Mínimo Produto Viável

Sinais e Sistemas 2022.2

Nosso time

Guilherme Pontes

Claudio Roberto

Nathália Bacalhau

Conteúdo

01.

Problema vs. objetivo

Dificuldades encontradas em contraste com o resultado esperado

02.

Estratégia

Funções e abordagens escolhidas para resolução do problema

03.

Resultados

Resultados seguindo as estratégias

04.

Evoluções futuras

Pontos que precisam de aprimoramento

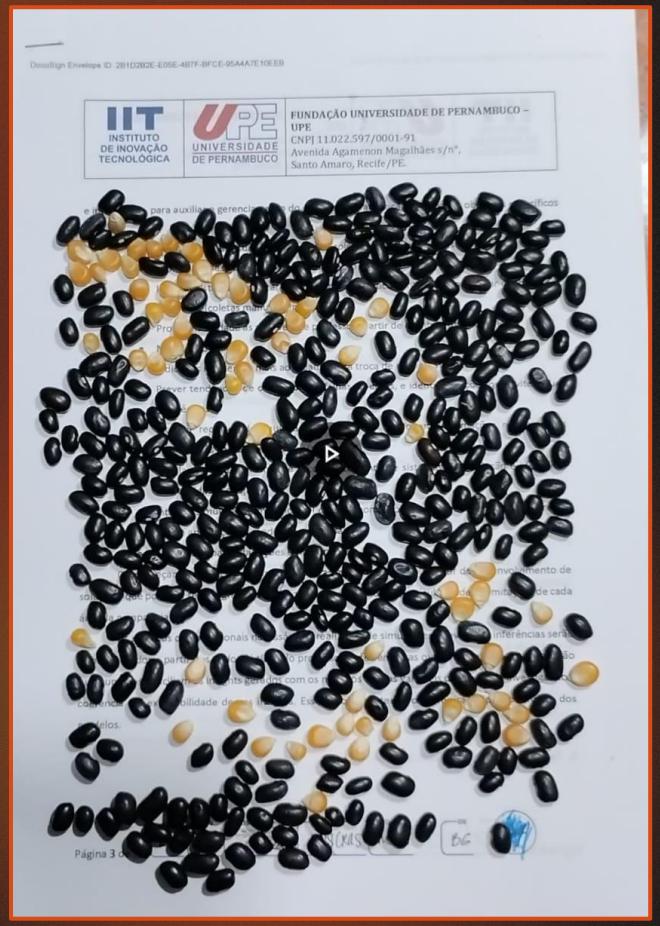
05.

Código Utilizado

06.

Transformada de Fourier

01.
Problema vs. objetivo



Problema

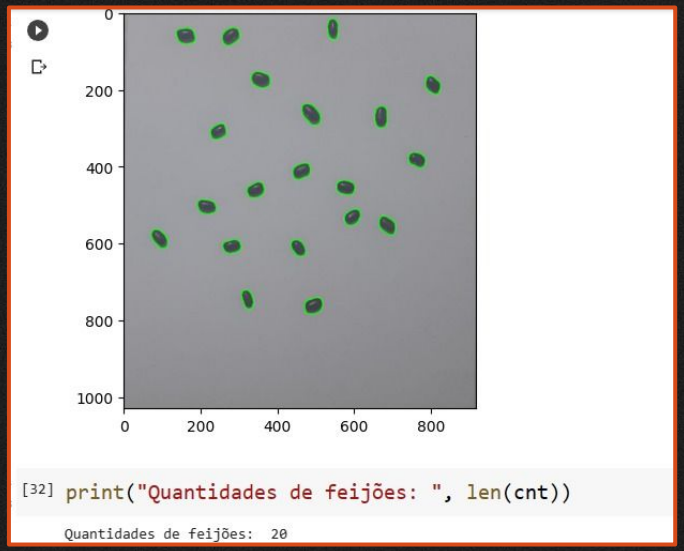
- Amontoado aleatório de feijões e milhos
- Cabeçalho, texto e imagens na página

