Aplicação de inteligência computacional para navegação autônoma na robótica educacional*

Alfredo Felipe Lopes Neto[1], **Silvio Roberto Fernandes de Araujo**[2], **Danniel Cavalcante Lopes**[2]

[1]Discente da Graduação em Ciência da Computação - Universidade Federal Rural do Semiárido

[2]Professor do Magistério Superior - Departamento de Computação Universidade Federal Rural do Semiárido

alfredo.lopes@ufersa.edu.br, silvio@ufersa.edu.br, danniel@ufersa.edu.br

Abstract. *This article describes the implementation of a neural network with the purpose of navigating a robot avoiding obstacles, discriminating two versions, a simulation developed using the Python language and a robot prototype using LEGO pieces and its own language, the EV3-G.*

Resumo. *Este artigo descreve a implementação de uma rede neural com o propósito da navegação de um robô desviando de obstáculos, discriminado duas versões, uma simulação desenvolvida utilizando a linguagem Python e um protótipo robô utilizando peças LEGO e sua linguagem própria o EV3-G.*

1. Introdução

A tecnologia evoluiu, tornando o ser humano dependente do computador em diversas áreas da sua vida. A ideia que tarefas que são consideradas complexas serão realizadas comumente por sistemas providos de ”inteligência”está cada vez mais presente na nossa realidade. Carros serão capazes de realizar uma navegação sozinhos, selecionando as melhores rotas e prevenindo a maior parte dos acidentes. Sistemas inteligentes ajudarão no controle de cidades inteiras, otimizando cada aspecto da vida da sua população. Nesses exemplos, teremos agentes autônomos presentes realizando estas tarefas. A área da Inteligência Artificial define um agente como uma entidade que funciona de forma contínua e autônoma, capaz de sentir e atuar no ambiente em que se situa por meio de atuadores e sensores [Russell and Norvig 2003]. Por exemplo, em um robô pode-se considerar os motores como atuadores de movimento e um sensor que garante a visualização do ambiente que permitem realizar um deslocamento preciso nesse ambiente controlado.

Em ambientes não controlados com situações imprevistas e de incerteza, que se assemelham aos problemas do mundo real, requer do agente a capacidade de adaptação dos seus conhecimentos de maneira automática.

Nos últimos anos, um grande número de pesquisas vem demonstrando interesse na automação de processos por meios de sistemas robóticos com a visão de promover a otimização do tempo e melhoria na qualidade do produto final, seja na realização de um tarefa ou um manufaturado. A Inteligência Artificial está presente nesses sistemas, na sua

*Trabalho de Conclusão de Curso apresentado pelo aluno **Alfredo Felipe Lopes Neto** sob a orientação dos professores **Silvio Roberto Fernandes de Araujo** e **Danniel Cavalcante Lopes** como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação na UFERSA Campus Central

maioria, a construção por meio de algoritmos propostos para solucionar os problemas.Por isso, na robótica, sistemas autônomos de navegação têm sido explorados nessa área.

A Aprendizagem de Máquina é um campo de estudo da Inteligência Artificial que busca desenvolver algoritmos que utilizam a experiência como forma de aprender possíveis soluções, esta característica oferece uma resposta para problemas que exigem o conhecimento humano. Com a modelagem do problema como tarefa de aprendizagem, os algoritmos de Aprendizagem de Máquina permitem a elaboração de novos agentes que aprendem pela interação com o ambiente ou por estudos de exemplos.

Em um robô, utilizando a Aprendizagem de Máquina para navegação autônoma geralmente desperta grande admiração devido o seu deslocamento de maneira independente. Contudo, os robôs móveis, mesmo com toda tecnologia, ainda apresentam muitas limitações quanto à sua capacidade de navegação [Cazangi and Figueiredo 2000]. A navegação autônoma de robôs requer, entre outras coisas, aprender estratégias de navegação, adaptar-se a novas situações e construir conhecimento a partir de informações obtidas do seu ambiente [Sun 2002].

O desenvolvimento e estudo de um projeto de robôs autônomos pode estar ligado a aplicações práticas, com isso, grandes desafios vêm se mostrando devido as dificuldades que os ambientes de navegação apresentam, cada vez mais imprevisíveis e diversificados. Assim, técnicas que oferecem melhores resultados no desenvolvimento de agentes por meio da interação com o ambiente tendem a se destacarem, nesse caso a Aprendizagem por Reforço se mostra como uma das mais viáveis soluções.

A Aprendizagem por Reforço objetiva desenvolver agentes autônomos que realizam ações e aprendem com o efeito delas. Os agentes tem como objetivo encontrar uma política (sequência de ações) que eleve ao máximo a soma das recompensas ao longo do tempo de execução. Podemos considerar que a política é ótima se a sequência de ações realizadas for a melhor possível para se finalizar uma tarefa. A Aprendizagem por Reforço apresenta meios para o desenvolvimento de novos algoritmos que aprendem por tentativa e erro, sem a necessidade direta de um ”professor”, mas sim, de um crítico para avaliar os estados do aprendizado, ao contrário de outras formas que no lugar do crítico existe um professor que instrui o agente.

Dentre as diversas áreas da robótica, a robótica educacional, forma de ensino usada para motivar e facilitar a instrução de alunos, desenvolvendo habilidades que promovem autonomia e auxiliam sua integração na sociedade, se destaca mostrando diversas formas de utilização no ensino, do fundamental até o universitário, integrando variadas áreas de conhecimento permitindo aos professores mostrarem na prática aplicações teóricas aos estudantes.

