UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ESPECIALIZAÇÃO EM AUTOMAÇÃO INDUSTRIAL

DANIEL TAKESHI WATANABE

SISTEMA DE VISÃO COMPUTACIONAL PARA A CONTAGEM DE SEMENTES DE SOJA

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2019

DANIEL TAKESHI WATANABE

SISTEMA DE VISÃO COMPUTACIONAL PARA A CONTAGEM DE SEMENTES DE SOJA

Monografia de Especialização, apresentada ao Curso de Especialização em Automação Industrial, do Departamento Acadêmico de Eletrônica – DAELN, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Especialista.

Orientador: Prof. Dr. Gustavo Benvenutti Borba

CURITIBA
2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Curitiba

Diretoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Eletrônica
Curso de Especialização em Automação Industrial

TERMO DE APROVAÇÃO

SISTEMA DE VISÃO COMPUTACIONAL PARA A CONTAGEM DE SEMENTES DE SOJA

por

DANIEL TAKESHI WATANABE

Esta monografia foi apresentada em 27 de Fevereiro de 2019 como requisito parcial para a obtenção do título de Especialista em Automação Industrial. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Gustavo Benvenutti Borba
Orientador

_____
Prof. Dr. Kleber Kendy Horikawa Nabas
Membro titular

_____
Prof. M. Sc. Omero Francisco Bertol
Membro titular

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

AGRADECIMENTOS

Agradeço à empresa Similar Tecnologia e Automação por dar a oportunidade de realizar este estudo e utilizar a esteira do laboratório de testes. Ao professor e orientador Gustavo Benvenutti Borba pelo acompanhamento deste TCCE. À minha família, amigos e namorada pelo suporte dado durante este estudo.

RESUMO

WATANABE, Daniel Takeshi. **Sistema de visão computacional para a contagem de sementes de soja**. 2019. 48 p. Monografia de Especialização em Automação Industrial, Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2019.

O procedimento de contagem de sementes é importante para a correta precificação e comercialização do produto. Os métodos tradicionais de contagem de sementes são baseados no peso ou volume, o que resulta em erros. Assim, este procedimento convencional de contagem de sementes pode ser aperfeiçoado visando uma maior acurácia. Neste trabalho, foi desenvolvido um sistema de contagem de sementes de soja baseado em visão computacional. O sistema utiliza uma esteira transportadora para movimentação das sementes, uma câmera RGB para a aquisição das imagens e bibliotecas *open source* para a implementação do algoritmo de contagem. As principais etapas do algoritmo são: remoção de ruído, segmentação baseada em cor no espaço HSV, operações morfológicas, *watershed* e rotulação. Os experimentos demonstraram que o sistema é capaz de realizar a contagem de sementes de soja com um erro médio absoluto de 5,32%. Este resultado comprova o funcionamento satisfatório desta primeira versão do sistema de contagem de sementes de soja desenvolvido. No entanto, é importante destacar que o sistema necessita de aperfeiçoamentos, com o objetivo de diminuir este erro e viabilizar a adoção do sistema em um ambiente industrial.

Palavras-chave: Watershed. Segmentação. Contador. Sementes de soja. Visão Computacional.

ABSTRACT

WATANABE, Daniel Takeshi. **Soybean counter with computer vision system**. 2019. 48 p. Monografia de Especialização em Automação Industrial, Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2019.

Seed counting is an important step for the correct pricing and further commercialization of the seeds. The traditional methods of seed counting are based on the weigh or volume measurement, which is prone to errors. Thus, such traditional seed counting methods can be improved in order to provide a larger accuracy. This work presents a computer vision system for the counting of soybean seeds. The developed system is based on a conveyor belt, a RGB camera and open source image processing software libraries. Main components of the developed algorithm are: noise removing, HSV-based color segmentation, morphological operations, watershed and labeling. Results shows that the proposed system presents a mean absolute error of 5.32%. This error is adequate for the current stage of this work, where a first version of the system for the counting of soybean seeds. However, it is important to mention that improvements are required for the eventual adoption of the developed system in industrial applications, since a better accuracy is necessary.

Keywords: Watershed. Segmentation. Counter. Soybean. Computer Vision.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

BSD	*Berkeley Software Distribution*
CCD	*Charge-Coupled Device*
CIELAB	*Commission internationale de l'éclairage: Lightness, Green-Red, Blue-Yellow*
CMOS	*Complementary Metal Oxide Semiconductor*
CMYK	*Cyan, Magenta, Yellow, Key*
CUDA	*Compute Unified Device Architecture*
FOV	*Field of View*
GPL	*GNU General Public License*
IDE	*Integrated Development* Environment ou Ambiente de Desenvolvimento Integrado
MAE	*Mean Absolute Error*
OpenCL	*Open Computing Language*
OpenCV	*Open-source Computer Vision*
SXGA	*Super Extended Graphics Array*
TD	Transformada de Distância
VGA	*Video Graphics Array*
XGA	*Extended Graphics Array*

LISTA DE SÍMBOLOS

d	distância
fps	*frames* por segundos
mm	milímetros
pixels	*picture elements*
s	segundos
t	tempo
v	velocidade

SUMÁRIO

1 INTRODUÇÃO

Atualmente os métodos mais tradicionais de contagem de sementes são baseados no peso ou volume da semente. Estes são ineficientes, pois as sementes são comercializadas por unidade e as plantadeiras utilizam cada unidade para plantar. Entre os diversos métodos atuais de contagem, a contagem manual é trabalhosa e demorada (LIU et al., 2017).

Sistemas de visão têm sido aplicados à vários processos de inspeção da indústria agrícola (KIRATIRATANAPRUK; SINTHUPINYO, 2011), a aquisição e processamento de imagens digitais representa uma tecnologia viável para diferentes tipos de aplicações e de bom custo benefício (HOBSON; CARTER; YAN, 2009).

As máquinas de contagem baseadas em visão computacional existentes no mercado realizam esta tarefa com uma vazão muito baixa de sementes, portanto, são utilizadas principalmente em laboratórios controlados. As câmeras industriais são amplamente utilizadas em aplicações com alta taxa de aquisição, sendo uma solução apropriada.

Neste trabalho foi desenvolvido um sistema para a contagem de sementes de soja baseado em visão computacional. Foi utilizada uma esteira disponível na empresa Similar Tecnologia e Automação para acomodar as sementes e uma câmera de *smartphone* para a aquisição das imagens. Para o processamento das imagens foram utilizadas bibliotecas abertas (*open source*).

Alguns dos trabalhos científicos utilizados como base para o desenvolvimento do algoritmo de contagem de sementes de soja foram: grãos de arroz (LIU et al., 2017; HOBSON; CARTER; YAN, 2009; SAKAI et al., 1996; WANG; CHOU, 2004; YAO; ZHOU; WANG, 2010), imagens coloridas (ZHANG et al., 2012), braquiterapia (MOULT et al., 2012), eritrócitos no sangue (KHAJEHPOUR et al., 2013), entre outros. Estes trabalhos consideram apenas imagens estáticas (*still images*), e não imagens como no caso do projeto proposto capturadas com esteira em movimento. Ainda, pode-se mencionar o trabalho de contagem dinâmica (objetos em movimento) de raízes de mandioca realizado na *Kansas State University* (NEILSEN et al., 2017).

1.1 JUSTIFICATIVA

A população global irá aumentar para mais de 9 bilhões e a demanda por comida irá crescer mais de 50% (NEILSEN et al., 2017). O aumento da eficiência em processos agrícolas auxilia no aumento de produtividade.

O Brasil é o segundo maior produtor de soja do mundo e o estado do Paraná é o segundo maior produtor do país (CONAB, 2018). Portanto, estudos com grãos de soja são muito importantes para estas regiões.

Sistemas de visão computacional apresentam ainda um grande potencial para aplicações em diversas áreas (BEYERER; LEÓN; FRESE, 2016), como por exemplo aquela abordada neste trabalho.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Neste trabalho será realizada a contagem de sementes de soja dinamicamente numa esteira. Utilizando sistema de visão industrial e criação de um software de contador de sementes com as bibliotecas OpenCV (BRADSKI, 2000), Python (ROSSUM, 2018), NumPy (OLIPHANT, 2006) e SciPy (JONES; OLIPHANT; PETERSON, 2001).

1.2.2 Objetivos Específicos

Os objetivos são divididos em “Definição do Hardware”, “Definição do Software”, “Desenvolvimento” e “Análise de Resultados”:

- Na “Definição do Hardware” são escolhidos a câmera, iluminação, sensores e processador para aquisição de imagem.
- Na “Definição do Software” são escolhidos a linguagem de programação, ambiente de desenvolvimento (IDE), bibliotecas de programação.
- Na etapa de “Desenvolvimento” é realizada a programação do contador de sementes de soja.
- Na “Análise de Resultados” é escolhido o método estatístico, são realizados testes e a análise estatística dos resultados.

