IMPLEMENTAÇÃO DE UMA REDE NEURAL ARTIFICIAL EM AMBIENTE SIMULADO PARA O CONTROLE DE TRAJETÓRIA DE UM ROBÔ MÓVEL COM A ARQUITETURA ACKERMANN.

ARTUR F. MOREIRA[1], PEDRO H. A. MIRANDA[2], ALAN V. DE A. BATISTA[1], ANDRESSA DA S. FERNANDES[3], ANTÔNIO R. XAVIER[4]

[1]*Instituto Federal de Ciências, Educação e Tecnologia do Ceará (IFCE) – Alameda José Quintino, s/n, Prado, CEP 63400-000, Tel.: +55 (88) 3564-1000 – Cedro – CE*

[2]UNIVERSIDADE FEDERAL DO CEARÁ (UFC) – AV. DA UNIVERSIDADE, 2853, BENFICA, CEP 60020-181, TEL.: *+55 (85) 3366 7300* – FORTALEZA – CE

[3]*Universidade Federal do Ceará (UFC) – Rua Coronel Estanislau Frota, 563, Centro, CEP 62010-560, Tel.: +55 (88) 3695 4630 – Sobral – CE*

[4]*Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB) – Av. João da Mata, 256, Jaguaribe, CEP 58015-020, Tel.: +55 (83) 3612-9701– João Pessoa – PB*

E-mails: artur31415926@gmail.com, pedrohenriqbg@gmail.com, alan.ifce@gmail.com, andressafernandes06.af@gmail.com, xavierconect@gmail.com

Abstract— Advances in the field of artificial intelligence in the last decade have provided one of the most interesting times so far in the field of robotics, especially in mobile robotics. One of the factors that contributed to this phenomenon is the complexity involved in the control of the mobile robot during the autonomous navigation of more complex situations, which requires a controller that can generalize and have a good success rate in unprecedented situations. The present work aims to implement, in a simulated environment, an artificial neural network as the controller of the mobile robot used in the NXP Cup. From the data gathered from the simulated experiments it can be concluded that the trained artificial neural network was able to control the mobile robot in an unknown track without letting the mobile robot leave the route, requiring few adjustments of its synaptic weights. Implementação De Uma Rede Neural Artificial Em Ambiente Simulado Para O Controle De Trajetória De Um Robô Móvel Com A Arquitetura Ackermann.

Keywords— Ackermann, NXP Cup, Artificial Neural Network, Mobile Robot, Simulation.

Resumo— Os avanços no campo de inteligência artificial na última década têm proporcionado uma das épocas mais interessantes, até o momento, do campo da robótica, mais especialmente, da robótica móvel. Um dos fatores que contribuíram para este fenômeno é a complexidade envolvida no controle do robô móvel durante a navegação autônoma de situações mais complexas, o que exige um controlador que possa generalizar e ter boa taxa de sucesso em situações inéditas. O presente trabalho tem por objetivo implementar, em ambiente simulado, uma rede neural artificial como o controlador do robô móvel usado na NXP Cup. Dos dados provenientes dos experimentos simulados pode-se concluir que a rede neural artificial treinada foi capaz de controlar o robô móvel em um trajeto desconhecido sem deixar o robô móvel sair do percurso, requerendo poucos ajustes de seus pesos sinápticos.

Palavras-chave—Ackermann, NXP Cup, Rede Neural Artificial, Robô Móvel, Simulação.

1 Introdução

Dentre as problemáticas inerentes à robótica móvel, a que possui maior relevância é a navegação autônoma. Esta depende de vários fatores, desde os tipos de sensores usados para estimar o estado do meio até a complexidade a interação entre robô e o meio.

Existem diversas maneiras de implementar o controle de auto nível para a navegação autônoma de um robô móvel. Dentre os mais comuns estão FSM (*Finite State Machines*), *swarm behavior*, inteligência artificial (IA), e entre outros.

IA tem sido implementada significativamente na pesquisa em robótica na última década. Amer et al. (2017) demonstraram a viabilidade do uso de CNN (Convolution Neural Networks) para a localização de um drone em ambientes urbanos.