A robótica educacional pode ser ministrada por meio da utilização de diversos kits educacionais, o mais conhecido e utilizado é o LEGO Mindstorms EV3 que se mostra bastante versátil para o ensino, além de permitir que alunos participem de competições utilizando o mesmo. O kit EV3 além da sua utilização dentro da sala de aula é usufruído por empresas para criação de modelos e prototipagem.

O objetivo desse trabalho é utilizar técnicas de inteligência computacional, redes neurais e aprendizagem por reforço, no desenvolvimento de um robô autônomo que desvie de obstáculos até chegar ao seu destino, fazendo uso de uma simulação computacional

juntamente com a criação de um modelo robótico, operando um kit LEGO Mindstorms EV3.

A organização do restante do trabalho está da seguinte forma: A seção 2 retrata sobre *Inteligência Computacional*. A seção 3 retrata sobre o *LEGO Mindstorms EV3*. A seção 4 aborda os *Resultados Obtidos*, à medida que a seção 5 e 6 correspondem a *conclusão* e *trabalhos futuros*, respectivamente.

2. Inteligência Computacional

A inteligência computacional é uma das técnicas que mais se prestam a desenvolver aplicações dotadas de alto grau de inteligência e robustez, como é o caso da robótica.

2.1. Redes Neurais

Dentre as pesquisas desenvolvidas nos últimos anos, se destacam as de simulação de capacidades cognitivas de um ser humano, máquinas capazes de demonstrar um comportamento inteligente, tentando simular uma reação humana. No ser humano, o local responsável por desenvolver a inteligência é o cérebro, tendo como entidades básicas os neurônios, servindo como interconectores de um rede que permite troca de informações entre eles, provendo a nossa inteligência como conhecemos. Muitas são as tentativas de criar uma estrutura funcional do cérebro em um ambiente controlado, mapeando e desenvolvendo um estrutura artificial, as redes neurais artificiais.

O cérebro humano possui certas capacidade que garantem o comportamento inteligente, características importantes que são atrativas para uma possível simulação em uma rede neural artificial

- *Tolerância a falhas e Robustez*: Eventual eliminação de alguns neurônios não influenciam na funcionalidade global;
- *Capacidade de aprendizagem*: O cérebro possui a capacidade de aprender novas tarefas, mesmo que, nunca foram executadas anteriormente;
- *Processamento de informações incerta*: Ao receber uma informação que esteja incompleta, afetada por agentes externos ou parcialmente contraditória, o cérebro ainda pode formular um raciocínio correto;
- *Paralelismo*: Incontáveis neurônios estão ativos ao mesmo tempo, não existindo restrição como de um processador.

Uma estrutura relativamente simples é responsável pelo processamento local de informações no cérebro, o neurônio, o modelo de um neurônio real pode ser visualizado na figura 1, sua composição celular é um núcleo e corpo (soma) onde acontecem reações elétricas e químicas, que representam o processamento de informações, a informação localizada no soma, através de impulsos elétricos, se propaga pelo axônio. Ao termino do axônio, várias ramificações propagam a informação para outros neurônios, o dendrite do neurônio receptor realiza conexão com outros neurônios por meio das sinapses. Liberando uma substância química pela sinapse, a informação é processada, assim ocorrendo transmissão entre sinapse e dendrite é realizado a conexão entre dois neurônios.

Um neurônio artificial tenta emular as realizações biológicas que uma célula do sistema nervoso executa. A figura 2 contem o modelo de neurônio artificial de [McCulloch and Pitts 1943], comparando com a Figura 1, temos a tentativa de simular

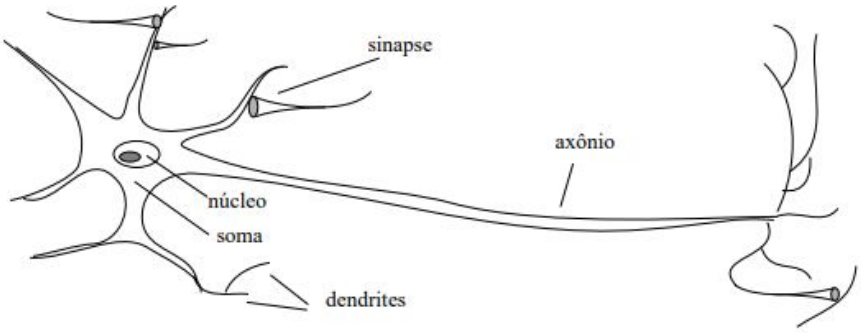

Figura 1. Neurônio biológico

as realidades biológicas de uma célula do sistema nervoso. As informações fornecidas por outros neurônios estão localizadas nas D entradas (x_j = sinapses) do neurônio processador. O processo compreende um somatório de combinações lineares das entradas net = $w_1x_1 + w_2x_2 + ... + w_Dx_D = \sum_{J=1}^{D} w_jx_j = w^Tx$. As entradas x_j possuem pesos w_1, que significam a sua importância, essa combinação linear resulta no valor net. Se esse valor passar o limiar μ, o neurônio encaminha o valor para a saída y, caso o valor não ultrapasse o limiar a saída fica com o valor $y = 0$.

