MONITORAMENTO DO CRESCIMENTO DE PLANTAÇÃO POR PROCESSAMENTO DE IMAGENS – UMA APLICAÇÃO INTEGRADA AO SMART CAMPUS E À HORTA AUTOMATIZADA DO IMT

Pedro Henrique Rodriguez Daniel [1]; Wânderson de Oliveira Assis [2]; Rogério Cassares Pires[3]; Fernando de Almeida Martins[3]

[1] Aluno de Iniciação Científica da Escola de Engenharia Mauá (EEM-CEUN-IMT);
[2] Professor da Escola de Engenharia Mauá (EEM-CEUN-IMT);
[3] Engenheiro do Centro de Pesquisas (CP-CEUN-IMT).

Resumo. *Este projeto visa o desenvolvimento de algoritmos para captura e processamento de imagens em horta automatizada, propondo monitorar o crescimento de plantação de soja. Para isso as imagens serão coletadas por meio de câmera IP (Internet Protocol) e tratadas utilizando aplicação desenvolvida em software pela biblioteca OpenCV (Open Source Computer Vision Library), sendo atribuídos filtros e algoritmos para detecção de contornos com a finalidade de determinar as características e dimensões da planta. As imagens captadas poderão ser visualizadas na interface do Smart Campus do IMT e os resultados serão integrados à aplicação de Internet das Coisas existente.*

Introdução

Nos últimos anos, o desenvolvimento de tecnologias na área da agricultura de precisão vem crescendo incessantemente devido à importância do agronegócio para a economia brasileira e ao incentivo público para a realização de pesquisas relacionadas ao tema. Esses projetos incluem metodologias de manejo sustentável e o emprego de tecnologias, como a Internet das Coisas, a qual é usada para designar processos que envolvam objetos físicos conectados em rede, capturando e processando informações instantaneamente e de forma autônoma, o que permite fornecer dados para o agricultor sobre o plantio e a safra, sendo os dados transmitidos em tempo real. Esses recursos tecnológicos trazem versatilidade no manuseio das lavouras, já que o acesso a informações de interesse permite benefícios como o acompanhamento do crescimento do plantio em tempo real, o rastreamento do maquinário agrícola, o monitoramento de informações meteorológicas e a otimização na distribuição de insumos. Tudo isso resulta em um melhor planejamento das ações e na tomada de decisões inteligentes e eficientes em relação à sua plantação, além do melhor aproveitamento da mão de obra.

A prática da agricultura no Brasil é um dos pilares centrais de sua economia, principalmente na exportação de grãos. De acordo com a Secretaria de Comércio e Relações Internacionais do Ministério da Agricultura, Pecuária e Abastecimento, o agronegócio

brasileiro exportou US $96,01 bilhões, em 2017, sendo a soja o produto mais exportado no país (GRANDCHAMP, 2021). Segundo o estudo realizado pelo IEDI (Instituto de Estudos para o Desenvolvimento Industrial), juntamente com dados coletados da OMC (Organização Mundial do Comércio), o valor da soja exportada no ano de 2020 foi em torno de US $ 28,56 bilhões, com aumento de 9,5% em relação ao ano anterior, totalizando 82.973.424 toneladas de soja exportadas (DATABRAS, 2020). Logo, para acompanhar esse crescimento significativo e continuar nesse progresso, é preciso inovar ano após ano em tecnologias de agricultura de precisão, visando sempre aprimorar a sustentabilidade ambiental e econômica da produção.

Considerando as aplicações tecnológicas existentes e voltadas para o ramo agrícola destaca-se o processamento e reconhecimento por meio de imagens de satélite e detecção de objetos. O Processamento de Imagens é um conceito relacionado à captação, tratamento e visualização de dados através de imagens capturadas por algum módulo de entrada (câmera, por exemplo) para reconhecer um determinado padrão. As aplicações deste conceito se baseiam na análise de sinais constituídos de dados referentes a imagens na busca por padrões e resultados quantitativos que representam uma "informação" estudada. Um exemplo é o projeto SISCOB (HONDA; JORGE, 2013), o qual se baseia em um software desenvolvido pela Embrapa (2009) que trabalha com técnicas de classificação e processamento de imagens digitais captadas por satélite, com o objetivo de fornecer informações ao agricultor sobre a cobertura do solo, quantidade de mudanças na cultura em um período determinado, doenças, pragas, e deficiências na lavoura, permitindo o monitoramento de sua safra em larga escala e em locais específicos do cultivo.

