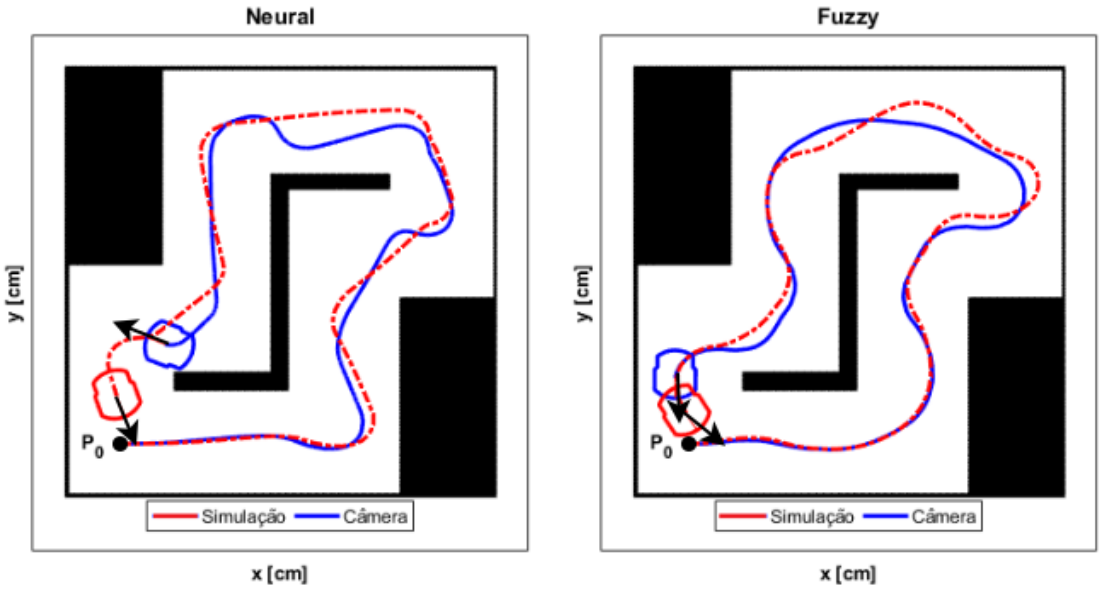
Figura 38 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 20cm/s.



(a) (b)

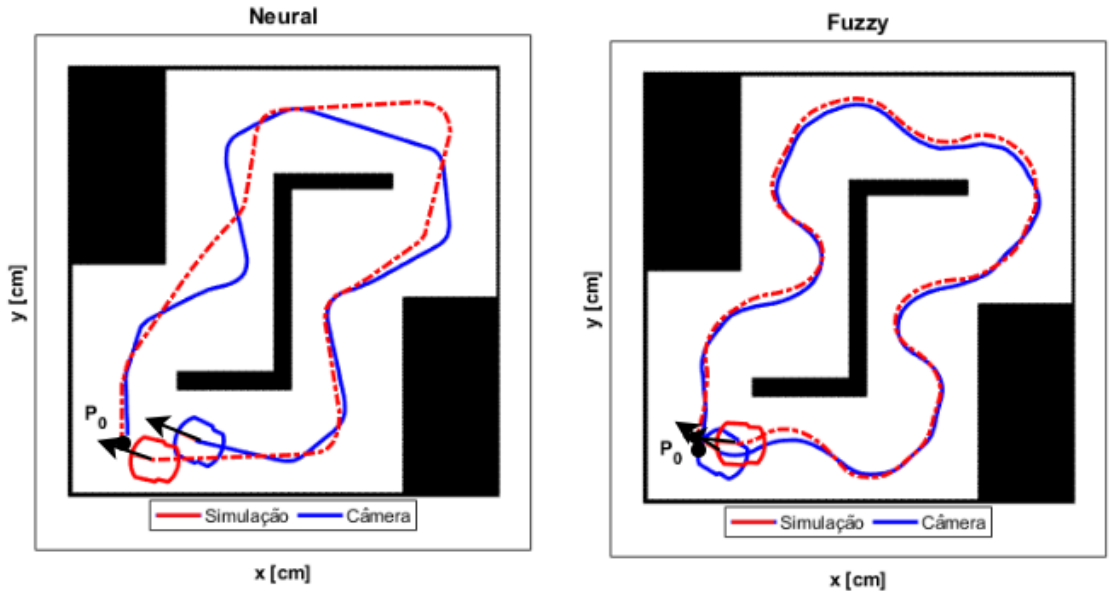
Figura 39 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 15cm/s.

fuzzy apresenta menor variação de desempenho com a transferência para as métricas de *fitness*, MS1, MS2, distância mínima e suavidade. Já o controlador neural apresenta menor variação para a velocidade linear média. Por outro lado, ocorrem desempenhos iguais nos experimentos para as métricas de sucesso, velocidade angular média, distância percorrida e maior distância entre dois pontos. Portanto, é notória a vantagem que o



(a) (b)

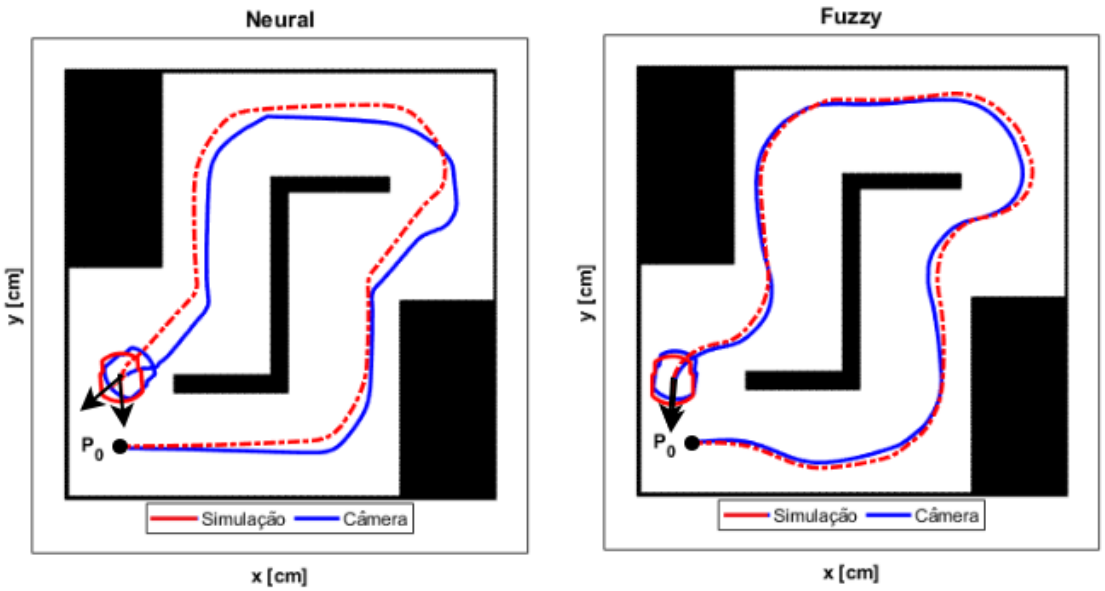
Figura 40 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 30cm/s.