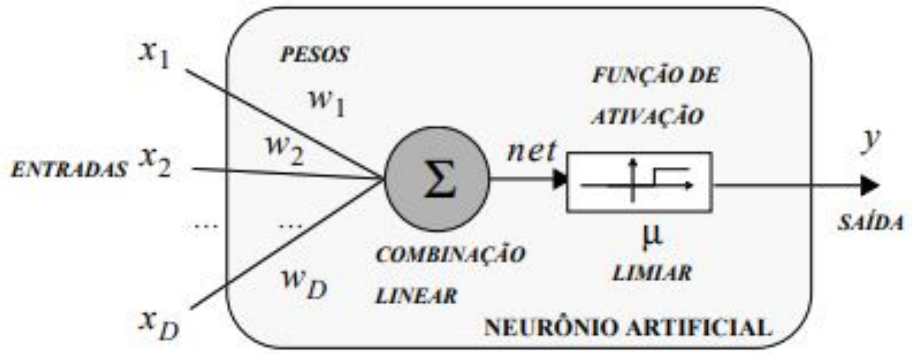

Figura 2. Modelo de um neurônio de McCulloch e Pitts

No modelo de [McCulloch and Pitts 1943] a função de ativação não é a única forma de produzir o valor de saída do neurônio, existe diferentes tipos de funções de ativação como mostra a figura 3. A função linear produz uma saída linear contínua, a função de escada, uma saída binária (não-linear discreta) e a função sigmoidal, uma saída não-linear contínua.

Ao definirmos uma entidade de processamento que calcula uma função de saída y a partir das entradas x_j e dos pesos w_j, tendo a função de ativação já definida. Uma rede neural possui cálculos flexíveis e de grande potencial devido ao conjunto de neurônios que estão interligados entre si, fazendo esse paralelismo de elementos um processamento local desenvolvendo a inteligência global da rede. A comunicação dentre da rede se realiza em um elemento receber um valor nas suas entradas, processa esse valor e encaminha o resultado que será recebido por outros elementos.

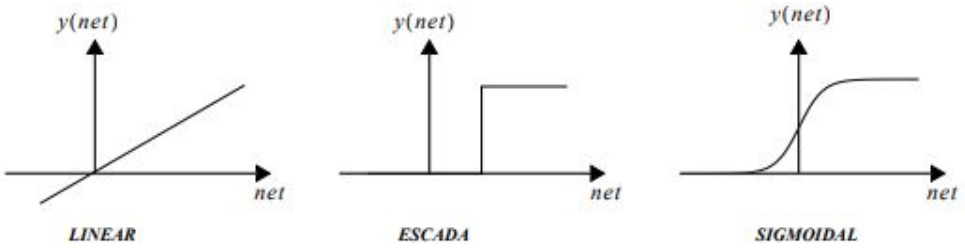

Figura 3. Funções de Ativação

Os neurônios artificiais possuem uma categorização de topologia em relação ao método como a informação recebida é propagada, a Figura 4 mostra as redes de propagação para frente (feedforward) e redes realimentadas (recurrent). Sendo as redes de propagação para frente com o fluxo de informação unidirecional, os neurônios recebem informações simultâneas agrupando-se em camadas, as camadas que as entradas e saídas estão ligadas a outras camadas chamam-se *camadas escondidas*.

Nas redes realimentadas, os neurônios têm ligações de restrições, diferente das redes sem realimentação o desempenho dinâmico possui um papel fundamental no modelo.

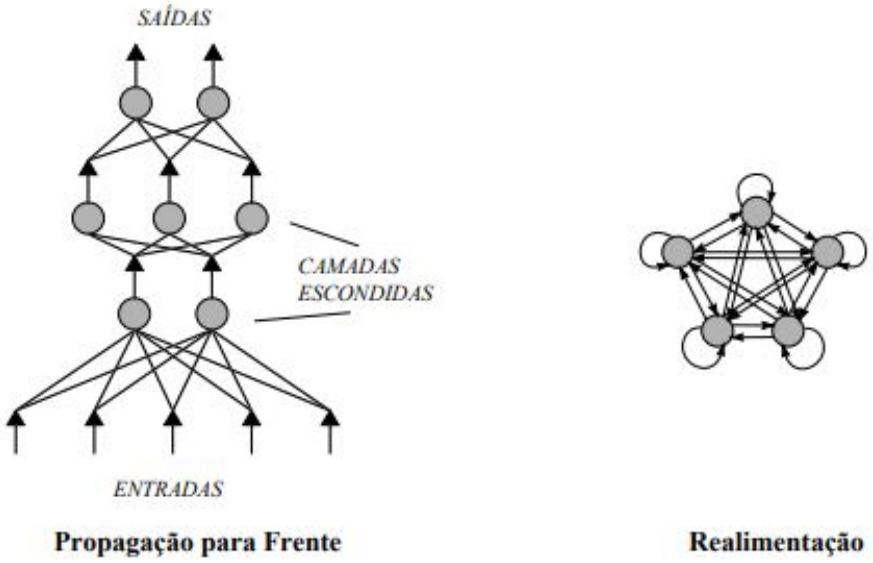

Figura 4. Topologias principais de redes neurais artificiais

Ao finalizar a escolha da rede neural essa deve ser treinada, isso significa que os graus de liberdade que fazem parte da rede, para solucionar a tarefa em questão, têm que ser adequados de maneira otimizada. Logo, isso significa que os pesos w_j, localizados entre os neurônios, serão modificados.

No decorrer do processo de aprendizagem os pesos sofrem modificações de maneira iterativa. Na figura 5, o peso w_{ij} que está entre o neurônio *i* e o neurônio *j* influencia a função calculada pela rede, interação *l*. O algoritmo de aprendizagem conjectura a qualidade do peso e determina eventual modificação no valor $\Delta w_{ij}^{(l)}$ para a próxima iteração

$l + 1$, definindo assim a regra básica de adaptação dos pesos:

$$\text{Adaptação de peso: } w_{ij}^{(l+1)} = w_{ij}^{(l)} + \Delta w_{ij}^{(l)}$$

Por convenção inicializa os pesos aleatoriamente, o algoritmo de aprendizagem percorre um número fixo de iterações ou até atingir uma condição de parada.