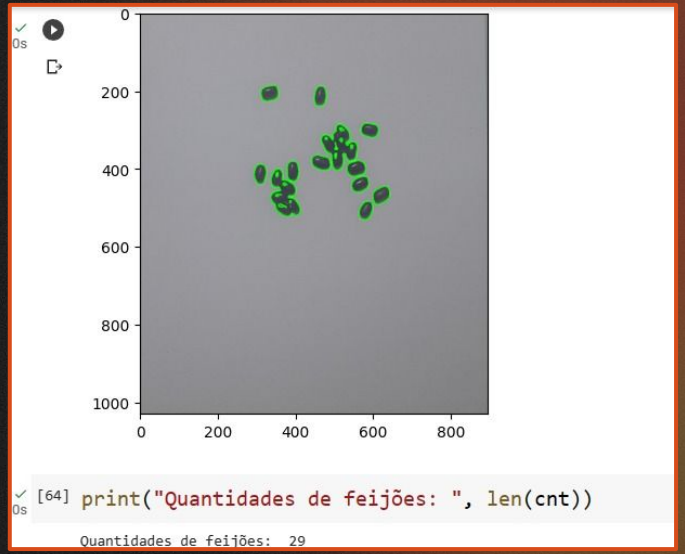
Objetivo

- Separação de cada grupo de grão
- Cálculo do valor total de grãos
- Quantificação dos feijões
- Contagem dos milhos

02.
Estratégia

Treinamento





- Uso de menor quantidade de feijão
- Na formatação junta e separada

Cabeçalho

- Corte de imagem
  - Definindo bordas
  - Eliminado as áreas que estejam fora da borda

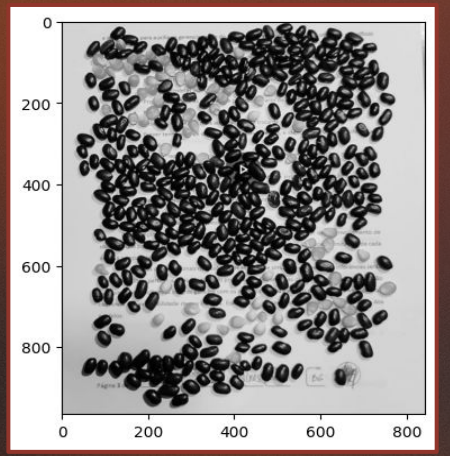
Amontoado

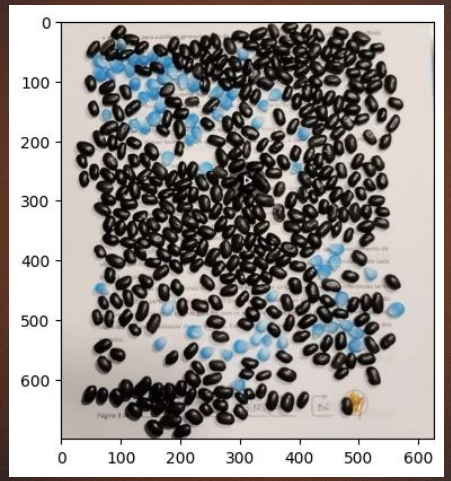
Para os Feijões:

- Aplicação de filtro de cor cinza
- Gaussiana para borrar a imagem
- Realçador de contorno

Para os Milhos:

- Conversão da imagem para escala HSV
- Aplicação de uma máscara para identificar objetos com as cores amarelas
- Realçador de contorno





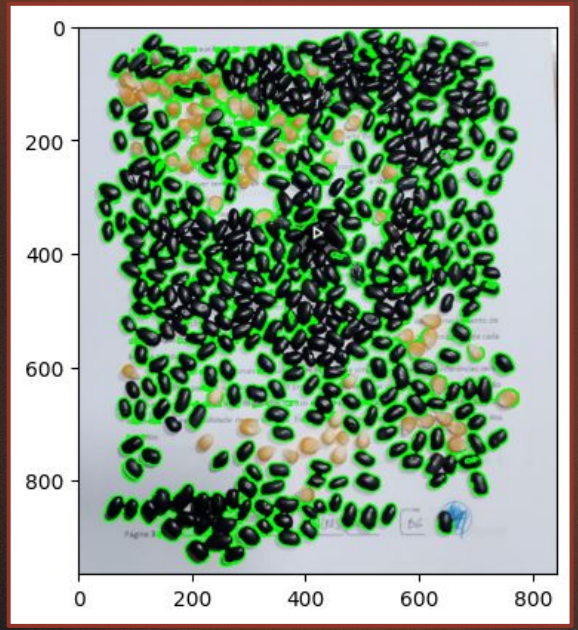
03.
Resultados

Quantidade de Total de Grãos

- Para esse objetivo foi pensado o caminho de dividir para conquistar ou seja seria encontrado a quantidade de feijões e de milho de forma separada e a quantidade de grãos seria então a soma deles

