GIMu 2.0: Descrição da solução para o desafio da IEEE Open 2017

Camila Barbosa, Gabriel Vantuil, Samuel Cavalcanti,
Tiago Almeida, Wendell Alves, William Kossmann e Orivaldo Santana

***Abstract*— Esse TDP apresenta o esboço do plano para solução do problema proposto pela IEEE Latin American Robotics Competition. São descritos os principais aspectos para resolução do desafio, bem como os recursos mais importantes a serem utilizados pela equipe.**

I. Introdução

Os desafios propostos pela IEEE Latin American Robotics Competition têm como objetivo levar às competições problemas reais encontrados no cotidiano das pessoas e indústrias. Nas edições de 2016 e 2017 da competição é abordada a questão da qualidade na produção do setor alimentício.

O processo atual da produção de leite envolve muito estresse sobre os animais, o que acarreta em um leite de qualidade inferior. Podemos melhorar isso ao mudar alguns fatores. A proposta desse desafio é fazer com que o habitat natural (pasto) deixe o animal mais confortável, para que assim os níveis de estresse do animal diminuam.

O problema proposto apresenta um cenário interessante para explorar técnicas de processamento de imagens, principalmente para a localização da miniatura de vaca. O método desenvolvido tem como saída o centro de massa do objeto e é baseado em alguns algoritmos conhecidos, como Shi-Tomasi Corner Detector, o clustering K-means e local binary patterns.

Dado o problema, propomos uma solução: um robô com uma garra e um sistema de visão que também será responsável pela ordenha e transporte do leite. Com recursos como Arduino, Raspberry Pi, OpenCV e ArUco [2], será possível apresentar essa solução.

As próximas seções deste artigo estão divididas como a seguir, a Seção II apresenta a estratégia adota para atacar o problema proposto no desafio IEEE *Open*. A Seção III introduz os principais recursos de Hardware que devem ser utilizados na estratégia descrita na Seção II, os softwares e bibliotecas usadas para implementação estão presentes na Seção IV. A Seção IV-A apresenta as técnicas de processamento de imagens adotadas para rastreamento do objeto de crucial importância para solução do problema. Os caminhos para validar partes da solução propostas são explicados na Seção V. A conclusão é apresentada na Seção VI.

II. Solução

A estratégia de solução do problema consiste em um único robô: especializado tanto na identificação e ordenha da vaca, quanto no transporte dos copos e armazenamento do ”leite” no local especificado.

O robô deverá ir em direção à zona de copos vazios, recolher um copo e seguir em direção a uma vaca, retirar o leite, colocá-lo no copo, e seguir em direção ao tanque de leite. A sequência de tarefas a serem realizadas pelo robô está descrita na lista abaixo:

1) Localizar a zona de copos vazios;
2) Pegar um copo;
3) Localizar uma vaca;
4) Navegar em direção a vaca;
5) Aproximar-se da vaca;
6) Retirar o leite da vaca;
7) Ir em direção ao tanque de leite
8) Despejar o leite no tanque
9) Retornar ao passo 1;

A primeira ação do robô é encontrar a região dos copos, a estratégia principal é localizar a tag que indica a região dos copos, aproximar-se desta região com auxilio de sensores de distância e de visão. Ao chegar na região dos copos o robô deverá alinhar-se em relação a parede dos copos usar a garra e sensores para fazer um ajuste fino da localização de um copo, em seguida acionar a garra para pegar o copo.

Para a localização e a navegação em direção à vaca será usado sensor de visão para encontrar uma região aproximada das vacas. A medida que o robô aproximar-se desta região uma vaca será escolhida. Para aproximar-se com precisão de uma vaca os sensores de distância serão utilizados para ajudar no ajuste fino do posicionamento do robô em relação a vaca.

Com o robô próximo e alinhado em relação a vaca, entra em ação o sistema de ordenha. Após ordenhar, o robô segue, com auxílio da orientação por tags (ArUco) e por cor, para o tanque de leite. Onde ele despejará o que ordenhou e retornará para o passo inicial. Esse ciclo deve se repetir até o final do tempo estipulado para uma rodada.