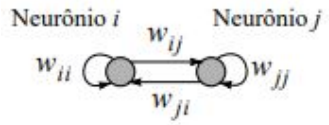

Figura 5. Pesos entre neurônios, caso geral com realimentação

2.2. Aprendizagem

A aprendizagem possibilita que um agente adquira capacidade ou conhecimento que anteriormente não estava disponível na construção do projeto. A aprendizagem por reforço é um paradigma computacional de aprendizagem em que um agente aprendiz procura maximizar uma medida de desempenho baseada nos reforços (recompensas ou punições) que recebe ao interagir com um ambiente desconhecido [Ribeiro 1999].

De modo específico, a atuação do agente no ambiente consiste de conjuntos de possíveis estados, podendo escolher ações dentro de um conjunto de ações possíveis. Recebendo um valor de reforço cada vez que executa uma ação, resultando em um valor imediato que indica uma transição de estado. Ao longo do processo, se forma uma sequência de pares estado-ação com os seus respectivos valores de reforço, o agente tem como objetivo desenvolver um política de controle (sequência de ações) que otimiza a quantidade total de recompensas recebidas na interação com o ambiente.

2.2.1. Processos de Decisão Markovianos

Em essência, a Aprendizagem por Reforço é restrita a processos de decisão Markovianos, contudo, os métodos e ideias podem ser aplicadas de forma mais genérica. Um ambiente satisfaz a propriedade de Markov se o seu estado resume o passado de forma compacta, sem perder a habilidade de prever o futuro. Isto é, pode-se predizer qual será o próximo estado e próxima recompensa esperada dado o estado e ação atuais [Sutton and Barto 1998].

Um processo de Markov é uma sequência de estados, com o atributo de que as predições de valores do estado futuro necessita apenas do estado e ação atuais, e não da sequência de estados passados. Um processo de Aprendizagem por Reforço, quando satisfaz as características de Markov, é chamado de processo de decisão Markoviano (MDP - *Markov Decision Process*). Se o espaço de estados e ações for finito, então ele é chamado de processo de decisão Markoviano finito, a Aprendizagem por Reforço assume o ambiente como sendo deste tipo, em sua base teórica.

Um processo de decisão Markoviano, formalmente, é definido por um conjunto (S, A, P, R), onde temos: um conjunto finito de estados S do sistema, um conjunto finito de ações A, um modelo de transição de estados $P : S\text{x}A \rightarrow \Pi(S)$, que mapeia os pares

estado-ação em uma distribuição de probabilidades sobre o conjunto de estados, e uma função de recompensa R : SxA, que especifica o reforço que o agente recebe por escolher uma determinada ação $a \in A$ no estado $s \in S$. O estado s e a ação a atuais determinam: 1) de acordo com a probabilidade $P(s'|s,a)$ de realizar transição de estado s para o estado s' quando se executa a ação a, e 2) a recompensa $r(s,a)$ associada.

Se o modelo de transição de estados for conhecido, técnicas de Controle Ótimo baseadas em Programação Dinâmica podem - ao menos em tese - ser utilizados para determinar uma política ótima de ações a ser seguida pelo agente [Bellman 1957], já a Aprendizagem por Reforço pode ser utilizada quando o modelo não está disponível (aprendizagem autônoma) ou, alternativamente, quando um modelo de simulação não permite a formulação analítica necessária para a construção de um algoritmo de programação dinâmica.

2.2.2. Aprendizagem por Reforço

A Aprendizagem por Reforço, tem como objetivo principal solucionar o problema de um agente aprender, interagindo com o ambiente via percepção e ação para atingir o objetivo. Sendo o domínio modelado como MDP, ao longo de uma sequência discreta de passo no tempo $t = 0, 1, 2, ..., n$, o agente interage com o ambiente, resultando em dado instante $(s_t \in S$ e $a_t \in A)$, que constroem as probabilidades para o estado s_{t+1} e o reforço r_t. O agente tem como objetivo padrão escolher ações de modo a maximizar a soma dos reforços subsequentes:

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k}$$

sendo a taxa de desconto $0 < \gamma \leq 1$ determina o peso temporal relativo dos reforços.

O agente escolhe sua ações com base em uma função do estado , denominada política, $\pi : S \mapsto A$. Um estado, com o seu valor de utilidade, é o reforço esperado partindo do estado e seguindo a política:

$$V^{\pi}(s) = E^{\pi}\{R_t|s_t = s\}$$

e a política ótima de ações é aquela que maximiza o valor de estado:

$$V*(s) = \max_{\pi} V^{\pi}(s)$$

Existe sempre ao menos uma política ótima π*, que produz o valor de utilidade máximo em todos os estados $s \in S$.

Em paralelo com essas funções, existem as funções de valor de ação:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t|s_t = s, a_t = a\}$$

e a sua maximização

$$Q*(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

2.2.3. Q-learning

Sendo um método de Aprendizagem por Reforço o Q-learning é um algoritmo que estabelece, autonomamente, de maneira interativa uma política de ações. O algoritmo Q-learning tende a um procedimento de controle ótimo, quando a possibilidade de aprendizagem dos pares estado-ação Q for representada em formatado de uma tabela completa, contendo informações de cada par. Essa tendência ocorre tanto para processos de decisão Markovianos determinísticos quanto não-determinísticos.

