Algoritmo para segmentação da amêndoa de cacau utilizando visão computacional com OpenCV

Patrick S. Ferraz[1], **Prabhát K. de Oliveira**[1], **Marta Magda Dornelles**[2]

[1]Ciência da Computação – Universidade Estadual de Santa Cruz (UESC)
45662-900 – Ilhéus – BA – Brasil

[2]Departamento de Ciências Exatas e Tecnológicas
Universidade Estadual de Santa Cruz (UESC) – 45662-900 – Ilhéus – BA – Brasil

patrick.ferraz@outlook.com, {bartkoliveira,marta.uesc}@gmail.com

Abstract. *This paper describes the process of segmentation of the different classes of cocoa beans with the development of an algorithm using the concepts of computer vision with the OpenCV open source library.*

Resumo. *Este artigo descreve o processo de segmentação das diferentes classes de amêndoas de cacau com o desenvolvimento de um algoritmo utilizando os conceitos de visão computacional com a biblioteca de código aberto OpenCV.*

1. Introdução

O cacau é uma palavra que deriva do termo *Kakaw*, de origem maia. Não se sabe ao certo quem foram os primeiros povos a cultivar o fruto, embora histórias apontem para os Astecas, no México, e os Maias, na América Central. De acordo com historiados, o cacaueiro, chamado de *cacahualt*, era considerado sagrado e muitas vezes seu cultivo era acompanhado de solenes cerimônias religiosas, por conta disso o botânico sueco Carlos Linneo denominou a planta de *Theobroma cacao*, que significa ”manjar dos deuses”, provavelmente, inspirado em toda a simbologia que envolvia o cultivo do cacau [CEPLAC 2018].

Com o passar do tempo, o cacau se tornou um fruto muito popular devido não só a produção do chocolate, a partir de suas sementes, sendo um dos alimentos mais apreciados do mundo [Anuário Brasileiro do Cacau 2016], mas também da manteiga de cacau, liquor, torta e pó. A qualidade do chocolate pode ser sentida ao mordê-lo e depende diretamente não apenas do sabor e da pureza das amêndoas de cacau, nomeação dada às sementes do cacau, mas também as características físicas como: salubridade, produção de material comestível e características da manteiga de cacau [Ferreira et al. 2013]. Seu notório sabor, apesar dos atributos genéticos, devem-se também as modificações que ocorrem durante sua primeira etapa de produção, o beneficiamento, que se estende desde o seu cultivo, colheita, abertura dos frutos, retirada das sementes, extração da polpa ou do mel, fermentação das sementes, secagem e armazenamento das amêndoas [Efraim et al. 2010].

Existem metodologias que podem ser aplicadas em cada fase do beneficiamento para melhorar a qualidade da amêndoa. Essa qualidade pode ser verificada pela prova de corte, que tem como objetivo avaliar se o beneficiamento foi realizado corretamente e em que fase do processo ocorreram problemas [Feitosa 2016]. Basicamente serão obtidas imagens das amêndoas durante a prova de corte, e a partir delas será realizado a

segmentação utilizando os conceitos de visão computacional, originando então uma imagem individual para cada amêndoa para posterior análise e classificação.

Em linhas gerais, a prova é realizada sobre 300 metades de amêndoas de cacau cortadas longitudinalmente. Nestas são analisadas algumas características internas, além do odor. Após a análise, o lote de amêndoas é classificado em Tipo 1, Tipo 2, Tipo 3, e Fora do Tipo, de acordo com a instrução normativa nº 38/2008 do Ministério da Agricultura, Pecuária e Abastecimento (MAPA, 38/2008), que defini o padrão oficial de classificação de amêndoas de cacau.

Uma imagem digital é composta por um número finito de elementos que possui localização e valor específicos. Esses elementos podem ser representados como *pixels*, elementos pictóricos e elementos de imagem, sendo o *pixel* mais utilizado para essa representação. A segmentação consiste na subdivisão da imagem em regiões ou objetos, o nível de detalhe que essas regiões são subdivididas depende do problema a ser resolvido. Ou seja, a condição de parada deve ser quando os objetos ou regiões de interesse forem detectados.