Neste sentido o objetivo deste projeto de iniciação científica é unificar os conceitos de Agricultura de Precisão, Processamento de Imagens e Internet das Coisas para produzir um algoritmo capaz de coletar imagens de diversas plantas, processá-las, reconhecer o tipo de plantio e apresentar os resultados, por meio de uma interface, ao produtor agrícola, em tempo real, o qual terá dados como dimensões das folhas e das plantas, suas características e colorações, e suas fases de crescimento. A proposta é integrar esta solução ao "Smart Campus Mauá", uma interface acessível na internet que incorpora diversos projetos realizados no Instituto Mauá de Tecnologia envolvendo conceitos de Internet das Coisas, comunicação por meio do protocolo LoraWAN™ e diversas aplicações relacionadas.

Materiais e Métodos

O projeto segue um fluxo de quatro etapas que integram o estudo abordado, as quais estão esquematizadas na Figura 1.

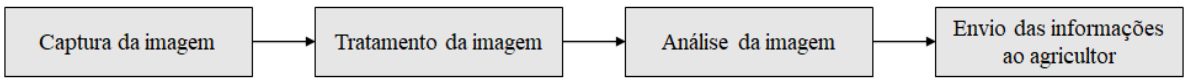

Figura 1 - Etapas do projeto

Os materiais utilizados neste projeto, tanto softwares como hardwares, estão representados abaixo e são detalhados ao longo do artigo:

1. Câmera IP Bullet da Intelbras VIP 1130 B, responsável pela captura das imagens;
2. Visual Studio Code - Software editor de código-fonte responsável por disponibilizar o ambiente de programação;
3. Python - Linguagem de programação usada para a escritura do código
4. OpenCV (*Open Source Computer Vision Library)* - Biblioteca que possui recursos de visão computacional para tratamento da imagem;
5. NumPy - Biblioteca que possui recursos de arranjos e matrizes

Sabendo que o objetivo do projeto é o reconhecimento de plantas a partir das dimensões, características e colorações de suas folhas, assim como as fases de crescimento em que se encontram no momento de captura das informações, foi escolhida a aplicação de Processamento de Imagens para alcançar este objetivo, e a soja como objeto de estudo.

Uma das principais etapas nesse processo é a representação da imagem, englobando fases de estrutura de dados e algoritmos, em que um software recebe algoritmos que interpretam a imagem digital. Uma imagem digital é composta por inúmeros pixels, e um pixel é o menor elemento da imagem, além da sua forma mais comum ser a retangular. Logo, uma matriz de pixels tem a função de organizar uma imagem digital, a qual é comumente feita em uma simetria quadrada e o modo como essa imagem é organizada é influenciado pela facilidade de uma implementação eletrônica. Entretanto, este arranjo apresenta dois problemas, os quais estão associados diretamente a algumas propriedades dos pixels. O primeiro é o fato dos pixels serem anisotrópicos, ou seja, não apresentam as mesmas características em todas as direções (borda e diagonal), o que influencia no tipo de conectividade a ser trabalhada, levando em consideração apenas os vizinhos de borda ou os de borda e diagonal. O segundo problema diz respeito à distância entre um pixel e seus vizinhos, a qual não é sempre a mesma, sendo igual a uma unidade (1) para vizinhos de borda e $\sqrt{2}$ para vizinhos na diagonal. Por conta disso,

aplicam-se pequenas matrizes (máscaras), que ponderam ou ajustam estas distâncias em função da direção, para solucionar estes problemas referentes às propriedades dos pixels e manipular as imagens (DE ALBUQUERQUE; DE ALBUQUERQUE, 2000).