(a) (b)

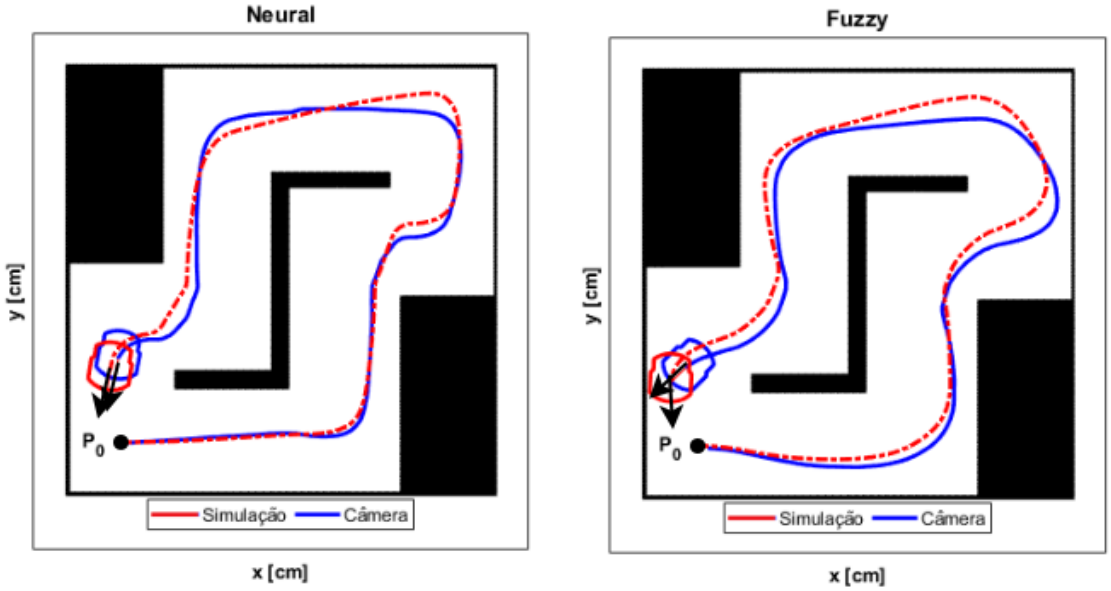
Figura 41 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 25cm/s.

controlador *fuzzy* proporciona quanto às métricas de segurança, se mantendo mais afastado dos obstáculos e também apresentando uma suavidade expressiva em suas ações de controle, característica intrínseca a este tipo de controlador diante das incertezas geradas pela transferência. Esse comportamento é esperado devido a capacidade do controlador *fuzzy* em lidar com incerteza. Por outro lado, o controlador neural tem a capacidade de generalizar qualquer problema em conformidade com a descrição da função de *fitness*, que



(a) (b)

Figura 42 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 20cm/s.



(a) (b)

Figura 43 – (a) Controlador neural simulado e real (câmera), (b) Controlador *fuzzy* simulado e real (câmera) no ambiente 1 com velocidade linear máxima das rodas de 15cm/s.

de acordo com a descrição realizada neste trabalho, sugere maior exploração do ambiente, levando-o a aproximação com os obstáculos.

Ambos os controladores possuem resistência a ruído. Porém, a rede neural funciona como uma função que a partir de uma entrada pontual gera uma saída também pontual, podendo ser interpretada como uma função com domínio e imagem específicas,

enquanto que o controlador *fuzzy* reage sobre determinada região associada aos conjuntos das variáveis *fuzzy*. Em contrapartida, o controlador neural proporciona características ligadas ao comportamento desejado com a função de *fitness* devido a sua capacidade de generalizar a solução do problema, como velocidade linear maior, fato que o torna menos suave em suas ações de controle. Em decorrência de sua velocidade linear maior, o percurso total trafegado é superior ao *fuzzy*, e isso também resulta em uma maior exploração do ambiente, que faz com que o robô se aproxime mais dos obstáculos. Logo, é possível concluir que o controlador *fuzzy* se mostra mais robusto com a transferência se comparado ao controlador neural, levando-se em consideração os controladores simples utilizados no projeto, bem como as condições nas quais os testes foram realizados.

Tabela 37 – Análise dos experimentos com base na variação de desempenho com a transferência.

	Ambiente 1				**Ambiente 2**			
Métricas	**30cm/s**	**25cm/s**	**20cm/s**	**15cm/s**	**30cm/s**	**25cm/s**	**20cm/s**	**15cm/s**
Fitness	F	F	N	F	F	F	F	N
Velocidade linear média	F	N	N	N	F	N	F	N
Velocidade angular média	F	N	N	N	F	F	F	N
Sucesso	N/F	F	N/F	N/F	N/F	N	N/F	N/F
Distância percorrida	F	N	N	N	F	F	F	N
Maior distância entre dois pontos	F	N	N	F	N	N	F	F
MS1	F	F	F	N	N	F	F	N
MS2	F	N	F	N	F	F	F	N
Min	F	F	F	N	N	F	F	F
Suavidade	F	N	F	F	F	F	N	F

De maneira geral controladores neurais têm como característica a necessidade de bastante informação para gerar uma boa sintonia. Deste modo, foi averiguado se as limitações impostas ao controlador neural estavam limitando seu desempenho perante o controlador *fuzzy*. Então foi realizada uma nova busca exaustiva com uma reestruturação do controlador neural. Essa reestruturação objetivou ampliar o tamanho da rede neural inserindo os 8 sensores de distância como entradas no controlador, consequentemente aumentando o tamanho do cromossomo (dobro do anterior). O Apêndice B apresenta esta análise, da qual pode-se concluir que diante dessa reestruturação o controlador mantém características similares quanto ao desempenho já obtido. Vale destacar que no novo controlador reestruturado são necessárias apenas 500 gerações para estabilização do processo evolutivo enquanto no controlador com 3 entradas foram requeridas 900 gerações. Tal fato é justificado pelos pesos resultantes da busca exaustiva serem bem diferentes.

A comparação do desempenho dos controladores, no que diz respeito às características evoluídas e representadas pelas métricas, é realizada por meio da Tabela 38. As características desejáveis e que são pontuadas a favor do controlador são o maior valor de *fitness*, que indica uma avaliação global do comportamento, maior velocidade linear média, menor velocidade angular, que é ponderada negativamente na função de *fitness*, maior taxa de sucesso com a transferência, maior distância percorrida, maior distância entre dois pontos do trajeto, que avalia a exploração do ambiente, e as métricas de se-

gurança, maior MS1 (média entre as leituras dos sensores de todas as amostras), maior MS2 (média das leituras mínimas de todas as amostras), maior distância mínima, e menor suavidade, que avalia a "agressividade"das ações do controlador. O preenchimento leva em consideração a mesma formatação da Tabela 37.

Neste sentido, observa-se que o controlador neural apresentou melhor desempenho para as métricas de *fitness*, velocidade linear média e MS1. Apresentou desempenho similar ao controlador *fuzzy* para as métricas de distância percorrida, MS2 e distância mínima. Já controlador *fuzzy* se destaca nas métricas de velocidade angular média, maior distância entre dois pontos e suavidade. Em relação a taxa de sucesso o controlador neural obteve somente 1 choque com obstáculos (no Ambiente 1) durante todos os experimentos e o controlador *fuzzy* 3 choques (no Ambiente 2), sendo esses choques oriundos de um único experimento (ou seja, de um controlador específico).

Tabela 38 – Análise dos experimentos com base no comportamento evoluído e apresentado nos experimentos em realidade.

	Ambiente 1				**Ambiente 2**			
Métricas	**30cm/s**	**25cm/s**	**20cm/s**	**15cm/s**	**30cm/s**	**25cm/s**	**20cm/s**	**15cm/s**
Fitness	N	N	N	N	N	N	N	N
Velocidade linear média	F	F	N	F	N	N	N	N
Velocidade angular média	F	F	F	F	F	N	F	F
Sucesso	N/F	F	N/F	N/F	N/F	N	N/F	N/F
Distância percorrida	F	F	N	F	N	F	N	N
Maior distância entre dois pontos	N	F	F	F	N	F	F	N
MS1	F	N	N	N	F	F	N	N
MS2	F	F	F	F	N	N	N	N
Min	F	F	F	F	N	N	N	N
Suavidade	N	F	F	F	F	N	F	F

De acordo com este resultado, tem-se que os controladores *fuzzy*, pontua em três das métrica, o controlador neural também e em 4 métricas o desempenho são semelhantes. Portanto, dentro das restrições impostas pela metodologia aplicada, nota-se uma homogeneidade no desempenho das estratégias de controle em estudo. O controlador *fuzzy* trás consigo atributos intrínsecos a essa estratégia de controle, como a suavidade na trajetória do robô.

