VISÃO COMPUTACIONAL APLICADA À OLIMPÍADA BRASILEIRA DE ROBÓTICA

Vitor dos Santos Silva - 3º ano do Curso Técnico Integrado ao Ensino Médio em Automação Industrial1

Derek Vieira Silva - 1º ano do Curso Técnico Integrado ao Ensino Médio em Automação Industrial1

Guilherme Fortunato Miranda - 1º ano do Curso Técnico Integrado ao Ensino Médio em Automação Industrial1

Nicolas Gabriel Bomfim Souza Santos - 1º ano do Curso Técnico Integrado ao Ensino Médio em Automação Industrial1

Pedro Eduardo dos Santos - 2º ano do Curso Técnico Integrado ao Ensino Médio em Automação Industrial1

Pedro Murilo Silva - 2º ano do Curso Técnico Integrado ao Ensino Médio em Automação Industrial1

Tutor: Vera Lúcia da Silva1, Professores Colaboradores: Wagner Roberto Garo Júnior1, Masamori Kashiwagi1 e Raphael Antônio de Souza1

verals@ifsp.edu.br, wagner.garo@ifsp.edu.br, masamori@ifsp.edu.br, raphael@ifsp.edu.br

[1] INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO - IFSP
Suzano – SP

Categoria: ARTIGO BÁSICO

Resumo: Este trabalho tem como objetivo propor uma solução ao desafio da Olimpíada Brasileira de Robótica (OBR), no que diz respeito ao resgate das esferas presentes na Sala 3. Com a utilização de visão computacional, as esferas são localizadas através da análise das imagens adquiridas por uma câmera. Após adquiridas, as imagens passam por diversos procedimentos para torná-las ótimas para o processamento que retornará o resultado desejado. Foram utilizados a câmera de um celular para o recebimento das imagens, o computador de placa única Raspberry Pi com o Sistema Operacional Raspbian e a biblioteca OpenCV para o processamento e demonstração de resultados ao usuário. Também foi utilizado um bloco de processamento Lego EV3 com a plataforma LeJOS para controle do robô. A comunicação entre os dois componentes é por meio de WiFi, processo no qual o celular funciona como ponto de acesso e Raspberry Pi como cliente, estabelecendo uma rede local entre os dois, independente de Internet. A visão computacional se caracteriza como uma abordagem diferente, pois a maior parte das soluções utiliza-se sensores de distância, como o ultrassônico. O resultado foi satisfatório, embora houvesse um pouco de ruído, o que não teve um impacto muito grande no processo.

Palavras Chaves: Visão Computacional, Processamento de Imagem, Análise de Imagem, OBR, C++, Java.

Abstract: *The purpose of this work is to propose a different solution to the challenge from the Brazilian Olympiad of Robotics (OBR), concerning the rescue of the spheres on the third room. Utilizing computational vision, the spheres are located through the analysis of the images acquired with the camera. After they are acquired, the images go through many procedures to make them optimal for the processing that will return the desired result. A camera from a phone was used to retrieve the images and the single board computer Raspberry Pi with the operational system Raspbian and the library OpenCV for the processing, and to show the user the results, besides the usage of a processing brick Lego EV3 running the Lejos interface to control the robot. The communication between these two components is established through WiFi, process in which the phone works as an access point, and the Raspberry Pi as a client, establishing a local network between the two, regardless of internet access. The computational vision characterizes as a different approach since most of the solutions use distance sensors, such as the ultrasonic. The result was satisfactory, although there was a small amount of noise, which did not have a big impact on the process.*

Keywords: *Computational Vision, Image Processing, Image Analysis, OBR, C++, Java.*

1 INTRODUÇÃO

A Olimpíada Brasileira de Robótica (OBR) é uma competição que visa estimular os alunos da rede pública e privada a desenvolver robôs autônomos para a resolução de determinados desafios que simulam possíveis situações reais de busca e resgate. Entre esses desafios estão curvas sinuosas, falhas no percurso, redutores de velocidade, obstáculos, entre outros.

No entanto, um desafio em específico tem sido uma grande dificuldade entre os estudantes: A sala 3. Essa propõe que o robô localize esferas prateadas que simbolizam vítimas de um acidente, capture-as e coloque-as em um local seguro, chamado Zona de Resgate. Porém, com os sensores comumente usados, esse desafio tem se tornado uma tarefa difícil [Nascimento e Lacerda, 2018]. Muitas das soluções apresentadas para a sala 3

não envolve a identificação das vítimas. Os robôs projetados fazem uma varredura por toda a sala 3, pega todas as esferas, armazenando-as em compartimentos no robô e depois as colocam na zona de resgate.