A IA possui vários subcampos, entre estes o mais significativo é o campo da rede neural artificial (RNA) (GALASSI et al., 2017). RNA é um modelo matemático de como os circuitos neuronais orgânicos funcionam em organismos vivos, onde a maior fonte de inspiração foi o cérebro humano (SHUAI et al., 2009).

Uma das arquiteturas mais estudadas de RNA é o FMLP (*Feedforward Multilayer Perceptron*) (Hassan; Hamada, 2017). Esta arquitetura implementa um dos mais simples modelos de neurônio artificial, o perceptron. A simplicidade computacional do perceptron foi um dos principais motivos do rápido avanço no campo de RNA na última década. Entretanto, existem outros modelos de neurônios artificiais.

A NXP Cup é uma competição organizada pela NXP Semiconductos na área de robótica móvel, que possui etapas anuais nacionais e uma etapa internacional em 9 países desde 2012 (NXP, 2018a). O principal objetivo desta competição é que estudantes desenvolvam, construam e programem um robô móvel que usa o mecanismo Ackerman, o mesmo de veículos automotores (Moreira et al., 2017). Para vencer, o robô móvel tem que terminar um trajeto, sem sair do mesmo, no menor tempo possível (NXP, 2018b).

Portanto, o presente trabelho tem como objetivo implementar em ambiente simulado uma RNA do tipo FMLP no controle de trajetória do robô móvel usado na NXP Cup.

2 Metodologia

2.1 Rede Neural Artificial

Considerando que a RNA possui somente uma camada escondida e que a RNA possui N_i nódulos na camada de entrada, N_h neurônios na camada escondida e N_o neurônios na camada de saída e definindo $\boldsymbol{P}_{nm}(k^N)$ como a m-ésima entrada da n-ésima amostra apresentada a RNA na iteração k^N do algoritmo e $\boldsymbol{W}_{im}^H$ o peso sináptico entre o m-ésimo nódulo da camada de entrada e o i-ésimo neurônio da camada escondida, o potencial do i-ésimo neurônio da camada escondida pode ser computado por:

$$\boldsymbol{H}_i^P(k^N) = \sum_{m=o}^{N_i} \boldsymbol{W}_{im}^H \boldsymbol{P}_{nm}(k^N) \quad (1)$$

Consequentemente a saída do i-ésimo neurônio da camada escondida pode ser calculada da seguinte maneira:

$$\boldsymbol{H}_i^O(k^N) = \varphi_h\left(\boldsymbol{H}_i^P(k^N)\right) \quad (2)$$

Analogamente, definindo $\boldsymbol{W}_{ji}^O$ como o peso sináptico entre o i-ésimo neurônio da camada escondida e o j-ésimo neurônio da camada de saída, o potencial e a saída do j-ésimo neurônio da camada de saída na iteração k^N do algoritmo podem ser computados como se segue.

$$\boldsymbol{O}_j^P(k^N) = \sum_{i=o}^{N_h} \boldsymbol{W}_{ji}^O \boldsymbol{H}_i^O(k^N) \quad (3)$$

$$\boldsymbol{O}_j^O(k^N) = \varphi_h\left(\boldsymbol{O}_j^P(k^N)\right) \quad (4)$$

No processo de *backpropagation* (Rashid; Langenau, 2017) usando *momentum*, definindo $\boldsymbol{D}_{nj}^O$ como a j-ésima saída desejada da RNA da n-ésima amostra dos dados apresentada a RNA, o erro do j-ésimo neurônio da camada de saída pode ser computado por:

$$\boldsymbol{E}_j^O(k^N) = \boldsymbol{D}_{nj}^O(k^N) - \boldsymbol{O}_j^O(k^N) \quad (5)$$