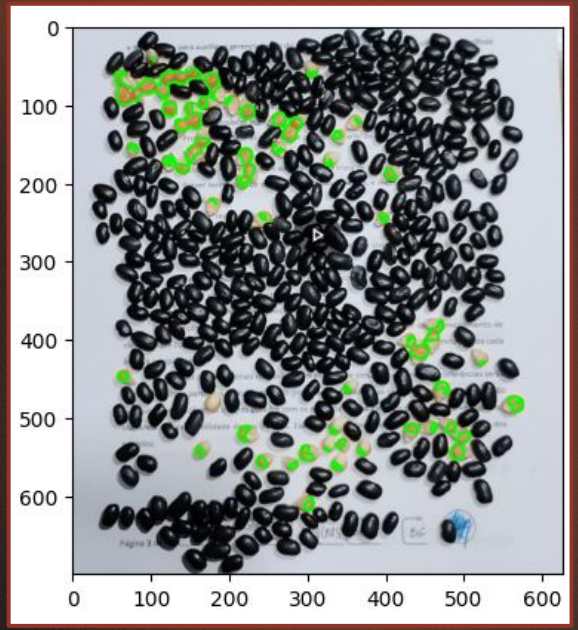
Quantidade de Feijões

- Eliminação de contornos que possui um tamanho inferior ao mínimo esperado



Quantidade de Milhos

- Aplicação de uma máscara para captar apenas a cor amarela e desta forma quantificar os milhos



Valores

63	397	433
Milhos	Feijões	Total
98% de acerto	100% de acerto	99,7% de acerto

04.
Evoluções Futuras

Pensando além do Problema

Utilizando fotos tiradas por nossa Startup(Grupo) encontramos alguns problemas nas imagens que um trabalhador qualquer ao tirar e enviar uma foto

05.

Código Utilizado

Código

https://colab.research.google.com/drive/1DTCXISIWIw9stn-rML4mlVOaeoVt7Rys?usp=sharing
https://colab.research.google.com/drive/1it6fweEqEhVThmuacTMkMBeIH_BZx3jq?usp=sharing

Bibliotecas Utilizadas

```
import matplotlib.pyplot as plt
import numpy as np
import cv2
import scipy.fftpack as fp
import imageio
from PIL import Image,ImageOps
```

Código do Cabeçalho

```
img = Image.open('/content/feijao.jpg')
border = (0, 265, 0, 50)
cropped_img = ImageOps.crop(img, border)
cropped_img.save('output.jpg')
```

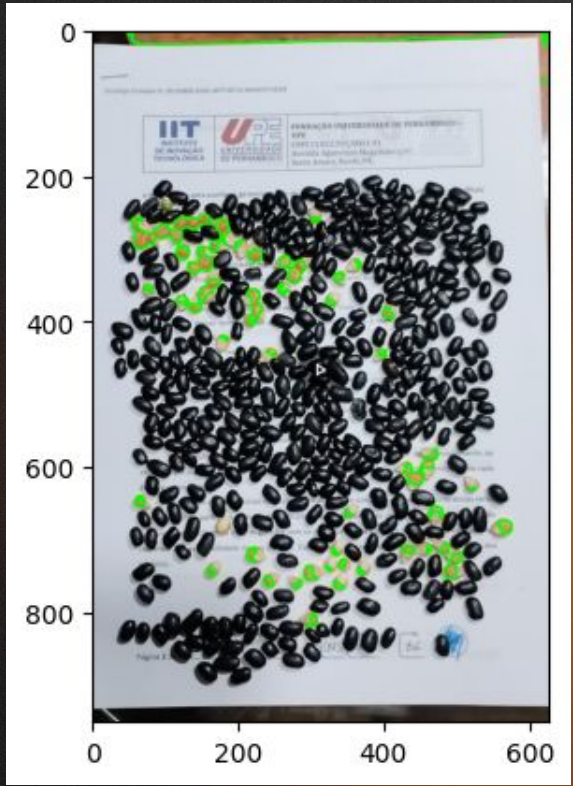
Observação do Cabeçalho

Apesar da retirada do cabeçalho o código conseguiria contar normalmente sem a sua retirada, apenas com uma taxa de erro de aproximadamente 0,5%