Alguns comportamentos importantes para a execução dos algoritmos do robô envolvido neste problema são:

- Localização de objetos no cenário tais como as vacas, o tanque e os copos;
- Localização de marcadores no cenário;
- Navegação em direção a um desejado objeto;
- Identificação de locais pelos quais o robô não deve seguir

A localização de objetos no cenário será feita através do sistema de visão do robô (câmera, Raspberry Pi e OpenCV). Esse deve ser capaz, através de algoritmos de atribuir *labels* a objetos e locais específicos no tabuleiro. A localização de marcadores no cenário também será realizada com o auxílio desse sistema. Além disso, a identificação de locais pelos quais o robô não deve seguir também utiliza de imagens

processadas pelo sistema de visão. Entretanto, conta com o auxílio de sensores posicionados estrategicamente no chassis do robô.

III. Hardware

O projeto do hardware do robô contêm um sistema de pneumático para sucção do leite e um sistema de braço e garra para pegar copos. Além disso, o robô contem um sistema de visão, um sistema de processamento de informações, um sistema de controle de atuadores e de coleta de dados dos sensores. O mecanismo de movimentação utilizará um conjunto de quatro rodas e quatro motores.

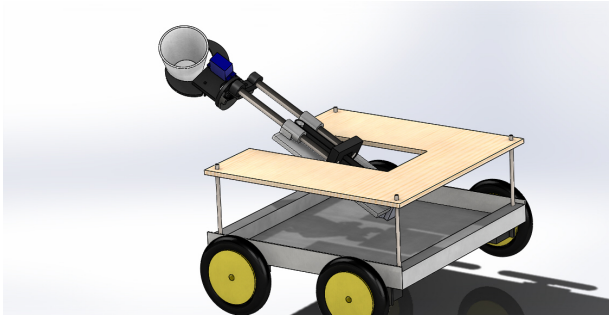


Fig. 1: Modelagem inicial do conjunto robô-garra

A construção de peças para o robô será realizada com o auxílio de uma impressora 3D. A impressora 3D será responsável pela produção de partes mecânicas presentes em várias partes do robô, tais como o braço robótico, suportes e adaptações para componentes.

O robô possui as seguintes características mecânicas principais:

- Um braço linear com uma garra para coletar e depositar o copo;
- Um sistema pneumático para retirada do ”leite” da vaca;
- Um tubo para direcionar o ”leite” do sistema de coleta para o copo;
- Um sistema linear de elevação para alcançar a altura do balde;
- Uma câmera e um conjunto de sensores para orientar as ações do robô;

O sistema de ordenha será composto de três partes:

1) bomba pneumática
2) acoplador
3) tubo de condução

A bomba pneumática, ver Figura 2, constituída por um motor de sucção, terá como principal função no sistema de ordenha a geração de uma pressão negativa sobre a luva. O acoplador será a parte responsável por encaixar o robô nos dedos da luva. Ele deve possuir a melhor vedação possível para evitar perda de pressão manométrica. O tubo de condução será por onde o fluxo de ar e de água fluirá da luva para o copo, passando pelo interior do motor.

O sistema de movimentação será composto por um conjunto de atuadores: servo motores e motores DC. A locomoção do robô tem como força motriz motores DC.

Fig. 2: Sistema de sucção.

O mecanismo de acionamento da garra é composto por servomotores e o de movimentação do braço é suportado por motores de passo.

IV. Software

A parte mais crítica para atingir o objetivo da IEEE *Open* são os algoritmos desenvolvidos com o auxílio da biblioteca OpenCV. Com a maioria do código em C++, planejamos realizar as tarefas descritas na secção II, com a implementação de rotinas de processamento de imagens. O OpenCV é uma biblioteca de visão computacional *Open Source* com foco em aplicações em tempo real [1].

O primeiro passo foi identificar objetos por cores, com esse recurso, torna-se possível para os robôs localizarem o tanque de leite e deslocarem-se em direção a ele. Esse algoritmo recebe cada *frame* do vídeo, em seguida analisa a cor de cada pixel buscando a desejada e substituindo por um pixel preto em uma imagem binária. Após esse procedimento, é aplicado um algoritmo de redução de ruídos, baseado em funções próprias do OpenCV (Erode e Dilate), facilitando, desse modo, a localização do maior objeto visível, de acordo com a quantidade de *pixels*.