Consequentemente o gradiente do j-ésimo neurônio da camada de saída pode ser obtido por:

$$\nabla \boldsymbol{E}_j^O = \frac{\partial \boldsymbol{E}_j^O(k^N)}{\partial \boldsymbol{W}_{ji}^O} = \dot{\varphi}_h\left(\boldsymbol{O}_j^O(k^N)\right)\boldsymbol{E}_j^O(k^N) \quad (6)$$

O erro do i-ésimo neurônio da camada escondida é calculado levando em consideração todos os caminhos reversos dos N_o neurônios da camada de saída.

$$\boldsymbol{E}_i^H(k^N) = \sum_{j=o}^{N_o} \nabla \boldsymbol{E}_j^O \boldsymbol{W}_{ji}^O \quad (7)$$

Logo, o gradiente do i-ésimo neurônio da camada escondida é:

$$\nabla \boldsymbol{E}_i^H = \frac{\partial \boldsymbol{E}_i^H(k^N)}{\partial \boldsymbol{W}_{im}^H} = \dot{\varphi}_h\left(\boldsymbol{H}_i^O(k^N)\right)\boldsymbol{E}_i^H(k^N) \quad (8)$$

Definindo λ_R como a taxa de aprendizado e γ_R como a taxa do *momentum*, a correção dos pesos $\boldsymbol{W}_{ji}^O(k^N)$ e $\boldsymbol{W}_{im}^H(k^N)$ é calculada como:

$$\Delta \boldsymbol{W}_{ji}^O(k^N) = \lambda_R \nabla \boldsymbol{E}_j^O \boldsymbol{H}_i^O(k^N) + \gamma_R \Delta \boldsymbol{W}_{ji}^O(k^N - 1) \quad (9)$$

$$\Delta \boldsymbol{W}_{im}^H(k^N) = \lambda_R \nabla \boldsymbol{E}_i^H \boldsymbol{P}_{nm}(k^N) + \gamma_R \Delta \boldsymbol{W}_{im}^H(k^N - 1) \quad (10)$$

Logo na iteração k^N, os pesos sinápticos da RNA são corrigidos por:

$$\boldsymbol{W}_{ji}^O \leftarrow \boldsymbol{W}_{ji}^O + \Delta \boldsymbol{W}_{ji}^O(k^N) \quad (11)$$

$$\boldsymbol{W}_{im}^H \leftarrow \boldsymbol{W}_{im}^H + \Delta \boldsymbol{W}_{im}^H(k^N) \quad (12)$$

Durante a inicialização, $\boldsymbol{W}_{im}^H$e $\boldsymbol{W}_{ji}^O$ são inicializados de forma aleatória dentro de um intervalo $[-0{,}3, 0{,}3]$ (Rey; Wender, 2010), ou seja:

$$\boldsymbol{W}_{ji}^O = random(-0{,}3, 0{,}3) \quad (13)$$

$$\boldsymbol{W}_{im}^H = random(-0{,}3, 0{,}3) \quad (14)$$

Para o controle do robô móvel, $\boldsymbol{P}(k^N)$ é a k^N-ésima imagem da câmera e $\boldsymbol{O}_0^O(k^N)$ é usado como a variável de controle na correção de orientação do robô móvel na k^N-ésima iteração do algoritmo. Com $N_o = 1$, o ângulo correspondente ao valor de $\boldsymbol{O}_0^O(k^N)$ é obtido interpolando $\boldsymbol{O}_0^O(k^N)$ para estar dentro do intervalo $[-\pi/4, \pi/4]$, devido a limitação do mecanismo Ackermann (Amarendra; Mathew; Hiremath, 2017). Definindo este ângulo como $u_A^N(k^N)$, tem-se:

$$u_A^N(k^N) = -\pi/4 + (\boldsymbol{O}_0^O(k^N) - (-1))\left(\frac{\pi/4 - (-\pi/4)}{1 - (-1)}\right) \Rightarrow$$

