Execução de Self-Organizing Maps em FPGA aplicada ao cálculo de trajetória de deslocamento de robôs

Kenny Shogo Nakamura[1], Alan Belem Santos[1], Miguel Angelo de Abreu de Sousa[1]

Instituto Federal de Educação, Ciência e Tecnologia São Paulo (IFSP)
Caixa postal 01109-010 – São Paulo – SP – Brazil

kennynakamura@hotmail.com, alanbelems94@gmail.com, angelo@ifsp.edu.br

Abstract. *The research aims to develop and analyze the functionality of the Self Organizing Maps, an artificial intelligence technique, applied to path planning of mobile robots, using the FPGA chip as the implementation platform. The developed project includes create virtual environments, adaptation of the algorithm to the proposed objective, simulations of the algorithm performance and analysis of the results obtained on the FPGA chip. The use of the FPGA chip and the application of artificial intelligence techniques are, together, promising for the challenges faced by mobile robotics.*

Resumo. *A pesquisa tem como objetivo desenvolver e analisar a funcionalidade da técnica de inteligência artificial Self Organizing Maps, aplicada ao cálculo de trajetória de robôs móveis, utilizando o chip FPGA como plataforma de execução. O projeto desenvolvido passou pelas etapas de criação dos ambientes a serem utilizados, a adequação do algoritmo para o objetivo proposto, simulações de desempenho do algoritmo e análise dos resultados obtidos no chip FPGA. A utilização do chip FPGA e a aplicação das técnicas de inteligência artificial se mostram, juntas, promissoras para os desafios enfrentados pela robótica móvel.*

1. Introdução

O estudo da robótica móvel é um tema bastante relevante e atual. A aplicação prática de robôs móveis em diferentes atividades em nossa sociedade vem demonstrando o quão promissor é o futuro desta área. Por exemplo, seu uso em aplicações domésticas (aspiradores de pó e cortadores de grama robóticos), industriais (transporte automatizado e veículos de carga autônomos), urbanas (transporte público, cadeiras de rodas robotizadas), militares (sistemas de monitoramento aéreo remoto, transporte de suprimentos e de armamento em zonas de guerra, sistemas táticos e de combate) e defesa civil (controle e patrulhamento de ambientes, resgate e exploração em ambientes hostis) confirma a grande gama de aplicações atuais dos robôs móveis e os interesses econômicos envolvidos em relação ao seu desenvolvimento e aplicação [Wolf et al. 2009].

Nesse amplo campo da robótica móvel existem diversos desafios a serem suplantados. Uma das adversidades enfrentadas é a navegação do robô móvel, um problema que passa por especificações mecânicas, elétricas e computacionais. A navegação pode ser considerada como a capacidade que um robô móvel tem para, partindo do conhecimento do ponto de partida e de chegada, calcular uma trajetória que lhe permita deslocar-se entre esses pontos, evitando os obstáculos que possa encontrar [Silva 2010].

Com o uso progressivo de técnicas de inteligência artificial nos mais variados problemas de otimizações, o cálculo de trajetória torna-se também um candidato a ser solucionado. Sob a perspectiva da plataforma de execução do algoritmo de navegação, o uso do chip FPGA (Field Programmable Gate Array) configura-se como uma alternativa interessante. Pois, além de possuir um processamento paralelo, que é indicado para as técnicas conexionistas de inteligência artificial, tem o consumo de energia menor do que as GPUs (Graphics Processing Unit) e os processadores sequenciais de alto desempenho [Sousa 2018].

Trabalhos como [Faigl 2016], [Siqueira 2005] e [De Abreu et al 2015] abordam o tema da otimização de trajetórias utilizando técnicas de inteligência artificial em softwares. Este trabalho utiliza uma arquitetura desenvolvida em FPGA para processar a metodologia proposta em [Brassai 2012].