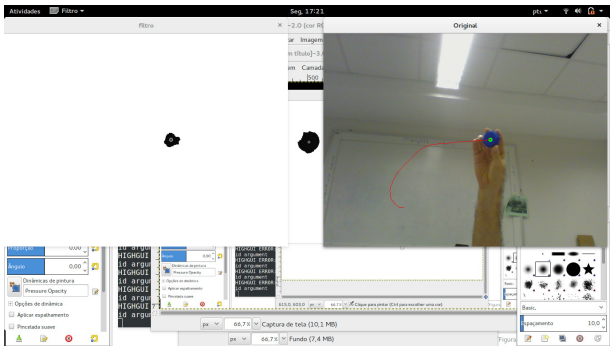
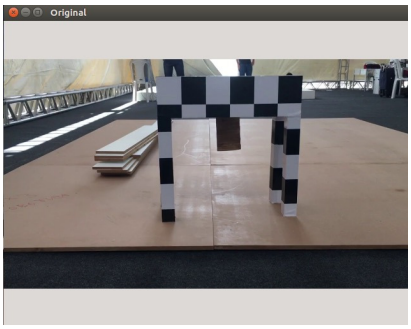


Fig. 3: Teste do codigo de rastreamento de objetos por cor.

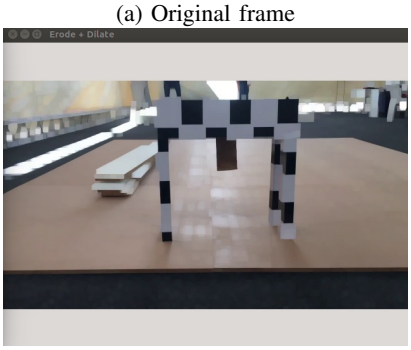
A localização dos objetos por cores e por tags é relativamente simples de implementar. As tags são localizadas com o uso da biblioteca ArUco, já que duas das tags presentes no desafio estão incluídas na biblioteca já consolidada. [2]

A. Algoritmo de visão computacional para reconhecimento inicial da vaca

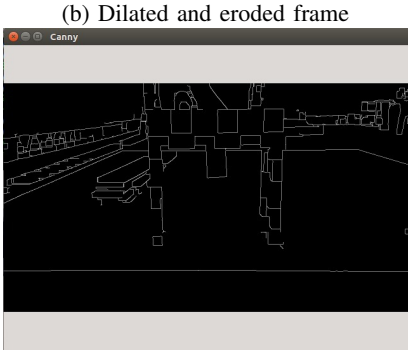
A vaca é representada como uma miniatura de madeira com 55 cm de altura [7]. A textura do objeto é um padrão xadrez com retângulos pretos e brancos de tamanho irregular (Fig. 4a). Esses aspectos foram cruciais para identificação do alvo.



(a) Original frame



(b) Dilated and eroded frame



(c) Frame with Canny Edge Detector

Fig. 4: Imagens da sequência de transformação aplicadas aos frames: a) Frame Original; b) Erode e Dilate; c) Cannz Edge Detector

Nós propomos um algoritmo que identifica e rastreia a miniatura de acordo com suas características visuais. Esses atributos diferenciam o alvo do ambiente ao seu entorno e filtram possíveis ruídos que possam ser causados por objetos com estrutura semelhante à da vaca.

O algoritmo implementado consiste em uma série de transformações na imagem, bem como associações geométricas baseadas nas características visuais do objeto. Tudo isso com o propósito de selecionar o que chamamos de *pontos cruciais*. Esses *pontos cruciais* são pontos (*pixels*) que têm uma alta probabilidade de pertencer ao alvo. Esse conjunto de pontos é, na verdade, o *training set* de um método de *clustering (machine learning)*, o qual tem como saída o centro de massa da vaca. Nosso algoritmo consiste em:

1) transformações de imagens para adequar os *frames* à associação geométrica;
2) identificação de linhas e cruzamentos (cantos, quinas);
3) intersecção dos conjuntos de pontos originados pelos métodos do passo anterior;
4) algoritmo de *clustering*.

