CONEXÕES
CIÊNCIA E TECNOLOGIA

INTELIGÊNCIA ARTIFICIAL APLICADA À OLIMPÍADA BRASILEIRA DE ROBÓTICA

VITOR SILVA[1], RICARDO PIRES[2]

[1]Universidade de São Paulo - USP
[2]Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - IFSP
<silva.vitor@usp.br>, <ricardo_pires@ifsp.edu.br>
10.21439/conexoes.v15i0.2100

Resumo. A Olimpíada Brasileira de Robótica é uma competição entre estudantes, na qual um dos desafios é que os robôs capturem esferas dispostas numa sala. Este trabalho propõe uma solução para esse problema utilizando visão computacional e redes neurais artificiais, as quais fornecem como saída final um vetor contendo a localização do ponto do centro da esfera mais próxima e seu raio na imagem. Isso permite que os pontos que delimitam a esfera sejam determinados. O modelo obtido ao final do trabalho em um computador de uso geral foi satisfatório, obtendo uma acurácia de detecção de aproximadamente 90%. Para seu desenvolvimento, foi utilizado um banco de dados com 1440 imagens de situações possíveis com as quais o robô pode se deparar. A rede já foi testada numa possível placa a ser embarcada, porém resta ainda fazer os testes em um robô real.

Palavras-chaves: Olimpíada. Robótica. Redes neurais artificiais. Visão Computacional. Keras.

ARTIFICIAL INTELLIGENCE APPLIED TO THE BRAZILIAN ROBOTICS OLYMPIAD

Abstract. The Brazilian Robotics Olympiad is a competition among students, in which one of the challenges is that the robots capture spheres distributed in a room. This work proposes a solution for this problem using computer vision and artificial neural networks, which produces as output a vector containing the location point of the center of mass from the nearest sphere, and its radius on the image. This allows the points that delimitate the sphere to be determined. The model obtained by the end of the work was satisfactory in a laptop, obtaining an accuracy of approximately 90%. For its development, a database with 1440 images of possible situations was used. The neural network was already tested on a board which is possible to be embedded, but there are still tests left to be done on a real robot.

Keywords: Olympiad. Robotics. Artificial neural networks. Computer Vision. Keras.

1 INTRODUÇÃO

A Olimpíada Brasileira de Robótica é uma competição para alunos do ensino médio e técnico que tem como objetivo estimulá-los a atuarem nas áreas de Ciência e Tecnologia por meio da construção de um robô autônomo capaz de superar diversos desafios.

Os desafios propostos pela Olimpíada simulam um ambiente no qual seres humanos não poderiam intervir, o que gera a necessidade de que um robô faça as atividades. Entre esses desafios, encontra-se a Sala de Resgate, na qual o robô deve recuperar diversas esferas de isopor, que simbolizam vítimas de um acidente, e colocá-las em uma zona segura, a partir da qual os seres humanos poderiam assumir os cuidados. Nesse desafio, há dois tipos de esferas: as prateadas (cobertas com papel alumínio), que representam uma vítima viva, e as pretas (pintadas com tinta), que representam vítimas mortas, ambas com diâmetro aproximado de 5 cm. As esferas prateadas valem 60 pontos, enquanto as pretas valem 50 pontos. O objetivo do desafio é maximizar a quantidade de vítimas vivas resgatadas.

Tentativas anteriores de solução desse mesmo desafio, em geral, envolvem a detecção das esferas medindo-se a variação de distância até elas utilizando sensores ultrassônicos e sensores a laser. No entanto, esses tipos de sensores fornecem pouca informação e estão muito suscetíveis a variações do robô, o que dificulta a elaboração de um algoritmo que seja capaz de gerar resultados

Artigo submetido em 31 maio 2021 e aceito em 06 set. 2021

consistentes (SILVA; NASCIMENTO; KASHIWAGI, 2019).