$$\Rightarrow u_A^N(k^N) = \frac{\pi}{4}\boldsymbol{O}_0^O(k^N) \quad (15)$$

Assim para um PWM (*Pulse Width Modulation*) de 8 bits, $\boldsymbol{O}_0^O(k^N)$ é interpolado linearmente para estar no intervalo $[0{,}255]$ e posteriormente, seu valor é arredondado usando a função *floor*. Definindo $u_C^N(k^N)$ como o valor usado no controle do servo motor pela RNA na k^N-ésima iteração do algoritmo, tem-se que:

$$u_C^N(k^N) = floor\left(0 + (\boldsymbol{O}_0^O(k^N) - (-1))\left(\frac{255 - 0}{1 - (-1)}\right)\right) \Rightarrow$$

$$\Rightarrow u_C^N(k^N) = floor(127.5(\boldsymbol{O}_0^O(k^N) + 1)) \quad (16)$$

Logo o algoritmo para o controle de trajetória do robô móvel usando uma *FMLP* é como se segue:

1. Algoritmo 1: Controle de Trajetória usando *FMLP* $(N_i, N_h, N_o, \lambda_R, \gamma_R)$:
2. Inicialize os pesos sinápticos $\boldsymbol{W}^O$ e $\boldsymbol{W}^H$ de acordo com (13) e (14)

3. Com $k^N = 0$, repita 4 até 10 enquanto o robô móvel estiver navegando o trajeto:
4. a. Capture um frame da imagem $\boldsymbol{P}(k^N)$
5. b. Apresente $\boldsymbol{P}(k^N)$ a RNA e compute a saída da RNA $\boldsymbol{O}_0^O(k^N)$.
6. c. Compute o erro de trajetória $\boldsymbol{D}_0^O(k^N)$usando o modelo matemático, derivado previamente, da câmera usada para obter $\boldsymbol{P}(k^N)$
7. d. Use *backpropagation* e atualize os pesos sinápticos $\boldsymbol{W}^O$ e $\boldsymbol{W}^H$ de acordo com o erro da RNA
8. e. Use $\boldsymbol{O}_0^O(k^N)$ para controlar o robô móvel, de acordo com (15) ou (16)
9. f. Se o robô móvel sair do trajeto, reposicione o robô móvel no início do trajeto
10. g. $k^N \leftarrow k^N + 1$

3 Resultados e discussão

3.1 Ambiente Simulado

O simulador de robótica V-REP (*Virtual Robot Experimentation Platform*) da Coppelia Robotics é usado para os ensaios experimentais, a versão usada é a PRO EDU V3.5.0 rev4 (Coppelia Robotics, 2018), usando a API (*Application Programming Interface*) interna na linguagem LUA. Este simulador é considerado um dos melhores da área da robótica, sendo usado até em trabalhos com resultados relevantes (Caldas et al., 2017; Ciszewski et al., 2017; Bazydło; Kacprzyk; Lasota, 2017). A interface do simulador pode ser vista na Figura 1.

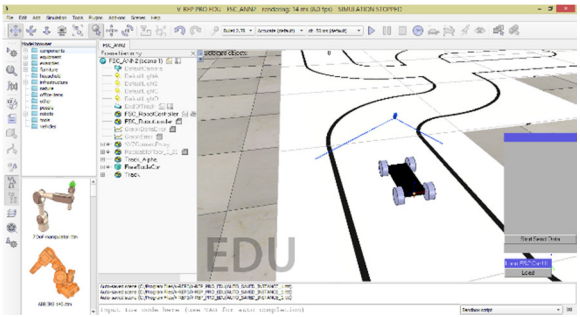

Figura 1. Ambiente do V-REP.

No simulador, quatro trajetos de diferentes formatos foram selecionados para os testes, construídos dentro do regulamento. Cada um dos três trajetos possui número de células N_{Cel} diferente. Sendo estes os trajetos I, II, III e IV, a relação de números de células assim como N_{Cel} de cada pode ser visto na Tabela 1.

Tabela 1. Caracteristicas dos trajetos usados em simulação.