2 REVISÃO BIBLIOGRÁFICA

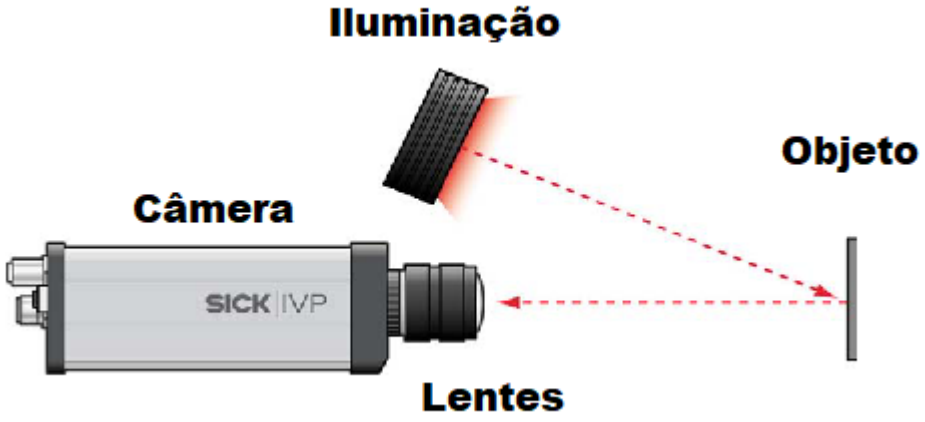
2.1 CONCEITOS BÁSICOS DE VISÃO

Nesta seção serão apresentados os conceitos de visão necessários para a para o desenvolvimento do sistema de contagem de sementes de soja.

2.1.1 Configuração Básica

Uma configuração simplificada de um sistema de câmera consiste em câmera, lente, iluminação e objeto, como pode ser visto na Figura 1. A iluminação atinge o objeto e a luz refletida é vista pelo sensor de imagem (SICK, 2010).

Figura 1: Exemplo de uma configuração simplificada de sistema de câmera



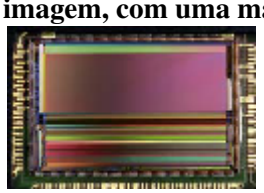
Fonte: Sick (2010).

2.1.2 Aquisição de Imagem Digital

Nas câmeras digitais, um sensor de imagem é usado para adquirir uma imagem digital. Um *chip* de sensor de imagem pode ser vista na Figura 2. No sensor, há uma matriz de pixels

sensíveis à luz. Num caso simples, cada pixel do sensor corresponde à um pixel na imagem (SICK, 2010).

Figura 2: Chip do sensor de imagem, com uma matriz de pixels sensíveis à luz



Fonte: Sick (2010).

O sensor de imagem está localizado logo após a lente, como pode ser visto na Figura 3.

Figura 3: Sensor de imagem na câmera

Fonte: Sick (2010).

As duas tecnologias principais de sensores de imagem digital são:

- **CCD (*Charge-Coupled Device*);**
- **CMOS (*Complementary Metal Oxide Semiconductor*).**

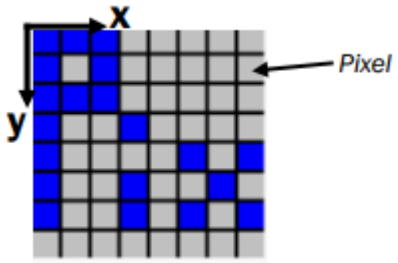
2.1.3 Pixel

Um pixel é o menor elemento de imagem de uma imagem digital. Normalmente, o pixel na imagem corresponde diretamente ao pixel físico do sensor. Geralmente, os pixels são tão pequenos que podem ser reconhecidos somente com a ampliação da imagem.

Na Figura 4 há um exemplo de uma imagem muito pequena, de dimensões de 8 x 8 pixels. Os pixels podem possuir as dimensões representadas em coordenadas x e y. Em que x são as colunas e y as linhas (SICK, 2010).

A representação das coordenadas de uma imagem é dada conforme a Figura 5. Em que x e y são números inteiros, m o número de colunas e n o número de linhas. Onde, $1 \leq x \leq m$ e $1 \leq y \leq n$ (BATCHELOR; WHELAN, 1997).

Figura 4: Pixel



Fonte: Sick (2010).

Figura 5: Representação da imagem

$$\mathbf{F} = \begin{vmatrix} f(1,1), & f(1,2), & \dots & f(1,n) \\ f(2,1), & f(2,2), & \dots & f(2,n) \\ . & . & \dots & . \\ . & . & \dots & . \\ f(m,1), & f(m,2), & \dots & f(m,n) \end{vmatrix}$$

Fonte: Batchelor e Whelan (1997).

2.1.4 Resolução

Os sensores de imagem possuem resoluções em pixels, que normalmente possuem as seguintes dimensões em câmeras industriais (SICK, 2010):

- **VGA (*Video Graphics Array*):** 640x480 pixels ou 0,3 Megapixels;
- **XGA (*Extended Graphics Array*):** 1024x768 pixels ou 0,8 Megapixels;
- **SXGA (*Super Extended Graphics Array*):** 1280x1024 pixels ou 1,3 Megapixels.

Já câmeras profissionais de fotografia ou *smartphones* possuem sensores de imagem e resoluções maiores.

2.1.5 Intensidade

É a intensidade luminosa naquele ponto da cena que está armazenada em cada pixel da imagem. Ela depende do tipo de dado, em que os mais comuns são (BRADSKI, 2000):

- 8-bit unsigned integer (uchar);

- 8-bit signed integer (schar);
- 16-bit unsigned integer (ushort);
- 16-bit signed integer (short);
- 16-bit floating-point number (float);
- 32-bit signed integer (int);
- 32-bit floating-point number (float);
- 64-bit floating-point number (double).

Quanto mais bits por pixel, melhor a representação de diferenças de intensidade ou cor. As imagens de intensidade também são chamadas de imagens de níveis de cinza (*grayscale*). As imagens do tipo preto e branco também são chamadas de imagens binárias – apenas pixels 0 e 1. Para o caso das imagens coloridas, pode-se adotar diferentes espaços de cores para a sua representação, sendo os mais usuais (BEYERER; LEÓN; FRESE, 2016):

- Escala de cinza;
- Binário;
- RGB (*Red, Green, Blue*);
- HSV (*Hue, Saturation, Value*);
- HSL (*Hue, Saturation, Lightness*);
- CIELAB (*Commission internationale de l'éclairage: Lightness, Green-Red, Blue-Yellow*);
- CMYK (*Cyan, Magenta, Yellow, Key*).

Em geral, os tipos de dados de dados mais utilizados são: i) 8-bits para a representação de imagens em nível de cinza, com os níveis de cinza variando entre 0 e 255 (Figura 6). ii) O tipo de dado de 1-bit, que pode assumir os valores 0 e 1, para imagens binárias (Figura 7). iii) Para imagens coloridas, costuma-se utilizar também o tipo de dados 8-bits, para representação de cada um dos canais de cores. Na Figura 8, por exemplo cada um dos canais do sistema de cores RGB pode variar entre 0 e 255.

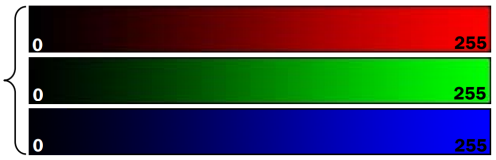
Figura 6: Escala de cinza

Fonte: Sick (2010).

Figura 7: Binário

Fonte: Sick (2010).

Figura 8: RGB



Fonte: Sick (2010).

2.1.6 HSV

O HSV é um espaço de cores do modelo *Hexcone*, possui o objetivo de capturar as noções comuns de matiz (*hue*), saturação (*saturation*) e valor (*value*) em três dimensões para descrever uma cor (SMITH, 1978). O matiz é o atributo da cor que descreve a cor pura (por exemplo, amarelo puro, laranja puro ou vermelho puro). A saturação fornece uma medida da diluição da cor pura por luz branca (variando-se a saturação, varia-se da cor pura até o branco). O valor refere-se à intensidade luminosa. Por exemplo, para uma cor com determinado H e S, quando o V for máximo, esta cor está "completamente iluminada". Já, independentemente de H e S, caso V seja zero não há sinal (cor preta) (GONZALEZ; WOODS, 2017).

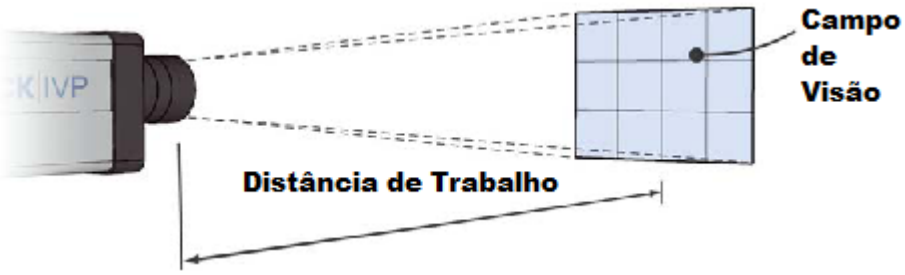
No OpenCV, o intervalo de *Hue* é de 0 a 179, o intervalo de *Saturation* e *Value* é de 0 a 255. Outras bibliotecas podem possuir outros intervalos, então a normalização destes intervalos deve ser realizada para comparar os valores entre bibliotecas diferentes (OPENCV, 2014).

2.1.7 Distância de Trabalho

A distância de trabalho (Figura 9) é a distância entre a lente e o objeto. Esta é importante para determinar parâmetros relacionados à aquisição das imagens, como o campo de visão.