2. Material e Métodos

2.1. Ambiente e equipamento

Uma das proposta do algoritmo de segmentação é ser capaz de realizar o processamento de imagens obtidas em ambientes não controlados, aproveitando a característica multiplataforma da biblioteca OpenCV, subseção 2.2, para que seja possível a sua portabilidade, utilizando-o em sistemas operacionais Linux, Windows, Mac OS, iOS e Android.

O algoritmo de segmentação foi desenvolvido utilizando as linguagens C++ e Python em ambiente Linux. Para tal, foi necessário o uso de alguns recursos físicos e lógicos afim de obter as imagens e posterior tratamento com visão computacional, sendo descritos abaixo:

- Recursos físicos
  - Corte longitudinal das amêndoas do espaço amostral
  - Tábua branca contendo espaço para 300 amostras de amêndoas
  - Celular com câmera para capturar a imagem da tábua com as amêndoas
  - Computador
- Recursos lógicos
  - Sistema operacional Debian GNU/Linux 9 Stretch 64bits
  - Compilador para C++ (gcc 6.3.0)
  - Interpretador para Pyhton (python3.5)
  - Biblioteca OpenCV 3.4.1

Ressalta-se que, embora o desenvolvimento tenha sido realizado em ambiente Linux, o mesmo código foi compilado, para C++, e executado, para C++ e Python, em Windows 10 64bits obtendo resultado equivalente. O mesmo vale para a construção dos algoritmos. Foram utilizados as funções de visão computacional equivalentes do OpenCV para linguagem C++ e Python.

2.2. OpenCV

O OpenCV (Open Source Computer Vision) é uma biblioteca de código aberto e multiplataforma desenvolvida inicialmente pela Intel, utilizada para visão computacional, processamento de imagem, aprendizagem de máquina e operações em tempo real com aceleração de GPU. É liberada sob uma licença de BSD e foi escolhida devido a grande presença em uso acadêmico e comercial. Embora tenha sido inicialmente desenvolvida em C/C++, também oferece suporte para utilização nas linguagens Java, Python e sistemas operacionais Windows, Mac OS, iOS e Android [Passarelli 2017]. Conta com uma grande presença em aplicações de tempo real e processamento *multi-core*, sendo adotada em todo o mundo com mais de 47 mil pessoas da comunidade e número estimado de downloads superior a 14 milhões [OpenCV 2018].

> Visão computacional é o estudo da extração de informação de uma imagem; mais especificamente, é a construção de descrições explícitas e claras dos objetos em uma imagem . Difere do processamento de imagens porque, enquanto ele se trata apenas da transformação de imagens em outras imagens, ela trata explicitamente da obtenção e manipulação dos dados de uma imagem e do uso deles para diferentes propósitos. [Rios 2010]

A segmentação de imagens, utilizando o OpenCV, podem ser adquiridas das seguintes formas:

1. Thresholding: O *threshold* se baseia na diferença de tons de cinza que compõem diferentes objetos na imagem. A partir das características dos objetos que se quer isolar (obtidos por meio de um histograma por exemplo), a imagem será segmentada em dois grupos: os que possuem níveis de cinza abaixo do valor estabelecido, e os que possuem níveis de cinza acima do valor estabelecido. Para a geração de uma imagem limiarizada, atribui-se um valor fixo para todos os *pixels* de um mesmo grupo. A imagem gerada será binária, ou seja, terá apenas dois valores possíveis para cada *pixel*.
2. Detecção de bordas: Primeiro aplica-se o método da morfologia matemática para detecção de bordas, que podem ser o de Sobel, Canny, Laplaciana, Prewit ou Roberts. Em seguida é feita um agrupamento de *pixels* detectados como bordas, a partir de um algoritmo de união ou realce de bordas, que permite determinar de maneira mais precisa o contorno dos objetos de uma imagem.
3. Regiões: Um conjunto de *pixels* que possuem determinado grau de similaridade, são tidos como regiões. No método de segmentação baseado em regiões, cada região é composta por *pixels* com um valor similar, baseado em um critério de similaridade.
4. Divisória (Watersheed): Esses algoritmos estão baseados na morfologia matemática, que permite extrair as bordas que existem na imagem. Também é considerada uma técnica de segmentação baseada em regiões. Os *pixels* são classificados de acordo com suas proximidades espaciais, gradiente de seus níveis de cinza e homogeneidade de suas texturas. O objetivo principal desse algoritmo é encontrar as linhas divisórias em uma imagem para separar as diferentes regiões. A ideia é imaginar que os valores dos *pixels* de uma imagem como sendo um gráfico topográfico em 3D, onde "x" e "y" denotam as coordenadas do plano, e "z" denota o valor do *pixel* (altura do terreno). O algoritmo começa por preencher