Desta forma, este trabalho propõe uso de técnicas de processamento de imagem para resolver os desafios propostos pela sala 3. O processamento de imagem exige maior capacidade de processamento, fazendo-se necessário a construção de um robô híbrido, utilizando o Raspberry Pi e o controlador EV3. Devido à necessidade de rapidez no processo, foi escolhida a Linguagem C++ para o programa que processa as imagens no Raspberry Pi e para a detecção da vítima foi utilizada a transformada de Hough para círculos [Dorini, Rocha, 2011]. A programação do controlador EV3 foi realizada com a Linguagem Java, plataforma LeJOS [Silva, 2018].

A seção 2 apresenta o trabalho proposto e a seção 3 os materiais e métodos utilizados. Em seguida são descritos as técnicas para o processamento de imagem, os resultados e a conclusão do trabalho.

2 O TRABALHO PROPOSTO

Foi proposto ao grupo o desafio de desenvolver um robô autônomo, utilizando o kit de desenvolvimento Mindstorms EV3 da Lego, para resolver os desafios propostos pela OBR. O robô projetado deveria ser capaz de seguir uma linha, fazer curvas, seguir por um percurso com diversos obstáculos, subir uma rampa e resgatar as vítimas para um local seguro.

A primeira parte dos desafios (seguir linha e o percurso) foi resolvida sem grandes problemas, embora com eventuais falhas. No entanto, a última parte do desafio levou um grande tempo, e uma solução satisfatória não tinha sido encontrada. Assim, este trabalho propõe aplicar técnicas de processamento de imagem para resolver os desafios da Sala 3 e o resgate de vítimas. A Figura 1 exibe o robô projetado com uma vítima resgatada, assim como o acoplamento do celular com a câmera logo abaixo da garra.

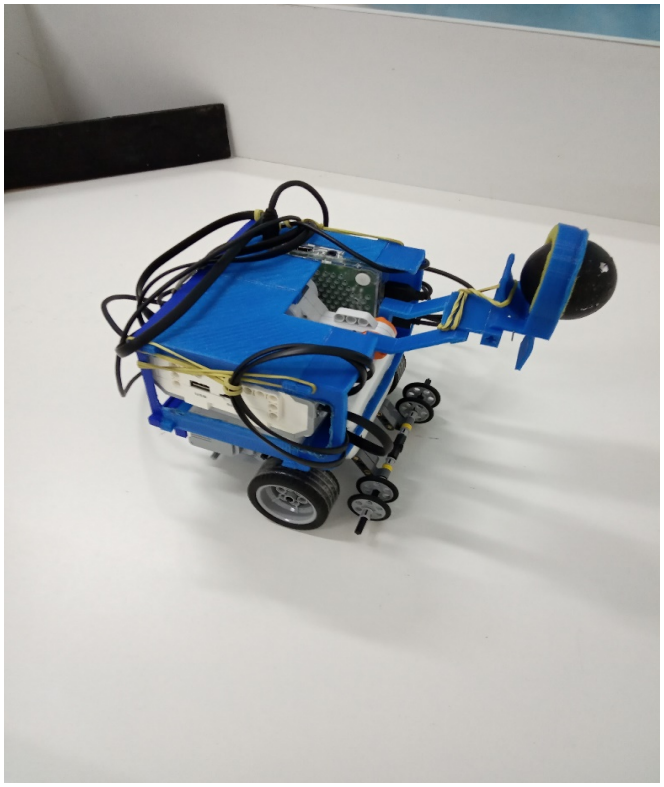

Figura 1 - Robô com vítima

3 MATERIAIS E MÉTODOS

O robô projetado é composto pelos componentes: um sensor de distância ultrassônico, dois sensores de cor e três motores como atuadores, um controlador EV3, um celular e um computador de placa única Raspberry Pi. Também foram usados um computador *desktop* para visualização remota da Raspberry Pi, e uma réplica da sala 3, localizada no laboratório de robótica da instituição.

Para a programação do robô foram usadas as seguintes IDEs: Eclipse para a programação em Java, e Geany para a programação em C++. Para o processamento de imagem foi utilizada a biblioteca OpenCV [OpenCV, 2019].

