Competição. Mínimo Produto Viável para a Indústria Agrícola

Integrantes da Startup: 1 - Eduardo Távora Rocha

2 - Gustavo Antônio Porto Cardoso

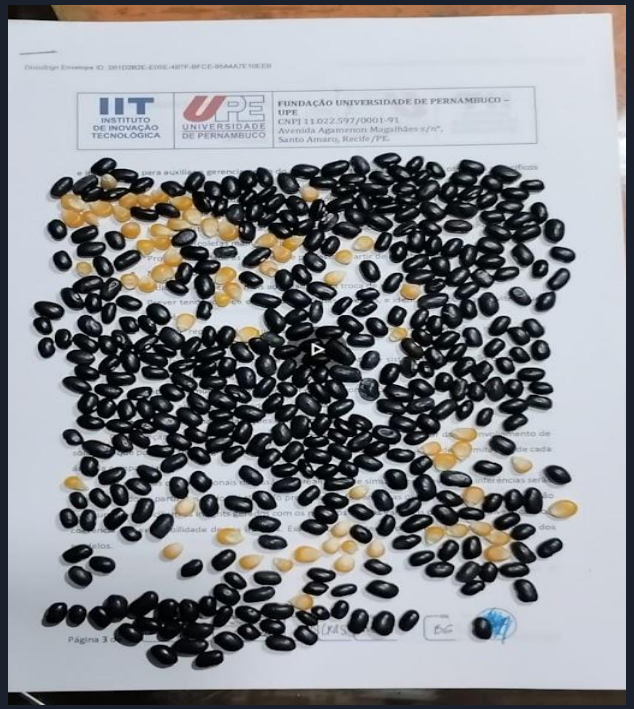
3 - Hiago Vinícius Domingues Marques Ribeiro

INTRODUÇÃO

O presente relatório aborda técnicas de implementação em Python visando a borragem do cabeçalho do documento, a separação e a contagem dos grãos de milho e de feijão apresentados na imagem fornecida.

Esse projeto está organizado em uma POC (prova de conceito), objetivando borrar o cabeçalho da imagem, e em um Mínimo Produto Viável, no qual estará presente o código da contagem dos grãos (juntos e separados).

IMAGEM ORIGINAL



CONVOLUÇÃO DA IMAGEM (Borrão do cabeçalho)

```python
import os
from PIL import Image
from scipy import signal
import matplotlib.pyplot as plt
import numpy as np
from skimage.color import rgb2gray
from skimage import io
from scipy.signal import convolve2d
```

```python
def show_convolve2d(imagem, kernel, mask=None):

    # Aplica o filtro de média apenas na região definida pela máscara
    imagem_filt = np.zeros(imagem.shape)
    for i in range(3):
        if mask is not None:
            imagem_filt[:, :, i] = imagem[:, :, i] * (1 - mask)
            imagem_filt[:, :, i] += convolve2d(imagem[:, :, i] * mask, kernel, mode='same', boundary='symm')
        else:
            imagem_filt[:, :, i] = convolve2d(imagem[:, :, i], kernel, mode='same', boundary='symm')

    # Converte a imagem de volta para o tipo uint8 e exibe as imagens
    imagem_filt = imagem_filt.astype(np.uint8)
    imagem = imagem.astype(np.uint8)

    fig, axs = plt.subplots(1, 2, figsize=(8, 6))
    axs[0].imshow(imagem_filt)
    axs[0].set_title('Imagem com filtro de média')
    axs[0].axis('off')
    axs[1].imshow(imagem)
    axs[1].set_title('Imagem original')
    axs[1].axis('off')
    plt.show()

    # Salva a imagem filtrada em um arquivo JPEG
    io.imsave(os.path.join('feijão e milho com filtro.jpeg'), imagem_filt)
```

Definindo o local da máscara e fazendo a convolução com o filtro de média

```python
mask = np.zeros(imagem.shape[:2])
mask[85:160, 75:430] = 1
```