Para isso, surgiram diversos desafios na tentativa de desenvolvimento de um algoritmo tradicional, que não envolvesse inteligência artificial. Dentre eles, os mais acentuados foram lidar com a diferença de luminosidade e evitar falsos positivos. A abordagem com inteligência artificial e visão computacional visa a prover diversas situações diferentes para que a rede neural seja treinada em múltiplos cenários. Ela foi realizada utilizando a linguagem de programação Python na IDE Visual Studio Code.

A solução aqui proposta consiste na análise do ambiente por meio de imagens obtidas por uma câmera na frente do robô, análise esta realizada por meio de redes neurais artificiais treinadas especificamente para essa situação. O tipo de rede neural escolhido foi a rede neural convolucional, a qual é um tipo de rede especializada em processar dados que possuam uma estrutura do tipo grade, como ocorre com imagens digitalizadas. O que caracteriza esse tipo de rede é a presença, dentre suas camadas, de ao menos uma camada que realize a operação de convolução (GOODFELLOW; BENGIO; COURVILLE, 2016).

2 DADOS PARA TREINAMENTO

2.1 Obtenção dos dados

Durante o ano de 2019, foram tiradas por um dos autores 120 fotos de diferentes situações com as quais o robô poderia se deparar. Entre elas, fotos nas quais não havia nenhuma esfera e fotos em que havia várias esferas (SILVA et al., 2019).

As fotos possuem a resolução de 320x240 pixels. Porém, para o processo de treino dos modelos, elas foram reduzidas para 160x120 pixels, para que o processo fosse mais rápido. Caso seja utilizada uma máquina mais potente, a resolução original pode ser mantida.

Na Figura 1, podem-se ver alguns exemplos de fotos tiradas. Foram também introduzidos elementos disponíveis no laboratório, de modo que esses objetos produzissem variação nas fotos e ruído que poderia, de fato, estar presente durante a competição. Também foram utilizadas diferentes fontes de iluminação artificial.

Com o intuito de otimizar a quantidade de pontos obtidos em função do tempo, que é fixo, optou-se por definir como a esfera correta a ser resgatada a que estivesse mais próxima do robô.

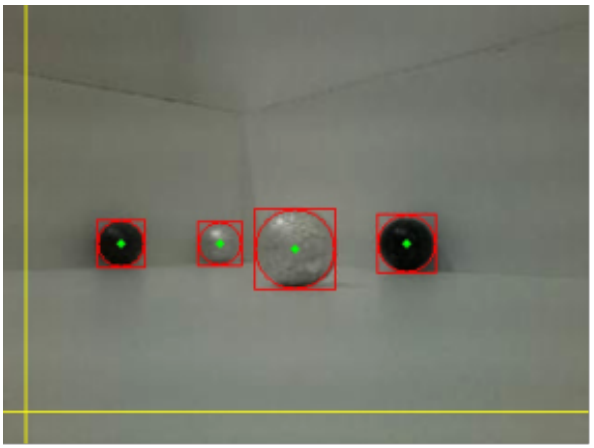
Nessas 120 fotos, foi realizado um processo manual de demarcação das esferas, pela seleção do ponto que representa seu centro, e de um outro ponto, no extremo da esfera, para que pudesse ser calculado o seu raio.

Figura 1: Fotos tiradas de ambiente similar ao da competição.

Obteve-se, portanto, para cada imagem, o centro da esfera (*xe*, *ye*) e seu raio *re*.

Além de seu centro e raio, foram obtidos também os 4 pontos que definem aproximadamente o quadrilátero mínimo que engloba a esfera. Esse quadrilátero não será utilizado neste trabalho, mas poderá ser útil para trabalhos futuros. Vê-se, na Figura 2, o resultado desse processo. Nela, as linhas amarelas representam a posição do cursor, já que esse teve um papel fundamental para a definição dos dados das esferas.

Figura 2: Esferas definidas.