O computador recebe as imagens da câmera, que são passadas por um tratamento *frame* por *frame*, e na sua etapa final envia ao bloco de processamento EV3 a posição no eixo X da imagem do centro da esfera, para que o robô possa ajustar-se, até que a esfera esteja centralizada. Então, após ter a vítima centralizada, o robô pode capturá-la.

Após ter a vítima recuperada, o controlador envia um sinal ao computador, informando que a vítima está em sua posse, e que deve procurar o local em que ela deverá ser colocada em segurança.

Assim, em vez de procurar por círculos, agora o computador deve analisar as imagens em busca de um retângulo preto, para que possa centralizar-se. O processamento é realizado, e quando é detectado um retângulo preto, a posição de seu centro em X é enviada ao robô, até que este o tenha centralizado, para então poder se dirigir até a posição indicada, e colocar a vítima em segurança. O processo então se reinicia até que todas as vítimas sejam resgatadas.

O algoritmo passou por diversos testes, nos quais o robô era colocado no centro da sala 3 e as esferas eram posicionadas em lugares aleatórios na sala. O algoritmo era iniciado e o robô começava a girar em torno de si mesmo, com o celular em sua parte dianteira.

Enquanto girava, o robô procurava por círculos. Quando encontrava um, centralizava-o, andava até a esfera por um determinado tempo, resgatava-a e reiniciava o giro, procurando pela zona de resgate. Quando essa era encontrada, o robô andava até ela por um determinado tempo, e colocava a esfera em segurança.

Para avaliar a precisão e funcionamento do algoritmo, um membro da equipe observava o processo, percebia as partes com eventuais problemas, e ajustava o algoritmo para resolver o problema em questão.

Os testes foram realizados por diversas vezes, até que o algoritmo atingisse um desempenho aceitável.

4 PROCESSAMENTO DE IMAGEM

Essa seção apresenta as técnicas de processamento de imagem adotadas.

4.1 Desfoque Gaussiano

O desfoque gaussiano é uma técnica de redução de ruído, que consiste em aplicar a cada *pixel* da imagem uma máscara de convolução com uma distribuição normal de Gauss, em relação aos *pixels* em volta. Também pode ser definido como um "operador de convolução 2-D que é usado para 'desfocar'

imagens e remover detalhes e ruídos" [Fisher, 2003]. Isso resulta em uma imagem menos detalhada, o que reduz a quantidade de ruído, e melhora o resultado do algoritmo. No entanto, caso a máscara de convolução seja muito grande, o desfoque é muito forte, e perde-se esferas que estejam longe da câmera, além de levar um tempo maior para o processamento.

4.2 Localização de Vítimas

Para o robô detectar a vítima, primeiro ele recebe um *frame* da câmera, que é guardada no objeto de uma matriz, definido pela biblioteca OpenCV. A Figura 2 ilustra a leitura da imagem sem o processamento.

Figura 2 - Vítima sem processamento

Essa primeira imagem é convertida para escala de cinza, diminuindo a diversidade de cores encontradas na imagem, e reduzindo a chance de falhas, como demonstrado na Figura 3.

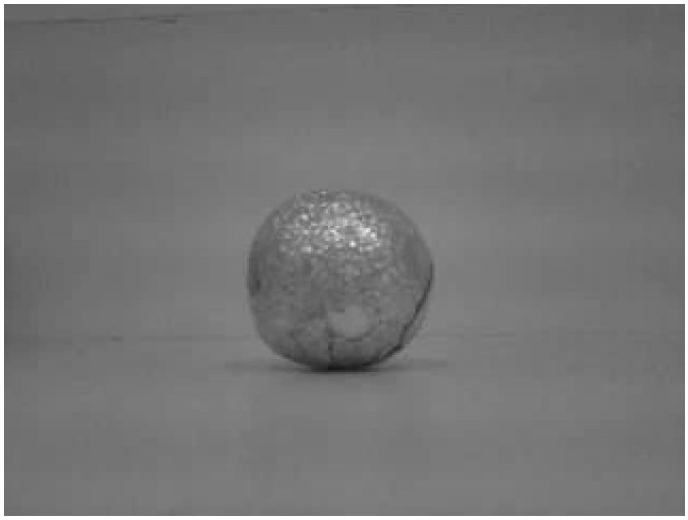

Figura 3 - Vítima em cinza

Após ser convertida para escala de cinza, é aplicado o desfoque gaussiano, que foi citado previamente. Deste modo, o ruído é reduzido. Na Figura 4 pode-se ver a imagem após o desfoque.