O algoritmo de Q-learning tem como base aprender uma função de avaliação ótima ao longo de todos os pares estado-ação S x A. Um mapeamento ocorre pela função Q de forma: $Q : S \text{ x } A \mapsto V$, onde V é o valor de utilidade esperado ao se executar uma ação a no estado s. Avaliando que o particionamento do espaço de estados do robô e o particionamento do espaço de ações não apresentem informações relevantes, uma vez que a função Q, no seu estado ótimo, seja aprendida o agente saberá precisamente que ação resultará na maior recompensa futura em uma situação particular s.

Uma recompensa futura espera ao se escolher a ação a no estado s, gera uma função $Q(s, a)$ que é aprendida através de tentativa e erro segundo a equação:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha\big[r_t + \gamma V_t(s_{t+1}) - Q_t(s_t, a_t)\big] \qquad (1)$$

sendo α a taxa de aprendizagem, r a recompensa ou custo, resultante de tomar a ação a no estado s, γ o fator de desconto temporal e o termo $V_t(s_{t+1}) = \max_\alpha Q_t(s_{t+1}, a)$ é o valor Q correspondente à ação com maior valor de utilidade no estado futuro.

O algoritmo Q-learning tem essa forma procedimental:

- Para cada s, a inicialize $Q(s, a) = 0$
- Teste s
- Repita
    1. Selecione ação a usando a política de ações atual
    2. Execute a ação a
    3. Recebe a recompensa imediata $r(s, a)$
    4. Teste o novo estado s'
    5. Atualize o item $Q(s, a)$ de acordo com a equação (1)
    6. $s \leftarrow s'$
- Até que critério de parada seja satisfeito

Usualmente, a política de ações converge para uma política ótima em tempo finito, ainda que de forma lenta. Uma vez que todos os pares estado-ação tenham sido visitados um número infinito de vezes, garante-se que o método gerará estimativas Q_t que convergem para o valor de Q [Watkins and Dayan 1992]

3. LEGO Mindstorms EV3

Nesta seção é apresentado o kit de robótica educacional *LEGO Mindstorms EV3*, suas principais características e seus componentes básicos. Esse modelo de kit foi utilizado para a criação e programação do protótipo robótico, utilizado nos experimentos realizados.

3.1. Kits LEGO Mindstorms EV3

Os kits LEGO Mindstorms EV3 são kits de robôs programáveis patenteados pela $LEGO^{®}$ em setembro de 2013, substituindo a segunda geração do kit LEGO Mindstorms NXT 2.0.

Existem duas versões do EV3, a 31313 (Home Edition) e a 45544 (Education), sendo a versão Education, figura 6, utilizada nos projetos desenvolvidos na UFERSA.

Figura 6. LEGO MINDSTORMS EV3 Education

O kit LEGO MINDSTORMS EV3 Education se caracteriza por ser um kit básico amplamente utilizado no ensino da robótica educacional, este kit é composto por cerca de 540 peças: 2 motores grandes e 1 motor médio (peças que apresentam movimento proporcional baseado em um comando), 5 sensores (2 de toque, 1 de cor, 1 ultrassônico e 1 giroscópio), 7 cabos para conexões com motores e sensores, 1 cabo USB, 1 brick EV3 (microcontrolador do robô), bateria recarregável, base giratória, rodinhas, pneus, blocos, vigas, eixos, engrenagens e polias (tipo de roda para correia transmissora de força e movimentos).

No kit da LEGO o microcontrolador, comumente chamado de brick, pode ser considerado o cérebro dos robôs, figura 7. Ele viabiliza autonomia na execução de diferentes tarefas tais como a criação, programação e montagem dos robôs, passando noções de distância, capaz de reagir a movimentos e ruídos, tudo com base em programação.

No brick existem 4 portas de saída (output), onde se conectam os motores, 4 portas de entrada (input). Se faz necessário verificação se os motores e sensores estão conectados corretamente nas devidas portas, pois eles não são permutáveis. A porta USB, utilizada para comunicação com o software, está localizada ao lado das portas de comunicação com os motores, através dela é possível copiar os programas do computador para o robô, e também do robô para o computador. Esta comunicação pode ser realizada por bluetooth, também disponível no EV3.

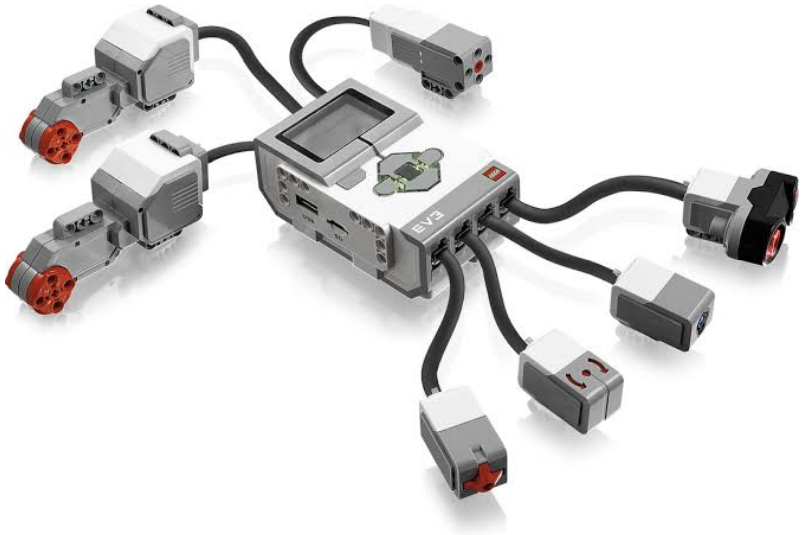

Figura 7. Brick EV3

A linguagem utilizada na programação do EV3, o EV3-G, tem como base blocos[3] o que proporciona uma compreensão simplificada de sua utilização. Analogamente a outras linguagens de programação permite a utilização de *loops* de repetição, módulos condicionais e criação de métodos, devido a programação sequencial em EV3-G, muitas vezes essa programação fica muito extensa, para facilitar isso, o software do EV3 disponibiliza uma funcionalidade chamada *My Block*, figura 8, que permite criar métodos, como qualquer linguagem de programação de modo que possibilita a modelagem para reaproveitar código, que pode ser utilizado em qualquer parte da programação realizando exatamente as mesmas funcionalidades dos blocos utilizados na sua construção. Encontramos nele a possibilidade de adicionarmos até 10 (dez) parâmetros, entre entradas e saídas.