A semelhança entre os controladores demonstra que o controlador *fuzzy* possui potencial para ser aplicado em robótica evolutiva. Nesse sentido, a depender da necessidade do projeto, estruturas mais elaboradas, com mais conjuntos *fuzzy* e possibilidade do algoritmo genético sintonizar outros parâmetros desse controlador (como funções de pertinência e seu posicionamento, estrutura e operadores das regras, metodologia de fuzzificação e deffuzificação, dentre outras) podem propiciar uma solução ainda melhor.

Em complemento, no Apêndice A, foi realizada uma avaliação quanto a robustez destes controladores quanto a variação da função de *fitness*. Os testes foram realizados sobre os Ambientes 1 e 2 com velocidade linear máxima de $30cm/s$ em ambiente simulado. Como resultado obteve-se que o controlador *fuzzy* se mostrou mais robusto que o

controlador neural.

5 Conclusões

Neste trabalho, foi inicialmente apresentada uma ampla revisão bibliográfica sobre a Robótica Evolutiva. Nesta revisão bibliográfica foram apresentadas as principais características que causam o *reality gap*, como as simplificações de modelo que não englobam características dinâmicas do robô e do ambiente.

Através da revisão bibliográfica foi observado que desde a década de 1990, pesquisas têm investigado o problema da transferência de sistemas de controle de robôs autônomos da simulação para a realidade. Foi verificado que as propostas de solução deste problema são divididas em dois tipos de abordagens: a primeira em que o processo de otimização é realizado unicamente em simulação, e a segunda na qual o processo de otimização é feito parte por simulação e parte em ambiente real.

Em suma, as soluções apresentadas aplicam estruturas de controle neurais integradas a algoritmos genéticos para sua otimização com a função de incorporar inteligência e adaptabilidade aos robôs. O uso de redes neurais é evidenciado na área pela sua facilidade de explorar várias formas de aprendizado com menor influência do projetista e pela sua robustez a ruído nas interações robô-ambiente. Por outro lado, a lógica *fuzzy* também pode ser usada para o desenvolvimento de controladores no âmbito da robótica evolutiva, pois é outra estratégia de controle que possibilita emular a inteligência humana por intermédio da sua descrição baseada na experiência do projetista, além de ser robusta a ruído e incerteza. Até o momento não foi encontrado na literatura uma avaliação quanto ao *reality gap* com a aplicação de controladores *fuzzy*.

Logo, a presente dissertação teve como objetivo investigar se sistemas de controle *fuzzy* são mais robustos que sistemas de controle neural, ambos otimizados por um algoritmo genético em simulação, e posteriormente transferidos para um robô real em ambiente físico. Neste sentido, foi adotado um simulador simples, que contempla atraso cinemático e dinâmica do motor, para o robô Pioneer 3-DX com sensoriamento laser. A tarefa a ser otimizada foi a navegação autônoma evitando obstáculos, que é a mais investigada nesta área.

Foi possível verificar com os experimentos que mesmo sendo utilizado um simulador simples com parametrização fixa, pôde-se obter controladores transferíveis e que conseguiram realizar a tarefa em ambiente real nos dois cenários avaliados.

Diante da transferência para o robô real e dentro das limitações impostas pela simplicidade das estruturas de controle aplicadas no projeto, a estratégia de controle *fuzzy* mostrou-se mais robusta com a transferência se comparada ao controlador neural. Quando comparados em relação ao desempenho, no que diz respeito às características evoluídas

e apresentadas através das métricas, obteve-se uma equivalência entre as estratégias de controle.

Além disso, o controlador *fuzzy* se mostrou mais rápido na estabilização do processo evolutivo, além de apresentar maior suavidade na trajetória do robô em quase todos os cenários analisados, bem como uma menor variação entre a as trajetórias reais e simuladas.

Estas características indicam que controladores *fuzzy* também pode ser aplicados a problemas em robótica evolutiva. De fato, a estratégia de controle *fuzzy* requer e/ou permite maior influência do projetista, como nas definições das funções de pertinência, termos linguísticos e na estrutura das regras. Portanto, permite a inclusão de características do sistema sabidas *a priori*, o que pode direcionar a otimização à solução mais rapidamente. Já o controlador neural não permite esse tipo de interação com o projetista. Porém, a complexidade e falta de conhecimento sobre o sistema torna-se um fator preponderante na escolha da estratégia de controle a ser usada em robótica evolutiva. Neste cenário, uma estrutura de controle baseada em rede neural pode ser indicada. Entretanto, mesmo sob a falta de conhecimento, uma estratégia de controle *fuzzy* pode ser adotada, permitindo ao algoritmo genético sintonizar vários outros parâmetros do mesmo.

Foram realizadas, à parte, duas análises em complemento aos resultados obtidos. Na primeira foi realizada uma avaliação quanto a robustez dos controladores à variação da função de *fitness* em que foi observado que o controlador *fuzzy* se mostrou mais robusto que o controlador neural. Na segunda análise, buscou-se averiguar se as limitações impostas na entrada do controlador neural estavam comprometendo seu desempenho. Então foi realizada uma reestruturação do controlador neural com a ampliação do tamanho da rede neural. Com isso, foi possível concluir que diante dessa reestruturação, o controlador manteve suas características.

Como recomendações para trabalhos futuros, indica-se a exploração de estruturas de controle *fuzzy* e neural mais complexas, dado que as estruturas aplicadas no presente trabalho são bem simples, e uma investigação quanto a variações de parâmetros no modelo do robô e da dinâmica incorporada podem tornar as estratégias de controle mais robustas. Este tipo de abordagem vem sendo explorada recentemente, por exemplo em Milano, Carvalho e Nolfi (2017), Peng et al. (2017), Claes (2017), com a finalidade de minimizar o *reality gap*.

Também percebe-se a necessidade de adequação da metodologia empregada, principalmente na sintonia da função de *fitness*, pois a sua descrição acaba por necessitar de um processo à parte do processo evolutivo para ajuste de parâmetros e também necessitou de ajuste com a mudança de ambiente. Neste sentido, também é proposto o teste da segunda função de *fitness* no primeiro ambiente.

Por outro lado, pretende-se também avaliar os controladores sobre outras tarefas

e também verificar a generalização de controladores evoluídos num ambiente e expostos a outro diferente.

Referências

ABDALLAH, C. et al. Survey of robust control for rigid robots. *IEEE Control Systems*, IEEE, v. 11, n. 2, p. 24–30, 1991. Citado na página 3.

ABRAHAM, A.; NATH, B. Evolutionary design of fuzzy control systems-an hybrid approach. In: *The Sixth International Conference on Control, Automation, Robotics and Vision,(ICARCV 2000), CD-ROM Proceeding, Wang JL (Ed.), ISBN*. [S.l.: s.n.], 2000. v. 1220499864. Citado 2 vezes nas páginas VII e 26.

BABUSKA, R. *Fuzzy modeling for control, international series in intelligent technologies*. [S.l.]: Kluwer Academic Press, 1998. Citado 2 vezes nas páginas 27 e 37.

BERNDT, D. J.; CLIFFORD, J. Using dynamic time warping to find patterns in time series. In: SEATTLE, WA. *KDD workshop*. [S.l.], 1994. v. 10, n. 16, p. 359–370. Citado na página 73.