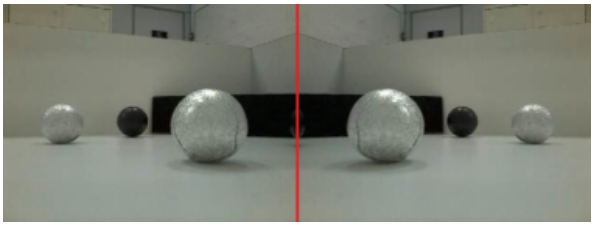


2.2 Ampliação de dados

O processo de ampliação de dados foi utilizado já que, quanto mais dados um algoritmo de aprendizagem de máquina tiver acesso, mais eficiente ele poderá ser (WANG; PEREZ, 2017). A quantidade de fotos obtidas (120 fotos) era muito pequena para que se pudesse treinar a rede neural, o que gerou a necessidade do processo de ampliação de dados, que consiste na aplicação de variações nas imagens originais, de modo que fossem geradas novas imagens, parecidas, porém não iguais, e que estas fossem introduzidas no treinamento da rede como situações diferentes das originais.

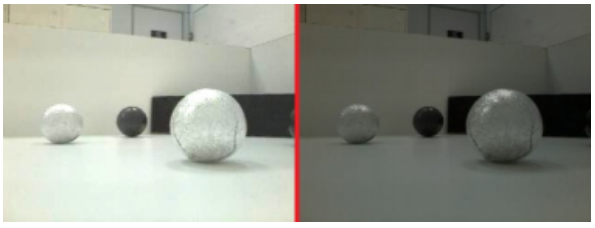
O processo de ampliação se deu da seguinte forma: Inicialmente, as 120 imagens originais foram invertidas horizontalmente, o que gerou mais 120 imagens, totalizando 240. Na Figura 3, pode-se ver, à esquerda, a imagem original, enquanto, na direita, pode-se ver a imagem invertida. A linha vermelha é a fronteira entre as imagens.

Figura 3: Inversão horizontal.



Com essas 240 imagens, geraram-se dois novos conjuntos de 240 imagens. O primeiro conjunto consiste das mesmas imagens, mas com o brilho aumentado. O segundo, com o brilho reduzido, para simular diferentes iluminações do ambiente, resultando em um total de 720 imagens. Na Figura 4, pode-se ver a mesma imagem da esquerda da Figura 3, porém com as alterações de brilho supracitadas.

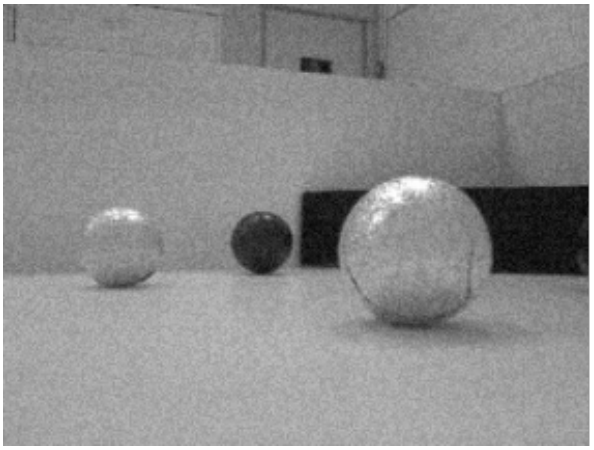
Figura 4: Alteração de brilho.



Por fim, às 720 imagens, foi acrescentado um ruído aleatório com a ajuda da biblioteca em Python Numpy. O resultado pode ser visto na Figura 5.

O resultado final do processo de ampliação de dados foram 1440 imagens diferentes entre si.

Figura 5: Imagem com ruído.



3 ABORDAGEM DO PROBLEMA

A abordagem do problema consistiu na utilização de uma rede neural. O procedimento adotado foi o seguinte: Primeiro, dividiu-se o problema em duas partes. Uma primeira análise na imagem seria necessária para confirmar se havia, de fato, esferas a vista e então, caso houvesse, uma segunda análise deveria determinar a localização da esfera mais próxima.