Observação do Cabeçalho

Para os Milhos:

Na contagem de grãos de milhos não existe diferença como pode ser observado e o valor encontrado continua sendo 63



Observação do Cabeçalho

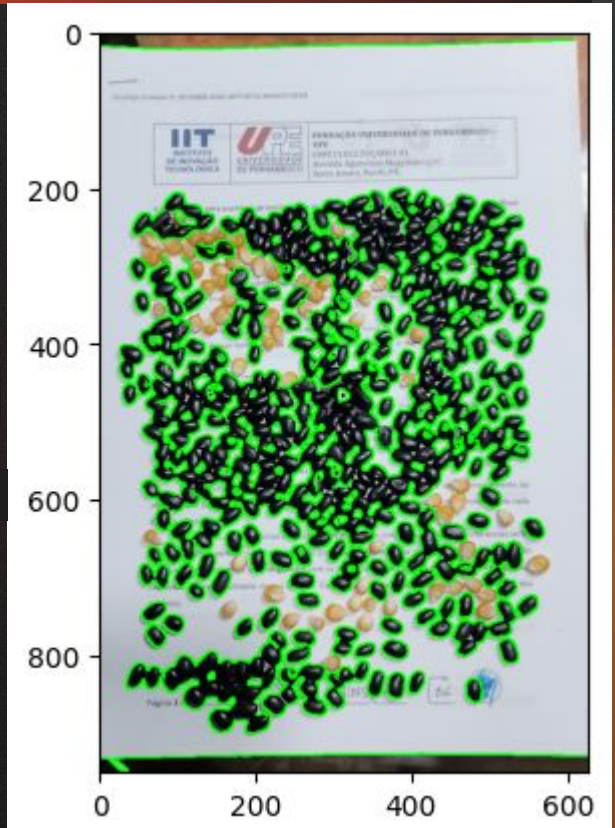
Para os Feijões:

Na contagem de feijões o código vai possuir uma diferença no valor do canny e desta forma uma taxa de acerto menor mas ainda perto do valor desejado

Sem Cabeçalho: `canny = cv2.Canny(blur, 45, 190, 1)`

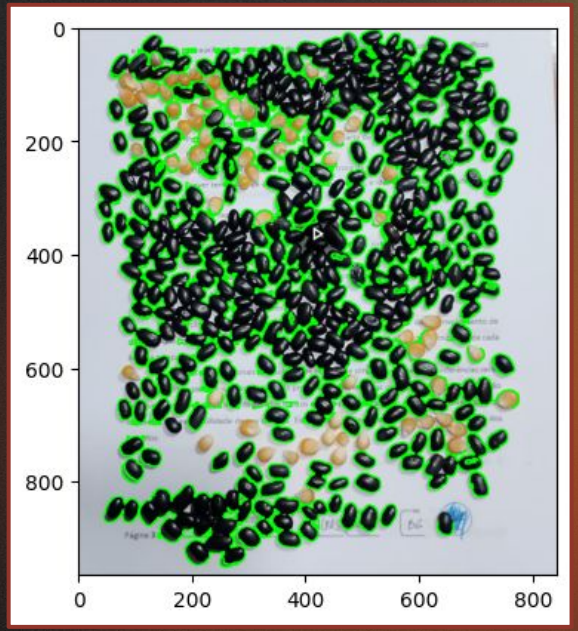
Com Cabeçalho: `canny = cv2.Canny(blur, 45, 195, 1)`