Considerando as variáveis descritas na Figura 10, a Equação 1 (VD, 2018) pode ser

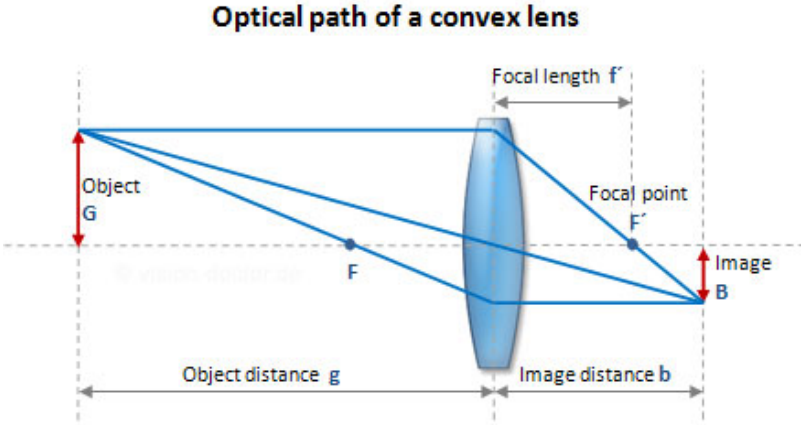
Figura 9: Distância de trabalho e campo de visão



Fonte: Sick (2010).

utilizada para obter a distância de trabalho.

Figura 10: Óptica geométrica



Fonte: VD (2018).

$$g = f' * (G/B + 1) \tag{1}$$

2.1.8 Campo de Visão

O campo de visão ou FOV (*Field of View*) é a área da cena que a câmera captura e é especificada pela largura e altura, como pode ser visto na Figura 9 (SICK, 2010).

O campo de visão é determinado pelos seguintes fatores (HORNBERG, 2017):

- Máximo tamanho do objeto;
- máxima variação da apresentação do objeto na translação e orientação;
- margem para o deslocamento do objeto;

- relação de aspecto do sensor de imagem.

Assim, o campo de visão pode ser expresso por meio da Equação 2 (HORNBERG, 2017):

$$FOV = maximo_{tamanho} + tolerancia + margem + relacao \tag{2}$$

No caso de múltiplos objetos, o máximo tamanho do campo de visão deverá possuir as dimensões que contenham a mínima quantidade de pixels necessários para reconhecer o objeto. Ou no caso de medições, cálculos da precisão e resolução devem ser realizados.

2.2 SEGMENTAÇÃO

Para interpretar uma imagem e gerar a descrição de seu conteúdo, uma das abordagens consiste em dividí-la em regiões disjuntas, preferencialmente correspondendo aos objetos de interesse. Este processo de divisão da imagem em regiões significativas é chamado de segmentação.

Uma região significativa pode corresponder a um objeto, parte de um objeto ou ao fundo homogêneo de uma imagem em que os objetos sobressaem. Os objetivos de cada aplicação condicionam os tipos de regiões a serem extraídas e os tipos de técnicas utilizadas. Por essa razão, não é possível afirmar que há um método de segmentação perfeito, no sentido de que apresentará excelentes resultados para qualquer tipo de imagem. Assim, técnicas de segmentação *ad hoc* são bastante comuns (TORRAS, 1992).

2.3 TRANSFORMADA DE DISTÂNCIA

A Transformada de Distância (TD), introduzida por Rosenfeld e Pfaltz (1968), é uma operação que, a partir de uma imagem binária contendo objetos e não-objetos (fundo), fornece outra imagem, em níveis de cinza, cujo valor de cada pixel, representa a sua distância à fronteira mais próxima.

Normalmente a distância é calculada a partir de cada pixel pertencente à fronteira ao pixel mais próximo, pertencente tanto ao fundo, quanto aos objetos. As aplicações da TD incluem a segmentação de imagens, a obtenção de esqueletos, a análise de *clusters*, e o cálculo da dimensão fractal, entre outras (LUPPE; COLOMBINI; RODA, 2014).

A Equação 3 é utilizada para obter a TD (JAIN, 1989):

$$u_k(m,n) = u_0(m,n) + min_{\Delta(m,n;i,j)}\{u_{k-1}(i,j);((i,j):\Delta(m,n;i,j)\leq 1)\}$$
$$u_0(m,n) \stackrel{\Delta}{=} u(m,n) \quad \text{e} \quad \text{k = 1,2, ...} \tag{3}$$

Onde $\Delta(m,n;i,j)$ é a distância entre (m, n) e (i, j). A transformada é realizada quando é igual à máxima espessura da região.

A esqueletização são o conjunto de pontos dados pela Equação 4 (JAIN, 1989):

$$\{(m,n):u_k(m,n)\geq u_k(i,j),\Delta(m,n;i,j)\leq 1)\} \tag{4}$$

2.4 WATERSHED

Deseja-se identificar bacias hidrográficas, definidas por mínimos regionais e suas regiões de domínio. Intuitivamente, a transformada *watershed* trata de encontrar os pontos em uma superfície onde uma gota d'água possa escorrer para dois mínimos regionais diferentes. Esta analogia pode ser também criada inversamente, onde um nível d'água é elevado através de uma superfície, inundando a partir dos mínimos regionais, e, nos pontos onde águas provenientes de mínimos diferentes se tocarem ergue-se uma barreira, que constitui a linha de divisão das bacias (KORBES, 2010).

2.5 OPENCV

O OpenCV (*Open-source Computer Vision*) (BRADSKI, 2000) é uma biblioteca livre e multiplataforma, originalmente desenvolvida pela Intel em 2000.

É uma biblioteca de visão computacional, que possui uma grande variedade de módulos que podem ajudar muito em problemas de visão computacional. Mas, possivelmente, a parte mais útil do OpenCV está em sua arquitetura e gerenciamento de memória. Ela fornece uma estrutura para trabalhar com imagens e vídeos sem a necessidade de alocar ou desalocar memória para as imagens (BRAHMBHATT, 2013).

2.6 NUMPY E SCIPY

O NumPy (OLIPHANT, 2006), é uma biblioteca para a linguagem de programação Python (ROSSUM, 2018), que fornece suporte para matrizes grandes e multidimensionais. Com

o NumPy, as imagens são demonstradas como matrizes multidimensionais. Além da eficiência computacional e de recursos, muitas outras bibliotecas também utilizam as representações de matrizes do NumPy.

O SciPy (JONES; OLIPHANT; PETERSON, 2001), fornece suporte adicional para computação científica e técnica (ROSEBROCK, 2016).

2.7 ERRO ABSOLUTO MÉDIO

O cálculo do erro absoluto médio, ou do inglês MAE (*Mean Absolute Error*), envolve o somatório das magnitudes (valores absolutos) dos erros para obter o erro total e depois a divisão pela quantidade n (WILLMOTT; MATSUURA, 2005). Conforme a Equação 5:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n} \tag{5}$$

O erro absoluto médio é medido nas mesmas unidades que os dados. Ele é menos sensível à erros ocasionais muito grandes, porque ele não eleva ao quadrado os erros no cálculo (NAU, 2016).

3 MATERIAIS E MÉTODOS

3.1 VISÃO GERAL

A Figura 11 descreve brevemente cada uma das etapas da elaboração do sistema de contagem de sementes de soja. Os próximos tópicos deste capítulo descrevem em detalhes estas etapas.

Figura 11: Etapas de desenvolvimento

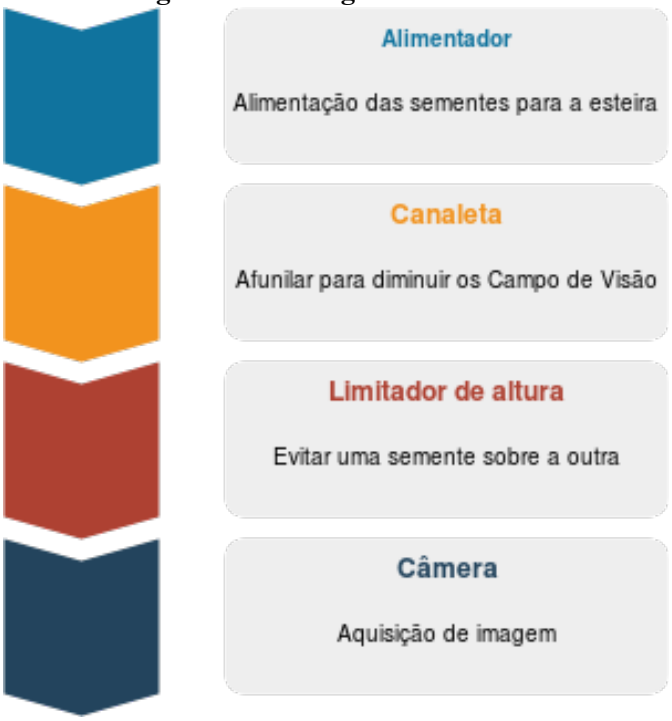
Fonte: Autoria própria.

3.2 HARDWARE

3.2.1 Esteira

O fluxograma da esteira pode ser visto na Figura 12, em que é composto por um alimentador, canaleta, limitador de altura e câmera.

Figura 12: Fluxograma da esteira



Fonte: Autoria própria.