Para desenvolver a aplicação primeiramente foi elaborado um código com a linguagem de programação Python e usando a biblioteca OpenCV como base de código, visto que nela encontram-se as funções necessárias para tratamento das imagens das plantas, como, por exemplo, filtragem, limiarização, que é um método de segmentação de imagem definida por preto ou branco, reconhecimento de objetos por cascatas, detecção de contornos, ROI (Região de Interesse), etc. Em seguida, optou-se por utilizar uma câmera IP (Internet Protocol) para a captura das imagens em tempo real, modelo IP Bullet da Intelbras VIP 1130 B, já que ela permite o tráfego das imagens via rede WAN (*Wide Area Network*), Intranet ou Internet, podendo ser acessada por meio de *tablets*, *smartphones*, aplicativos, *smart TV* e por meio do Smart Campus Mauá, através de um endereço próprio de IP.

Retomando a utilização do OpenCV, o *Deep Learning* foi utilizado neste projeto de Iniciação Científica para detecção da soja, visto que esta forma de aprendizado é responsável por configurar parâmetros principais sobre os dados estudados e treinar o software a aprender sozinho com base no reconhecimento padronizado, passando por várias camadas de processamento. A técnica usada neste projeto foi o método em **cascata (*cascade*)**, que, de forma resumida, se baseia em treinar uma árvore de decisão a qual analisa diferentes estágios de atributos de algum objeto estudado. Em cada estágio há classificadores fracos (*weak classifiers*), que, através de algoritmos voltados para *boosting*, classificam e determinam se o objeto é positivo ou negativo a cada iteração e, por fim, no término deste treinamento, um arquivo .xml é gerado contendo todos os parâmetros de reconhecimento do objeto estudado. O método em cascata se relaciona com o OpenCV pela biblioteca, visando proporcionar uma "*feature*" denominada "*Haar*", a qual subtrai os pixels das regiões brancas dos pixels da região preta associada (vizinha) da imagem, de acordo com o tipo de "máscara" escolhida. O software seleciona os melhores aspectos do objeto e rejeita aspectos desnecessários para o estudo. Tabora (2019) apresenta um exemplo de aplicação utilizando *Deep Learning* para reconhecimento facial.

No desenvolvimento do algoritmo para reconhecimento da soja, primeiramente é necessário montar um banco de dados com imagens positivas e negativas, ou seja, imagens de interesse, no caso desse projeto, a soja, e objetos que não são de interesse, mas que têm alguma relação e que pertencem ao universo da soja. A proporção ideal para o treinamento do algoritmo

é de 1 ROI (Região de Interesse) positiva para 2 imagens negativas, sem conter o objeto estudado aparecendo na imagem, para melhor precisão de detecção.

Neste projeto, foi usado o software Cascade Trainer GUI (AHMADI, 2016), desenvolvido pelo Amin Ahmadi, para a geração do arquivo .xml, em virtude dele ser um programa *boosting* eficaz no treinamento de classificadores em cascata e, consequentemente, necessário para gerar os traços e parâmetros para reconhecimento da soja. O programa principal lê a imagem captada pela câmera e armazena na variável "img". Logo após realiza o tratamento dela, convertendo-a para escala de cinza e aplicando filtros que a desfocam, deixando-a manchada e realçando os seus contornos, através da função "cv2.GaussianBlur" e "cv2.bilateralFilter". Após isso, são aplicados efeitos de limiarização para binarizar a imagem, por meio do "cv2.threshold" e facilitar o reconhecimento da soja. Também é aplicada uma função de convolução e dilatação para realçar os contornos e os traços da soja: "cv2.Canny". Este procedimento foi escrito em Python, no editor de código Visual Studio Code, e foi utilizada, como teste, uma foto grátis para uso comercial, adquirida na internet, de uma plantação de soja no nome de "Soja (38).jpg". O trecho do código que realiza este procedimento, Código 1, está mostrado na Figura 3. Resultados são apresentados na Figura 4.

```
img = cv2.imread('Soja (38).jpg')
cv2.imshow('Soja.jpg', img)
gray = cv2.cvtColor(img, cv.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (5,5), cv.BORDER_DEFAULT)
bilateral = cv2.bilateralFilter(gray, 10, 35, 15)
ret, thresh = cv2.threshold(bilateral, 150, 255, cv.THRESH_BINARY)
canny = cv2.Canny(bilateral, 50, 100)
```