BOEING, A.; BRÄUNL, T. Leveraging multiple simulators for crossing the reality gap. In: IEEE. *Control Automation Robotics & Vision (ICARCV), 2012 12th International Conference on*. [S.l.], 2012. p. 1113–1119. Citado na página 7.

BRULE, R. The search for a bridge over the reality gap: Effects of interleaving interval duration and quantity. 2008. Citado na página 10.

CAPPELLE, C.; BERNATSKIY, A.; BONGARD, J. Reducing training environments in evolutionary robotics through ecological modularity. In: SPRINGER. *Conference on Biomimetic and Biohybrid Systems*. [S.l.], 2017. p. 95–106. Citado na página 7.

CLAES, S. Transferring learned control from simulation to real. 2017. Citado 3 vezes nas páginas 8, 17 e 83.

CORDÓN, O. et al. Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy sets and systems*, Elsevier, v. 141, n. 1, p. 5–31, 2004. Citado na página 13.

COUCEIRO, M. S.; VARGAS, P. A.; ROCHA, R. P. Bridging the reality gap between the webots simulator and e-puck robots. *Robotics and Autonomous Systems*, Elsevier, v. 62, n. 10, p. 1549–1567, 2014. Citado na página 11.

DONCIEUX, S. et al. Evolutionary robotics: what, why, and where to. *Frontiers in Robotics and AI*, Frontiers, v. 2, p. 4, 2015. Citado 2 vezes nas páginas VII e 19.

DULLERUD, G. E.; PAGANINI, F. *A course in robust control theory: a convex approach*. [S.l.]: Springer Science & Business Media, 2013. Citado 4 vezes nas páginas 1, 2, 3 e 4.

FLOREANO, D.; MONDADA, F. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, IEEE, v. 26, n. 3, p. 396–407, 1996. Citado 5 vezes nas páginas 6, 35, 39, 43 e 44.

FREIRE, E. O. et al. Human based benchmark for robot navigation assessment. In: *Proceedings of the ISSNIP Biosignals and Biorobotics Conference*. [S.l.: s.n.], 2010. Citado na página 41.

GOMIDE, F.; GUDWIN, R. R.; TANSCHEIT, R. Conceitos fundamentais da teoria de conjuntos fuzzy, lógica fuzzy e aplicações. In: *Proc. 6 th IFSA Congress-Tutorials*. [S.l.: s.n.], 1995. p. 1–38. Citado 4 vezes nas páginas VII, 11, 24 e 25.

HALAL, F.; DUMITRACHE, I. Genetic optimization of fuzzy controller for mobile robots. In: *Proceedings of the 16 th International Conference on Control Systems and Computer Science CSCS16, Bucharest*. [S.l.: s.n.], 2007. p. 433–438. Citado na página 13.

HANNA, J. P. Bridging the gap between simulation and reality. In: INTERNATIONAL FOUNDATION FOR AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS. *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. [S.l.], 2017. p. 1834–1835. Citado na página 9.

HAYKIN, S. S. *Redes neurais*. [S.l.]: Bookman, 2001. Citado 4 vezes nas páginas VII, 22, 23 e 24.

HOFFMANN, F. Evolutionary algorithms for fuzzy control system design. *Proceedings of the IEEE*, IEEE, v. 89, n. 9, p. 1318–1333, 2001. Citado 2 vezes nas páginas 14 e 27.

HOFFMANN, F.; PFISTER, G. Evolutionary learning of a fuzzy control rule base for an autonomous vehicle. In: *Proceedings of the Fifth International Conference IPMU: Information Processing and Management of Uncertainty in Knowledge-Based Systems, Granada, Spain*. [S.l.: s.n.], 1996. p. 659–664. Citado 2 vezes nas páginas 14 e 27.

HOLLAND, J. H. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press, 1975. Citado na página 18.

INOUE, R. *Controle robusto de robôs móveis com rodas*. Tese (Doutorado) — Master's thesis, Universidade de São Paulo, 2007. Citado na página 5.

JAKOBI, N.; HUSBANDS, P.; HARVEY, I. Noise and the reality gap: The use of simulation in evolutionary robotics. In: SPRINGER. *European Conference on Artificial Life*. [S.l.], 1995. p. 704–720. Citado na página 6.

JAKOBI, N. et al. The minimal simulation approach to evolutionary robotics. *Proceedings of ER*, Citeseer, v. 98, 1998. Citado 2 vezes nas páginas 1 e 6.

KIM, S. H.; PARK, C.; HARASHIMA, F. A self-organized fuzzy controller for wheeled mobile robot using an evolutionary algorithm. *IEEE Transactions on Industrial Electronics*, IEEE, v. 48, n. 2, p. 467–474, 2001. Citado na página 13.

KONG, S.-G.; KOSKO, B. Comparison of fuzzy and neural truck backer-upper control systems. In: IEEE. *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*. [S.l.], 1990. p. 349–358. Citado 4 vezes nas páginas VII, 11, 12 e 13.

KOOS, S.; MOURET, J.-B.; DONCIEUX, S. The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 17, n. 1, p. 122–145, 2013. Citado 3 vezes nas páginas 1, 2 e 4.

LIGOT, A.; BIRATTARI, M. On mimicking the effects of the reality gap with simulation-only experiments. 2018. Citado na página 7.

MAMDANI, E. H.; ASSILIAN, S. An experiment in linguistic synthesis with a fuzzy logic controller. *International journal of man-machine studies*, Elsevier, v. 7, n. 1, p. 1–13, 1975. Citado na página 26.

MATELLÁN, V.; FERNÁNDEZ, C.; MOLINA, J. Genetic learning of fuzzy reactive controllers. *Robotics and Autonomous Systems*, Elsevier, v. 25, n. 1-2, p. 33–41, 1998. Citado 2 vezes nas páginas 13 e 44.

MIGLINO, O.; LUND, H. H.; NOLFI, S. Evolving mobile robots in simulated and real environments. *Artificial life*, MIT Press, v. 2, n. 4, p. 417–434, 1995. Citado na página 10.

MILANO, N.; CARVALHO, J. T.; NOLFI, S. Moderate environmental variation promotes adaptation in artificial evolution. *arXiv preprint arXiv:1710.07913*, 2017. Citado 3 vezes nas páginas 8, 17 e 83.

MONTANA, D. J.; DAVIS, L. Training feedforward neural networks using genetic algorithms. In: *IJCAI*. [S.l.: s.n.], 1989. v. 89, p. 762–767. Citado 2 vezes nas páginas 21 e 39.

MOURET, J.-B.; CHATZILYGEROUDIS, K. 20 years of reality gap: a few thoughts about simulators in evolutionary robotics. In: *Workshop"Simulation in Evolutionary Robotics", Genetic and Evolutionary Computation Conference*. [S.l.: s.n.], 2017. Citado 3 vezes nas páginas 5, 29 e 31.

MOURET, J.-B.; KOOS, S.; DONCIEUX, S. Crossing the reality gap: a short introduction to the transferability approach. *arXiv preprint arXiv:1307.1870*, 2013. Citado na página 10.

MUÑOZ, N.; VALENCIA, J.; LONDONO, N. Evaluation of navigation of an autonomous mobile robot. In: ACM. *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems*. [S.l.], 2007. p. 15–21. Citado na página 41.

NELSON, A. L.; BARLOW, G. J.; DOITSIDIS, L. Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, Elsevier, v. 57, n. 4, p. 345–370, 2009. Citado 4 vezes nas páginas 11, 28, 44 e 91.

NOLFI, S. et al. How to evolve autonomous robots: Different approaches in evolutionary robotics. In: MA: MIT PRESS. *Artificial life IV: Proceedings of the 4th International Workshop on Artificial Life*. [S.l.], 1994. p. 190–197. Citado na página 11.