2. Fundamentação Teórica

A técnica Self Organizing Maps, SOM, é uma alternativa de modelagem adequada para tratamento de dados que apresentam relações complexas; pois se trata de um método não supervisionado, ou seja, que não requer conhecimento a *priori* das relações subjacentes entre as variáveis investigadas ou designação de padrões de saída [Kohonen 1990]. Em relação ao aprendizado não supervisionado aplicado à robótica, destaca-se que a dinâmica de RNAs auto-organizáveis evolui dependendo unicamente dos estímulos oriundos do ambiente externo. À medida que estes estímulos, sejam eles sequenciais ou não, vão sendo apresentados à rede, espera-se que ela extraia *por si só* algum tipo de regularidade a partir desses estímulos [Barreto et al. 2002]. Desta forma o método não supervisionado pode auxiliar em aspectos relevantes da robótica móvel, como o tratamento de dados do ambiente se a interferência humana. A navegação utilizada para o robô móvel neste estudo foi a movimentação em direções cardeais e colaterais, assim como em [Brassai 2012].

O processo de treinamento do SOM fornece uma maneira de representar dados de alta dimensionalidade em um espaço de dimensionalidade reduzida, menor que o conjunto de dados original, formando agrupamentos de neurônios competitivos na chamada de camada de saída [Kohonen 2013]. A Figura 1 mostra a configuração do SOM utilizada neste trabalho, com entrada bidimensional (coordenadas do ambiente) e saída unidimensional (trajetória do robô), de forma a possibilitar o cálculo de deslocamento no ambiente mapeado.

O princípio básico do processo de aprendizado competitivo é a concorrência entre os neurônios. O neurônio vencedor (BMU, em inglês Best Matching Unit) tem o ajuste de seu peso, proporcionalmente aos valores do padrão de entrada apresentado, visando-se assim aperfeiçoar o seu estado para a próxima competição (próximo padrão a ser apresentado) [Da Silva et al. 2010].

O neurônio é considerado vencedor quando possuir a menor distância em relação à entrada dentre todos neurônios presentes na rede. A *Expressão* (1) mostra a distância euclidiana como método de cálculo para determinação do neurônio vencedor. Na *Expressão* (1) temos que $dist_j{}^{(k)}$ quantifica a distância entre o vetor de entrada, representando a k-ésima amostra $x^{(k)}$em relação ao vetor de pesos do j-ésimo neurônio $w^{(j)}$ com dimensão n, e admitindo uma rede com *m* neurônios.

$$dist_j^{(k)} = \sqrt{\sum_{i=1}^{n}(x_i^{(k)} - w_i^{(j)})^2}, \text{ onde j = 1, ..., } m \quad (1)$$

Após efetuado o cálculo da distância entre a entrada e todos os neurônios, o neurônio vencedor tem seu vetor peso ajustado, assim como os neurônios próximos ao vencedor, porém com intensidade menor. Dessa forma o neurônio vencedor e seus neurônios próximos têm as suas posições alteradas na camada de saída. A intensidade do ajuste pode ser configurada através da taxa de aprendizagem do processamento do SOM. A taxa de aprendizagem é composta pelo fator de aprendizagem e pela função de vizinhança.

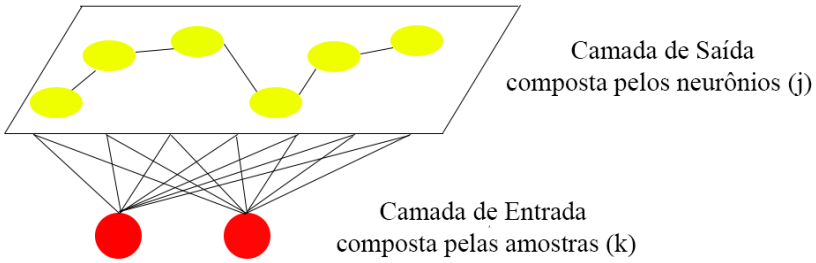


Figura 1 – Representação simplificada do SOM. Os círculos amarelos representam os neurônios (j) de saída unidimensional (trajetória do robô) enquanto os círculos vermelhos são as entradas (k) bidimensionais de dados (coordenadas do ambiente).