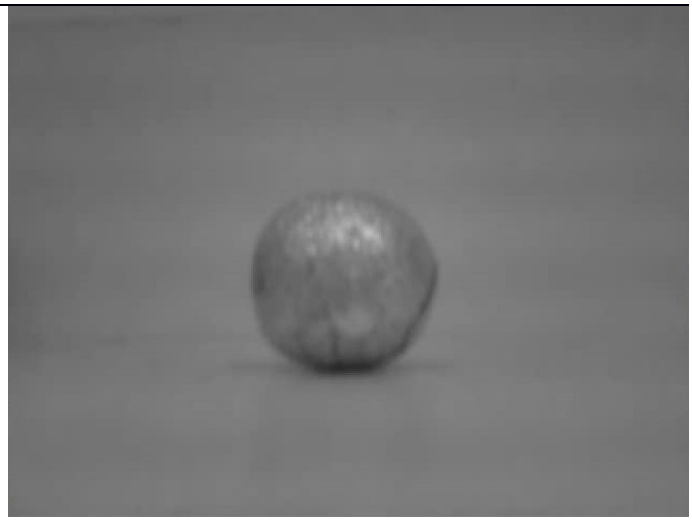

Figura 4 - Vítima desfocada

A transformada de Hough, função disponível na biblioteca OpenCV, e cujo funcionamento é além do escopo desse artigo, é aplicada, e um círculo vermelho é traçado em torno da vítima, e o ponto de seu centro é marcado com um círculo verde. Na Figura 5 abaixo pode-se ver o círculo encontrado, que é traçado na imagem original.

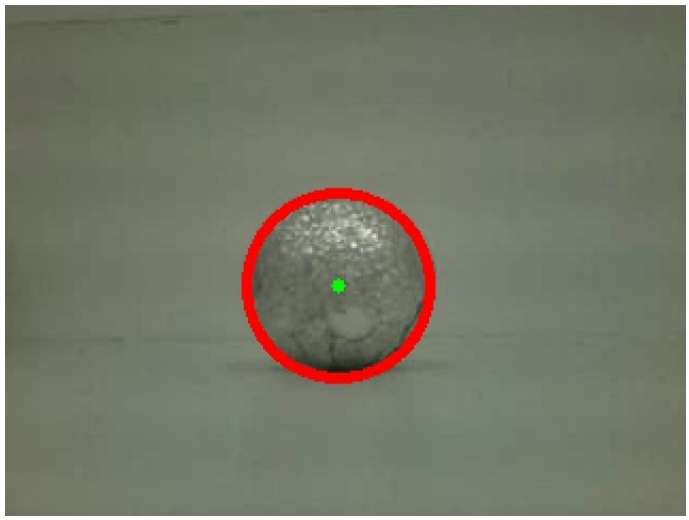

Figura 5 - Transformada de Hough

O valor de X desse ponto é então passado ao robô, que calcula a diferença desse valor e o valor desejado, e ajusta o robô até que os valores sejam iguais. Existe uma tolerância de aproximadamente 2 *pixels* para mais ou para menos.

4.3 Localização da zona de resgate

Após recuperar a vítima, o robô envia uma mensagem pra o computador, que começa a aplicar um processo diferente às imagens recebidas, agora procurando por um retângulo preto.

Primeiro, recebe um *frame* sem nenhum processamento, que novamente é guardada em um objeto fornecido pela biblioteca OpenCV. A Figura 6 ilustra a imagem obtida.

Figura 6 - Zona de Resgate sem processamento

Em seguida é aplicado um desfoque gaussiano nessa imagem, para que se reduza o ruído, conforme Figura 7.

Figura 7 - Zona de Resgate desfocada

A imagem é passada por uma operação que gera uma nova matriz. É fornecida uma faixa de cores, e essa operação gera uma nova matriz, na qual os *pixeis* da imagem de entrada que se encontrem nessa faixa são colocados como branco, e os que não se encontram são colocados como preto. Isso é realizado para que apenas o retângulo preto (zona de resgate) seja localizado, e os outros retângulos não interfiram no processo. A Figura 8 ilustra esse resultado.