Qtd de Feijões sem cabeçalho: 397
Qtd de Feijões com cabeçalho: 395



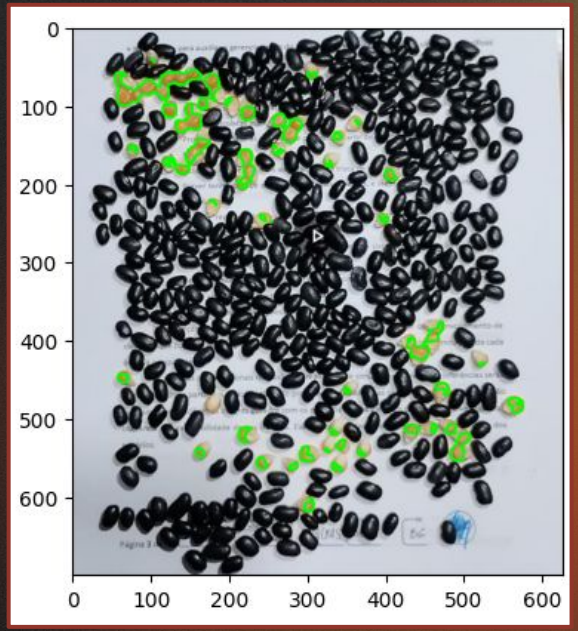
Código de Contar Feijões

```
image = cv2.imread('/content/output.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (15, 15), 0)
canny = cv2.Canny(blur, 45, 190, 1)
dilated = cv2.dilate(canny, (1, 1), iterations=0)
(cnt, hierarchy) =
cv2.findContours(dilated.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
cv2.drawContours(rgb, cnt, -1, (0, 255, 0), 2)
print("Quantidade de feijões: ", len(cnt))
```



Código de Contar Milhos

```
amarelo = [(4, 99, 128), (21, 255, 255)]
image_HSV = cv2.imread('/content/output.jpg')
hsv_img=cv2.cvtColor(image_HSV,cv2.COLOR_BGR2HSV)
mask_amarela = cv2.inRange(hsv_img,
np.array(amarelo[0]), np.array(amarelo[1]))
(cnt_amarelo,hierarchy)=cv2.findContours(mask_amarela
.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
rgb2 = cv2.cvtColor(image_HSV, cv2.COLOR_BGR2RGB)
cv2.drawContours(rgb2, cnt_amarelo, -1, (0, 255, 0),
2)
print("Quantidade de milhos: ", len(cnt_amarelo))
```



Código do Cálculo do valor total de grãos

```
print("Quantidade de graos sao:",len(cnt + cnt_amarelo))
```

05.

Transformada de Fourier

Transformada de Fourier

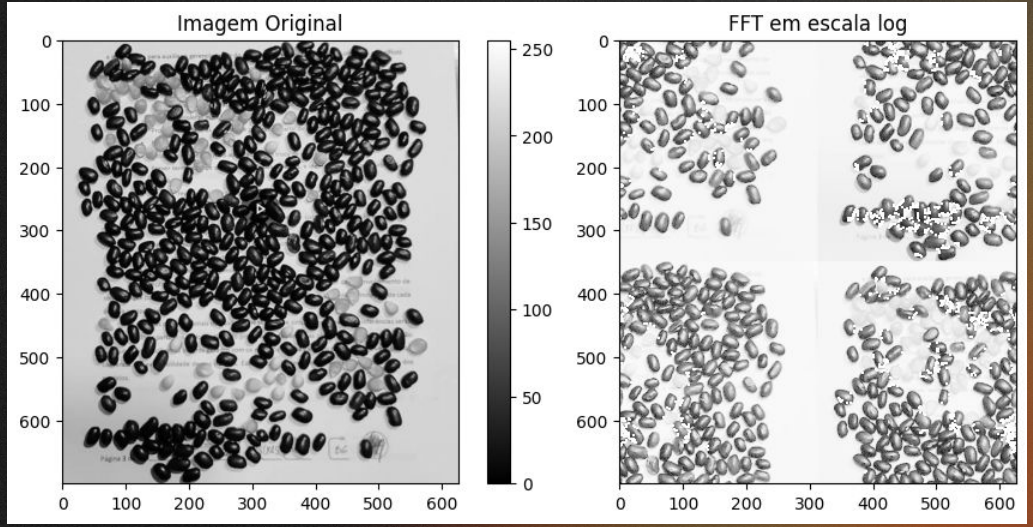
Pesquisando na internet para encontrar maneiras que pudessem aplicar a transformada de fourier foi encontrada uma que se aplica o filtro de gaussiana através da transformada onde basicamente definimos uma função gaussiana

S é a imagem
sigma a ser utilizado na gaussiana

```
def gaussian(S,sigma = 1, vmax =.05):

  Nu, Nv = S.shape
  u = Nu * np.linspace(-vmax,vmax,Nu)
  v = Nv * np.linspace(-vmax,vmax,Nv)
  U, V = np.meshgrid(v, u)

  sigma2 = sigma ** 2
  G = np.exp((U*U + V*V) / 2. /
sigma2)
  G = fp.fftshift(G)

  return G / sigma2
```