PENG, X. B. et al. Sim-to-real transfer of robotic control with dynamics randomization. *arXiv preprint arXiv:1710.06537*, 2017. Citado 3 vezes nas páginas 8, 17 e 83.

PEREZ, A. L. F. Robótica evolutiva. *IX Escolha Regional de Computação Bahia-Alagoas-Sergipe (ERBASE). Universidade Estadual de Santa Cruz, Ilhéus, BA*, 2009. Citado 4 vezes nas páginas 1, 2, 18 e 91.

PESSIN, G.; OSÓRIO, F.; MUSSE, S. Utilizando redes neurais artificiais no controle robusto de navegação de robôs móveis. *IN Anais da Feira Mato Grosso Digital*, 2008. Citado 3 vezes nas páginas 2, 4 e 5.

RIEFFEL, J. et al. *Introduction to the evolution of physical systems special issue*. [S.l.]: MIT Press, 2017. Citado na página 6.

RUUD, E.-L.; SAMUELSEN, E.; GLETTE, K. Memetic robot control evolution and adaption to reality. In: IEEE. *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*. [S.l.], 2016. p. 1–7. Citado na página 9.

SCHEPER, K. Y.; CROON, G. C. de. Abstraction as a mechanism to cross the reality gap in evolutionary robotics. In: SPRINGER. *International Conference on Simulation of Adaptive Behavior*. [S.l.], 2016. p. 280–292. Citado 2 vezes nas páginas 7 e 18.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. *Introduction to autonomous mobile robots*. [S.l.]: MIT press, 2011. Citado 3 vezes nas páginas VII, 16 e 17.

SILVA, A. J. M. *Implementaçao de um Algoritmo Genético utilizando o modelo de ilhas*. Tese (Doutorado) — UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, 2005. Citado 3 vezes nas páginas VII, 18 e 20.

SILVA, F. et al. Open issues in evolutionary robotics. *Evolutionary computation*, MIT Press, v. 24, n. 2, p. 205–236, 2016. Citado 2 vezes nas páginas 1 e 2.

SIMOES, M. G.; SHAW, I. S. Controle e modelagem fuzzy. *Blucher: FAPESP, São Paulo, BRA*, 2007. Citado 2 vezes nas páginas 25 e 26.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its applications to modeling and control. *IEEE transactions on systems, man, and cybernetics*, IEEE, n. 1, p. 116–132, 1985. Citado 2 vezes nas páginas 26 e 37.

TOBIN, J. et al. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*, 2017. Citado na página 9.

TREMBLAY, J. et al. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. *arXiv preprint arXiv:1804.06516*, 2018. Citado na página 6.

WONG, S. C. et al. Performance metrics for robot coverage tasks. In: *Proceedings of Australasian Conference on Robotics and Automation*. [S.l.: s.n.], 2002. v. 27, p. 29. Citado na página 41.

YAMAMOTO, M. M. et al. Um simulador dinâmico para mini-robôs móveis com modelagem de colisoes. *SBAI 2003-Simpósio Brasileiro de Automaçao Inteligente, Bauru SP, Brazil*, p. 852–857, 2003. Citado na página 4.

ZADEH, L. A. Fuzzy sets. *Information and control*, Elsevier, v. 8, n. 3, p. 338–353, 1965. Citado na página 24.

ZAGAL, J. C.; SOLAR, J. Ruiz-del; VALLEJOS, P. Back to reality: Crossing the reality gap in evolutionary robotics. In: *IAV 2004 the 5th IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal*. [S.l.: s.n.], 2004. Citado na página 10.

ZHOU, K.; DOYLE, J. C. *Essentials of robust control*. [S.l.]: Prentice hall Upper Saddle River, NJ, 1998. Citado 3 vezes nas páginas 2, 3 e 17.

ZUBEN, F. J. V. Computação evolutiva: uma abordagem pragmática. *Anais da I Jornada de Estudos em Computação de Piracicaba e Região (1a JECOMP)*, v. 1, p. 25–45, 2000. Citado 4 vezes nas páginas VII, 18, 19 e 21.

Apêndices

APÊNDICE A – Robustez quanto a variações na função de *fitness*

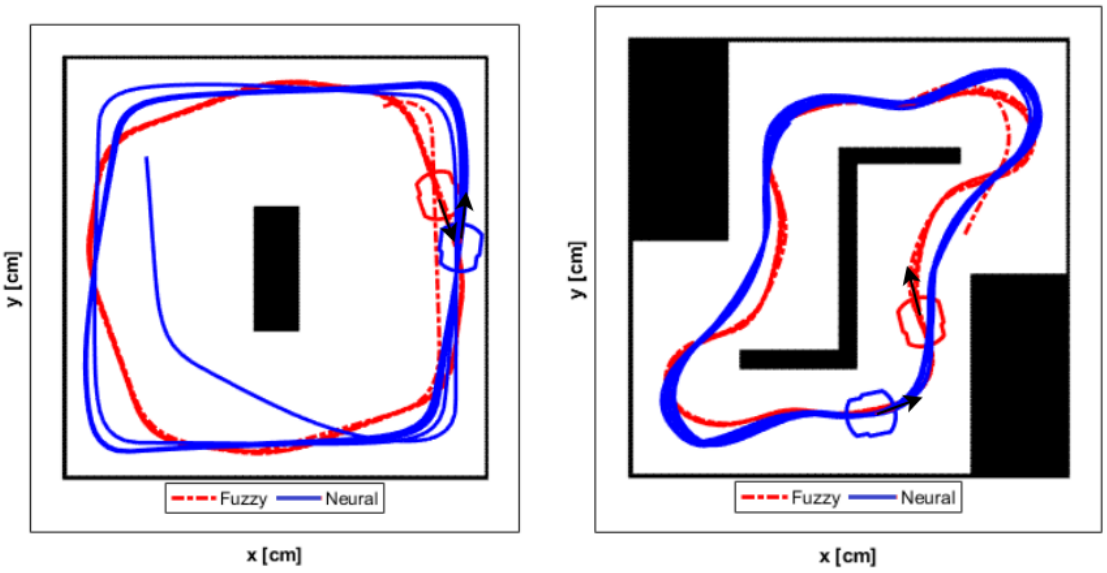
A definição da função de avaliação ou *fitness* é muito importante em um algoritmo evolutivo, pois é responsável por determinar o quão apto ou próximo da solução do problema o indivíduo está (PEREZ, 2009). O resultado do processo evolutivo depende muito do formato da função de avaliação (NELSON; BARLOW; DOITSIDIS, 2009). Neste sentido, foram realizados alguns experimentos (processos evolutivos) a fim de se avaliar a robustez dos controladores quanto a variações da função *fitness*. Os testes foram realizados sobre os Ambientes 1 e 2 com velocidade linear máxima de $30cm/s$ em simulação. Ao todo, foram testadas 5 variações baseadas na função de *fitness* adotada nos experimentos anteriores.

Os valores atribuídos às constantes presentes nas funções de *fitness* (ou seja, Kv, $K\omega$, K_d) variam entre $0,0$, $0,5$ ou $1,0$ e pôde ser verificado que essa atribuição contribuiu para que os controladores evoluídos atingissem o comportamento requerido, ou seja, navegassem sem se chocar com obstáculos, tanto para o controlador *fuzzy* quanto para o controlador neural, nos dois ambientes em questão.