Figura 3 - Código 1 em Python para leitura e tratamento da imagem

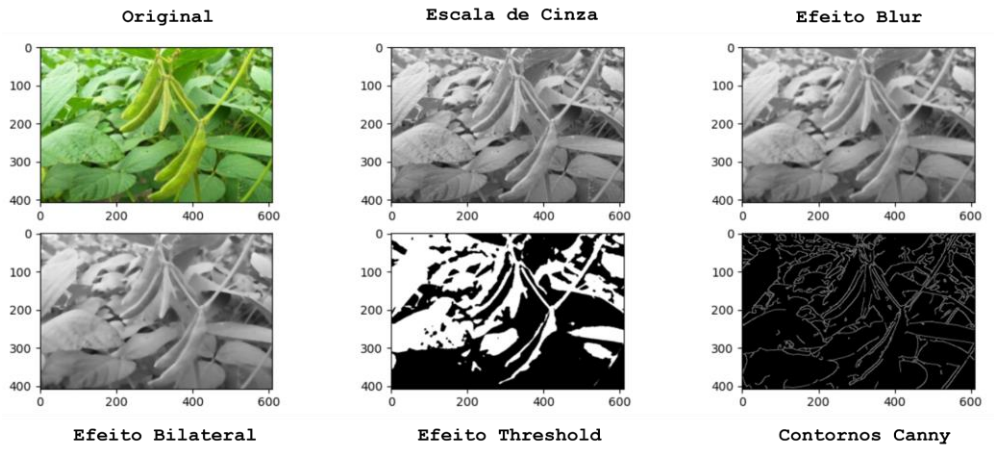

Figura 4 - Display do Código 1

Sequencialmente, carrega-se o arquivo 'haarcascade_soja.xml', gerado pelo software Cascade Trainer GUI, através da função "cv2.CascadeClassifier". Em seguida executam-se

duas funções “detectMultiScale”, tomando como parâmetros as imagens sob efeito de contornos *canny* e efeito *threshold*, e um loop “cv2.rectangle”, que gera retângulos ao reconhecer o objeto na imagem, apresentando a soja detectada como mostrado no trecho do código (Código 2) da Figura 5 e nos resultados da Figura 6.

```
haar_cascade = cv2.CascadeClassifier('haarcascade_soja.xml')

soja_rectangleCanny = haar_cascade.detectMultiScale(canny, scaleFactor=1.01, minNeighbors=3)
for (x,y,w,h) in soja_rect:
    cv2.rectangle(img, (x,y), (x+w,y+h), (0,0,255), thickness=2)

soja_rectangleThreshold = haar_cascade.detectMultiScale(thresh, scaleFactor=1.01, minNeighbors=3)
for (x,y,w,h) in soja_rect2:
    cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), thickness=2)

cv2.imshow('Detected Soja', img)
```

Figura 5 - Código 2 em Python para detectar a soja pelo método Deep Learning

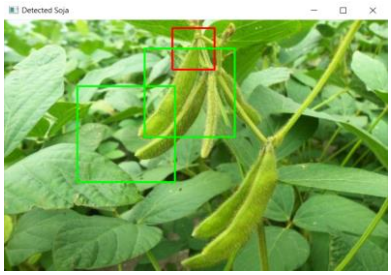

Figura 6 - Display do Código 2: retângulos na cor vermelha se referem aos contornos *canny* e os retângulos na cor verde pertencem ao efeito de limiarização (*thresholding*)

Com isso, temos não só a soja identificada pelo programa principal, mas também todo o procedimento técnico para reconhecimento de qualquer objeto estudado. Entretanto, apenas o reconhecimento da planta não é suficiente para otimizar o trabalho do agricultor. São igualmente necessários alguns parâmetros identificados e apresentados após todo esse procedimento de reconhecimento da imagem, como as dimensões do caule, das folhas, dos ramos, sementes, etc., visto que é a partir deles que será possível realizar um plano de ação certeiro sobre a safra. Logo, é percebido que, para a coleta dos indicadores de interesse do agricultor, sejam as dimensões do caule ou coloração da folha, necessita-se de uma segregação do objeto identificado, ou seja, de um desmembramento da imagem estudada por meio de filtros. Para tal fim, foi estudada a eficácia da conversão do modelo matemático de cores da imagem, de BGR, o qual é construído por pixels a partir das cores primárias Vermelho (*Red*), Verde (*Green*) e Azul (*Blue*), para HSV, cuja escala de cor é definida por valores de Matiz (*Hue*), Saturação (*Saturation*) e Brilho (*Value*). O parâmetro Matiz verifica o tipo de cor, a tonalidade da imagem, percorrendo todas as cores do espectro, do vermelho ao violeta. A