Foram utilizadas as seguintes camadas nas redes: convolucional, *pooling*, *dropout* e densa.

Decidiu-se que, como o trabalho é com imagens, seria útil utilizar ao menos uma camada convolucional na rede, já que as redes convolucionais são usadas para aprendizado progressivo, ao invés de sofisticados modelos baseados em níveis progressivos de abstração. Isso relembra os modelos de visão que evoluíram ao longo de milhões de anos dentro do cérebro humano (GULLI; PAL, 2017).

Também seriam necessárias camadas de *pooling*, que são usadas para reduzir a dimensão de uma rede convolucional, conforme camadas de convolução são adicionadas (BERNICO, 2018).

As camadas de *dropout* foram utilizadas, de modo a também resolver o problema de sobreajuste, já que elas desativam aleatoriamente as saídas dos neurônios da camada anterior, interrompendo o fluxo de informações, fazendo com que nem todos os dados da camada anterior prossigam na rede (TENSORFLOW, 2020).

Por fim, há as camadas densas, sendo elas camadas de neurônios completamente conectados, fazendo com que as informações se transmitam entre todos os neurônios da camada (GULLI; PAL, 2017).

Levando em conta a divisão do problema, foram elaborados dois modelos, que funcionam em série. O primeiro modelo, doravante chamado de Rede Alpha, tem

a função de realizar um pré-processamento da imagem recebida, de modo a determinar se, naquele instante, alguma esfera está no campo de visão do robô. A Rede Alpha foi treinada de modo a gerar uma classificação binária, tendo como saída dois neurônios, sendo que o primeiro indica o quanto ela está confiante de que há alguma esfera e o segundo o quanto ela está confiante de que não há. A proporção dos dados de treino é de 3 a cada 4, sendo os demais para teste. Caso o nível de confiança da Rede Alpha esteja acima de um limiar definido (para esse trabalho foi usado 80%), a imagem é então transmitida para uma segunda rede, que chamamos de Rede Beta, que é responsável por determinar a posição da esfera mais próxima do robô. A Rede Beta foi treinada com a mesma proporção de dados que a Rede Alpha, porém com uma quantidade menor de imagens. Como 216 das 1440 imagens não possuíam esferas a vista, a Rede Beta foi treinada com 1224 imagens. Seu formato de saída é o vetor V, tal que V = (P,r), P = (x, y), sendo P o ponto na imagem que indica o centro da esfera mais próxima, e r o raio dessa esfera. Considerando as informações mencionadas anteriormente e após a realização de alguns testes, a arquitetura final do modelo foi aquela das Tabelas 1 e 2.

Tabela 1: Arquitetura da Rede Alpha

Tipo de camada	Quantidade de neurônios
Convolucional	30
Pooling	30
Dropout	30
Flatten	Não se aplica
Densa	20
Densa	20
Dropout	20
Densa	2

Tabela 2: Arquitetura da Rede Beta

Tipo de camada	Quantidade de neurônios
Convolucional	40
Pooling	40
Convolucional	50
Pooling	50
Convolucional	40
Pooling	40
Dropout	40
Flatten	Não se aplica
Densa	35
Densa	25
Densa	3

Essa construção foi feita por meio da biblioteca de inteligência artificial Keras (GULLI; PAL, 2017) sobre a plataforma Tensorflow (TENSORFLOW, 2020). Cada rede foi treinada por aproximadamente 3000 épocas, utilizando a plataforma *online* Google Colab. A rede só precisa ser treinada uma vez. Portanto, essa parte do processo não dependeria da capacidade de processamento do computador embarcado no robô.

O algoritmo resultante dos dois modelos em série foi chamado de NeuralRescue.