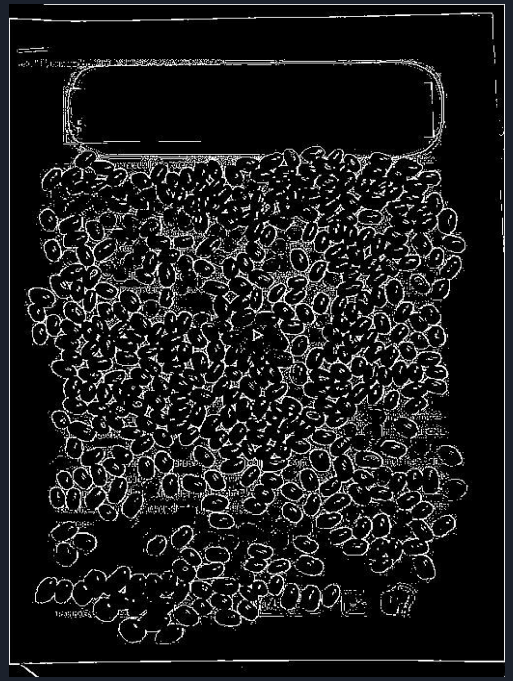
```python
# Define o tamanho do kernel do filtro de média
tam = 50
kernel = np.ones((tam, tam)) / (tam ** 2)

show_convolve2d(imagem, kernel, mask)
```

CABEÇALHO BORRADO

SEPARANDO O FEIJÃO DO MILHO

Aplica o filtro para separar o feijão do milho:

```
# Carrega a imagem
imagem = Image.open('feijão e milho com filtro.jpeg')

# Converte a imagem para escala de cinza
imagem_cinza = imagem.convert('L')

# Converte a imagem em um array numpy
imagem_array = np.array(imagem_cinza)

# Cria um filtro (kernel) que realce os grãos de feijão
filtro_feijao = np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])

# Aplica o filtro à imagem para realçar os grãos de feijão
imagem_realcada_feijao = convolve2d(imagem_array, filtro_feijao, mode='same', boundary='fill', fillvalue=0)


# Cria uma imagem binária (preto e branco) com base na imagem realçada dos grãos de feijão
threshold_feijao = 275
imagem_binaria_feijao = np.where(imagem_realcada_feijao > threshold_feijao, 255, 0).astype('uint8')

# Exibe a imagem filtrada resultante no notebook
plt.imshow(imagem_binaria_feijao, cmap='gray')
plt.title('Feijão separado do Milho')
plt.axis('off')
plt.show()
```



Com o cabeçalho borrado, convoluímos a imagem com um kernel de filtro gaussiano após a convertermos para escala de cinza e em um array, e, depois, escurecemos a imagem usando o threshold de “275” para separarmos o feijão do milho.

Essa imagem nos trouxe alguns problemas quanto à detecção das bordas de alguns feijões, por isso, decidimos realizar um corte na imagem original para que não houvesse problemas de detecção.

CORTE DA IMAGEM

```python
import numpy as np
import matplotlib.pyplot as plt
from skimage import io

def crop_image(imagem, mask=None):
    # Define a região de interesse (ROI) que exclui o cabeçalho
    roi = np.zeros(imagem.shape[:2])
    roi[160:-36, :] = 1
    if mask is not None:
        roi *= mask

    # Encontra os limites da ROI
    rows, cols = np.nonzero(roi)
    top, bottom = rows.min(), rows.max()
    left, right = cols.min(), cols.max()

    # Corta a imagem com base na ROI
    imagem_cortada = imagem[top:bottom+1, left:right+1]

    # Exibe a imagem cortada
    plt.imshow(imagem_cortada)
    plt.show()

    # Salva a imagem cortada
    io.imsave('imagem_cortada.jpeg', imagem_cortada.astype(np.uint8))


    return imagem_cortada

# Carrega a imagem
filename = os.path.join('foto', 'feijão e milho.jpeg')
imagem = io.imread(filename)

# Chama a função crop_image para cortar a imagem
imagem_cortada = crop_image(imagem)

# Exibe a imagem cortada
plt.show()
```