Para identificar a vaca, a seguinte estratégia foi utilizada: selecionar todas as linhas e extrair apenas as paralelas. Após isso, uma quantidade razoável de pontos é adquirida (com detecção de *cantos* [3]) e então, aqueles que pertencerem a essas retas paralelas são considerados *pontos cruciais*. Consequentemente, temos um grupo de marcadores que são, em sua maioria, compostos de pontos que definem os limites da vaca.

Em quadros semelhantes à Fig. 4c (Com o Canny Operator [4]), o algoritmo Hough Lines é aplicado [6]. A teoria por trás desse método é que qualquer ponto em uma imagem binária poderia ser parte de um conjunto de linhas possíveis. Quando cada linha é parametrizada por, por exemplo, uma inclinação a e uma interceptação b, então um ponto na imagem original é transformado em um locus de pontos no plano (a, b) que corresponde a todas as linhas que passam por esse ponto [5]. Em outras palavras, o algoritmo vai analisar as bordas da figura e selecionar as que correspondem à equação de uma linha. É por isso que este método também pode ser usado para reconhecer outros elementos, desde que possam ser descritos com uma equação.

O conjunto inicial de linhas adquiridas com Hough Lines [6] certamente teve bastante ruído. Em qualquer ambiente onde o algoritmo poderia ser testado, seria muito provável a existência de uma grande quantidade de outras linhas que definissem elementos aleatórios no *frame*. Por esse motivo, mais filtros são necessários para diminuir ruídos na coleção de linhas.

Uma afirmação importante para resolver o problema é que o objeto alvo é composto apenas com bordas paralelas e perpendiculares [7]. Portanto, para diminuir o número de *linhas cruciais*, somente aquelas com a mesma inclinação que outras serão, agora, consideradas *cruciais*.

Nesse ponto, um pouco do ruído é retirado, mas é possível que, nos arredores, linhas paralelas que não delimitam as bordas do nosso alvo sejam identificadas pelo algoritmo. Como mencionado, um padrão determinado pode ocorrer em vários objetos naturais e artificiais. Portanto, outra abordagem é necessária: o Shi-Tomasi Corner Detector [3].

Nessa etapa do algoritmo existem duas grandes coleções de pontos: as que definem as linhas de 90 graus e as extraídas com o algoritmo de Shi-Tomasi (Fig. 7). O segundo conjunto ainda passa por uma filtragem mais: apenas os pontos que pertencem a uma linha são mantidos como *cruciais*. Esses dois grupos mesclados são os dados de treinamento do algoritmo de aprendizagem de máquina (K-means).

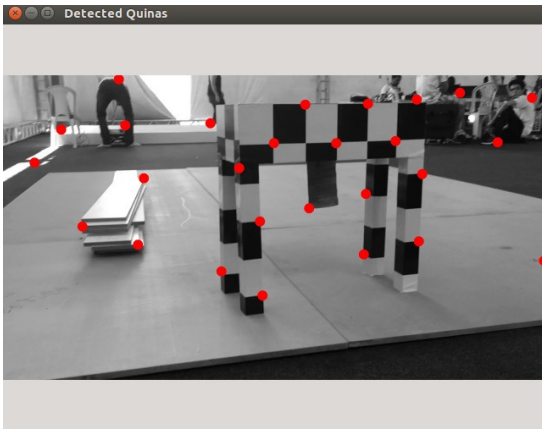


Fig. 5: Frame da vaca após aplicação do algoritmo Good features to Track

B. Algoritmo de aprendizado de máquina para rastreamento do centro de massa da vaca

Depois da implementação de alguns algoritmos de visão computacional, a maior parte dos *pontos cruciais* remanescentes estão no objeto a ser rastreado. Mas, ainda há pontos que são erroneamente selecionados. Isso acontece devido a características semelhantes entre o alvo e o ambiente ao redor dele. Esses ruídos, dependendo num variado número de fatores, bem como luminosidade, contraste, figuras semelhantes e intenso movimento, podem causar turbulência no resultado final. Por isso, um algoritmo de clustering é aplicado.