O fator de aprendizagem (α) é descrito na *Expressão* (2) com valores inicial e final, onde T representa o número total de interações e t, o valor atual da iteração [Da Silva et all 2010]. A presença de valores iniciais e finais se dá pela necessidade do algoritmo se estabilizar no final do processamento de agrupamento para obtenção de um ajuste fino.

$$\alpha(t) = \alpha_{inicial}(t)\left(\frac{\alpha_{final}}{\alpha_{inicial}}\right)^{\frac{t}{T}} \quad (2)$$

A função de vizinhança é a configuração no processo de treinamento que regula a intensidade do ajuste do vetor peso dos neurônios próximos ao vencedor, ela é descrita pela *Expressão* (3). Na qual a função σ (t) determina o alcance de influência da função de vizinhança, ela é descrita na *Expressão* (4) com as mesmas características do fator de aprendizagem.

$$h_{c_j} = e^{\left(\frac{-(dist)^2}{2\sigma^2(t)}\right)} \quad (3)$$

$$\sigma(t) = \sigma_{inicial}(t)\left(\frac{\sigma_{final}}{\sigma_{inicial}}\right)^{\frac{t}{T}} \quad (4)$$

Finalmente, a *Expressão* (5) descreve a equação de atualização do vetor peso do neurônio vencedor e seus neurônios próximos ao final de cada etapa de iteração do algoritmo. Todos os parâmetros possuem valores sugeridos, no entanto cada problema proposto possui suas particularidades, sendo assim eles devem ser ajustados conforme testes e avaliações.

$$w_{atual}^{(v)} = w_{anterior}^{(v)} + \alpha(t).h_{c_j}(t).(x^{(k)} - w_{anterior}^{(v)}) \quad (5)$$

Tradicionalmente as técnicas de inteligência artificial como o SOM são desenvolvidas para aplicação em software, entretanto, recentemente na robótica móvel, o uso da computação reconfigurável (FPGA) se tornou uma alternativa para as necessidades enfrentadas de redução de energia e dimensões da robótica em geral [Sousa 2018].

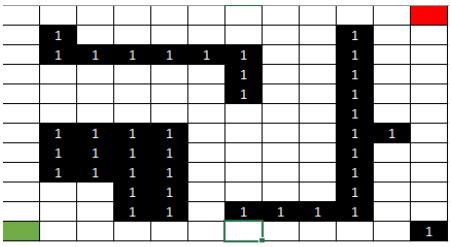
A computação reconfigurável permite combinar o desempenho do processamento em hardware com a flexibilidade dos desenvolvimentos em software [Silva 2016]. Neste trabalho, todo o desenvolvimento da programação em VHDL e as simulações do programa para o mapeamento e cálculo de trajetória dos ambientes foram realizados no software VIVADO. Esse software simula o funcionamento de chips FPGA, possibilitando que se mensure tempo de processamento, consumo de energia e área ocupada do circuito.

3. Metodologia

Nas seguintes subseções são descritas as etapas do desenvolvimento do SOM para a execução do algoritmo, desde a coleta de dados até os ajustes no algoritmo.

3.1 Obtenção dos dados de entrada

Os ambientes virtuais para obtenção das entradas bidimensionais de dados foram criados, utilizando o software Excel, como o exemplo de ambiente desenvolvido mostrado a Figura 2(A). As células pretas representam obstáculos, ou seja, são as células em que o trajeto a ser calculado não pode percorrer. A célula verde representa o início obrigatório do trajeto, e a célula vermelha o fim do trajeto. A Figura 2(B) mostra as coordenadas das células livres que podem ser consideradas no cálculo do trajeto, sendo "L" a letra que acompanha o número da linha e "C" o número da coluna. Essas coordenadas foram utilizadas como dados de entrada na camada de entrada do SOM desenvolvido.