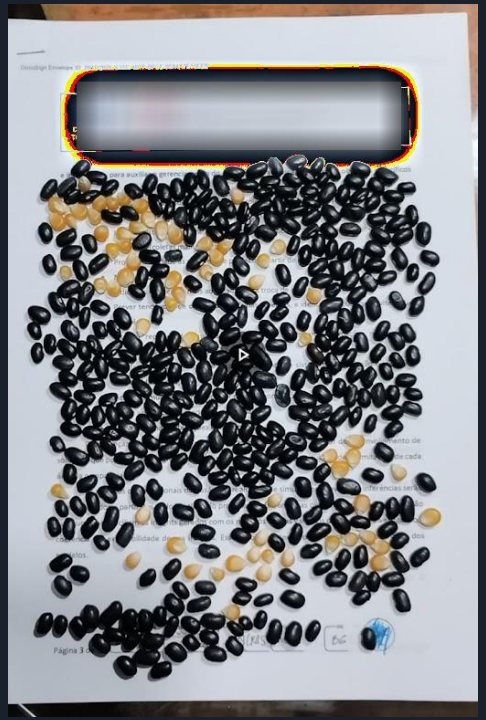
IMAGEM CORTADA

MUDANÇAS PARA O CABEÇALHO:

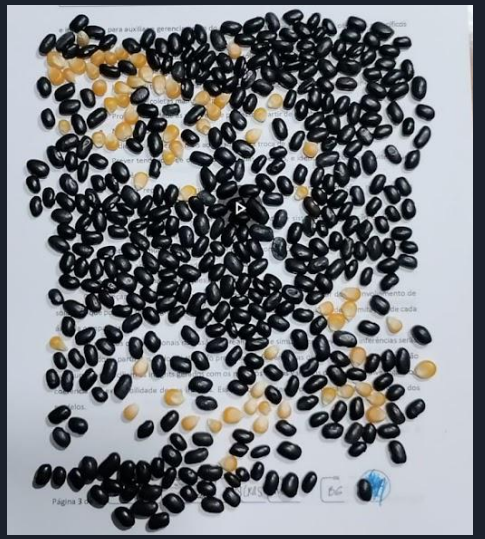
No nosso POC, decidimos aplicar uma máscara no cabeçalho, realizando a convolução com um kernel para borrá-lo. Porém, esse borrão atrapalhou a contagem dos grãos e, por isso, resolvemos rever o que faríamos com a imagem.

Tendo isso em mente, decidimos cortar a imagem, removendo o cabeçalho no corte, de forma que apenas a área com os grãos seja analisada.

Cabeçalho borrado



Cabeçalho cortado



TENTATIVA DE SEPARAÇÃO DOS GRÃOS USANDO TRANSFORMADA DE FOURIER

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt
```

```python
# Carregar a imagem
img = cv2.imread('imagem_cortada.jpeg')
if img is None:
    print('Erro ao carregar a imagem')
    exit()

# Converter para escala de cinza
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Verificar se a imagem em escala de cinza não está vazia
if np.size(gray) == 0:
    print('Erro ao converter a imagem para escala de cinza')
    exit()
```

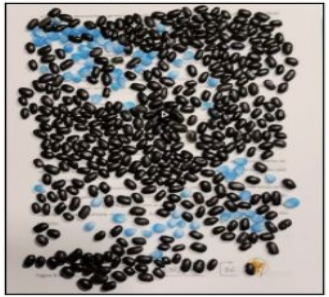
```python
# Aplicar filtro passa-alta
kernel = np.array([[-1,-1,-1],[-1,1,-1],[-1,-1,-1]])
highpass = cv2.filter2D(gray, -1, kernel)

# Aplicar a transformada de Fourier
fft = np.fft.fft2(highpass)
fftshift = np.fft.fftshift(fft)
magnitude_spectrum = np.abs(fftshift)

# Normalizar a magnitude do espectro
magnitude_spectrum = magnitude_spectrum / np.max(magnitude_spectrum) * 255

# Converter para uint8
magnitude_spectrum = magnitude_spectrum.astype(np.uint8)

# Limiarização
thresh = cv2.threshold(magnitude_spectrum, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# Encontrar contornos
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Contar objetos
n_objects = len(contours)

# Segmentação de cores
lower_color = np.array([0,0,0])
upper_color = np.array([50,50,50])
mask = cv2.inRange(img, lower_color, upper_color)
masked = cv2.bitwise_and(img, img, mask=mask)
gray_masked = cv2.cvtColor(masked, cv2.COLOR_BGR2GRAY)
```

TENTATIVA DE SEPARAÇÃO DOS GRÃOS USANDO TRANSFORMADA DE FOURIER