A *fitness* F_1 (Equação A.1) apresenta a constante $K_v = 1$ que é responsável pela ponderação de velocidade linear na primeira parcela. As demais constantes foram atribuídos $K\omega = 0,5$ e $K_d = 0,5$. Com isso, pode ser verificado na Figura 44, que os controladores conseguiram executar a tarefa, sendo que no Ambiente 1 (Figura 44(a)) a parcela da *fitness* que contribui para exploração do ambiente leva os controladores a se comportarem de forma similar a controladores seguidores de parede.

$$F_1(v_n, \omega_n, d_n, Pos, mapa) = \frac{1}{2}\left(\frac{\sum_{n=1}^{n_a}\left[\frac{v_n}{v_{max}} - K_\omega\left(\frac{|\omega_n|}{\omega_{max}}\right) - K_d d_n\right]}{t_{lim}/t_s} + d_t\ d_{\Delta s}\right) \quad \text{(A.1)}$$

Na *fitness* F_2 (Equação A.2) foi atribuída a todas constantes referentes a primeira parcela (ou seja, Kv, $K\omega$, K_d) valores iguais a 1, pois sendo a velocidade linear diretamente relacionada com a velocidade angular, deve haver uma relação de compromisso entre elas e a distância ao obstáculo para que o indivíduo ou controlador seja bem avaliado na execução da tarefa, independentemente dos pesos de cada componente. Diante deste fato, o controlador neural no Ambiente 1 foi quem apresentou um comportamento ruim

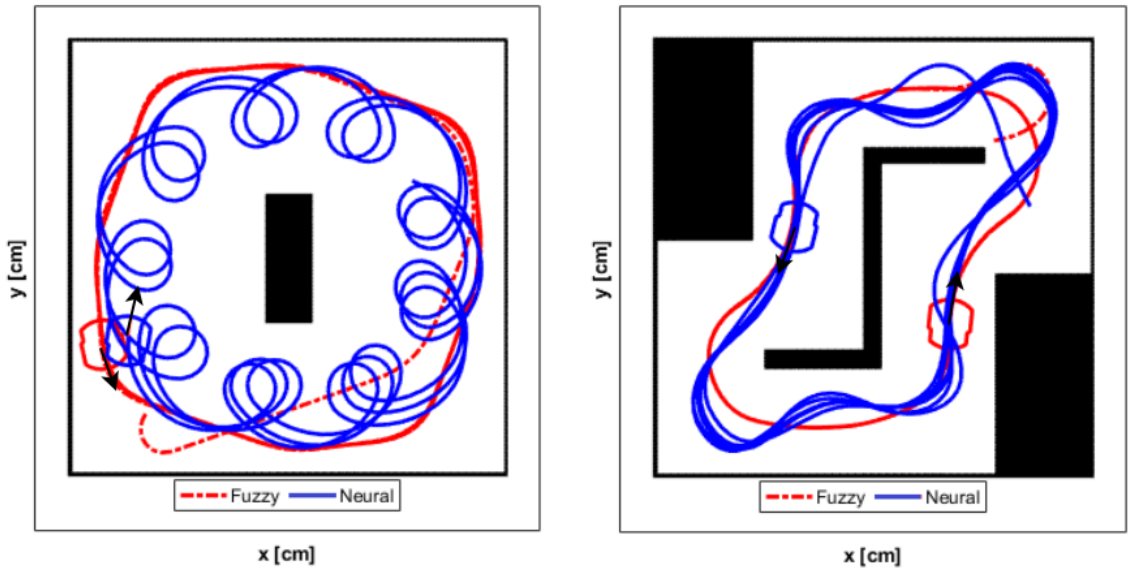


(a) (b)

Figura 44 – (a) *Fitness* 1 no Ambiente 1 simulado, e (b) *Fitness* 1 no Ambiente 2 simulado.

(Figura 45(a)), girando enquanto executava a trajetória.

$$F_2(v_n, \omega_n, d_n, Pos, mapa) = \frac{1}{2}\left(\frac{\sum_{n=1}^{n_a}\left[\frac{v_n}{v_{max}} - \frac{|\omega_n|}{\omega_{max}} - d_n\right]}{t_{lim}/t_s} + d_t\ d_{\Delta s}\right) \tag{A.2}$$



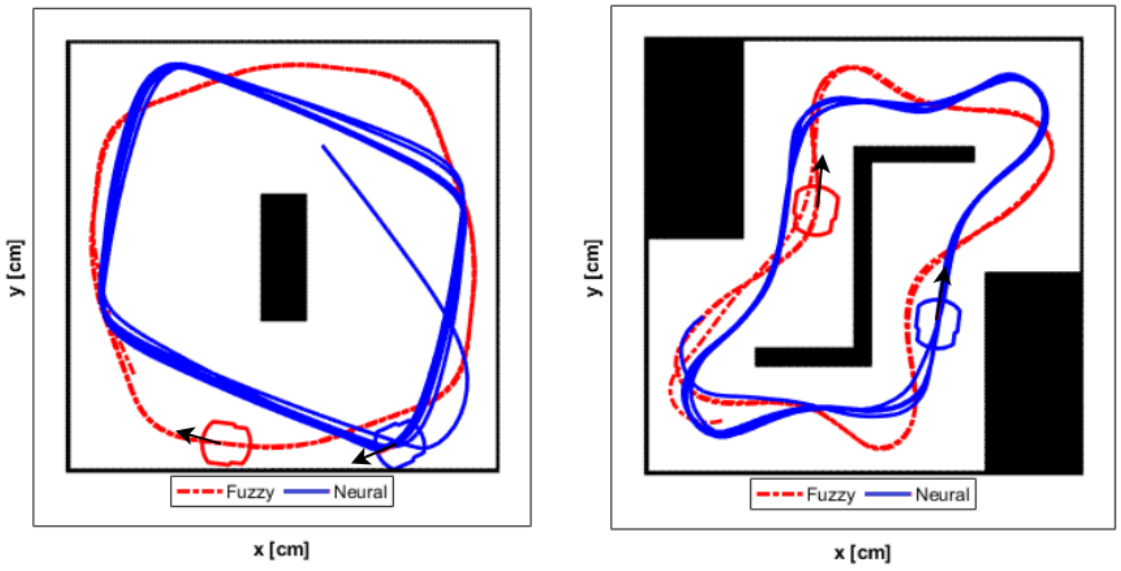
(a) (b)

Figura 45 – (a) *Fitness* 2 no Ambiente 1 simulado, e (b) *Fitness* 2 no Ambiente 2 simulado.

Na *fitness* F_3 (Equação A.3) foi removida a distância percorrida (d_t) da segunda parcela, pois esta componente está diretamente atrelada à velocidade linear, também

contemplada na primeira parcela, e nela as constantes $K_\omega = 0,5$ e $K_d = 0,5$ retornam à equação, além de $K_v = 1$ e $K = 0,5$. Como resultado do processo evolutivo (Figura 46), tem-se que nos dois ambientes os robôs conseguiram evoluir o comportamento desejado, porém o controlador *fuzzy* apresenta uma trajetória mais adequada.

$$F_3(v_n, \omega_n, d_n, Pos, mapa) = (1 - K)\left(\frac{\sum_{n=1}^{n_a}\left[\left(\frac{v_n}{v_{max}}\right) - K_\omega\left(\frac{|\omega_n|}{\omega_{max}}\right) - K_d d_n\right]}{t_{lim}/t_s}\right) + K\ d_{\Delta s} \tag{A.3}$$