Figura 8 - Limiar de imagem

Após esse procedimento, o programa procura por contornos, e após encontrar o maior contorno, verifica se ele possui quatro lados. Caso a condição seja satisfeita, é desenhado um contorno vermelho em volta desse retângulo, e seu ponto central é marcado com um círculo verde, conforme ilustra a Figura 9.

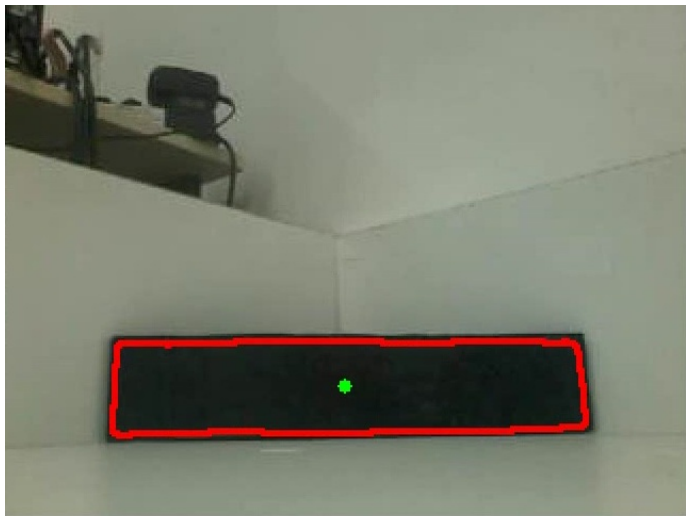

Figura 9 - Zona de Resgate final

O ponto identificado é informado ao robô, para que ele possa ser centralizado, e se dirija até a Zona de Resgate e coloque a vítima em segurança.

5 RESULTADOS E DISCUSSÃO

O processamento de imagem se mostrou bem eficaz para a localização do que era necessário como pode ser visto nas imagens anteriores. Apesar de sua eficiência, algumas situações se provaram problemáticas para o algoritmo.

Quando a esfera se encontra muito longe da câmera, o desfoque impede que a transformada de Hough detecte-a. Caso o desfoque seja diminuído, quando a esfera se encontra muito perto, há muito ruído na imagem.

A iluminação também causa muita influência. Em ambiente mais escuros, a zona de resgate pode ser localizada mais facilmente, pois mais *pixels* se encontram na faixa de procura, enquanto em ambientes com uma iluminação mais forte, os

pixels ficam mais claros, portanto, menos *pixels* estão posicionados na faixa de procura.

6 CONCLUSÕES

Em suma, o grupo chegou à conclusão de que este método para resolução da sala 3, embora ainda possua diversos ajustes a serem feitos, tem um grande potencial de sucesso se desenvolvido corretamente, e aprimorado ao longo do tempo. Pode-se considerar também a utilização de um módulo de câmera para Raspberry Pi, em vez de um celular, devido ao seu volume, que ocupa mais espaço do que um módulo de câmera. O grupo, no entanto, optou por usar o celular devido às outras necessidades do projeto, que poderiam ser supridas com o uso do mesmo.

Para uma maior simplicidade na programação, pode-se considerar a utilização da Linguagem Python, a qual possui uma versão da biblioteca OpenCV.

Mais camadas de filtragem também podem ser adicionadas, na tentativa de obter melhores resultados.

REFERÊNCIAS BIBLIOGRÁFICAS

Dorini, L. E. B., Rocha, A. de R. (2011). Detecção de círculos em imagens através da transformada de Hough. Disponível em: <https://www.ic.unicamp.br/~rocha/msc/ipdi/deteccao_formas_circulares.pdf>. Acesso em: 18 ago. 2019

Fisher, R., et al. (2003). Gaussian Smoothing. University of Edinburgh. Disponível em: <https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm>. Acesso em: 18 ago. 2019.

Nascimento, E. C; Lacerda, A. O. (2018). Método para detecção de rampa e vítimas no desafio proposto pela Olimpíada Brasileira de Robótica. *Mostra Nacional de Robótica, 2018.*

OpenCV. (2019). OpenCV Tutorials. Disponível em: https://opencv.org/. Acesso em: 18 ago. 2019

Silva, V. S; Jesus, A. C; Coura, A. L. S; Rodrigues, A. S. S. (2018). Robô seguidor de linha autônomo utilizando Java. *Mostra Nacional de Robótica, 2018.*

Observação: O material multimídia deste trabalho encontra-se disponível em: www.mnr.org.br/mostravirtual.