o gráfico topográfico com água desde a bacia mais baixa (valor mínimo local) até o pico mais alto. Nesse processo, os objetos resultantes da segmentação correspondem aos mínimos do gradiente morfológicos e aos contornos das linhas divisórias [Chipana and Iano 2012].

3. Resultados

3.1. Testes de parâmetros e algoritmos

As amêndoas estão divididas em 7 classes, sendo elas: achatada, ardósia, germinada, inseto, marrom, parcialmente marrom e violeta, mostrada na Figura 1.

Figura 1. Amêndoas divididas em classes. Achatada, ardósia, germinada, inseto, marrom, parcialmente marrom e violeta, respectivamente.

Foram realizados testes com diferentes algoritmos de segmentação, e para cada um, uma variação nos parâmetros para comprovar quais se mostrariam as melhores escolhas para a segmentação de amêndoas.

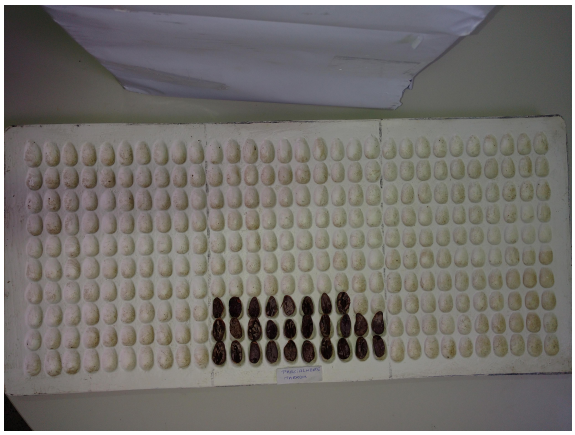
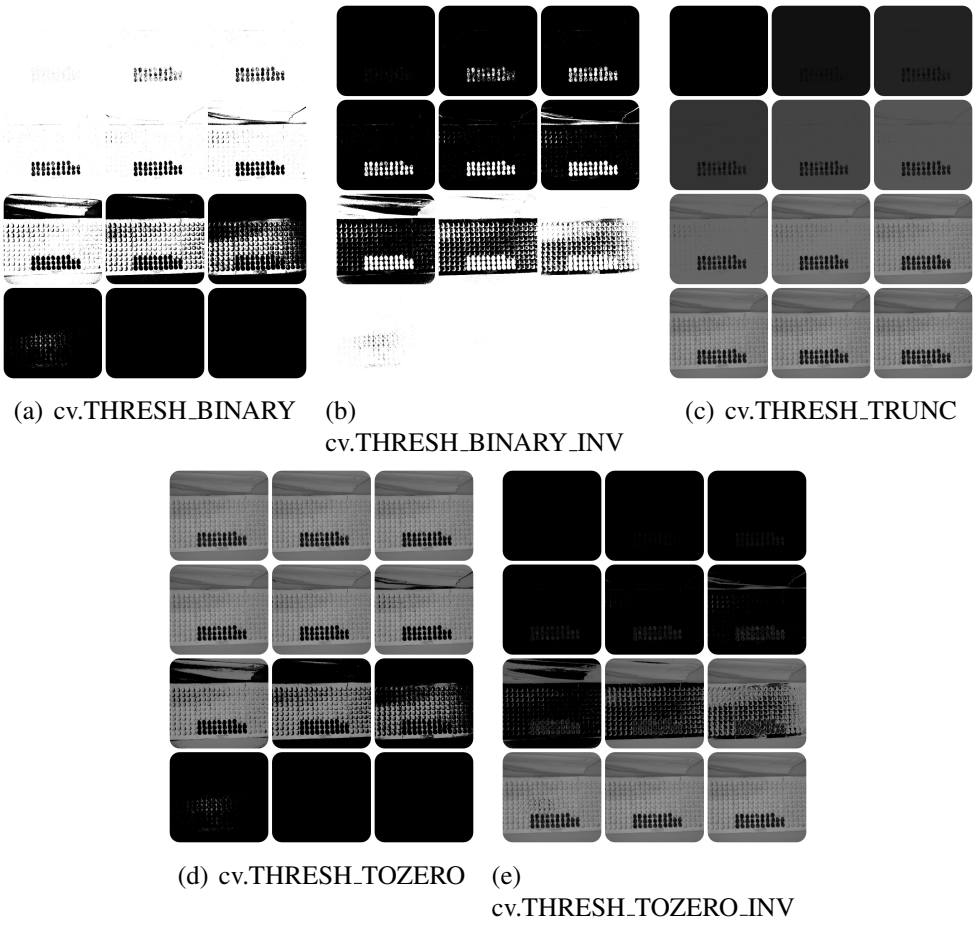