Figura 8. *My Block*

4. Resultados Obtidos

Nesta seção tratamos a implementação da simulação computacional e do robô. A simulação consiste na criação de um modelo similar de um robô construído utilizando LEGO, executando a implementação de uma rede neural para aprender o caminho entre dois pontos e desviar de possíveis obstáculos, o objetivo é encontrar os pesos da rede neural para ser aplicado no protótipo robótico. O robô consiste na montagem de um protótipo

[3]Detalhamento de todos os blocos utilizados na programação podem ser encontrados no seguinte site: https://ev3-help-online.api.education.lego.com/Retail/en-us/index.html

utilizando peças LEGO, para realizar a navegação entre dois pontos sobre uma mesa de competição da LEGO, e sua programação utilizando EV3-G baseada em redes neurais.

A utilização de redes neurais e aprendizagem por reforço, tanto na simulação quanto no protótipo, são usadas para permitir que o robô se adapte durante sua trajetória ao aparecer obstáculos desconhecidos.

4.1. Simulação Computacional

Para a simulação, na parte gráfica, foi criado um ambiente utilizando a biblioteca Kivy, desenvolvida majoritariamente em Python, sua execução se dá em cima da biblioteca gráfica OpenGL garantindo que o código será executado nos processadores da placa gráfica, garantindo uma performance igual ou superior a aplicações nativas.

Podendo adicionar obstáculos a qualquer momento da execução do simulador, pressionando e segurando o botão esquerdo do mouse, basta desenhar os contornos das formas.

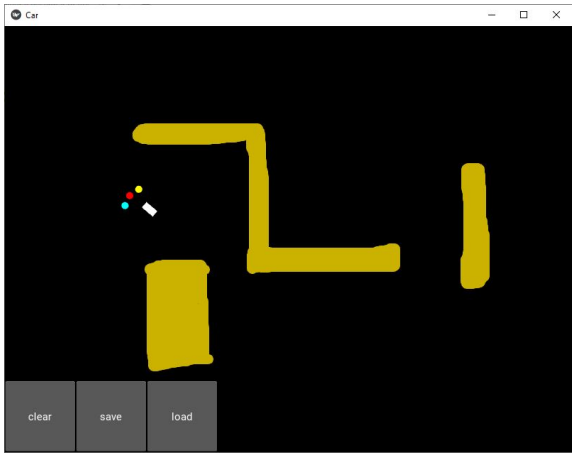

Figura 9. Ambiente da simulação

O simulador do robô, possui formato retangular e círculos na parte dianteira que representam os sensores para saber se ocorreu alguma colisão ou não. Esse sensores fazem a leitura em linha reta, no formato de círculos para frente, determinando que sensor central esteja no ângulo 0 o sensor da direita e da esquerda estão a uma distancia de $30°$, garantindo um abertura de leitura de $60°$ a frente do robô, equivalente ao sensor ultrassônico do robô físico.

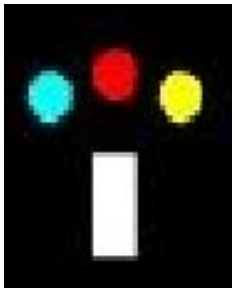

Figura 10. Simulador do Robô

Na programação foi utilizada a biblioteca Torch, para auxiliar na criação da rede neural, utilizando 5 (cinco) neurônios na camada de entrada, 30 (trinta) neurônios na camada oculta e 3 (três) na camada de saída, essa rede neural é *full connection* (cada neurônio se comunica com todos neurônios da camada seguinte).

A implementação do algoritmo de Q-learning se dá com a inicialização de uma função Q(s, a), tendo como função de ativação a ReLU (se a entrada for negativa, ela será convertida em zero e o neurônio não será ativado, garantindo uma rede mais eficiente e fácil). O robô inicia sua movimentação pelo ambiente alimentando a função Q(s, a), os novos resultados passam por uma função, que multiplica esses resultados com a variável de temperatura, se utiliza dessa variável para encontrar a porcentagem de cada resultado, e efetuar o sorteio de escolha, com essa escolha o robô conhece qual ação será tomada.

Ao tomar conhecimento de qual ação será realizada e dos valores de probabilidade de todas as possíveis ações, uma outra função com essas informações calcula o erro utilizando a variável gamma γ e realiza o processo de *backpropagation*, calculando o gradiente da função de erro e modificando os pesos dos neurônios da rede na tentativa que a probabilidade de todos possuam valores parecidos de escolha

Esses procedimentos garantem uma não estagnação em um máximo local ou mínimo local, corrigindo os pesos e realimentando a rede, dando ao robô a chance de continuar aprendendo.