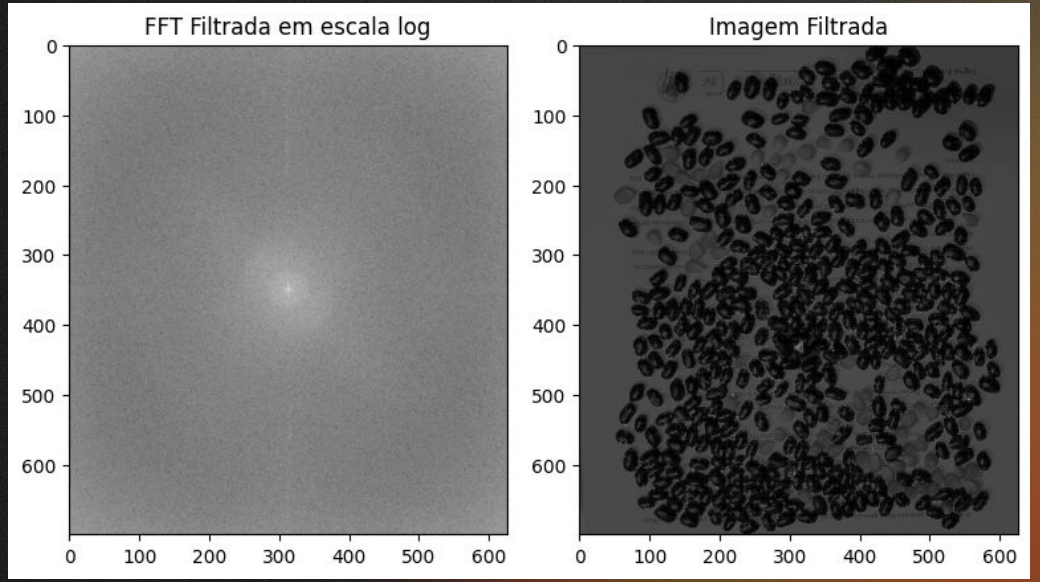
Transformada de Fourier

- Recebemos uma imagem e a convertemos para cor cinza
- Aplicamos a transformada e convertemos para escala log para ser possível a visualização
- Definimos um valor do sigma a ser utilizado
- Aplicamos a no função gaussiana
- Depois aplicamos o filtro através da multiplicação da transformada e da função gaussiana



Transformada de Fourier

- Então temos a transformada com o filtro aplicado em escala log
- E a imagem da transformada inversa que assim mostra a imagem com o filtro



Transformada de Fourier

```
f = imageio.imread('/content/output.jpg',
pilmode='L')
plt.figure()
cmap = 'gray'
plt.title('Imagem Original')
plt.imshow(f,cmap = cmap)
plt.colorbar()
```

```
TF = fp.fft2(f) -> transformada de fourier
```

```
Fm = np.absolute(f)
Fm =Fm/Fm.max()
Fm = fp.fftshift(Fm)
Fm = np.log(Fm)
```

```
plt.figure()
plt.title('FFT em escala log')
plt.imshow(Fm,cmap = cmap)
```

```
sigma = 25
G = gaussian(f,sigma) -> aplicando a função
```

```
Fg = TF * G -> aplicação do filtro
```

Transformada de Fourier

```
plt.figure()
plt.title('FFT Filtrada em escala log')

Fga =  np.absolute(Fg)
Fga = fp.fftshift(Fga)
Fga = np.log(Fga+1e-6)
plt.imshow(Fga,cmap = cmap)
```

Código da FFT Filtrada em escala de log

```
f_blurred = fp.fft2(Fg)
f_blurred = np.absolute(f_blurred)

plt.figure()
plt.title('Imagem Filtrada')
plt.imshow(f_blurred,cmap = cmap)
```

Código da transformada inversa e assim a imagem filtrada

Considerações

Quando pesquisamos e tentamos aplicar a transformada, percebemos que os resultados encontrados eram muito abaixo do esperado, seja por não saber aplicar de forma correta, ou pela não aplicabilidade para nosso caso em específico de contar grãos.

Notamos que existem outros caminhos com menor curva de aprendizado, como o filtro de gaussiana, e que poderiam ser aplicados à transformada, porém a questão da luminosidade não ficou bem resolvida, em relação a fotografias em outros tipos de ambiente.

Diante disso, escolhemos não utilizar a Transformada de Fourier a fim de apontar o resultado com maior precisão.

Obrigado pela atenção!