4 RESULTADOS E DISCUSSÕES

4.1 Acurácia

Após a conclusão dos modelos, foram realizados testes com as 1440 imagens, das quais o NeuralRescue foi capaz de detectar esferas em 1225, havendo 1 falso positivo, considerando que a quantidade real de imagens com esferas é 1224. Usando como métrica a diferença entre o ponto conhecido (xe, ye, re) e o ponto resultante da rede (xo, yo, ro) foram calculadas algumas informações. A partir das diferenças calculadas entre os pontos, foram gerados histogramas de erro em cada componente do vetor. Foram consideradas as imagens cujo erro pertencia ao intervalo [-20,20].

Na Figura 6, está apresentado o histograma do erro na componente X. A quantidade de imagens consideradas, dada a restrição de intervalo imposta, citada no parágrafo anterior, é de 1208, o que corresponde a aproximadamente 98,7% do total de imagens com esferas.

Para a componente Y, 1221 imagens obedeciam à restrição, o que corresponde a aproximadamente 99,7% das imagens com esferas. O histograma é apresentado na Figura 7.

Por fim, para erro dos raios, as mesmas 1221 imagens do histograma anterior satisfizeram a condição. O histograma de R está apresentado na Figura 8.

Para se ter uma melhor visualização do quão acurado foi o algoritmo, a Figura 9 compara em algumas imagens o vetor definido como correto (em azul), com o vetor estimado pelo algoritmo (em vermelho).

As informações mais detalhadas sobre os histogramas estão apresentadas na Tabela 3. Os dados foram arredondados para duas casas decimais, por finalidades práticas.

Por meio da análise do histograma e dos dados na Tabela 3, pode-se observar que, para X, em aproximadamente 81,29% dos casos, o erro se encontra no intervalo [-3,3], enquanto para Y, nesse mesmo intervalo, estão aproximadamente 92,89% dos casos. Para R, nesse intervalo estão 94,28% dos casos.

Figura 6: Histograma do erro em X.

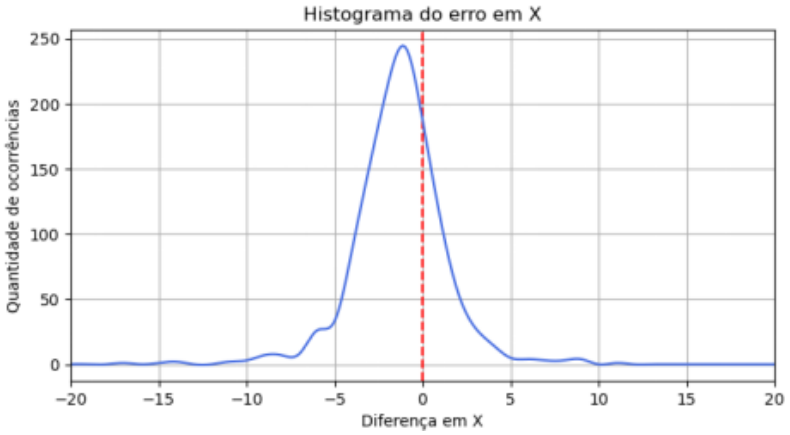


Figura 7: Histograma do erro em Y.

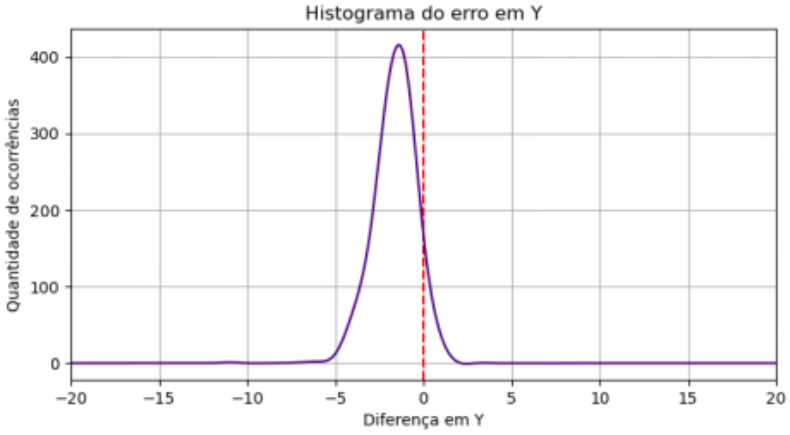


Figura 8: Histograma do erro em R.