T	N_{Cel}	N_{Curva}	N_{Reta}	$N_{Junção}$	N_S
I	46	20	17	1	5
II	34	18	7	6	3
III	25	12	7	3	3
IV	56	27	15	6	5

Denominando P_i como o ponto em que o robô começa e termina o trajeto, os trajetos I, II, III e IV podem ser vistos na Figura 2, onde os mesmos estão ordenados no sentido anti-horário a partir do canto superior esquerdo.

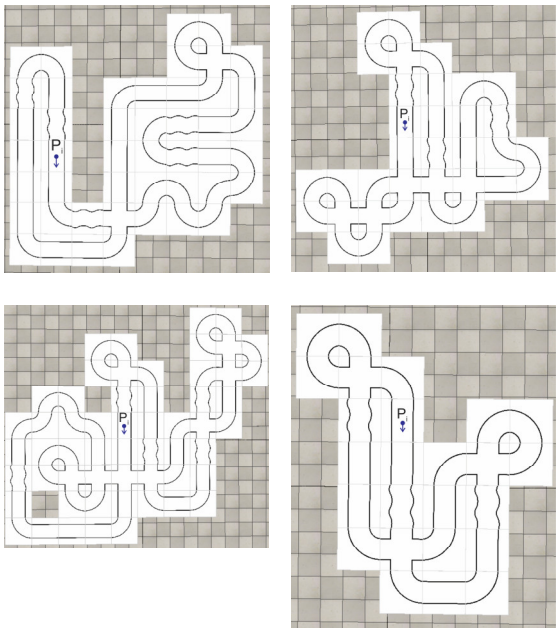

Figura 2. Trajetos usados nos testes em ambiente simulado (Fora de escala).

Na Figura 2 a seta indica o sentido em que o robô móvel completa o trajeto. Para o controle do robô móvel usando uma rede neural *FMLP*, fez-se uso dos trajetos I, II e III para o treinamento e o trajeto IV para a validação do treinamento usando o Algoritmo 1 com as velocidades nominais da Tabela 2. A *FMLP* é treinada de forma *online*, ou seja, enquanto o robô móvel está navegando o trajeto. Como professor do treinamento, fez-se uso do algoritmo derivado para o controle do robô móvel usando uma câmera unidimensional, a mesma usada nestes testes com a *FMLP*.

Tabela 2. Valores de ω_I, ω_{II} e ω_{III}.

ω	rad/s	RPM
ω_I	25	238,732
ω_{II}	30	286,478
ω_{III}	40	381,971

3.2 Ensaios Experimentais

A RNA possui os seus parâmetros com os valores de: $N_i = 128$, $N_h = 12$, $N_o = 1$, $\lambda_R = 0{,}001$ e $\gamma_R = 0{,}01$ (Omidvar; Smagt, 2012). Cada trajeto foi percorrido com e sem pré-treinamento no mesmo trajeto usando um valor para ω, ou seja, o robô móvel percorre cada um dos trajetos I, II e III usando os pesos sinápticos depois que o robô móvel conseguiu completar uma volta no mesmo trajeto usando uma determinada velocidade ω (com pré-treinamento C_I) e

usando os pesos sinápticos usados no mesmo trajeto para uma velocidade anterior da Tabela 2 (sem pré-treinamento C_{II}). A única exceção é usando ω_I, pois não há pesos sinápticos anteriores.

Portanto, C_I usando ω_I, os pesos sinápticos são iniciados de forma aleatória enquanto que C_I para as demais velocidades que estão na Tabela 3, ω_{II} e ω_{III}, os pesos sinápticos iniciais são os pesos sinápticos da velocidade anterior do mesmo trajeto. Quando o robô móvel sai do trajeto, o mesmo é reposicionado na posição inicial de cada trajeto, porem os pesos sinápticos continuam os mesmos.

Definindo N_s como o número de vezes que o robô móvel sai do trajeto, os dados provenientes da simulação do robô móvel enquanto a *FMLP* aprende a efetuar o controle de orientação dos trajetos I, II e III usando os valores de ω da Tabela 2 para os casos C_I e C_{II} podem ser vistas nas Tabelas 3-5.

Tabela 3. Resultados experimentais para o trajeto I.