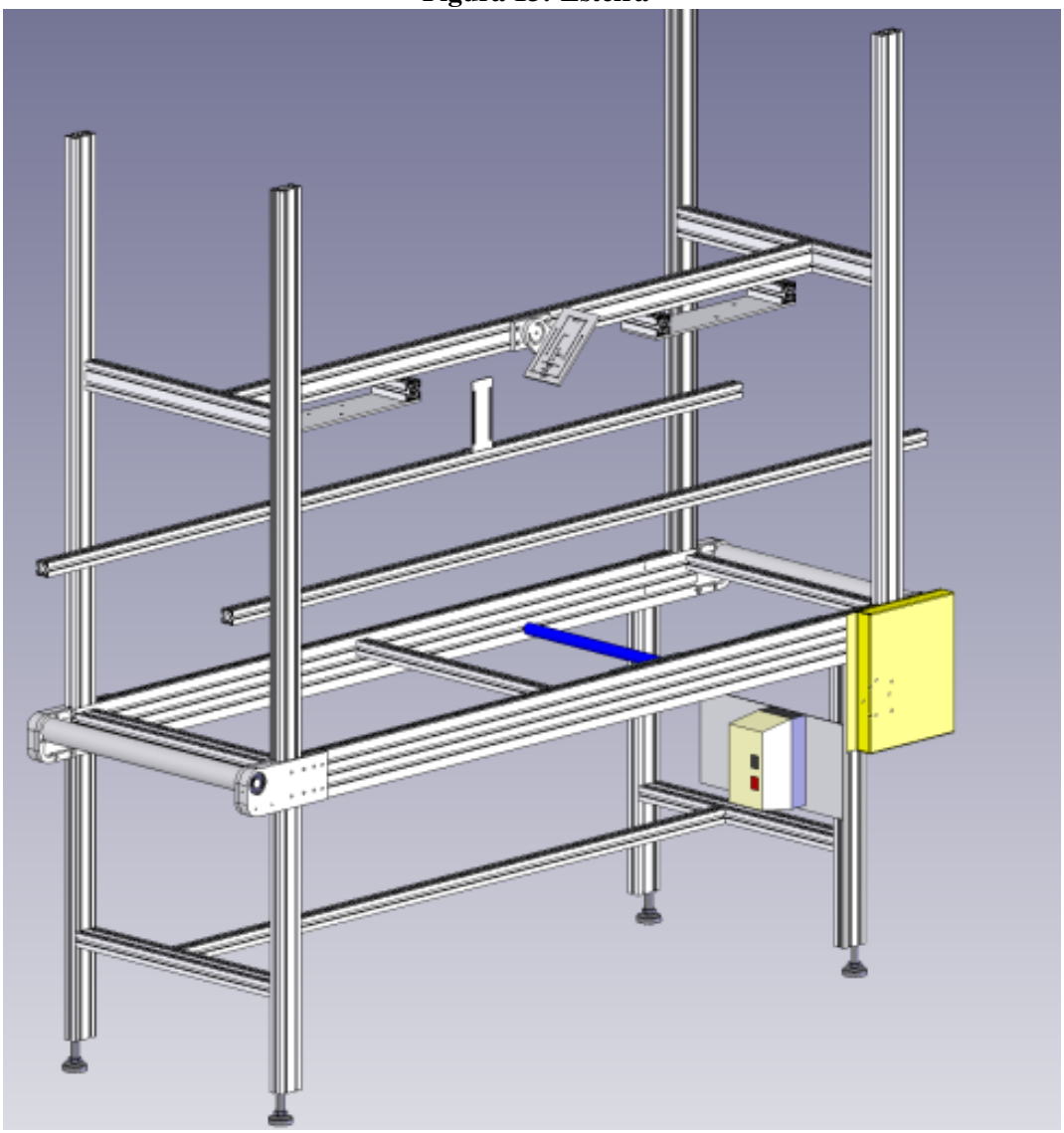
A esteira utilizada (Figura 13) é de perfil de alumínio, possui 2 m de comprimento e 0,56 m de largura. Com um motor de 0,5 cv e um inversor de frequência do modelo IG5A, para o controle de velocidade.

A correia da esteira é de tecido poliéster de cor branca, possibilitando o contraste entre a superfície e as sementes de soja.

Foram instaladas canaletas (Figura 14) para afunilar os grãos e diminuir o campo de visão. Além de evitar que as sementes de soja caiam da esteira.

Um limitador de altura (Figura 15) foi posicionado no início da esteira para evitar que sementes sobrepostas passem pela esteira. Ou seja, desagrupar as sementes de soja, para que não fiquem sobre a outra, impossibilitando a contagem e segmentação.

Figura 13: Esteira



Fonte: Autoria própria.

3.2.2 Câmera em Escala de Cinza e Iluminação

Foram realizados estudos para a utilização de uma câmera industrial da SICK AG, modelo IVC-2DM1121, que pode ser visto na Figura 16.

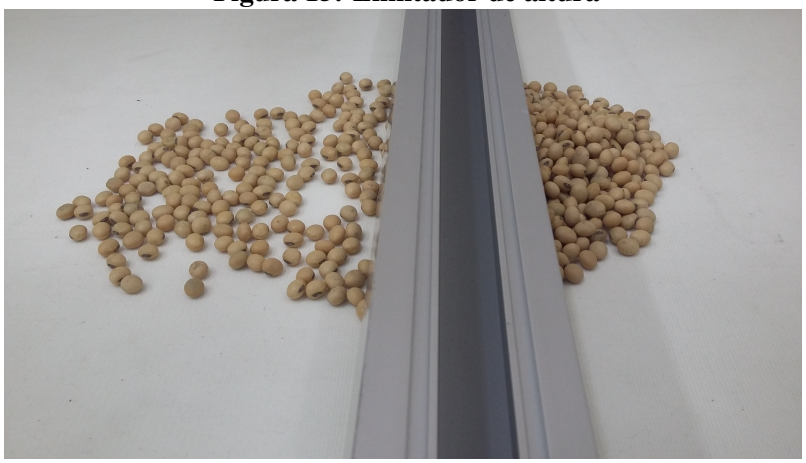
A câmera em escala de cinza possui os dados técnicos da Tabela 1. Foi utilizada uma lente objetiva C-Mount e com distância focal de 8 mm (Figura 17).

As iluminações do estudo foram de LED diretas, com ângulo de 45° relação aos objetos ou LED de baixo ângulo em anel. Ambas se mostraram falhas e um *backlit conveyor* seria o melhor tipo de iluminação para a aplicação. Porém, a esteira utilizada nos testes não era adaptável para a instalação deste tipo de iluminação.

Figura 14: Canaletas

Fonte: Autoria própria.

Figura 15: Limitador de altura



Fonte: Autoria própria.

Figura 16: Câmera IVC-2D

Fonte: Sick (2010).

A falta de iluminação adequada resultou em sombras entre os grãos de soja. Como a câmera é em escala de cinza, as sombras possuíam a mesma intensidade que as sementes de

Tabela 1: Dados técnicos

Dado	Valor
Sensor de Imagem	1/3”
Lente	C-Mount ou CS-Mount
Resolução	1.024 px x 768 px (XGA)
Processador	800 MHz

Fonte: Autoria própria.

Figura 17: Lente

Fonte: Sick (2010).

soja (Figura 18). Impossibilitando a segmentação da imagem.

Para contornar o problema, uma câmera colorida deve ser utilizada.

3.2.3 Câmera Colorida

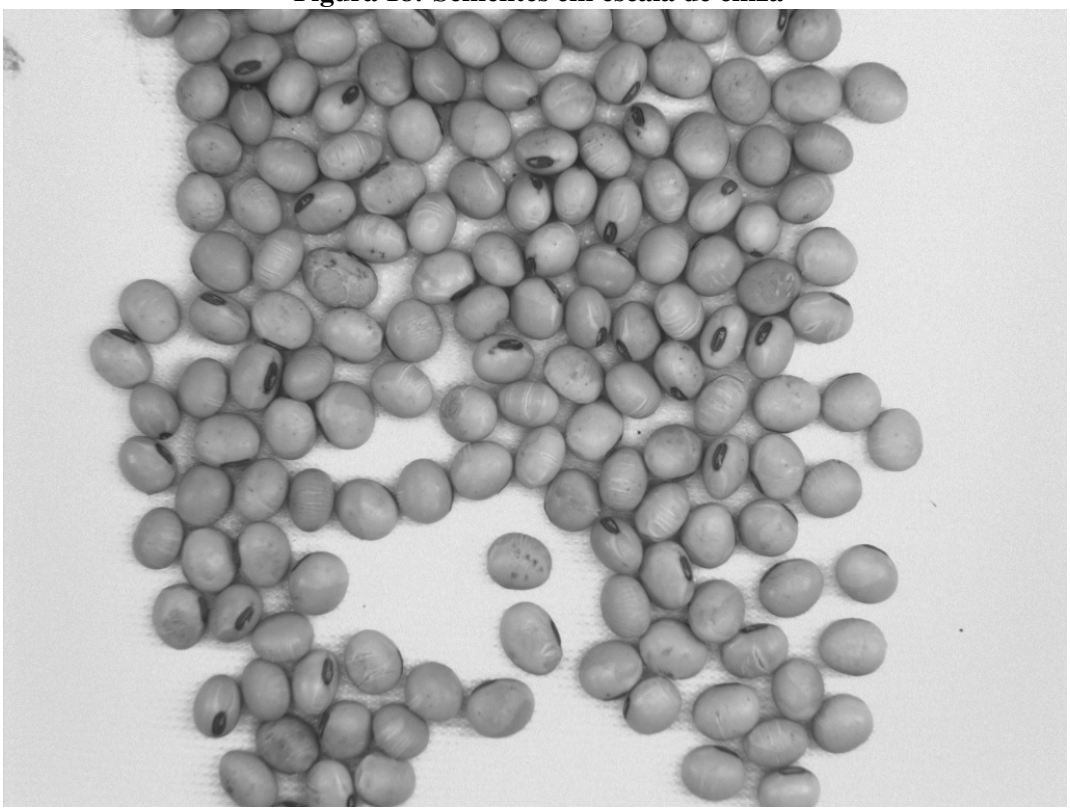
A câmera colorida escolhida foi de um celular com gravação de vídeo em *Full HD*, com resolução de 1920 x 1080 px ou 2.07 Megapixels e 30 fps.