```python
# Limiarização
thresh = cv2.threshold(gray_masked, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# Encontrar contornos
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Contar feijões
n_feijoes = len(contours)

plt.subplot(1, 2, 1)
plt.imshow(img)
plt.title('Imagem original com contornos')
plt.xticks([]), plt.yticks([])

print('Número de grãos:', n_objects)
print('Número de feijões:', n_feijoes)
```

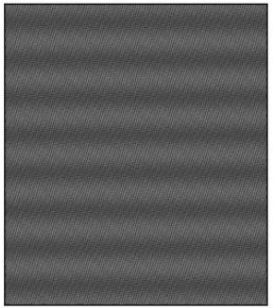
```
Número de grãos: 6238
Número de feijões: 3268
```

Imagem original com contornos



```python
# Normaliza a imagem para 0-255
normalized_spectrum = cv2.normalize(magnitude_spectrum, None, 0, 255, cv2.NORM_MINMAX)

# Plota as imagens

plt.subplot(1, 2, 2)
plt.imshow(normalized_spectrum, cmap='gray')
plt.title('Espectro')
plt.xticks([]), plt.yticks([])

plt.show()
```

Espectro



Pode-se observar, no código, que o número de grãos ficou muito distante do que deveria, mesmo com o ajuste dos parâmetros dos filtros. Portanto, decidimos mudar nossa abordagem, sem o uso da Transformada de Fourier.

CONTANDO OS GRÃOS DA IMAGEM

Como a implementação por série de Fourier não funcionou, aplicamos uma abordagem com a utilização de filtros para buscar os contornos, tanto do milho, quanto do feijão. Para isso, convertemos a imagem em uma escala de cinza e a suavizamos. Depois, aplicamos um filtro binário adaptativo.

Como a imagem tem muitos detalhes, o programa pode contornar o que não é grão. Para resolver esse problema, definimos um limite mínimo e máximo de área de contorno, onde tudo que estiver dentro desse limite de área é considerado grão e contabilizado.

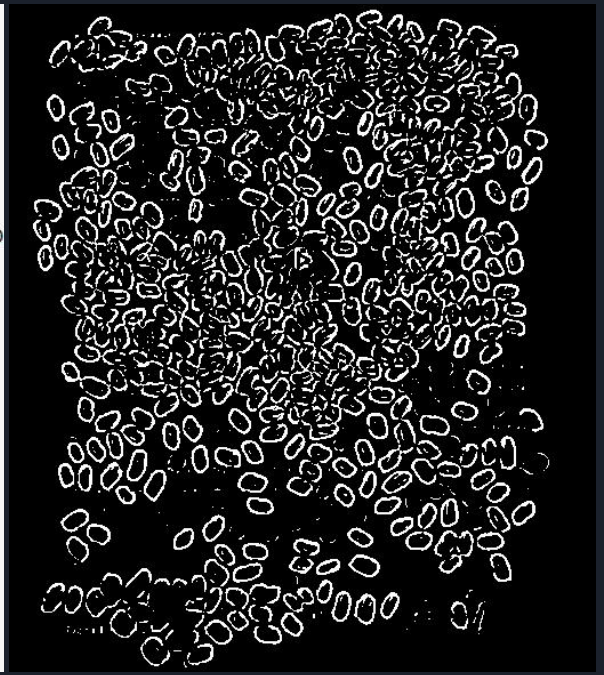
Conseguimos contabilizar 460 grãos de 461 possíveis, gerando uma taxa de acerto de 99,78%.

CONTANDO OS GRÃOS DA IMAGEM

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```python
# carregar a imagem
img = cv2.imread('imagem_cortada.jpeg')

# converter para escala de cinza
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# aplicar filtro gaussiano para suavizar a imagem
blur = cv2.GaussianBlur(gray, (5,5), 0.9)

# aplicar filtro binário adaptativo
thresh = cv2.adaptiveThreshold(blur, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV,5,14)

# encontrar contornos
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# contar os grãos
total_graos = 0
for cnt in contours:
    area = cv2.contourArea(cnt)
    if area > 10 and area < 1000:
        total_graos += 1

plt.imshow(thresh, cmap='gray')
plt.title("Imagem binarizada")
plt.show()
# imprimir o total de grãos
print("Total de grãos na imagem: ", total_graos)

#salva a imagem binarizada
cv2.imwrite("Imagem binarizada.jpeg", thresh)
```