A simulação apresentou um resultado satisfatório, o robô se deslocava do ponto A ao ponto B desviando de obstáculos depois de 6 (seis) horas do inicio da execução. Inicialmente o robô estava conhecendo o ambiente, ao longo do tempo o robô estuda as possibilidades de chegar ao objetivo, somente com 4 (quatro) horas do inicio da execução o robô apresenta evolução, chegando ao seu objetivo depois das 5 (cinco) horas e mantendo até o fim dos testes, resultando em um gráfico de evolução:

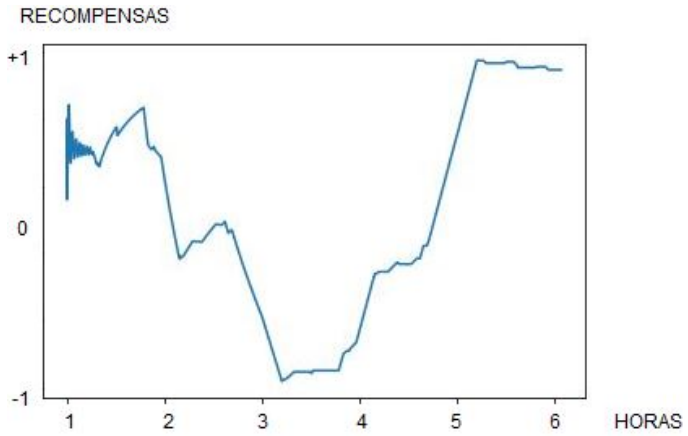

Figura 11. Simulador do Robô - Gráfico

4.2. Robô

O robô foi construído utilizando peças LEGO, juntamente com 1 (um) brick EV3, 2 (dois) motores grandes, 1 (um) sensor giroscópio, 1 (um) sensor ultrassônico, 1 (um) motor médio, 2 (dois) rodas de borracha, 5 (cinco) cabos de transmissão. As medidas do robô são 15cm de largura, 21cm de comprimento e 13cm de altura, figura 12.

Figura 12. Robô

Durante a programação do robô foi verificado que a implementação da rede neural tal qual está presente na simulação não seria viável para o robô, devido a quantidade neurônios presentes na rede neural. Foi necessário realizar um adaptação nas configurações da rede neural.

Na programação foi utilizado a linguagem EV3-G para construção de uma rede neural, devido a sua capacidade de processamento (baixa, comparado a um computador) e limitações da linguagem no desenvolvimento da programação, a implementação da rede neural foi feita utilizando somente 3 (três) neurônios na camada de entrada, 4 (quatro) neurônios na camada oculta e 3 (três) neurônios na camada de saída, a conexão da rede neural *full connection*.

Comparando com a simulação, a rede neural do robô precisou somente de 1 (um) neurônio na camada de entrada para tratar as colisões enquanto na simulação utilizava 3 (três), na camada oculta da simulação ao longo dos testes encontramos que 30 (trinta) era um número de neurônios satisfatório enquanto no robô, devido a sua capacidade de funcionamento, 4 (quatro) neurônios apresentaram um desenvolvimento satisfatório, a tentativa de implementar mais neurônios deixou a execução bastante debilitada.

Na programação, foram construídos *My Blocks* para facilitar o entendimento da mesma e modularizar o código. O *My Block* coordenadas, figura 13, recebe como entrada 2 (dois) ponto iniciais e 2 (dois) pontos finais, e calcula o ângulo entre os pontos iniciais e os pontos finais, utilizando a formula arco tangente

$$\arctan(x) = \tan^{-1}(x) = \frac{y_2 - y_1}{x_2 - x_1}$$

e calcula a distância entre os pontos iniciais e os pontos finais, utilizando a formula da hipotenusa

$$H = \sqrt{A^2 + B^2}$$

Esses valores são armazenados em variáveis.

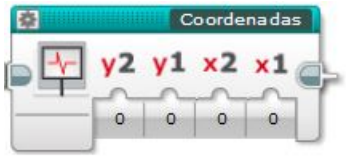

Figura 13. *My Block* coordenadas

Dessa forma o robô pode encontrar qualquer ponto na mesa de competição LEGO, utilizando o princípio de Pitágoras, figura 14.

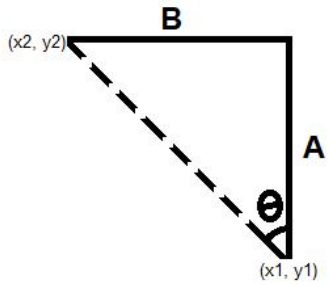

Figura 14. Triângulo de Pitágoras

Com as variáveis do ângulo e da hipotenusa passamos para o próximo passo pegamos essa variáveis e alimentamos um novo *My Block*, figura 15 que calcula o movimento do robô, como pesos para cada motor, encontramos a força e direção para o deslocamento do robô.

Figura 15. *My Block* calcula movimento

Com todas essas informações salvas nas variáveis, passamos para um terceiro *My Block* juntamente com as informações do sensor ultrassônico, o *My Block* rede, figura 16, recebe essas informações para iniciar o desvio de algum objeto que se encontre no caminho entre o ponto inicial e final, passando os resultados para os motores fazendo com que o robô realize um desvio quando necessário. O *My Block* rede realiza o calculo do ponto onde encontra um obstáculo, derivando os valores desse ponto inicial até um ponto final criando uma parábola para desviar do obstáculo.