Saturação ajusta a pureza da imagem, quanto menor seu valor, mais tom de cinza aparecerá na imagem; em contraposição, quanto maior o seu valor, mais "pura" é a imagem. O Brilho define a intensidade do brilho na imagem. Dito isso, ajustando os parâmetros do sistema de cores HSV (Matiz, Saturação e Brilho) em um intervalo específico de valores, é possível filtrar as partes da planta estudada com o resto da imagem, como, por exemplo, selecionando apenas valores da tonalidade verde para segregar a planta do restante. Desta forma, define-se um padrão de tonalidades para cada parte da soja, seja um verde mais claro para filtrar apenas a vagem dela ou um tom mais escuro para adquirir a folha dela.

Com o objetivo de aplicar este filtro por padronização de cores a fim de facilitar a detecção da soja, foi desenvolvida, no código, uma função "DetectSoja", que recebe como parâmetros a variável da imagem e valores de máximo e mínimo, ou seja, intervalados, da matiz, saturação e brilho. Nesta função, em associação com a biblioteca OpenCV, foi chamada a biblioteca NumPy, por conta da sua facilidade em trabalhar com matrizes e arranjos e, principalmente, por organizar os valores máximos e mínimos dos parâmetros da escala HSV em uma lista com "np.array". Retornando para a manipulação da imagem com OpenCV, a imagem original BGR é convertida para a escala HSV na variável "hsv_img", com o objetivo de filtrar a imagem para expor apenas o conteúdo relacionado ao intervalo de cores de interesse, por meio da função "cv2.inRange". Salvando-a na variável "mask", junto com a operação "cv2.bitwise". O procedimento da função "DetectSoja" está descrito no Código 3 e segue na Figura 7.

```
def DetectSoja(img, lowerHue, upperHue, lowerSat, upperSat, lowerVal, upperVal):

    hsv_img = cv2.cvtColor(img, cv.COLOR_BGR2HSV)
    lowerGreen = np.array([lowerHue, lowerSat, lowerVal])
    upperGreen = np.array([upperHue, upperSat, upperVal])
    mask = cv2.inRange(hsv_img, lowerGreen, upperGreen)
    result1 = cv2.bitwise_and(img, img, mask = mask)
    result2 = cv2.bitwise_and(img, img, mask = mask)
```

Figura 7 - Código 3 em Python para aplicação de filtro por cor

Após a aplicação do filtro, é preciso delimitar as áreas onde o objeto pode estar localizado, permitindo o reconhecimento de múltiplas sojas dentro de uma foto. Além disso, é preciso realizar o cálculo das dimensões das partes da planta, visto que abrangem as informações mais valiosas para o agricultor. Este cálculo é feito em duas etapas. Primeiramente é necessário de um objeto de referência com dimensões já conhecidas posicionado ao lado do objeto de estudo. Portanto, como este projeto envolve temática agrícola, pode ser usada uma haste como objeto de referência ao lado da plantação da soja. Em seguida, calcula-se a razão