(A)

L1C1	L1C2	L1C3	L1C4	L1C5	L1C6	L1C7	L1C8	L1C9	L1C10	L1C11	L1C12
L2C1		L2C3	L2C4	L2C5	L2C6	L2C7	L2C8	L2C9		L2C11	L2C12
L3C1							L3C8	L3C9		L3C11	L3C12
L4C1	L4C2	L4C3	L4C4	L4C5	L4C6		L4C8	L4C9		L4C11	L4C12
L5C1	L5C2	L5C3	L5C4	L5C5	L5C6		L5C8	L5C9		L5C11	L5C12
L6C1	L6C2	L6C3	L6C4	L6C5	L6C6	L6C7	L6C8	L6C9		L6C11	L6C12
L7C1					L7C6	L7C7	L7C8	L7C9			L7C12
L8C1					L8C6	L8C7	L8C8	L8C9		L8C11	L8C12
L9C1					L9C6	L9C7	L9C8	L9C9		L9C11	L9C12
L10C1	L10C2	L10C3			L10C6	L10C7	L10C8	L10C9		L10C11	L10C12
L11C1	L11C2	L11C3			L11C6					L11C11	L11C12
L12C1	L12C2	L12C3	L12C4	L12C5	L12C6	L12C7	L12C8	L12C9	L12C10	L12C11	

(B)

Figura 2 – Método utilizado para geração das coordenadas do ambiente virtual

3.2 Validação da técnica

Para validação da técnica do uso do SOM para cálculo de trajetórias, proposta inicialmente por [Brassai 2012], utilizou-se um ambiente simples de 12 coordenadas disponíveis, o mapeamento obtido está ilustrado na Figura 3(A) as linhas azuis representam as demarcações das coordenadas não disponíveis (obstáculo retangular), com os pontos pretos representando os neurônios, sendo que os pontos com apenas uma ligação são o primeiro e último neurônios do SOM unidimensional. As coordenadas do canto inferior esquerdo e superior direito foram escolhidas como pontos de início e fim. Isto foi obtido, fixando-se o primeiro e último neurônio do SOM, algo que não ocorre no algoritmo tradicional de treinamento da rede.

O cálculo de trajetória se dá com a formação de um itinerário, a partir do neurônio inicial, que se desloca para o neurônio adjacente com maior índice. Como os neurônios estão interligados, os índices são enumerados seguindo a ordem crescente. Sendo o neurônio inicial com índice 1 e por fim o neurônio final, no caso do ambiente com 12 coordenadas, com índice 12. Na Figura 3(B) os neurônios do ambiente estão enumerados de acordo com a seu índice entre o início e o fim do itinerário. Desta forma, a trajetória calculada é composta pelos neurônios 1, 4, 5, 6, 7, 8 e 12 respectivamente. É possível observar que, no início da trajetória, que itinerário se dá na escolha entre os neurônios 2 e 4 que estão adjacentes ao neurônio 1. Como o neurônio 4 possui maior índice, o trajeto passa pela coordenada em que o neurônio 4 está localizado. Seguindo essa metodologia por todo o percurso.

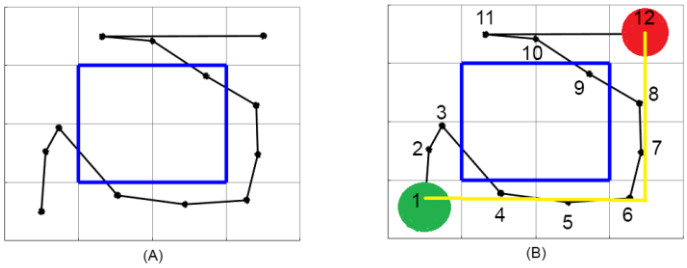


Figura 3 – Ambiente com 12 coordenadas. (A) mapeamento realizado pelo algoritmo SOM. (B) Cálculo da trajetória realizado

3.3 Proposta de inicialização alternativa do SOM

Verificando a funcionalidade da técnica no ambiente simples, foi criado um ambiente mais complexo, chamado ambiente A, como mostra a Figura 4(A). Nos testes, foi possível observar que, com a distribuição inicial aleatória dos neurônios, como mostra a Figura 4(B), o mapeamento total não era satisfatoriamente realizado, mesmo com milhares de iterações do algoritmo, como mostra a Figura 4(C).