Como queríamos eliminar o ruído, um algoritmo de agrupamento com centroides rotulados como ”in” e ”out” do objeto alvo é uma boa solução. O método implementado é um K-Means simples. Consiste na atribuição desses rótulos a cada ponto (*os pontos cruciais*) com base na distância euclidiana entre eles. Em outras palavras, para cada ponto crucial, um cluster seria associado e seria aquele com a menor distância euclidiana [8], [9].

O K-Means [10] procede selecionando *k* centros de cluster iniciais e depois refinando iterativamente. O algoritmo converge quando não há mais mudanças na atribuição de clusters aos pontos. Apenas dois centroides foram necessários para identificar a vaca, logo $k = 2$: cada um identifica uma das pernas da vaca, os limites do objeto no $eixo - x$.

Os dados de treinamento desse algoritmo de aprendizado de máquina é basicamente um conjunto de pontos e tem 5 características:

- a coordenada x do ponto;
- a coordenada y do ponto;
- uma representação baseada em histogramas dos pontos brancos ao redor de (x,y);
- uma representação baseada em histogramas dos pontos pretos ao redor de (x,y);
- uma representação baseada em histogramas dos pontos ”médios” ao redor de (x,y);

Os dados de treinamento que entram no *loop* do algoritmo k-means tem 5 características, e a saída são dois centroides, dois pares (x, y). Esses pontos definem os limites horizontais da vaca e a média deles representa o centro de massa da miniatura de vaca. O resultado do algoritmo pode ser visualizado na Figura 6.

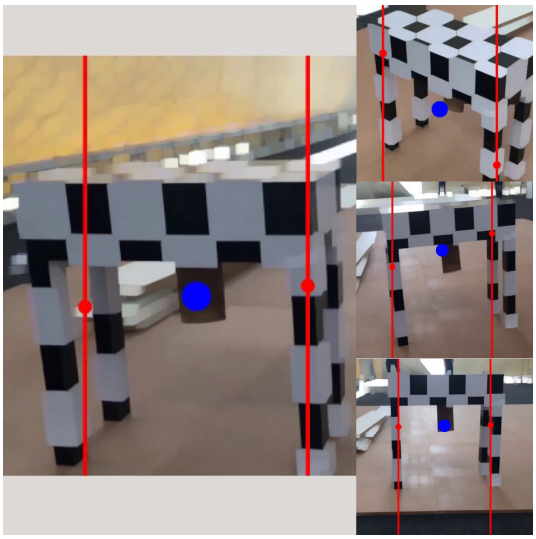


Fig. 6: Resultado final do código de rastreamento da vaca

V. Experimentos

A validação do sistema por completo ainda está em processo de realização. A princípio será dividida em duas partes:

1) Teste de algoritmos no ambiente de simulação V-Rep;
2) Teste final com o robô em um ambiente semelhante ao descrito pelo edital;

Desse modo, o sistema robótico passará por alguns pequenos testes. Logo, vários experimentos serão realizados e alguns -novos- protótipos serão propostos para resolver sub-tarefas dentro da solução geral do desafio. O robô e cada tecnologia descrita acima deverão ter suas funções testadas separadamente, bem como cada etapa principal para a solução do problema. A validação dessas etapas varia de acordo com o conjunto hardware/software que é responsável por tal função no robô. Segue uma lista das principais validações:

- Identificação dos principais objetos no terreno;
- Navegação por meio das principais áreas da arena;
- Reconhecimento da vaca;
- Reconhecimento das tags;
- Despejo do leite no depósito final.

A identificação dos principais objetos no terreno será feita através do sistema de visão de ambos os robôs. Através do reconhecimento por cores utilizando os recursos do OpenCV, cada robô deverá ser capaz de identificar os principais ”*spots*” do terreno e determinar o próximo passo a ser dado. Sendo assim, os testes partem do mais básico: identificar as cores e selecionar um range para independer da luminosidade; até a locomoção dos robôs ao local identificado através do sistema de visão.