entre a quantidade de pixels da haste na imagem e suas dimensões reais, que vai ser atribuída na proporção entre a quantidade de pixels da soja interpretada pelo código e suas dimensões originais. Por conseguinte, conhecendo as medidas 2D (altura e largura) do objeto de referência é possível achar as dimensões da soja, independentemente da distância da câmera até a planta. O Código 4 da Figura 8 mostra como isso funciona na prática:

```
contours1, hierarchies = cv2.findContours(mask, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)
cv2.drawContours(blank, contours1, -1, (0,0,255), thickness=1)
i = 0
maxArea = cv2.contourArea(contours1[0])

for contour in contours1:
    area = cv2.contourArea(contour)
    if area > 2000:
        (x, y, w, h) = cv2.boundingRect(contour)
        cv2.rectangle(result1, (x,y), (x+w,y+h), (0,0,255), thickness=3)
        alturaReferencia_cm = 5 # Medida da altura do objeto de referência
        larguraReferencia_cm = 5 # Medida da largura do objeto de referência
        alturaPixels = 50 # Quantidade de pixels referentes à altura do objeto identificado
        larguraPixels = 70 # Quantidade de pixels referentes à largura do objeto identificado
        razaoY = alturaReferencia_cm / alturaPixels
        razaoX = larguraReferencia_cm / larguraPixels
        alturaObjeto = h*razaoY # altura do objeto de estudo em centímetro
        larguraObjeto = w*razaoX  # largura do objeto de estudo em centímetro
        cv.putText(result1, "Largura {} cm".format(round(larguraObjeto,1)), (x+5, y+h-10), cv.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 0),2)
        cv.putText(result1, "Altura {} cm".format(round(alturaObjeto,1)), (x+w+10, y+h-15), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255),2)
        print(x, y, w, h)
    i += 1
```

Figura 8 - Código 4 em Python - cálculo de dimensões e reconhecimento das partes da soja

A partir desta sequência de códigos de programação, pode-se dizer que a técnica de Processamento de Imagens mais adequada e determinante para este projeto de Iniciação Científica concerne à combinação do método de *Deep Learning* com o tratamento e a filtragem da imagem na escala de cor HSV.

Resultados e Discussões

A câmera IP Intelbras foi instalada com sucesso na horta do Instituto Mauá de Tecnologia, e é possível acompanhá-la em tempo real através de *dashboards* e interface gráfica disponibilizadas pela plataforma *Smart Campus* da Mauá (Figura 9):

Devido à pandemia da Covid-19 e fatores climáticos, não foi possível plantar culturas de soja e, consequentemente, coletar imagens reais da soja para treinar o algoritmo *Deep Learning* a reconhecê-la, como explicado nos parágrafos anteriores. Logo, o plano de ação foi coletar 52 imagens de soja espalhadas pela Internet e 105 imagens negativas para a criação do arquivo xml. Após a coleta das imagens, foram separadas 15 amostras para testar a precisão do algoritmo por este método de cascata somado com o tratamento das imagens por *thresholding*, *canny* e escala de cinza, ou seja, junção do código 1 com o código 2. Entretanto, o programa não obteve tanto êxito no reconhecimento da soja. Das 15 imagens aleatórias de plantações de

soja, 12 tiveram alguma região reconhecida pelo programa, porém apenas 5 detectaram alguma soja, resultando em 33,33% de precisão.

Figura 9 - Interface da plataforma *Smart Campus* da Mauá

Partindo para a análise do filtro por região de coloração e cálculo das dimensões constituídas nos códigos 3 e 4, os valores mínimo e máximo da matiz fornecidos foram 30 e 40, respectivamente e, ao rodar o programa principal, que reconhece a soja em meio à plantação através do método de *Deep Learning* mencionado anteriormente e filtra as partes que são interessantes ao estudo, obteve-se o reconhecimento não só das 2 plantas de soja presentes na imagem, mas também das 4 vagens presentes na planta e suas dimensões com bastante êxito, como mostrado na Figura 10.