Figura 2. Exemplares na tábua de corte para testes dos algoritmos.

Para estes testes foi utilizado a Figura 2. O primeiro passo foi testar o algoritmo *threshold* com diferentes estilos de limiar da biblioteca OpenCV:

- cv.THRESH_BINARY
- cv.THRESH_BINARY_INV
- cv.THRESH_TRUNC
- cv.THRESH_TOZERO
- cv.THRESH_TOZERO_INV

Os parâmetros de valor limite para classificar os valores de *pixel* do *threshold* foram variados de 0 a 255, em passos de 17, gerando assim 15 imagens. Porém, as imagens com o valor acima de 200 não apresentaram variação visual significativa, então foram mantidas apenas 12 imagens (variação de 0 a 204). Resultados presentes na Figura 3.



(a) cv.THRESH_BINARY (b) cv.THRESH_BINARY_INV (c) cv.THRESH_TRUNC

(d) cv.THRESH_TOZERO (e) cv.THRESH_TOZERO_INV

Figura 3. Resultados dos valores de pixels para threshold com diferentes limiar.

Foram testados também algoritmos de detecção de borda, como: Canny, Sobel e Laplacian, mas o último não apresentou resultados significativos para esta segmentação.

3.2. Passo a passo do algoritmo de segmentação

1. O algoritmo inicia o processamento em uma imagem de escolha, para este caso, uma das imagens apresentadas na Figura 1, e faz uma transformação para escala de cinza utilizando a função cv2.cvtColor do pacote OpenCV. Esta função recebe como parâmetro a imagem de entrada e o modo de cor de saída (cv2.COLOR_BGR2GRAY nesse caso). Resultados em Figura 4.