A câmera possui um sensor de imagem de 1/2,5”ou 5,7 mm e distância focal de 3,93 mm. Então, com a Equação 10, é possível obter a distância de trabalho a partir dos dados de distância de trabalho e sensor de imagem. Possuindo uma distância de trabalho de aproximadamente 165 mm, para um campo de visão de 233 mm x 131 mm.

Um grão de soja possui muitas variações de forma e tamanho. Mas considerando que a menor dimensão do grão de soja possui aproximadamente 4 mm, a câmera precisa detectar uma elipsoide de pelo menos 4 mm no eixo vertical ou horizontal e possuir resolução para detectar este objeto.

Na distância de trabalho de 165 mm, a dimensão de 4 mm será de aproximadamente 33 px, aceitável para detectar um grão de soja.

Figura 18: Sementes em escala de cinza



Fonte: Autoria própria.

3.3 SOFTWARE

A escolha da linguagem de programação, ferramentas e bibliotecas serão explicadas nesta seção. Em que o principal requisito é a licença de software livre.

A biblioteca OpenCV será utilizada por causa dos seguintes motivos:

- Licença de software livre BSD (*Berkeley Software Distribution*);
- ampla utilização da comunidade de visão computacional;
- fácil integração com processadores para otimização do desempenho OpenCL (*Open Computing Language*) ou CUDA (*Compute Unified Device Architecture*).

As funções da OpenCV foram chamadas a partir de códigos em Python, pois esta linguagem apresenta:

- Licença de software livre GPL (*GNU General Public License*);

- extensiva lista de bibliotecas;
- curva de aprendizado pequena;
- uma das principais linguagens de programação da comunidade OpenCV.

A IDE utilizada é o PyCharm Community da JetBrains, pelos seguintes motivos:

- Licença de software livre Apache 2;
- auto completa o código;
- terminal local incorporado;
- suporte para ferramentas de controle de versão, como o Git;
- refatoração;
- verificação da PEP8.

As bibliotecas NumPy e SciPy foram utilizadas pelos seguintes motivos:

- Licença de software livre BSD;
- biblioteca científica com uma comunidade ativa e grande;
- eficiência no desempenho de operações com matrizes.

3.4 DESENVOLVIMENTO

3.4.1 Visão Geral

O programa para realizar a contagem de sementes é dividido em três partes (Figura 19). Com as etapas de aquisição de imagem, pré-processamento e segmentação.

O fluxograma da Figura 20 ilustra a lógica utilizada para criar o programa.

Figura 19: Visão geral de desenvolvimento

Fonte: Autoria própria.

3.4.2 Aquisição de Imagem

A câmera realiza a aquisição de vídeo com 30 *fps*, então, para não contar sementes múltiplas vezes e obter somente os *frames* desejados. Um *encoder* ou velocidade constante da esteira devem ser utilizados. A Figura 21 ilustra como é a sequência de *frames*.

A esteira foi configurada para uma velocidade constante de 28,27 mm/s, um *frame* possui 131 mm na direção da esteira e a velocidade de gravação de vídeo é de 30 fps, portanto, é possível calcular o tempo ou quantidade de *frames* entre cada *frame* com a Equação 6.

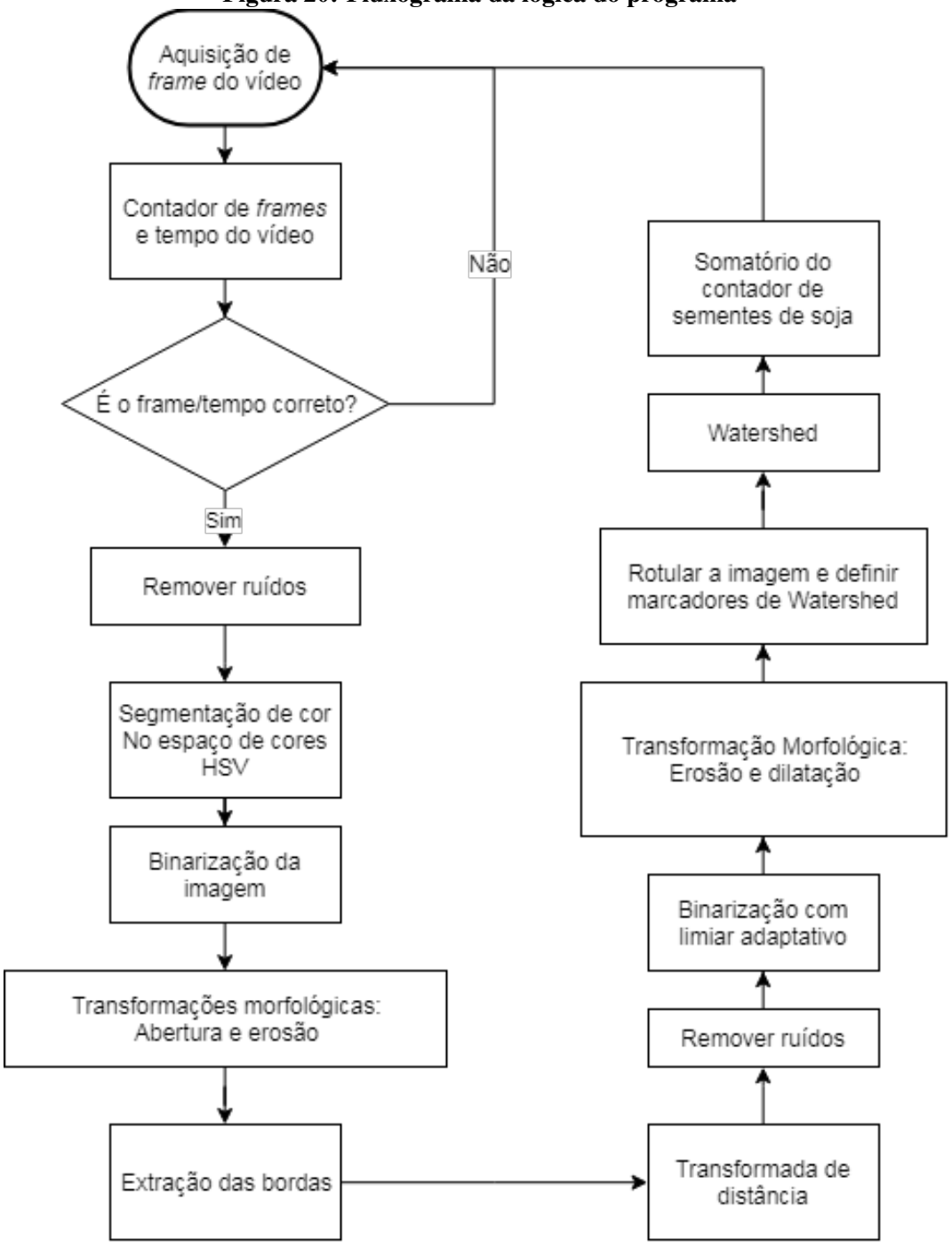
$$v = d/t \tag{6}$$

Então a cada tempo de 4,63 s ou a cada 139 *frames*, será utilizado a imagem para realizar a contagem de sementes de soja. Com a lógica da Equação 7, que utiliza a operação módulo, ou seja, encontra o resto da divisão de um número por outro. Em que a variável var_{video} é o contador de tempo ou de *frames* do vídeo e a variável var_{cte} é a constante de tempo ou de quantidade de *frames* entre cada *frame*.

$$var_{video} \bmod var_{cte} \tag{7}$$

Se o resto da operação módulo for igual a zero, a imagem é utilizada no programa, caso contrário carrega o próximo *frame* do vídeo.

Figura 20: Fluxograma da lógica do programa



Fonte: Autoria própria.

3.4.3 Remover Ruídos com a Função Blur

A função *blur* do OpenCV é utilizada para realizar uma leve suavização na imagem (filtragem passa-baixas), removendo ruídos pequenos. Em que os parâmetros são a imagem e o *kernel*.

Figura 21: Sequência de frames

Fonte: Autoria própria.

A suavização é obtida ao convoluir uma imagem com um *kernel* 2D de filtro passa-baixas. A função obtém a média de todos os pixels sob a área do *kernel* e substitui o elemento central (OPENCV, 2014).

O *kernel* é dado pela Equação 8 (OPENCV, 2014).

$$K = \frac{1}{k_{size}.width \times k_{size}.height} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ 1 & 1 & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix} \tag{8}$$

Um exemplo de utilização é um *kernel* $(3,3)$, como pode ser visto na Equação 9.

$$K = \frac{1}{3 \times 3} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{9}$$

3.4.4 Segmentação de Cor

A segmentação é realizada convertendo a imagem com os ruídos removidos para o espaço de cores HSV. Então, uma máscara com intervalo dos parâmetros de cores das sementes de soja é criada com a função *inRange*() do OpenCV.