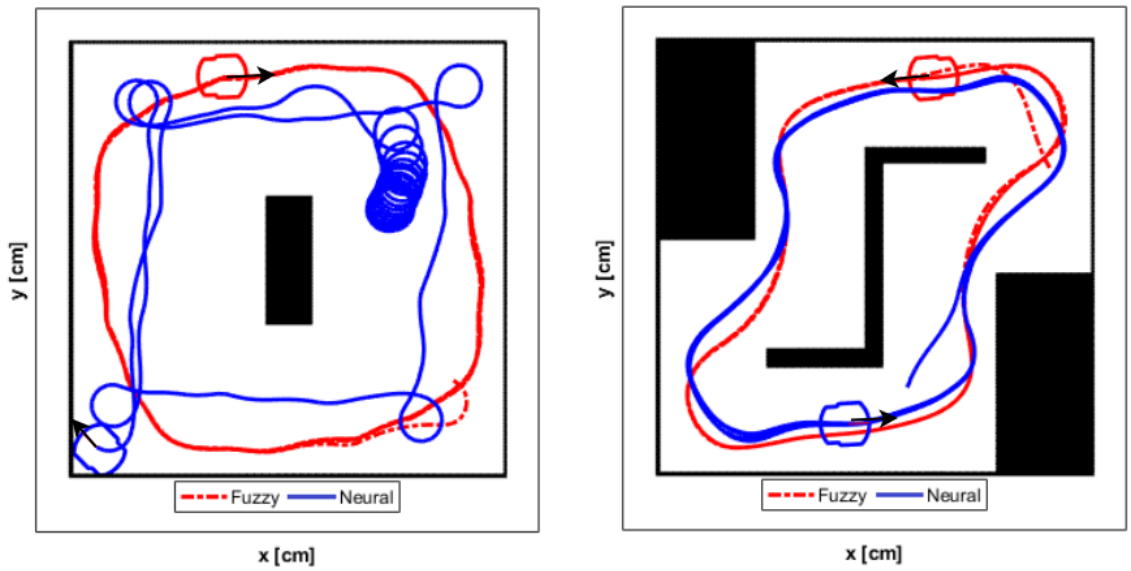
(a) (b)

Figura 46 – (a) *Fitness* 3 no Ambiente 1 simulado, e (b) *Fitness* 3 no Ambiente 2 simulado.

A *fitness* F_4 (Equação A.4) é idêntica à F_3, sendo atribuído às constantes K_v K_ω, K_d o valor 1 e $K = 0,5$, permitindo ao processo evolutivo atingir o comportamento desejado. A Figura 47 apresenta o desempenho atingido com o processo evolutivo no qual pode ser verificado que no Ambiente 1 (Figura 47(a)), o controlador neural apresentou um comportamento ruim, girando em alguns momentos.

$$F_4(v_n, \omega_n, d_n, Pos, mapa) = (1 - K)\left(\frac{\sum_{n=1}^{n_a}\left[\left(\frac{v_n}{v_{max}}\right) - \frac{|\omega_n|}{\omega_{max}} - d_n\right]}{t_{lim}/t_s}\right) + K\ d_{\Delta s} \tag{A.4}$$

E por fim, utilizou-se a segunda parcela da *fitness* como uma nova função de avaliação F_5, Equação A.5. Como resultado, Figura 48, pode ser verificado que os dois controladores conseguiram evoluir o comportamento requerido em projeto, porém, esta função de *fitness* resulta em uma trajetória semelhante à apresentada pelo robô no Ambiente 1 (Figura 48(a)), de um controlador seguidor de parede, justamente por requerer

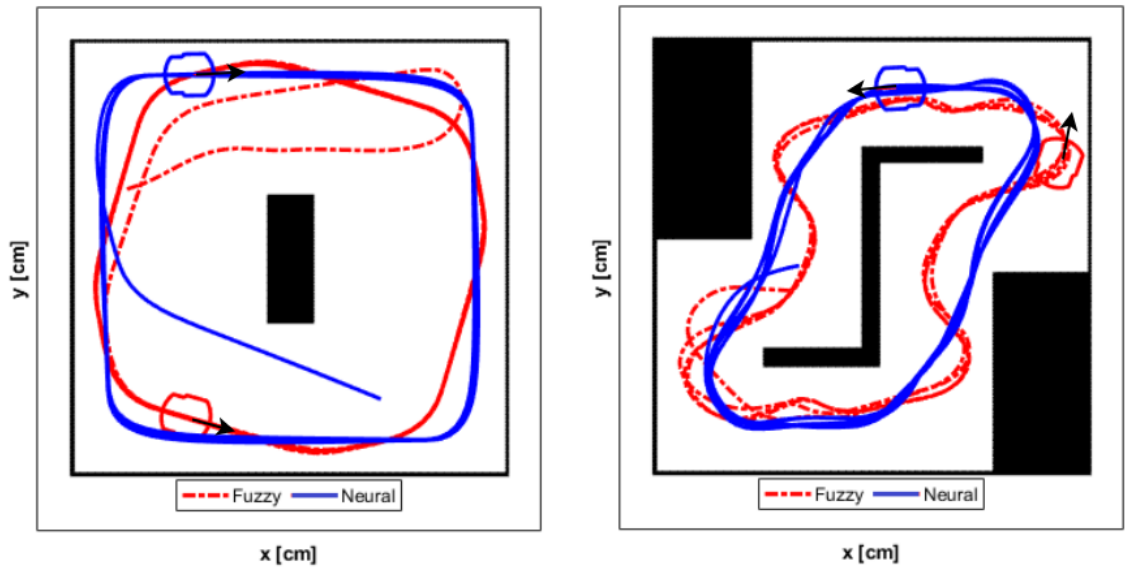


(a) (b)

Figura 47 – (a) *Fitenss* 4 no Ambiente 1 simulado, e (b) *Fitness* 4 no Ambiente 2 simulado.

a exploração do ambiente. O comportamento no Ambiente 2 dos controladores retrata a forte dependência da segunda parcela da *fitness* utilizada no projeto do controlador.

$$F_5(Pos, mapa) = d_t \; d_{\Delta s} \tag{A.5}$$



(a) (b)

Figura 48 – (a) *Fitness* 5 no Ambiente 1 simulado, e (b) *Fitness* 5 no Ambiente 2 simulado.

Com o objetivo de averiguar a aplicabilidade da função de *fitness* definida para o Ambiente 2 (Equação 4.11) no Ambiente 1, foi realizada uma busca exaustiva das suas constantes (K_v K_ω e K_d). Como resultado, notou-se que a segunda parcela desta *fitness*,

referente à componente do produto entre a exploração do ambiente e a distância percorrida entrava em conflito com o objetivo de andar afastado dos obstáculos, de maneira similar aos testes descritos acima. Portanto, é notória a necessidade de buscar os parâmetros que ponderam cada parcela desta *fitness* para assim melhor avaliar os indivíduos. Esta abordagem poderá ser avaliada em trabalhos futuros.

Portanto, com estes experimentos realizados com variações na função de *fitness* é possível verificar que o controlador *fuzzy* é mais robusto sob esse aspecto. E, como pôde ser visto, em todas as configurações analisadas este controlador gerou uma trajetória mais suave que o controlador neural, levando-se em consideração os dois ambientes.

APÊNDICE B – Controlador Neural Estendido

O controlador neural proposto no trabalho levou em consideração três entradas com base no agrupamentos dos 8 sensores disponíveis. Com o objetivo de averiguar se esta limitação no número de entradas afetou seu desempenho, já que limita a quantidade de informação passada ao mesmo, foi proposta a adoção dos 8 sensores como entradas do controlador (vide Figura 49). Nesta avaliação, foi realizada uma busca exaustiva com o limite de velocidade linear máxima de $30cm/s$ no Ambiente 2 e, posteriormente, escolhidas as constantes da *fitness*. Este procedimento foi o mesmo aplicado ao controlador com 3 entradas. De posse das constantes, foi evoluído um controlador com este nível de velocidade e realizados experimentos em ambiente real.