T	ω	*Caso*	N_s	$EQM(rad^2)$
I	ω_I	C_I	0	0.0554421
I	ω_I	C_{II}	72	1675.777712
I	ω_{II}	C_I	0	105286.792880
I	ω_{II}	C_{II}	2	0.374163
I	ω_{III}	C_I	40	2538.729432
I	ω_{III}	C_{II}	6	8.923943

Para demonstrar a eficácia do uso de uma *FMLP* como controle de orientação do robô móvel, os pesos sinápticos da *FMLP* que possuir o melhor desempenho nos trajetos I, II e III são usados para controlar o robô móvel no trajeto IV. A avaliação da eficácia dos pesos sinápticos de cada trajeto é feita pela diferença do EQM do erro da *FMLP* nos casos C_I e C_{II}.

Tabela 4. Resultados experimentais para o trajeto II.

T	ω	*Caso*	N_s	$EQM(rad^2)$
II	ω_I	C_I	2	43.731084
II	ω_I	C_{II}	86	3583.296896
II	ω_{II}	C_I	8	1144.004265
II	ω_{II}	C_{II}	0	0.113970
II	ω_{III}	C_I	26	1247.246464
II	ω_{III}	C_{II}	0	2162.155254

Tabela 5. Resultados experimentais para o trajeto III.

T	ω	*Caso*	N_s	$EQM(rad^2)$
III	ω_I	C_I	0	0.733317
III	ω_I	C_{II}	94	65441.599738
III	ω_{II}	C_I	16	326.916517
III	ω_{II}	C_{II}	22	159.465939
III	ω_{III}	C_I	40	2714.088377
III	ω_{III}	C_{II}	12	28.000036

Nos estudos comparativos do EQM do erro da *FMLP*, os dados obtidos quando ω_I é usada como a velocidade nominal não são usados. Pois em alguns destes casos, a rede neural é inicializada com pesos sinápticos aleatórios e o que se deseja avaliar é o desempenho da *FMLP* quando a mesma faz uso de pesos sinápticos provenientes de algum pré-treinamento. Que é a situação que a *FMLP* é implementada no hardware físico.

Definindo $EQM^{\rho}_{\omega_j}$ como a razão entre o EQM obtido para o caso C_I e o EQM obtido para o caso C_{II} usando a j-ésima velocidade da Tabela 2 e μ^{ρ}_{EQM} como a media aritmética dos valores de $EQM^{\rho}_{\omega_j}$ para $2 \leq j \leq 3$, $EQM^{\rho}_{\omega_j}$ e μ^{ρ}_{EQM} podem ser calculados como se segue:

$$EQM^{\rho}_{\omega_{II}} = \frac{EQM_{\omega_{II}}|_{C_I}}{EQM_{\omega_{II}}|_{C_{II}}} \quad (17)$$

$$\mu^{\rho}_{EQM} = \frac{1}{2}\sum_{j=2}^{3} EQM^{\rho}_{\omega_j} \quad (18)$$

Os valores de μ^{ρ}_{EQM} para o EQM do erro da *FMLP* nos trajetos I, II e III podem ser vistos na Figura 3.

Da Figura 3 pode-se ver que o menor μ^{ρ}_{EQM} foi o do trajeto I, com $\mu^{\rho}_{EQM} = 0.001759337923$. Logo os pesos sinápticos produzidos pela *FMLP* depois que a mesma aprendeu a navegar o robô móvel no trajeto I são usados para que a *FMLP* controle o robô móvel no trajeto IV. No controle do robô móvel no trajeto IV, a *FMLP* continua sendo treinada.

As posições do robô móvel enquanto a *FMLP* navega o trajeto IV usando os pesos sinápticos aprendidos no trajeto I usando os valores de ω podem ser vistas nas Figuras 4-6. Os dados provenientes do trajeto IV podem ser vistos na Tabela 6.