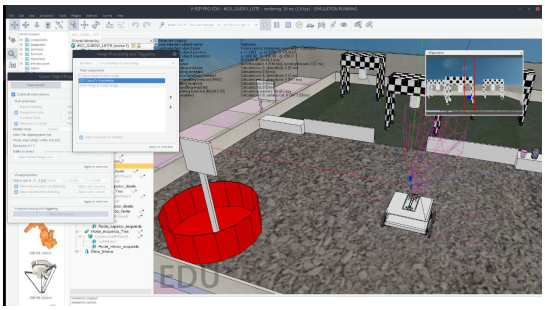


Fig. 7: *Print* da simulação integrada com a identificação da vaca

A partir da navegação por meio das principais áreas da arena será validada a API desenvolvida para proporcionar a locomoção dos robôs. Testes iniciais deverão ser feitos ainda sem automação (via Bluetooth ou ESP), para depois avançar com experimentos da locomoção baseada nos comandos enviados pelo OpenCV (já com integração de sistemas).

O reconhecimento das *tags* será feito com auxílio do módulo integrado ao OpenCV, ArUco [2]. Os experimentos realizados com essa parte do sistema serão semelhantes aos com o reconhecimento de cores. Logo, será necessário testar individualmente cada tag e certificar o reconhecimento correto dessas pelo robô.

Enquanto esses testes são realizados com partes independentes do sistema físico, a simulação tratará de testar a integração e a máquina de estados do robô. Até o momento, dentre outros testes, o principal que teve resultados satisfatórios, foi o de reconhecimento da vaca. Na Fig. 1 a demonstração do experimento no ambiente de simulação.

VI. Conclusão

Este artigo de descrição de equipe apresentou uma proposta de solução para o desafio IEEE *Open* 2016/2017. Essa solução consiste na construção de um robô para realizar a tarefa de coletar leite de uma vaca e depositá-lo em um tanque. O robô deve possuir habilidades de reconhecer marcas no ambiente para auxiliar a navegação e a execução da tarefa central do desafio. A solução proposta possui três principais sub-sistemas: sistema de visão, sistema de processamento de dados e um sistema de controle de atuadores. Uma sub-tarefa importante é a identificação da vaca, algumas técnicas de processamento de imagens e aprendizagem de máquinas serão exploradas e a mais adequada será utilizada para resolver esta sub-tarefa.

References

[1] Bradski, A. (2008), Learning OpenCV, [Computer Vision with OpenCV Library; software that sees], O'Reilly Media.

[2] Garrido-Jurado, S.; noz Salinas, R. M.; Madrid-Cuevas, F. & Marín-Jiménez, M. "Automatic generation and detection of highly reliable fiducial markers under occlusion Pattern Recognition", 2014, 47, 2280-2292;

[3] Shi, J. & Tomasi, C. "Good features to track", 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 1994, 593-600;

[4] Canny, J. A. "Computational Approach to Edge Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986, PAMI-8, 679-698;

[5] Kaehler, A. & Bradski, G. "Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library", O'Reilly Media, Inc., 2016;

[6] Duda, R. O. & Hart, P. E. "Use of the Hough Transformation to Detect Lines and Curves in Pictures", Commun. ACM, ACM, 1972, 15, 11-15;

[7] Rules of the IEEE Open category 2016-2017 [Online] Available: http://ewh.ieee.org/reg/9/robotica/Reglas/IEEE%20Open%202016-2017%20-%20English%20-%20Version%201.1.2.pdf

[8] Likas, A.; Vlassis, N. & Verbeek, J. J. "The global k-means clustering algorithm". Pattern Recognition, 2003, 36, 451 - 461;

[9] Kanungo, T.; Mount, D. M.; Netanyahu, N. S.; Piatko, C. D.; Silverman, R. & Wu, A. Y. "An efficient k-means clustering algorithm: analysis and implementation", IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, 24, 881-892;

[10] Wagstaff, K.; Cardie, C.; Rogers, S. & Schrödl, S. "Constrained K-means Clustering with Background Knowledge", Proceedings of the Eighteenth; International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., 2001, 577-584;