Ao longo dos testes foi constatado que o robô realiza a função de encontrar o ângulo e deslocamento, quando não encontra nenhum obstáculo, em todos os casos. Quando o robô encontra obstáculos similar a um poste ou não muito largo, o robô realiza a função em todos os casos. já quando o robô encontra obstáculos similar a uma

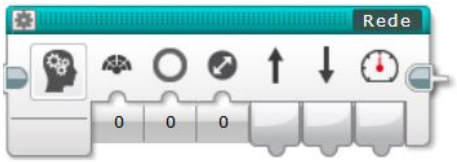

Figura 16. *My Block* Rede

parede ou tão largos quanto ele próprio, ele não consegue realizar o desvio esbarando com a lateral no obstáculo.

Acreditamos que o robô apresentou bons resultados e que devido falta de capacidade de processamento do EV3 a rede neural não apresentou todos os resultados esperados falhando no ultimo teste.

5. Conclusão

Esse trabalho apresentou a implementação de uma rede neural em um ambiente simulado e sua versão análoga em um robô LEGO. A escolha desses métodos não foi realizada por acaso, se fazia necessário a aplicação de redes neurais em programação LEGO para facilitar, em um ambiente competitivo, a realização de navegação entre pontos e para o ensino, em um ambiente acadêmico, o estudo de redes neurais.

Foi essencial compreender os diferentes métodos de implementação das redes neurais, suas formulas de ativação, tipos de aprendizagem de máquina e utilização dos resultados, e aplicar tudo isso em um ambiente já familiar, que se caracteriza o mundo da robótica com LEGO, todas essa etapas foram necessárias para desenvolver a solução obtida.

A partir do cenário de testes e análise dos resultados implementados no robô mostrou que existe um custo para desenvolver a capacidade de performance do robô e que encontrado o melhor custo benefício a aplicação se faz viável. É importante mencionar que a aplicabilidade da implementação em um cenário competitivo é possível, adaptando para a realidade do novo robô, número de motores e sensores.

6. Trabalhos Futuros

Como trabalho futuro, planeja-se adaptar a implementação para micropython, que pode ser utilizado no EV3 substituindo o EV3-G. A ideia é desenvolver a rede neural externa a programação e aplicar diretamente no brick do EV3, fazendo o uso de um cartão de memória.

Outro trabalho futuro consiste em desenvolver essa programação existente em EV3-G para que o robô possa estimar a possibilidade de passar entre dois objetos, tendo o robô consciência da sua largura estimar o tamanho da abertura entre os objetos e decidir desviar ou passar entre eles. No entanto, isso significa a criação de uma nova rede neural que pode acarretar no desempenho do robô.

Referências

Bellman, R. (1957). Applied dynamic programming. Princeton University Press.

Cazangi, R. R. and Figueiredo, M. F. (2000). Sistema de navegação autônoma de robôs baseado em sistema de classificação com aprendizado e algoritmos genéticos. In *Anais do II Fórum de Informática e Tecnologia de Maringá e V Mostra de Trabalhos em Informática da UEM, Maringá-PR*, page 29–29.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. In *Bulletin of Mathematical Biophysics, vol. 5*, pages 115–133. Anderson and Rosenfeld.

Ribeiro, C. H. C. (1999). A tutorial on reinforcement learning techniques. page 104. International Joint Conference on Neuronal Networks. Washington.

Russell, S. J. and Norvig, P. (2003). Artificial intelligence: a modern approach 2.ed. [s.l.]. Pearson Education.

Sun, Z. e. a. (2002). A real-time precrash vehicle detection system. In *Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision (WACV'02), Orlando-FL*, pages 01–06.

Sutton, R. S. and Barto, A. G. (1998). Reinforcement learning: An introduction. Massachusetts: MIT Press.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. machine learning. pages 279–292.

UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

Às 14 horas do dia 06 de Fevereiro do ano de dois mil e vinte, no LAACOSTE do prédio LCC – Laboratórios de Ciência da Computação, na Universidade Federal Rural do Semi-árido - UFERSA, sob a presidência do professor D. Sc. Sílvio Roberto Fernandes de Araújo, reuniu-se a Banca Examinadora de defesa de monografia de autoria do aluno ALFREDO FELIPE LOPES NETO, matrícula no. 2010201751, aluno do curso de Ciência da Computação desta Universidade, com o título "Aplicação de Inteligência Computacional para Navegação Autônoma na Robótica Educacional". A Banca Examinadora ficou assim constituída: Prof. D. Sc. Sílvio Roberto Fernandes de Araújo, presidente da banca e orientador do trabalho de conclusão de curso; Prof. D. Sc. Danniel Cavalcante Lopes, coorientador, Profa. D. Sc. Amanda Gondim de Oliveira, e Prof. D.Sc. Paulo Gabriel Gadelha Queiroz, como membros. Concluída a defesa, procedeu-se o julgamento pelos membros da banca examinadora, em reunião fechada, tendo o aluno obtido as seguintes notas: 7,0; 7,0; 7,0; e 7,0. Apuradas as notas verificou-se que o aluno foi APROVADO com média geral 7,0. E para constar, eu, Sílvio Roberto Fernandes de Araújo, lavrei a presente ata que, após lida e aprovada pelos membros da banca examinadora, será assinada por todos.

Mossoró, 06 de Fevereiro de 2020.

Assinatura dos membros da Banca Examinadora.

Sílvio R. Fernandes de Araújo
Prof. D.Sc. Sílvio Roberto Fernandes de Araújo – UFERSA
Presidente e Orientador

Danniel Cavalcante Lopes
Prof. D.Sc. Danniel Cavalcante Lopes
Membro e Coorientador

Prof. D. Sc. Amanda Gondim de Oliveira – UFERSA
Primeiro Membro

Prof. D.Sc. Paulo Gabriel Gadelha Queiroz – UFERSA
Segundo Membro