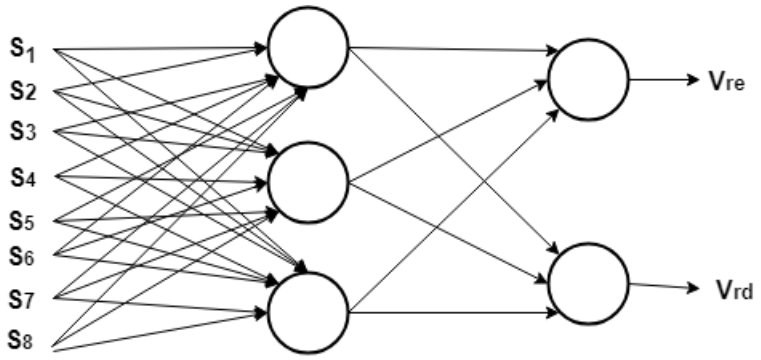


Figura 49 – Controlador Neural Estendido.

A Figura 50 apresenta a evolução da *fitness* para o controlador. Os parâmetros utilizados no processo evolutivo são apresentados na Tabela 39.

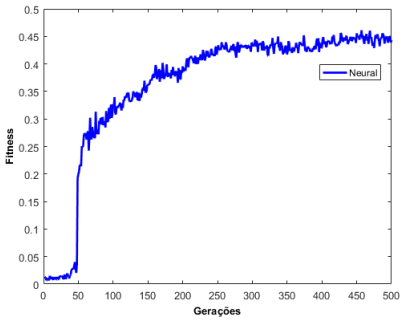


Figura 50 – Processo evolutivo neural do ambiente 2 com velocidade máxima de 30cm/s.

Tabela 39 – Parâmetros do algoritmo genético do controlador neural aplicado ao ambiente 2 com velocidade máxima de 30cm/s.

Parâmetros	Valor
Tamanho da população	100
Número de gerações	500
Comprimento do cromossomo	30
Taxa de *crossover*	0,2
Taxa de mutação	0,3
Taxa de elitismo	0,1
Range de mutação	±0,5 (Neural) / ±0,4 (Fuzzy)
d_n	$\mu = 10,\ \sigma = 20$
K_v	0,6
K_ω	0,8
K_d	0,4
Posição inicial	aleatória

Posteriormente à evolução, o melhor indivíduo da última geração foi transferido para o ambiente real e 10 experimentos foram realizados sob condições iniciais similares aos experimentos anteriores, ou seja, o robô com 100% da bateria carregada e pose inicial P_0 (0,5m; 0,5m; 0°). Na Figura 51 tem-se o resultado deste controlador em simulação e em ambiente real para um dos experimentos de sucesso dos 10 realizados, já que houve alguns choques com obstáculos em alguns deles. Sua trajetória se apresenta em sentido anti-horário. É notória nesta ilustração as diferenças nos trajetos com a transferência para o ambiente real, destacando uma fragilidade deste controlador. Os pesos obtidos para este controlador são apresentados em (B.1) e (B.2).

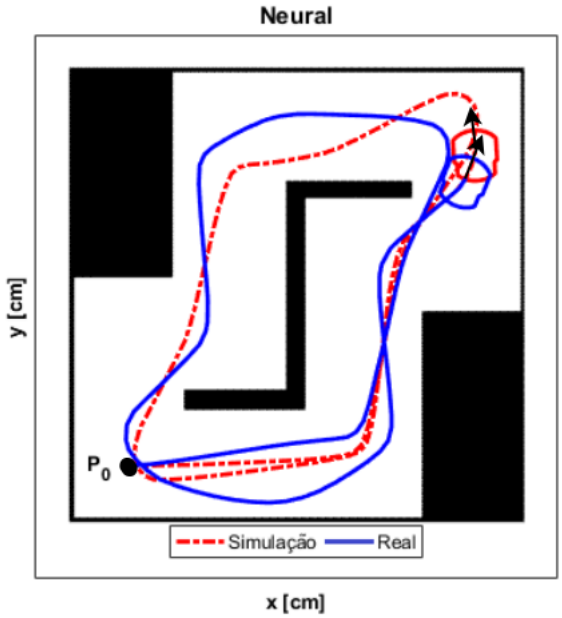


Figura 51 – Controlador neural simulado e real no ambiente 2 com velocidade linear máxima das rodas de 30cm/s.

$$\mathbf{W_1} = \begin{bmatrix} -2,9897 & -4,7212 & -0,0969 & -2,4893 & 5,1966 & 5,0344 & -2,2166 & 4,2614 \\ 4,6583 & 1,6729 & -0,5607 & 6,6743 & 1,3180 & -7,4516 & -6,5733 & -1,8401 \\ -1,5803 & -1,8077 & 1,1989 & 6,4465 & 2,6455 & 5,7091 & -6,2102 & -5,3541 \end{bmatrix} \tag{B.1}$$

$$\mathbf{W_2} = \begin{bmatrix} -3,9035 & 5,4242 & 5,6900 \\ 3,0256 & -5,7584 & 7,2953 \end{bmatrix} \tag{B.2}$$

A Tabela 40 traz o desempenho do controlador com base nas métricas definidas perante a transferência ao ambiente real. Comparando este controlador com a estrutura neural anterior, aplicada a este mesmo nível de velocidade linear e apresentada na Tabela 21, percebe-se que com o aumento das entradas houve uma maior diferença percentual com a transferência. Anteriormente, tinha-se 3 entradas formadas por conjuntos de sensores que forneciam a leitura utilizada pelo controlador com base na menor leitura entre os

sensores do conjunto, isso torna o controle mais agressivo, e para contornar essa característica, o controlador evolui uma velocidade menor. No caso do controlador com 8 entradas, o processo evolutivo resulta em uma velocidade linear maior, resultado dos pesos obtidos com a busca exaustiva para velocidade linear da *fitness*, que por consequência aumenta a manifestação de efeitos dinâmicos e, consequentemente, o risco de choques com obstáculos. Contudo, este novo controlador resultou numa trajetória mais suave, porém com distância mínima para o obstáculo menor. Com este novo controlador foram necessárias apenas 500 gerações para estabilização do processo evolutivo enquanto no controlador com 3 entradas foram requeridas 900 gerações. Tal fato é justificado pelos pesos resultantes da busca exaustiva serem distintos, inflenciando na dinâmica do processo evolutivo.Vale destacar que dos 10 experimentos realizados em ambiente real, 7 executaram o trajeto sem se chocar com obstáculos. Foi verificado que no caso dos insucessos houve uma derrapagem na partida do robô, ocasionada pelo nível médio de velocidade linear que é elevado. Tal derrapagem tirou ele de sua trajetória normal e com isso não foi possível corrigir completamente seu trajeto, resultando no choque.

Tabela 40 – Neural - melhor indivíduo no ambiente 2 com velocidade linear máxima das rodas de $30 cm/s$

Métricas - Controlador Neural	Simulação	Real (média) e Desvio Padrão	Diferença (%)
Fitness	0,4303	0,3824 ± 0,0062	11,1304
Velocidade linear média (cm/s)	25,6657	25,4137 ± 0,0622	0,9818
Velocidade angular média (rad/s)	0,1951	0,2022 ± 0,0033	-3,6503
Sucesso	1	7	-
Distância percorrida (cm)	1539,9429	1517,9160 ± 4,9049	1,4304
Maior distância entre dois pontos (cm)	442,6075	408,3310 ± 3,3644	7,7442
MS1 (cm)	94,1204	91,9383 ± 0,7826	2,3184
MS2 (cm)	28,8494	27,2654 ± 1,0362	5,4905
Min (cm)	1,5762	4,8429 ± 2,2082	-207,2489
Suavidade (°)	9,7757	10,4517 ± 0,1444	-6,9153