A Equação 10 é utilizada para obter os valores binários da segmentação, em que valores dentro do intervalo são iguais a 255 e fora igual a 0.

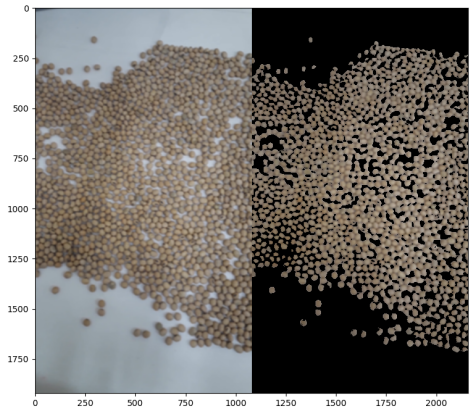
$$imascara = mascara > 0 \tag{10}$$

Uma matriz de imagem de variável *slicer* é criada com a função do NumPy *zeros_like*() e mesmas dimensões da imagem original.

A Equação 11 gera o resultado da Figura 22, que ilustra a imagem de um frame à esquerda e da segmentação de cor à direita.

$$slicer[imascara] = imagem[imascara] \tag{11}$$

Figura 22: Um frame da imagem original à esquerda e a segmentação de cor à direita



Fonte: Autoria própria.

3.4.5 Binarização

Primeiro, a imagem obtida com a segmentação de cor é convertida para escala de cinza. Então, a binarização é obtida com uma limiarização simples, com a função *threshold*, em que o tipo de limiarização utilizada é a binária ou $THRESH\_BINARY$, que é descrita pela Equação 12 (OPENCV, 2014). Onde *dst* é a imagem de destino, *src* é a imagem de origem e o *limiar* é um valor de intensidade em escala de cinza.

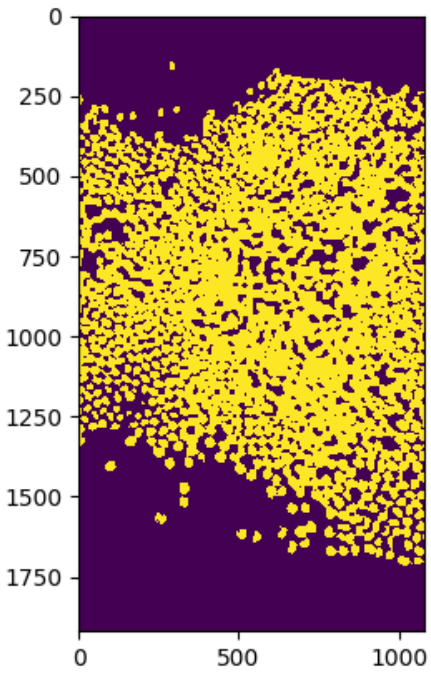
$$dst(x,y) = \begin{cases} maxval & \text{se src(x,y) } > \text{limiar} \\ 0 & \text{caso contrário} \end{cases} \tag{12}$$

O resultado da binarização pode ser visto na Figura 23.

3.4.6 Transformações Morfológicas

Transformações morfológicas são operações simples baseadas no formato da imagem, normalmente aplicadas em imagens binárias. Ela precisa de dois parâmetros de entrada, a

Figura 23: Binarização



Fonte: Autoria própria.

imagem e o *kernel*, ou o elemento estruturante. As duas operações básicas principais são a erosão e a dilatação (OPENCV, 2014).

A função do OpenCV $morphologyEx()$ realiza transformações morfológicas avançadas, em que os tipos são:

- $MORPH\_ERODE$;
- $MORPH\_DILATE$;
- $MORPH\_OPEN$;
- $MORPH\_CLOSE$;
- $MORPH\_GRADIENT$;
- $MORPH\_TOPHAT$;
- $MORPH\_BLACKHAT$;
- $MORPH\_HITMISS$.

A operação morfológica utilizada foi a abertura e em seguida a erosão, em que a abertura é a erosão seguida da dilatação. A Equação 13 (OPENCV, 2014), descreve a operação morfológica de abertura.

$$dst = abertura(src, elemento) = dilate(erode(src, elemento)) \tag{13}$$

A função de erosão pode ser vista na Equação 14 (OPENCV, 2014), em que erode a imagem utilizando o elemento estruturante especificado. Todos os pixels próximos à fronteira do objeto são descartados, em função do tamanho do elemento estruturante.

$$dst(x,y) = \min_{(x',y'):element(x',y')\neq 0} src(x+x', y+y') \tag{14}$$

Um elemento estruturante em forma de elipse é utilizado, em que um exemplo de matriz 3x3 pode ser visto na Equação 15.

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{15}$$

O resultado após as operações morfológicas de abertura e erosão pode ser visto na Figura 24. Observa-se que em comparação com a Figura 23, algumas sementes de soja estavam interligadas. Mas após a transformação morfológica, possuem forma mais definida e estão separadas.

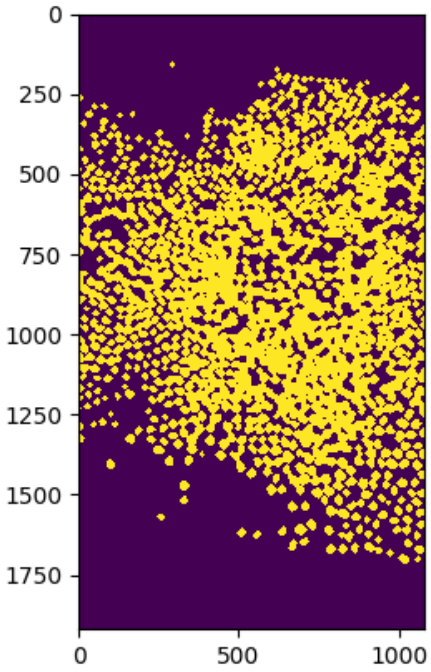
3.4.7 Extração das Bordas

As bordas são obtidas para auxiliar na definição dos marcadores do algoritmo *Watershed*. Para obtê-las é necessário dilatar a imagem do resultado da transformação morfológica, com um elemento estruturante maior, e depois extrair a erosão, com um elemento estruturante menor, da dilatação realizada. Como pode ser visto na Equação 16.

$$bordas = \text{dilatação}(elemento\_estruturante_{maior}) - \text{erosão}(elemento\_estruturante_{menor}) \tag{16}$$

A função de dilatação, apresentada na Equação 17 (OPENCV, 2014), é o oposto da erosão, em que os pixels próximos à fronteira do objeto são dilatados.

Figura 24: Transformação morfológica



Fonte: Autoria própria.

$$dst(x,y) = \max_{(x',y'):element(x',y')\neq 0} src(x+x',y+y') \tag{17}$$

Em que o resultado destas operações com um elemento estruturante elíptico pode ser visto na Figura 25.

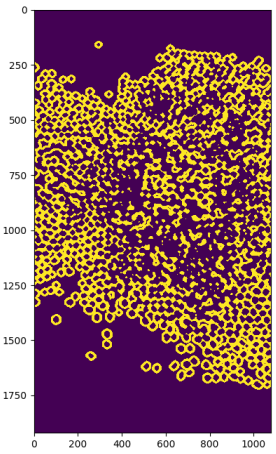
3.4.8 Transformada de Distância

A função $distanceTransform()$ do OpenCV realiza a transformada da distância de uma imagem binária. Na saída da transformada da distância os valores de cada pixel contém a distância daquele pixel até o pixel de fundo (*background*) mais próximo. Ou seja, calcular a distância precisa ou aproximada de cada pixel da imagem binária ao pixel zero mais próximo (OPENCV, 2014).

Os parâmetros de entrada da função são uma imagem binária, o tipo de transformada de distância e o tamanho da máscara.

Os tipos de transformada de distância são (OPENCV, 2014):

Figura 25: Resultado da extração das bordas



Fonte: Autoria própria.

- *DIST_USER*;
- *DIST_L1*;
- *DIST_L2*;
- *DIST_C*;
- *DIST_L12*;
- *DIST_FAIR*;
- *DIST_WELSCH*;
- *DIST_HUBER*.

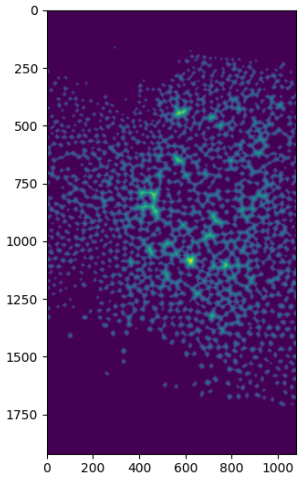
O tipo utilizado é o *DIST_L2* ou distância euclidiana, que normalmente utiliza uma máscara 3*x*3 para obter uma distância rápida e aproximada. Para uma estimativa mais precisa, uma máscara 5*x*5 é utilizada.

Os tipos de máscaras da transformada de distância são (OPENCV, 2014):

- *DIST_MASK_3*;
- *DIST_MASK_5*;
- *DIST_MASK_PRECISE*.

O resultado da transformada de distância, utilizando a máscara *DIST_MASK_3* pode ser vista na Figura 26.