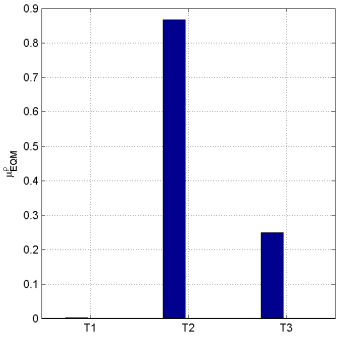

Figura 3. Valores de μ^{ρ}_{EQM} para o EQM do erro da *FMLP* nos trajetos I, II e III.

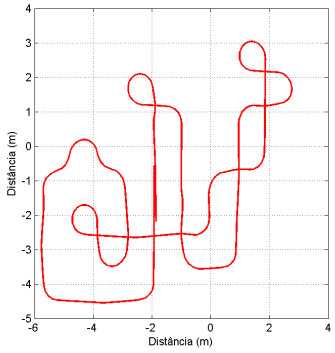

Figura 4. Posições do robô móvel enquanto a *FMLP* efetua o controle de orientação com o robô percorrendo o trajeto IV e usando ω_I.

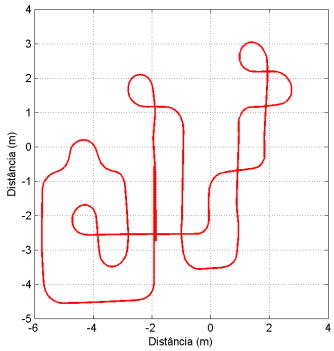

Figura 5. Posições do robô móvel enquanto a *FMLP* efetua o controle de orientação com o robô percorrendo o trajeto IV e usando ω_{II}.

Tabela 6. Resultados experimentais para o trajeto I.

T	ω	N_s	$EQM(rad^2)$
IV	ω_I	0	0.095077390392
IV	ω_{II}	0	0.233917407780
IV	ω_{III}	0	0.097099409577

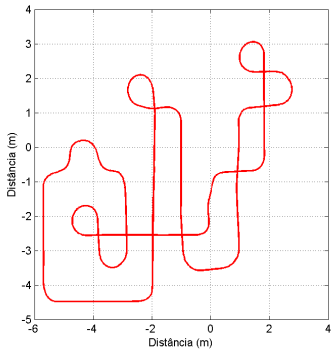

Figura 6. Posições do robô móvel enquanto a *FMLP* efetua o controle de orientação com o robô percorrendo o trajeto IV e usando ω_{III}.

Os valores do EQM do erro da *FMLP* e do erro de orientação robô móvel enquanto o mesmo navega o trajeto IV usando *FMLP* com os pesos sinápticos aprendidos no trajeto I usando ω_I, ω_{II} e ω_{III} podem ser vistos na Figuras 7 e 8.

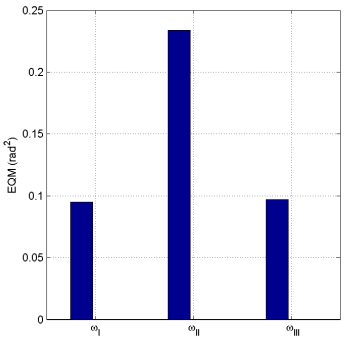

Figura 7. EQM do erro da *FMLP* e enquanto o robô móvel percorre o trajeto usando ω_I, ω_{II} e ω_{III}.

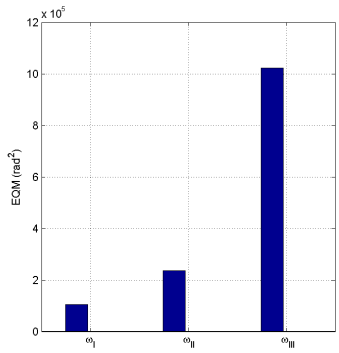

Figura 8. EQM do erro de orientação enquanto o robô móvel percorre o trajeto usando ω_I, ω_{II} e ω_{III}.

Logo, da Figura 8 pode-se ver que o *FMLP* pré-treinado efetua poucos ajustes dos seus pesos sinápticos durante a navegação, o que evidencia que a metodologia aqui proposta possui aplicabilidade viável para a implementação no hardware real.