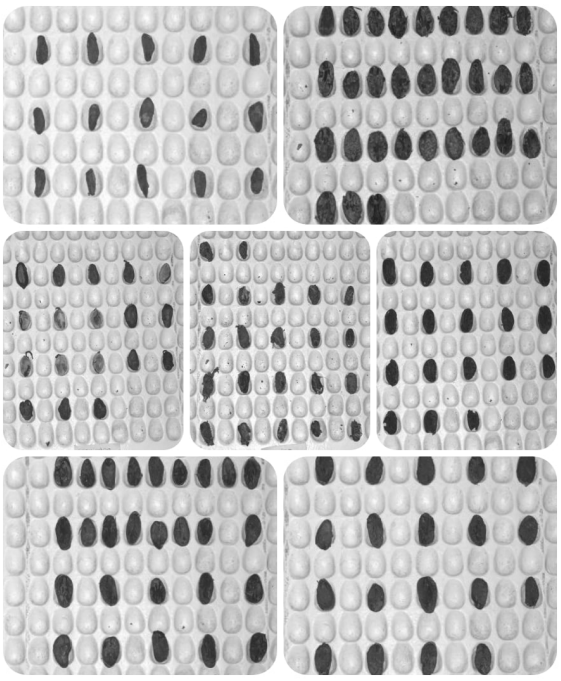
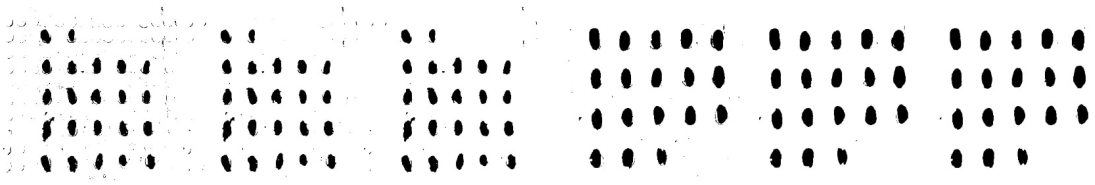


Figura 4. Conversão para escala de cinza.

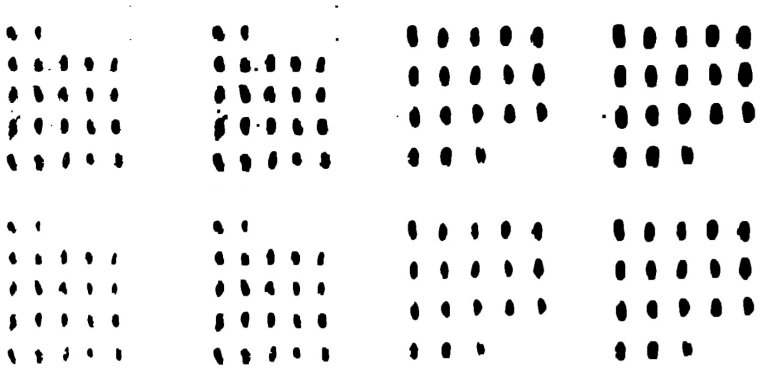
2. É realizado, a partir das imagens obtidas em Figura 4, a técnica do *thresholding*, mas constatou-se que a imagem apresentava "corpos estranhos" que não faziam parte das amêndoas, então viu-se a necessidade de utilizar uma técnica para reduzir ou remover esses "ruídos" da imagem. Primeiramente foi utilizado a função GaussianBlur, que recebe a imagem e o *kernel* como parâmetro. Para testar qual o melhor *kernel* (sendo necessário positivo e ímpar), foram variados de 1 a 5, respectivamente. A Figura 5 ilustra os resultados obtidos com a classe inseto e marrom.



(a) Inseto (b) Marrom

Figura 5. Resultados obtidos após aplicação do Gaussian blur com *kernel* 1, 3 e 5, respectivamente.

Nota-se que o *kernel* (5,5) foi o que apresentou melhores resultados, para todos os casos, mas ainda assim não eliminou todos os ruídos. Portanto, foi utilizado as funções de erosão e dilatação, respectivamente, com variação na quantidade de iterações. As iterações variam de 1 a 3, onde primeiro fixa-se 1 para erosão, depois fixa-se 2 para erosão, e 3, variando os 3 valores para dilatação em cada caso. A Figura 6 expõe os resultados obtidos no pior e melhor caso das classes inseto e marrom.