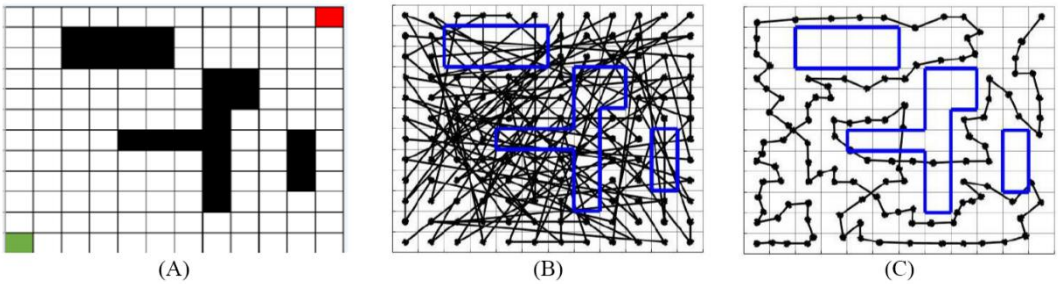


Figura 4 – (A) Ambiente A. (B) Ambiente A com início aleatório dos neurônios do SOM e (C) mapeamento após conclusão do algoritmo de treinamento

Deste modo, foi constatado que a distribuição aleatória dos neurônios no ambiente dificultaria um mapeamento adequado e rápido. Assim, foi proposto por este trabalho que a distribuição inicial dos neurônios ocorresse sem cruzamentos. Mantendo os mesmos pontos de início e fim, a Figura 5(A) mostra a distribuição forçada para o início de treinamento.

Utilizando o mesmo algoritmo aplicado ao ambiente anterior, foi obtido o mapeamento adequado do ambiente A com 60.000 ciclos do algoritmo, como mostra a Figura 5(B). Pode-se observar que o novo mapeamento possui menos cruzamentos e

maior distribuição nas coordenadas. Como o objetivo do projeto é o cálculo da trajetória de início e fim, a distribuição inicial dos neurônios é dada, pois o ambiente de deslocamento do robô é conhecido previamente.

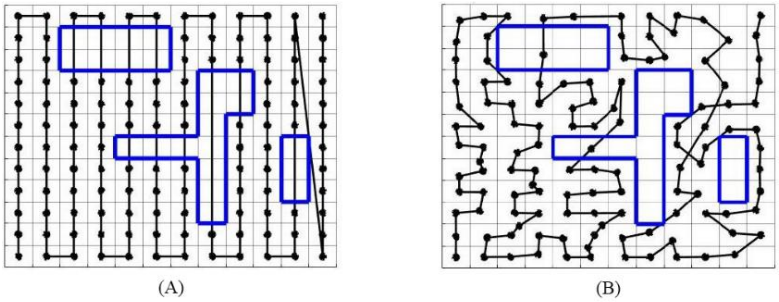


Figura 5 – Ambiente A. (A) Ambiente A com distribuição inicial forçada dos neurônios do SOM e (B) ambiente A mapeado após conclusão do algoritmo de treinamento.

4. Resultados

Com a metodologia desenvolvida e com as alterações propostas, foi executado o mesmo algoritmo aplicado ao ambiente A. A Figura 6 mostra o mapeamento e a trajetória calculada no ambiente A, com a linha amarela representando o percurso calculado.