4 Conclusão

O presente trabalho propôs a implementação de uma rede neural artificial para o controle de trajetória do robô móvel usado na NXP Cup. Dos dados coletados e mostrados nas Figuras 7 e 8 pode-se concluir que a rede neural obteve desempenho superior enquanto o robô móvel navega o trajeto IV. Logo o pré-treinamento efetuado nos trajetos I, II e III produziram uma rede neural artificial capaz de controlar o robô móvel em um trajeto desconhecido, portanto a presente metodologia obteve o resultado desejado.

Em um trabalho futuro pretende-se efetuar os ensaios experimentais usando o hardware físico do robô móvel para demonstrar a eficácia da presente metodologia.

Agradecimentos

Agradecemos ao Instituto Federal do Ceará (IFCE) - Campus Cedro e Campus Caucaia, a Universidade Federal do Ceará (UFC) e ao CNPQ pelo apoio e suporte financeiro.

Referências Bibliográficas

Amarendra, J. H.; Mathew, R.; Hiremath, S. S. A Mathematical Model to Estimate the Position of Mobile Robot by Sensing Caster Wheel Motion. International Conference on Research in Mechanical Engineering Sciences (RiMES), 2017.

Amer, K.; Samy, M.; ElHakim, R.; Shaker, M.; ElHelw, M. Convolutional Neural Network-Based Deep Urban Signatures with Application to Drone Localization. EEE International

Conference on Computer Vision Workshop (ICCVW), 2017.

Bazydło, P.; Kacprzyk, J.; Lasota, K. Global path planning for a specialized autonomous robot for intrusion detection in wireless sensor networks (WSNs) using a new evolutionary algorithm. Trends in Advanced Intelligent Control, Optimization and Automation, Vol. 577, pp. 503-513, 2017.

Caldas, K. A. Q.; Silva, J. A. R.; Escalante, F. M.; Grassi, V.; Terra, M. H.; Siqueira, A. G. A Comparison Between a Simulated and a Real Mobile Robot Path Tracking Application Using V-REP. XIII Simpósio Brasileiro de Automação Inteligente, 2017.

Ciszewski, M.; Mitka, Ł.; Buratowski, T.; Giergiel, M. Modeling and Simulation of a Tracked Mobile Inspection Robot in MATLAB and V-REP Software. Journal of Automation, Mobile Robotics & Intelligent Systems, Vol. 11, No. 2, pp. 5-11, 2017.

Coppelia Robotics. V-REP: Virtual Robot Experimentation Platform. Disponível em:< http://www.coppeliarobotics.com/>. Acesso em: 01 de jan. 2018.

Galassi, M.; Capodieci, N.; Cabri, G.; Leonardi, L. Evolutionary strategies for novelty-based online neuroevolution in swarm robotics. IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2017.

Hassan, M; Hamada, M. Performance Comparison of Feed-Forward Neural Networks Trained with Different Learning Algorithms for Recommender Systems. Computation, Vol. 5, No. 3, 2017.

Moreira, A. F.; Fernandes, A. S.; Silva, J. A. B.; Miranda, P. H. A. Utilização De uma Rede Neural Artificial Como Algoritmo De Controle De Um Robô Ackerman Em Ambiente Simulado. Mostra Nacional de Robótica (MNR), 2017.

Omidvar, O.; Smagt, P. Neural Systems for Robotics. [S.I.]: Elsevier Science. 2012.

Rashid, T.; Langenau, F. Neuronale Netze selbst programmieren: Ein verständlicher Einstieg mit Python. O'Reilly, 2017.

Rey, G. D.; Wender, K. F. Neuronale Netze: Eine Einführung in die Grundlagen, Anwendungen und Datenauswertung. Verlag Hans Huber, 2010.

Shuai, Z.; Yong, Y.; Qing-Kai, H.; Gang, G. 神经网络故障诊断算法及嵌入式系统应用. 微计算机信息 - 测控自动化, Vol. 25, No. 11. 2009.