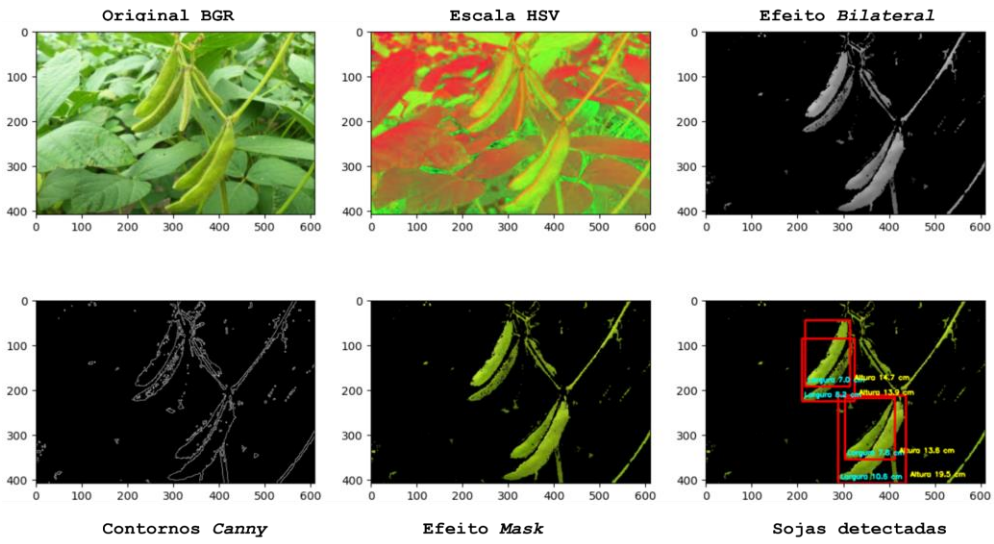

Figura 10 - Display dos códigos 3 e 4: detecção das sojas e suas dimensões

Após essa fase de análise da imagem, as informações coletadas deveriam ser enviadas e integradas à plataforma do *Smart Campus* da Mauá, no entanto, por conta das limitações impostas pela pandemia neste ano de 2021, o foco do projeto ficou concentrado no tratamento

e na análise das imagens, não havendo tempo suficiente para incorporar o código à plataforma, assim como a captação das informações em tempo real.

Conclusão

Com base no que foi apresentado e nos resultados obtidos, conclui-se que a solução computacional proposta neste projeto de Iniciação Científica, unindo os conceitos de Agricultura de Precisão, Processamento de Imagens e Internet das Coisas, tem grande potencial de ser implementada no mundo agrícola.

Em virtude do distanciamento social provido pela pandemia do Covid-19, não foi possível realizar testes presenciais na horta do algoritmo desenvolvido neste projeto, o que dificultou o seu progresso.

Portanto, para maior eficácia de projetos posteriores, é crucial a aquisição de imagens reais do objeto de estudo de diversos ângulos diferentes, ao invés de imagens da Internet como banco de dados, para o treinamento em cascata do método de reconhecimento de objetos, *Deep Learning*. Adicionalmente, é necessário implementar o código aqui apresentado na plataforma do *Smart Campus* da Mauá para a coleta dos dados em tempo real.

Referências Bibliográficas

AHMADI, Amin. CASCADE TRAINER GUI. *In*: AHMADI, Amin. **CASCADE TRAINER GUI**. 3.3.1. [*S. l.*], 2016. Disponível em: https://amin-ahmadi.com/cascade-trainer-gui/. Acesso em: 13 dez. 2021.

DATABRAS. Confira os produtos mais exportados pelo Brasil em 2020. [*S. l.*]: DataBras, 18 mai. 2020. Disponível em: http://databras.com.br/confira-os-produtos-mais-exportados-pelo-brasil-em-2020/. Acesso em: 12 out. 2021.

DE ALBUQUERQUE, Márcio Portes; DE ALBUQUERQUE, Marcelo Portes. Processamento de Imagens: Métodos e Análises. Processamento de Imagens: Métodos e Análises, Rio de Janeiro, Brasil, v. 12, p. 1-11, 1 jan. 2000.

GRANDCHAMP, Leonardo. Veja quais são os 10 principais produtos agrícolas do Brasil. [*S. l.*]: Dia Rural, 11 ago. 2021. Disponível em: https://diarural.com.br/veja-quais-sao-os-10-principais-produtos-agricolas-do-brasil/. Acesso em: 12 out. 2021.

HONDA, Bruna; JORGE, Lúcio André Castro. Computação aplicada à agricultura de precisão. Embrapa Instrumentação, [*S. l.*], p. 122-123, 24 jun. 2013. Disponível em: http://ainfo.cnptia.embrapa.br/digital/bitstream/item/186392/1/Computacao-aplicada-a-agricultura-de-precisao..pdf. Acesso em: 12 out. 2021.

TABORA, Vincent. Face Detection Using Open CV With Haar Cascade Classifiers. Medium, Open Platform. 1 fev. 2019. Disponível em: https://becominghuman.ai/face-detection-using-opencv-with-haar-cascade-classifiers-941dbb25177. Acesso em: 28 nov. 2021.