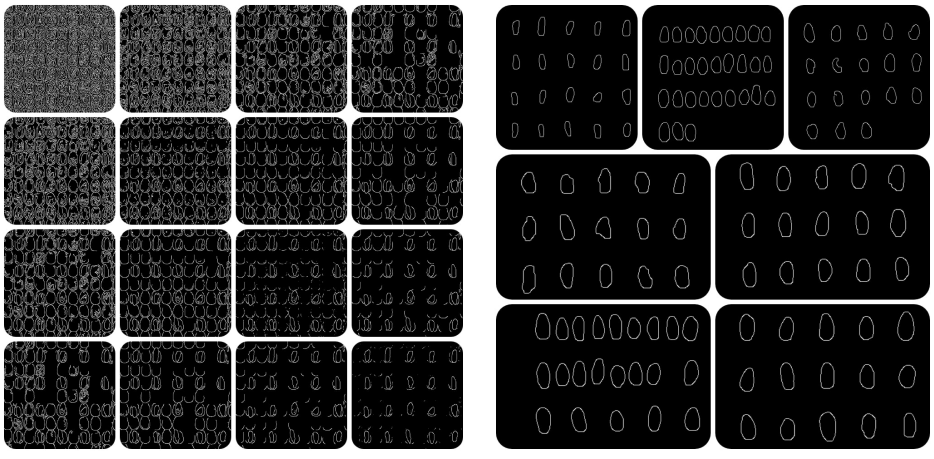


(a) Inseto (b) Marrom

Figura 6. Resultados obtidos após aplicação da erosão e dilatação para cada classe, no pior caso (cima), erosão 1 e dilatação 1, e no melhor caso (baixo), erosão 3 e dilatação 3, respectivamente.

Foi escolhido 3 iterações para erosão, e 3 para dilatação, que foram os valores que apresentaram os melhores resultados para todos os casos.

3. O passo seguinte foi utilizar o algoritmo de detecção de borda Canny, utilizando, como entrada, as imagens obtidas na Figura 6. Como já foi visto, o algoritmo de Canny possui parâmetros variáveis, e foi realizado testes com estes parâmetros para identificar a melhor opção. A seguir será apresentado apenas os testes para classe achatada, devido a semelhança entre os resultados das outras classes, e o resultado das demais classes, para o melhor caso, que foi escolhido para ser usada no passo seguinte. Os dois parâmetros variam de 0 a 255, sendo assim, o passo de 17, gera 16 imagens para apenas um parâmetro fixo, com um total de 256 imagens. Para exemplificar os testes, serão utilizadas apenas 16 dessas.



(a) Testes na variação dos parâmetros do Canny Edge na classe achatada, sendo o primeiro o par (0,0) e o ultimo o par (255,255)

(b) Resultados para o melhor caso (255,255), nas classes achatada, ardósia, germinada, inseto, marrom, parcialmente marrom e violeta, respectivamente.

Figura 7. Aplicação do Canny Edge na classe achatada (esquerda) e do melhor caso nas demais classes (direita).

4. Posteriormente, os resultados de borda obtidos da Figura 7(b) foram aplicados como máscara na função cv2.findContours, que analisa a imagem e encontra os contornos, para assim destacar as amêndoas encontradas na imagem original, conforme demonstrado na Figura 8.

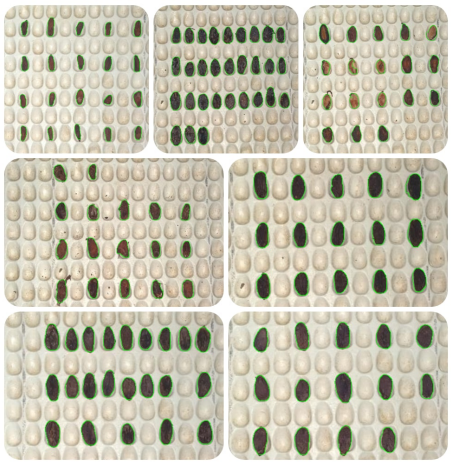


Figura 8. Amêndoas segmentadas e destacadas na imagem original.