Imagem binarizada



```
Total de grãos na imagem:  460
```

CONTANDO OS GRÃOS SEPARADAMENTE

Tendo os grãos contados em sua totalidade, realizamos, também, a contagem separada de cada um dos grãos. Para isso, delimitamos um limite de área para definir o que é grão e o que não é, assim como na contagem geral.

Posteriormente, fizemos uma média dos pixels da região do contorno:

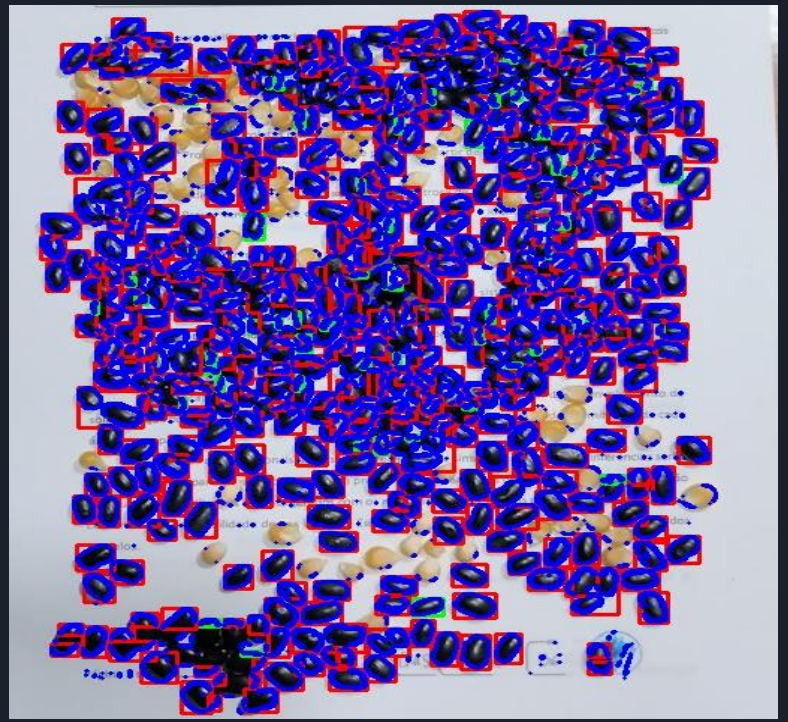
- Se a média desses pixels for maior que 111, consideramos um grão de milho;
- Se for menor, consideramos um grão de feijão.

CONTANDO OS GRÃOS SEPARADAMENTE

```python
import matplotlib.pyplot as plt

# plotar imagem com contornos
img_contornos = cv2.drawContours(img, contours, -1, (255, 0, 0), 2)
plt.imshow(cv2.cvtColor(img_contornos, cv2.COLOR_BGR2RGB))
plt.title("Imagem com contornos")
cv2.imwrite("imagem_contornos.jpeg", img)
plt.show()
```

```python
# contar grãos de milho e de feijão preto
milho = 0'
feijao = 0
for cnt in contours:
    area = cv2.contourArea(cnt)
    if area > 10 and area < 1000:
        x, y, w, h = cv2.boundingRect(cnt)
        roi = thresh[y:y+h, x:x+w]
        mean_val = cv2.mean(roi)
        if mean_val[0] >111: # se a média for maior que 111, é um grão de milho
            milho += 1
            cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
        else: # caso contrário, é um grão de feijão preto
            feijao += 1
            cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 2)

# imprimir o total de grãos de milho e de feijão preto
print("Total de grãos de milho: ", milho)
print("Total de grãos de feijão preto: ", feijao)
```

```
Total de grãos de milho:  64
Total de grãos de feijão preto:  396
```



Porcentagem de acerto na contagem do feijão: 99,75%

Porcentagem de acerto na contagem do milho: 100%