Figura 26: Resultado da transformada de distância



Fonte: Autoria própria.

Então os valores são normalizados linearmente para valores de 0 a 255 e *8-bit unsigned integer* com uma função a normalização do OpenCV *normalize* ou com a função da Equação 18.

$$I_N = (\text{novoMax} - \text{novoMin}) \times \frac{I - \text{Min}}{\text{Max} - \text{Min}} + \text{novoMin} \tag{18}$$

Com isso, a aplicação desta normalização na imagem da transformada das distâncias é obtida a partir da Equação 19 (dt_novo é a transformada das distâncias normalizada).

$$dt_{novo} = (255 - 0) \times \frac{dt - Min_{dt}}{Max_{dt} - Min_{dt}} + 0 \tag{19}$$

3.4.9 Remover Ruídos com a Função Gaussian Blur

A função *GaussianBlur*() do OpenCV é utilizada para remover os ruídos. Ao invés do elemento estruturante, um *kernel* Gaussiano é utilizado. A altura e largura do *kernel* devem ser especificados, sendo valores inteiros, positivos e ímpares.

O desvio padrão nas direções *x* e *y* também devem ser especificados, *sigmaX* e *sigmaY* respectivamente. Se somente o *sigmaX* for especificado, o valor de *sigmaY* será igual ao de *sigmaX* (OPENCV, 2014).

3.4.10 Binarização com Limiar Adaptativo

A limiarização simples utiliza um valor global para definir o limiar. Mas se as condições não são uniformes por toda a imagem, a limiarização adaptativa pode ser utilizada. Neste algoritmo o cálculo do limiar é realizado em regiões pequenas da imagem. Então, limiares diferentes são obtidos em diferentes regiões da mesma imagem.

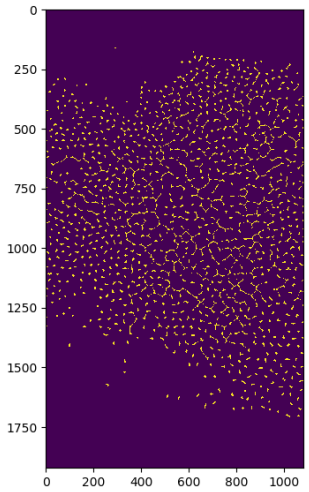
O limiar adaptativo é usado para obter o máximo regional da transformada de distância. O máximo regional é a região na imagem que todos os pixels vizinhos possuem valor menor. Muitos máximos locais são resultantes de ruídos, por isso é necessário remover estes ruídos da imagem.

A função possui sete parâmetros de entrada, imagem de origem, imagem de destino, valor máximo, método adaptativo, tipo de limiarização, tamanho do bloco e constante.

Há dois algoritmos de método adaptativo, o $ADAPTIVE\_THRESH\_MEAN\_C$ e o $ADAPTIVE\_THRESH\_GAUSSIAN\_C$, em que o segundo foi utilizado. O valor de limiarização $T(x,y)$ é uma soma ponderada da vizinhança $(tamanho\_bloco \times tamanho\_bloco)$ de (x,y), menos a constante C (OPENCV, 2014).

No resultado da figura 27, o valor máximo é 255 e o tipo de limiarização é o $THRESH\_BINARY$, descrito na Equação 12.

Figura 27: Resultado da binarização com limiar adaptativo



Fonte: Autoria própria.

3.4.11 Erosão e Dilatação

As transformações morfológicas de erosão e dilatação são utilizadas para limpar a imagem após a limiarização. A operação morfológica de abertura não é utilizada, pois tamanhos de elementos estruturantes diferentes serão utilizados em cada função.

Na etapa de extração das bordas foram utilizados dois elementos estruturantes, um maior e outro menor. Os mesmos serão utilizados para remover o ruído. Na erosão o elemento estruturante menor é usado e na dilatação o elemento estruturante menor.

3.4.12 Rotulação da Imagem e Definição dos Marcadores

A função *label*() do pacote *Multi-dimensional image processing* (ndimage) da biblioteca SciPy é utilizado para rotular os elementos em um vetor ou matriz, em que a função retorna um vetor ou matriz enumerando cada elemento único e o número de elementos.

Por exemplo, para a matriz dada pela Equação 20 (JONES; OLIPHANT; PETERSON, 2001).

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{20}$$

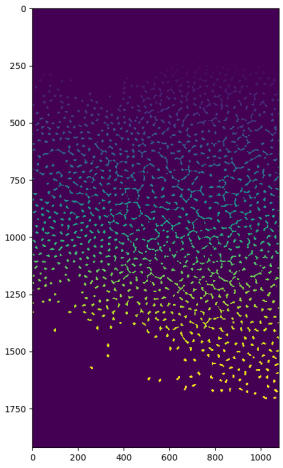
Retorna a matriz da Equação 21 (JONES; OLIPHANT; PETERSON, 2001) e o número de elementos, ou seja, quatro.

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 2 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \end{pmatrix} \tag{21}$$

O resultado pode ser visto na figura 28, onde cada elemento encontrado possui um número inteiro único, por isso, o gradiente de cores pode ser percebido na imagem.

A abordagem descrita na seção 2.4 fornece um resultado sobressegmentado devido aos ruídos ou quaisquer irregularidades na imagem. Então a implementação do algoritmo *watershed* baseado em marcadores foi implementado pelo OpenCV, onde é necessário especificar quais pontos de vale devem ser mesclados e quais não. Isso é realizado fornecendo rótulos diferentes

Figura 28: Resultado da rotulação

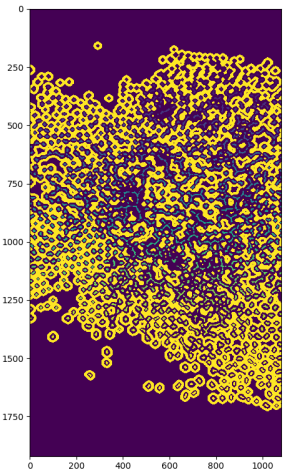


Fonte: Autoria própria.

para os objetos conhecidos. Rotulando as regiões em que há a certeza de que é um objeto de uma cor, outra região em que há a certeza de que é um plano de fundo ou que não é um objeto de outra cor e a região em que não há certeza, de zero. Estes são os marcadores segundo OpenCV (2014).

Para completar a criação dos marcadores do *watershed*, a união das imagens da rotulação e das bordas é realizada, resultando na figura 29.

Figura 29: Marcadores

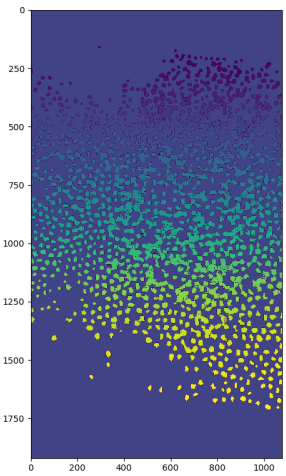


Fonte: Autoria própria.

3.4.13 Watershed

A função *watershed*() do OpenCV implementa o algoritmo descrito em Meyer (1992). Ela possui dois parâmetros de entrada, a imagem, que deve ser de 8-bits e possuir 3 canais e os marcadores, que deve ser de 32-bits e possuir um único canal. Ambas imagens devem ser do mesmo tamanho. O resultado desta segmentação pode ser visto na Figura 30.

Figura 30: Watershed



Fonte: Autoria própria.

O resultado do *watershed* é convertido para 8-bits e o contador é incrementado com o número de elementos encontrado. O programa retorna ao início do *loop* para realizar a aquisição do *frame* do vídeo.

3.5 ANÁLISE DE RESULTADOS

O erro total dos resultados obtidos foram calculados a partir do erro absoluto médio, da Equação 5. Em que foram realizados três experimentos, que podem ser vistos na Tabela 2.

Tabela 2: Resultados

Quantidade
4538
4782
5118

Fonte: Autoria própria.

O cálculo do erro absoluto médio pode ser visto na Equação 22. Em porcentagem, o erro é de 5,32%.

$$MAE = \frac{\sum_{i=1}^{n} |e_i|}{n} = \frac{|4538 - 5000| + |4782 - 5000| + |5118 - 5000|}{3} = 266 \qquad (22)$$

Embora estes resultados mostrem-se bastante satisfatórios no contexto deste trabalho, e também considerando que trata-se da primeira versão do algoritmo de visão desenvolvido para a contagem de sementes de soja, é importante destacar que este erro de 5% não é adequado para a utilização em um produto comercial. Este erro ocorreu por causa da baixa qualidade de iluminação, gerando muitos ruídos de sombras e sujeiras na esteira.

Uma possível solução para isso é a utilização de um *backlit conveyor*, que tornará o sistema em um ambiente mais controlado e uniforme. Sem muitas variações nas imagens e sem sombras. A utilização de uma câmera industrial RGB também é recomendada, para controlar a aquisição dos *frames* em conjunto com um *encoder*. Pois a velocidade pode não ser constante, devido ao desgaste das engrenagens da esteira ou à variações da carga sobre a esteira. Nas condições deste estudo, a velocidade era constante por se tratar de uma esteira de testes pequena no laboratório da Similar Tecnologia e Automação.

4 CONCLUSÃO

O estudo foi dividido em quatro etapas, a escolha do hardware, a escolha do software, o desenvolvimento do programa e a análise de resultados.