Embora as amêndoas tenham sido destacadas na imagem original, ainda é necessário a segmentação completa e individual das mesmas, para que possa ser extraído, posteriormente, suas características. Foi utilizado então as imagens resultantes da Figura 6, invertendo-se as cores, onde a silhueta das amêndoas passariam a ter a cor branca, e o plano de fundo a cor preta. Assim, com essa máscara obtida, aplicou-se a função cv2.bitwise_and junto a imagem original, resultando em amêndoas destacadas com as cores originais e todo o fundo convertido a branco (Figura 9).

Figura 9. Amêndoas segmentadas e destacadas na imagem original com fundo branco.

Por fim, foi utilizado os contornos obtidos na Figura 8 para extrair cada amêndoa individualmente, gerando amostras para posterior extração das características e podendo ser utilizadas para classificação das amêndoas.

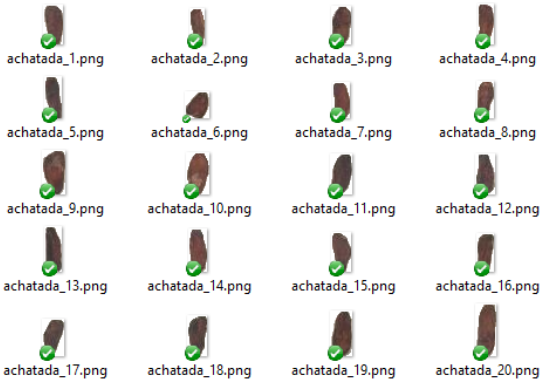


Figura 10. Exemplo de pasta contendo amêndoas extraídas da Figura 9, da classe achatada.

4. Conclusão

Conforme discutido, o cacau é um fruto de vital importância para economia, principalmente por conta da produção do chocolate. A prova de corte para classificação das amêndoas, apesar de ser um procedimento trivial, evolve inúmeros cuidados para atestar a qualidade da amêndoa. Ressalta-se que, embora os resultados obtidos através da visão computacional com OpenCV, que originou o algoritmo de segmentação aplicado aos processos descritos, tenham trazido os resultados esperados, gerando imagens individuais e de qualidade de cada amêndoa, estamos apenas no início do objetivo que se deseja alcançar. Tais resultados proporcionarão a implementação de métodos de baixo custo, também utilizando a visão computacional, para classificação dessas amêndoas, acarretando na autonomia e maior produtividade para as empresas de classificação de médio e pequeno porte, que muitas vezes, ainda realizam as atividade de forma manual ou não possuem recursos para compra de equipamentos que garantam a sua presença competitiva no mercado.

Referências

Anuário Brasileiro do Cacau (2016). Anuário brasileiro do cacau 2016. Editora Gazeta.

CEPLAC (2018). Cacau: história e evolução. `http://www.ceplac.gov.br/radar/radar_cacau.htm`. Online; acessado 18 de Junho de 2018.

Chipana, F. E. A. and Iano, Y. (2012). Segmentação de imagens: abordagens para reconhecimento de placas de veículos. Universidade Estadual de Campinas, Campinas - SP.

Efraim, P., Pezoa-García, N. H., Jardim, D. C. P., Nishikawa, A., Haddad, R., and Eberlin, M. N. (2010). Influência da fermentação e secagem de amêndoas de cacau no teor de compostos fenólicos e na aceitação sensorial. In *Ciência e Tecnologia de Alimentos*. SciELO Analytics.

Feitosa, R. (2016). Avaliação microscópica de características internas em amêndoas de cacau. Universidade Estadual de Santa Cruz, Ilhéus - BA.

Ferreira, A. C. R., Ahnert, D., de Melo Neto, B. A., and Mello, D. L. N. (2013). Guia de beneficiamento de cacau de qualidade. Instituto Cabruca.

OpenCV (2018). Open source computer vision library. https://opencv.org. Online; acessado 18 de Junho de 2018.

Passarelli, L. (2017). Aplicação de visão computacional com opencv. https://www.embarcados.com.br/aplicacao-de-visao-computacional-com-opencv/. Online; acessado 18 de Junho de 2018.

Rios, L. R. S. (2010). Visão computacional. Universidade Federal da Bahia, Salvador - BA.