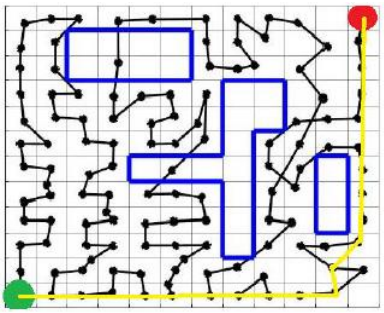


Figura 6 – Calcula da trajetória no ambiente A

A Figura 7(A) ilustra um ambiente de testes de robótica móvel inicialmente proposto por [Sousa 2004]. Baseado nesse ambiente foi criado o mapa nomeado de ambiente B. Nesse ambiente foram obtidas as coordenadas livres, conforme descrito na seção 3.1, e realizada a distribuição inicial dos neurônios, conforme descrito na seção 3.3. Deste modo, foi efetuado o teste final para a avaliação da metodologia desenvolvida no ambiente B. A Figura 7(B) e 7(C) mostram os resultados obtidos do programa neste ambiente, que o mapeamento do ambiente e a trajetória traçada forma adequados.

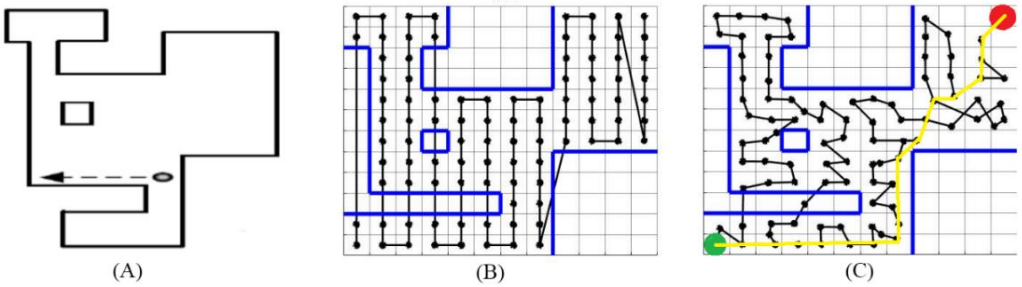


Figura 7 – (A) Ambiente B proposto para testes de robótica móvel por [Sousa 2004]. (B) Distribuição inicial forçada dos neurônios. (C) Trajetória calculada.

Pelos experimentos, foi possível analisar que no ambiente com 121 coordenadas livres (ambiente A) após aproximadamente 60.000 interações do algoritmo foi possível um mapeamento que não possui redundâncias nas trajetórias. Da mesma forma, no ambiente com 72 coordenadas livres (ambiente B) a trajetória sem redundâncias foi obtida a partir de 50.000 iterações.

4.1 Implementação do SOM em FPGA

A arquitetura proposta para execução de cálculo de trajetórias com o uso do SOM foi implementada no FPGA XC7A200T, da família Artix-7 (Fabricante Xilinx), com o objetivo de verificar suas características de tempo de processamento, taxa de ocupação do chip e potência utilizada. Desta forma, utilizou-se o software VIVADO para implementação do SOM com diferentes frequências de clock. A Tabela 1 mostra os resultados obtidos de acordo com a frequência utilizada. Foram verificados principalmente fatores importantes para robótica móvel como energia gasta, tempo de execução e utilização dos recursos internos do chip FPGA (*LUTs*).

Tabela 1 – Testes de desempenho do algoritmo no chip FPGA

Frequência de *clock*	Potência	Área ocupada do chip (% *LUTs* utilizados)	Tempo de execução
0,1 MHz	132 mW	31,3%	500 ms
1 MHz	140 mW	31,3%	50 ms
10 MHz	218 mW	31,3%	5 ms
100 MHz	987 mW	31,3%	0,5 ms

Pode-se perceber, pela Tabela 1, que a potência consumida pela implementação aumenta em função do crescimento da frequência de *clock*, entretanto, a área ocupada do chip permanece constante. Esta característica é importante para permitir que o FPGA seja compartilhado por outros algoritmos necessários para a mobilidade do agente robótico, tais como, localização e controle de deslocamento. A Tabela 1 permite também dimensionar o consumo de potência de acordo com a demanda de velocidade para o cálculo da trajetória de deslocamento do robô (tempo de execução).