Na definição do hardware, a esteira utilizada é do laboratório da empresa Similar Tecnologia e Automação, em que o estudo com uma câmera industrial em escala de cinza se mostrou ineficiente pela falta de iluminação adequada. A utilização de uma câmera colorida era a melhor opção para contornar este problema.

O principal requisito na escolha da linguagem de programação e bibliotecas foi a licença de software livre. O Python foi a linguagem utilizada, a IDE PyCharm e as bibliotecas NumPy, SciPy e OpenCV. Todas possuem uma comunidade de visão computacional e científica ativa e grande.

No desenvolvimento, o algoritmo *watershed* baseado em marcadores foi utilizado. Mas primeiro é necessário remover ruídos, com transformações morfológicas e definir os marcadores do algoritmo, com a transformada de distância e rotulação.

Conforme mencionado anteriormente, embora o erro absoluto médio de 5,32% obtido na contagem das sementes de soja seja satisfatório para a primeira versão deste projeto, melhorias são necessárias para a sua utilização como um produto comercial. Para a continuação do projeto é possível a utilização de um *backlit conveyor*, que tornará o sistema em um ambiente mais controlado e uniforme. Sem muitas variações nas imagens e sem sombras.

A utilização de uma câmera industrial RGB também é recomendada, para controlar a aquisição dos *frames* em conjunto com um *encoder*.

A falta de iluminação adequada resultou em muitos ruídos, impossibilitando uma segmentação confiável. Possíveis modificações do algoritmo são a utilização de segmentação de cor adaptativa ou aprendizado de máquina.

REFERÊNCIAS

BATCHELOR, Bruce G.; WHELAN, Paul F. **Intelligent vision systems for industry**. London, Springer-Verlag, 1997.

BEYERER, Jürgen; LEÓN, Fernando P.; FRESE, Christian. **Machine vision - Automated visual inspection**: Theory, practice and applications. Berlin, Heidelberg, Springer-Verlag, 2016. 798 p.

BRADSKI, Gary. **The OpenCV library**. Copyright© 2018 UBM, publicado em: 01 nov. 2000. Disponível em: <http://www.drdobbs.com/open-source/the-opencv-library/184404319>. Acesso em: 15 ago. 2018.

BRAHMBHATT, Samarth. **Practical OpenCV**. Apress Media, 2013.

CONAB. **Grãos**: Série histórica. Companhia Nacional de Abastecimento (Conab), 2018. Disponível em: <https://portaldeinformacoes.conab.gov.br/index.php/safra-serie-historica-dashboard>. Acesso em: 20 jul. 2018.

GONZALEZ, Rafael C.; WOODS, Richard E. **Digital image processing**. 4. ed. Pearson, 2017.

HOBSON, D. M.; CARTER, R. M.; YAN, Y. **Rule based concave curvature segmentation for touching rice grains in binary digital images**. IEEE Instrumentation and Measurement Technology Conference, Singapura, 2009. p. 1685-1689. Disponível em: <https://ieeexplore.ieee.org/document/5168727/authors#authors>. Acesso em: 25 jul. 2018.

HORNBERG, Alexander. **Handbook of machine and computer vision: The guide for developers and users**. 2. ed. Wiley, 2017. 860 p.

JAIN, Anil K. **Fundamentals of digital image processing**. Englewood Cliffs, NJ: Prentice Hall, 1989.

JONES, Eric; OLIPHANT, Travis E.; PETERSON, Pearu. et al. **SciPy**: Open source scientific tools for Python. Copyright© 2018 SciPy developers, 2001. Disponível em: <http://www.scipy.org/>. Acesso em: 01 nov. 2018.

KHAJEHPOUR, Hassan; et al. **Detection and segmentation of erythrocytes in blood smear images using a line operator and watershed algorithm**. Journal of medical signals and sensors, v. 3, n. 3, jul. 2013. p. 164-171. Disponível em: <http://europepmc.org/articles/PMC3959006>. Acesso em: 20 jul. 2018.

KIRATIRATANAPRUK, Kantip; SINTHUPINYO,Wasin. **Color and texture for corn seed classification by machine vision**. International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS), Chiang Mai, Thailand, dez. 2011. Disponível em: <https://ieeexplore.ieee.org/document/6146100>. Acesso em: 25 jul. 2018.

KORBES, André. **Análise de algoritmos da Transformada Watershed**. Orientador: Roberto de Alencar Lotufo. Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação. Campinas, SP: 2010. Disponível em: <http://repositorio.unicamp.br/jspui/handle/REPOSIP/259826>. Acesso em: 10 set. 2018.

LIU, Tao; et al. **Rice and wheat grain counting method and software development based on Android system**. Computers and Electronics in Agriculture, v. 141, set. 2017. p. 302-309. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0168169917303289>. Acesso em: 20 jul. 2018.

LUPPE, Maximiliam; COLOMBINI, Angelo C.; RODA, Valentin O. **Arquitetura para transformada de distância e sua aplicação para o cálculo da dimensão fractal**. ago. 2014.

MEYER, F. **Color image segmentation**. International Conference on Image Processing and its Applications, 1992. p. 303-306. Disponível em: <https://ieeexplore.ieee.org/document/785528?arnumber=785528>. Acesso em: 15 ago. 2018.

MOULT, Eric; et al. **Implicit active contours for automatic brachytherapy seed segmentation in fluoroscopy**. Medical Imaging 2012: Image-Guided Procedures, Robotic Interventions, and Modeling, v. 8316, fev. 2012. p. 8316-7. Disponível em: <https://doi.org/10.1117/12.911153>. Acesso em: 20 jul. 2018.

NAU, Robert. **Statistical forecasting**: Notes on regression and time series analysis. 2016. Disponível em: <https://people.duke.edu/~rnau/411home.htm>. Acesso em: 01 nov. 2018.

NEILSEN, Michel; et al. **A Dynamic, real-time algorithm for seed counting**. Trabalho disponibilizado em: out. 2017. Disponível em: <https://www.researchgate.net/publication/322317840_A_Dynamic_Real-time_Algorithm_for_Seed_Counting>. Acesso em: 15 jul. 2018.

OLIPHANT, Travis E. **Numpy**: A guide to NumPy. Copyright© 2005-2018, NumPy Developers, 2006. Disponível em: <http://www.numpy.org/>. Acesso em: 01 nov. 2018.

OPENCV. **The OpenCV reference manual**. Copyright© 2018, OpenCV team, jun. 2014.

ROSEBROCK, Adrian. **Practical Python and OpenCV**: An introductory, example driven guide to image processing and computer vision. 4. ed., PyImageSearch.com, 2016.

ROSENFELD, Azriel; PFALTZ, John L. **Distance functions on digital pictures**. Pattern Recognition, v. 1, n. 1, jul. 1968. p. 33-61. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/0031320368900137>. Acesso em: 18 jul. 2018.

ROSSUM, Guido van. **Python tutorial**: Release 3.7.0. Python Software Foundation, Amsterdam, set. 2018. Disponível em: <https://bugs.python.org/file47781/Tutorial_EDIT.pdf>. Acesso em: 27 jul. 2018.

SAKAI, N. et al. **Two-dimensional image analysis of the shape of rice and its application to separating varieties**. Journal of Food Engineering, v. 27, n. 4, dez. 1996. p. 397-407. Disponível em: <https://www.sciencedirect.com/science/article/pii/0260877495000224>. Acesso em: 17 jul. 2018.

SICK. **Machine Vision**: Introduction. Copyright© 2018 SICK AG. Germany, material de treinamento, v. 2.41, mar. 2010.

SMITH, Alvy R. **Color gamut transform pairs**. SIGGRAPH'78 Proceedings, v. 12, n. 3, ago. 1978. p. 12-19. Disponível em: <https://dl.acm.org/citation.cfm?id=807361>. Acesso em: 01 nov. 2018.

TORRAS, Carme. **Computer Vision**: Theory and industrial applications. Berlin, Heidelberg, Springer-Verlag, 1992. 455 p.

VD. **Vision Doctor**: Solutions for Industrial Machine Vision. Copyright© Vision-Doctor.com (VD), 2018. Disponível em: <https://www.vision-doctor.com/en/>. Acesso em: 01 dez. 2018.

WANG, Y.; CHOU, J. J. **Automatic segmentation of touching rice kernels with an active contour model**. American Society of Agricultural and Biological Engineers, St. Joseph, Michigan, v. 47, n. 5, set. 2004. p. 1803-1811.

WILLMOTT, Cort J.; MATSUURA, Kenji. **Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance**. Climate Research, v. 30, p. 79, dez. 2005. Disponível em: <https://www.int-res.com/abstracts/cr/v30/n1/p79-82/>. Acesso em: 18 jul. 2018.

YAO, Qing; ZHOU, Yingfeng; WANG, Jianging. **An automatic segmentation algorithm for touching rice grains images**. 2010 International Conference on Audio, Language and Image Processing, Shanghai, China, 2010. p. 802-805. Disponível em: <https://ieeexplore.ieee.org/document/5685114>. Acesso em: 28 jul. 2018.

ZHANG, C. et al. **An improved watershed algorithm for color image segmentation**. International Conference on Computer Science and Electronics Engineering, Hangzhou, China, 2012. p. 69-72. Disponível em: <https://ieeexplore.ieee.org/document/6187966>. Acesso em: 28 jul. 2018.