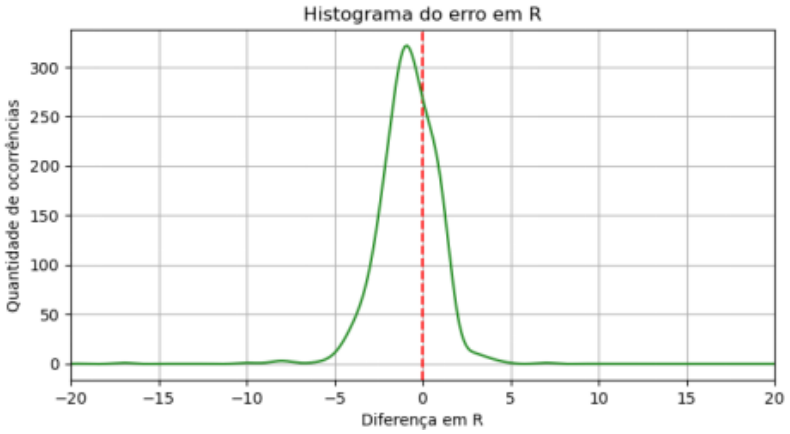


Figura 9: Comparação de detecção.

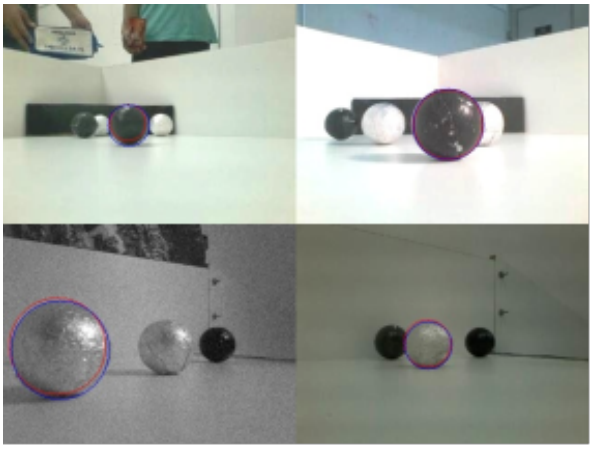


Tabela 3: Informações sobre os histogramas

Dado	Valor (em pixels)
Média do erro em X	-1,31
Média do erro em Y	-1,61
Média do erro em R	-0,81
Desvio padrão em X	2,62
Desvio padrão em Y	1,25
Desvio padrão em R	1,72

4.2 Desempenho

Durante o teste com as 1440 imagens, levou-se em consideração também o tempo utilizado para o seu processamento. Foi utilizado um notebook cujas especificações relevantes para a análise estão citadas na Tabela 4 e também uma Raspberry Pi modelo 3B, cujas especificações estão na Tabela 5.

Tabela 4: Especificações do primeiro sistema de testes (notebook)

Dado	Especificação
Sistema operacional	Windows 10 Home, 64 bits
Processador	AMD Ryzen 5
Placa gráfica	AMD Radeon Vega 8
Frequência do processador	2,5 GHz
Núcleos	4

Com essas especificações, o tempo médio levado para identificar se havia ou não esfera na imagem foi de aproximadamente 61,74 ms. Uma vez identificada uma esfera na imagem, foram levados, em média, mais 60,42 ms para encontrar a sua posição.

Nesse segundo sistema o desempenho foi bem inferior ao primeiro. O tempo médio levado para identificar se havia ou não esfera na imagem foi de aproximadamente 300,28 ms. Uma vez identificada uma esfera na imagem, foram levados, em média, mais 330,45 ms para encontrar a sua posição.