5. Conclusão

A aplicação da técnica de inteligência artificial SOM no cálculo da trajetória em ambientes conhecidos ocorreu de forma adequada como inicialmente suposto. A utilização do chip FPGA para execução do SOM se mostrou uma opção viável, com o tempo de processamento adequado e com possibilidade de expansão para ambientes mais complexos. As trajetórias obtidas podem ser classificadas como otimizadas, sendo elas bem próximas das menores distancias possíveis, pois se observou um direcionamento objetivo e sem movimentos redundantes durante o percurso. Com os testes de desempenho do chip FPGA foi possível observar as diferentes possibilidades de execução, que podem variar de acordo com a aplicação e as características do robô móvel.

Referências

Barreto, G. D. A., Dücker, C., Araújo, A. F., & Ritter, H. (2002). Implementação de um sistema de controle para o robô puma 560 usando uma rede neural auto-organizável. Sba: Controle & Automação Sociedade Brasileira de Automática, 13(2), 141-155.

Brassai, S. T., Iantovics, B., & Enachescu, C. (2012). Optimization of robotic mobile agent navigation. Stud. Inform. Control, 21(4), 403-412.

De Abreu, A. A. A. M., de Oliveira, S. L. G., & Lacerda, W. S. (2015). Uma resolução do problema do caixeiro-viajante por mapa auto-organizável com aprendizado winner takes all. *Revista Brasileira de Computação Aplicada*, *7*(1), 100-109.

Da Silva, I. N., Spatti, D. H., & Flauzino, R. A. (2010). Redes Neurais Artificiais para engenharia e ciências aplicadas curso prático. São Paulo: Artliber.

Faigl, J. (2016). An application of self-organizing map for multirobot multigoal path planning with minmax objective. Computational intelligence and neuroscience, 2016.

Kohonen, T. (1990). The self-organizing map. Proceedings of the IEEE, 78(9), 1464-1480.

Kohonen, T. (2013). Essentials of the self-organizing map. Neural networks, 37, 52-65.

Siqueira, P. H. (2005). Uma nova abordagem na resolução do problema do caixeiro viajante. Tese (doutorado) - Universidade Federal do Paraná, Setores de Tecnologia e Ciẹncias Exatas, Programa de Pós-Graduação em Métodos Numéricos em Engenharia. Defesa: Curitiba, 2005.

Silva, P. M. D. S. P. (2010). Movimentação autónoma de robôs móveis de baixo custo, com base no sistema NXT da Lego. Dissertação (Mestrado integrado em Engenharia Mecânica) – Faculdade de Engenharia, Universidade do Porto, Porto, 2010.

Silva, L. M. D. D. (2016). Proposta de arquitetura em hardware para fpga da técnica qlearning de aprendizagem por reforço. Dissertação (Mestrado em Engenharia Elétrica e de Computação) - Centro de Tecnologia, Universidade Federal do Rio Grande do Norte, Natal, 2016.

Sousa, M. A. A., Hirakawa, A. R., & Neto, J. J. (2004). Adaptive automata for mapping unknown environments by mobile robots. In Ibero-American Conference on Artificial Intelligence (pp. 562-571). Springer, Berlin, Heidelberg.

Sousa, M. A. A. & Del-Moral-Hernandez, E. Comparison of three FPGA architectures for embedded multidimensional categorization through Kohonen's Self-organizing maps. In 2017 IEEE Intl. Symposium on Circuits and Systems (ISCAS), IEEE 2017.

Sousa, M. A. A. (2018). Metodologias para desenvolvimento de mapas auto-organizáveis de Kohonen executados em FPGA. Doctoral Thesis, Escola Politécnica, University of São Paulo, São Paulo. doi:10.11606/T.3.2018.tde-06092018-091449. Retrieved 2020-03-21, from www.teses.usp.br.

Vivado Design Suite - HLx Editions Update 2 Version 2019.1 Xilinx.

Wolf, D. F., Simões, E. D. V., Osório, F. S., & Junior, O. T. (2009). Robótica móvel inteligente: Da simulação às aplicações no mundo real. In Mini-Curso: Jornada de Atualização em Informática (JAI), Congresso da SBC (p. 13). sn.