Tabela 5: Especificações do segundo sistema de testes (Raspberry Pi)

Dado	Especificação
Sistema operacional	Raspbian Buster, 64 bits
Processador	Broadcom BCM2837
Placa gráfica	Broadcom VideoCore IV
Frequência do processador	1,2 GHz
Núcleos	4

Pode-se considerar, para uma maior precisão, o tempo de aquisição da imagem. Para um novo teste foi utilizada uma câmera IP, de modo que o programa obtinha as imagens através do protocolo HTTP. O tempo médio de aquisição de imagem, para 1000 testes realizados, foi de 64,64 ms através da interface USB 2.0 e 64,43 ms através da interface Wifi 2.4 GHz.

5 CONCLUSÃO

Sobre a acurácia, diante dos resultados obtidos, uma possível conclusão é a de que o NeuralRescue é confiável, contanto que o método utilizado para a recuperação das esferas admita uma certa variação do algoritmo responsável por identificá-las. Na maior parte dos casos, a variação da detecção fica no intervalo [-3,3]. Portanto, pode-se projetar um sistema de recuperação das esferas que admita essa variação.

Quanto ao desempenho em tempo, nos casos médios em que há esfera na imagem e é utilizado o hardware do primeiro sistema, o tempo levado é de 186,59 ms (incluindo o tempo de aquisição e processamento da imagem). Portanto, em uma plataforma de hardware semelhante à utilizada, pode-se utilizar a câmera com uma frequência de 5 FPS de maneira segura.

No caso de um sistema embarcado com hardware semelhante à Raspberry Pi utilizada, o tempo médio total é de aproximadamente 695,16 ms, para o qual é recomendada uma frequência de 1 FPS.

Ainda há espaço para a busca de melhoria na acurácia. Uma alternativa que pode ser testada é a aquisição de mais dados, aumentar a complexidade do modelo, ou apenas trocar os parâmetros da rede.

Quanto à melhora do tempo, pode-se tentar obter um resultado melhor utilizando um hardware dedicado, mais potente, que seja projetado especificamente para essa finalidade, como alguns computadores da empresa Google, ou, no caso de sistemas embarcados, uma plataforma como a Latte Panda. Como trabalho futuro, portanto, há possibilidade de análise da viabilidade de implementação do algoritmo de forma embarcada em um robô real.

REFERÊNCIAS

BERNICO, M. **Deep learning quick reference: useful hacks for training and optimizing deep neural networks with TensorFlow and Keras**. 1. ed. Birmingham: Packt Publishing Ltd, 2018.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. 1. ed. Cambridge: MIT press, 2016.

GULLI, A.; PAL, S. **Deep learning with Keras**. 1. ed. Birmingham: Packt Publishing Ltd, 2017.

SILVA, V. L. d.; NASCIMENTO, E. C. d.; KASHIWAGI, M. Detecção de objetos em ambiente controlado por meio de mapeamento, utilizando uma plataforma robótica arduino. **Mostra Nacional de Robótica**, v. 9, n. 9, 2019.

SILVA, V. L. d.; SILVA, V. d. S.; SILVA, D. V.; MIRANDA, G. F.; SANTOS, N. G. B. S.; JUNIOR, W. R. G.; KASHIWAGI, M.; NASCIMENTO, E. C. d. Uso de algoritmo genético para sintonia de processamento de imagem. **Mostra Nacional de Robótica**, v. 9, n. 9, 2019.

TENSORFLOW. **Uma plataforma completa de código aberto para machine learning**. 2020. Disponível em: <https://www.tensorflow.org/>. Acesso em: 23 out. 2020.

WANG, J.; PEREZ, L. The effectiveness of data augmentation in image classification using deep learning. **Convolutional Neural Networks Vis. Recognit**, v. 11, n. 1, p. 1–8, 2017.