DCA 0445 Lista de Exercícios 1 1

Mateus de Assis Silva (mateus.d.assis.silva@gmail.com)

October 28, 2020

Question 1.

Note como a imagem g(x,y) é composta por vários fenômenos que ocorrem em adição: a cena que está sendo capturada r(x,y) e os fenômenos físicos que a acompanham (iluminação i(x,y) e o ganho do sensor h(x,y)); um ruído de característica aleatória n(x,y); e um erro "natural" e determinístico gerado pela implementação física do sensor p(x,y).

Vale saber que a polarização adicionada pelo sensor é naturalmente baixa, visto que as imagens geralmente capturadas em dispositivos eletroeletrônicos não possuem tons distorcidos. Por exemplo, se a polarização na banda do vermelho fosse alta, todas as fotos registradas com o referido sensor teriam um tom vermelho forte, independente do que estivesse sendo fotografado.

A reflectância é uma característica dos objetos em cena, sendo sua capacidade de refletir a luz incidida. A iluminação, por sua vez, pode ser controlada via lâmpadas e refletores adequados (como aqueles que encontramos em estúdios de fotografia).

Por fim, o ruído aditivo representa um elemento indesejado de natureza aleatória. Sua interação pode ser reduzida a partir de filtros adequados. Por exemplo, um ruído impulsivo do tipo sal-e-pimenta pode ser tratado com filtro espacial da mediana, enquanto que um ruído de natureza gaussiana pode ser corrigido com filtro espacial da média.

Question 2.

A seguinte resolução envolve códigos desenvolvidos em Python. Supondo a importação das bibliotecas de programação adequadas, salvemos na memória as matrizes a serem convoluídas:

```
imagem_2 = np.zeros((9,9))
for i in range(0,9):
  for j in range(0,9):
    if i in range(2,7) and j in range(2,7):
      imagem_2[i,j]=1

imagem_2
```

. O que leva ao seguinte resultado:

```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 1., 1., 1., 1., 1., 0., 0.],
       [0., 0., 1., 1., 1., 1., 1., 0., 0.],
       [0., 0., 1., 1., 1., 1., 1., 0., 0.],
       [0., 0., 1., 1., 1., 1., 1., 0., 0.],
       [0., 0., 1., 1., 1., 1., 1., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

e

```
mascara_2 = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])
mascara_2
```

o qual gera

```
array([[-1,  0,  1],
       [-1,  0,  1],
       [-1,  0,  1]])
```

E, ao convoluir usando

```
signal.convolve2d(imagem_2,mascara_2)
```

temos

```
array([[ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0., -1., -1.,  0.,  0.,  0.,  1.,  1.,  0.,  0.],
       [ 0.,  0., -2., -2.,  0.,  0.,  0.,  2.,  2.,  0.,  0.],
       [ 0.,  0., -3., -3.,  0.,  0.,  0.,  3.,  3.,  0.,  0.],
       [ 0.,  0., -3., -3.,  0.,  0.,  0.,  3.,  3.,  0.,  0.],
       [ 0.,  0., -3., -3.,  0.,  0.,  0.,  3.,  3.,  0.,  0.],
       [ 0.,  0., -2., -2.,  0.,  0.,  0.,  2.,  2.,  0.,  0.],
       [ 0.,  0., -1., -1.,  0.,  0.,  0.,  1.,  1.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.]])
```

Note os números negativos na metade esquerda da matriz. Isso ocorre porque no momento da convolução a matriz é invertida em ambos eixos. Isto é, a matriz "máscara" resultante é a seguinte:

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Question 3.

Como premissa inicial considere que a imagem digital em questão se encontra em tons de cinza (imagem cinzenta, não colorida). Assuma também que o fundo é apresentado com tom claro (branco) e os objetos que compõem a figura são apresentados no tom preto.

(a) De forma a remover os objetos que tocam a margem da figura, executa-se uma varredura sobre os pixels que compõem perímetro da imagem. Caso algum pixel encontrado tenha tom escuro, executa-se o algoritmo *floodfill* com semente igual ao valor do maior tom de brilho. Isso fará com que os objetos que tocam a margem sejam mesclados ao fundo da cena. Essa imagem de saída será denominada *imagem_sem_borda*

(b) De forma a realizar a contagem dos objetos que se encontram na figura, pode-se executar uma varredura linha-a-linha, partindo do topo esquerdo da imagem até se atingir o canto inferior direito. Também deve-se iniciar uma variável que chamaremos *contador_de_objetos*. Para cada pixel varrido, compara-se seu valor

com o menor tom possível. Caso o retorno dessa comparação seja verdadeiro, incrementa-se o valor da variável *contador_de_objetos* e realiza-se o algoritmo *floodfill* com semente igual ao valor dessa variável. Isso permitirá a contagem de objetos até o maior valor de tom cinza menos um. A imagem resultado será denominada *imagem_contada*.

(c) Como medida de área de objeto, podemos contar quantos pixels compõem o mencionado objeto. Para tanto, define-se um vetor denominado *vetor_de_areas* com tantas posições quanto objetos encontrados anteriormente. Varre-se a imagem linha a linha, partindo-se do topo esquerdo até alcançar-se o canto inferior direito. Checa-se o valor de cada pixel, e dependendo de tal quantidade, incrementa-se o respectivo valor no *vetor_de_areas* (note que ignora-se valores que correspondem ao fundo ou a 0). Ao fim do processamento, tem-se as áreas de cada objeto.

(d) Para podermos diferenciar um círculo de um quadrado de um triângulo, será necessário, em primeiro lugar, aferir um valor de perímetro para os objetos presentes na figura. Para podermos realizar a medição de tais valores, basta iterar sobre a imagem, mais uma vez, e, ao encontrar um pixel de uma figura, percorrer o trajeto ao redor da imagem, contando quantos pixels que pertencem a imagem tocam o fundo.

Concluída tal etapa, temos, para cada objeto encontrado,suas áreas e perímetros. Assim, basta computar a razão entre o quadrado do perímetro e a área. Note que, para um quadrado, esse número equivale a 16, enquanto que para um círculo , esse valor se aproxima de 12.57 (4 vezes o valor de pi). Caso não se aproxime de nenhum desses valores, atribui-se ao objeto o rótulo "triângulo".

(e) Para identificar as posições dos objetos, itera-se sobre a imagem, partindo do topo esquerdo até se atingir o canto inferior direito. Atingindo-se algum pixel que compõe um objeto, registra-se aquela posição (que vem a ser a posição do objeto) e aplica-se o algoritmo *floodfill* com semente igual ao valor do maior tom de brilho (mesclando, assim, o objeto ao fundo da imagem). Ao final teremos, como posição dos objetos o pixel do canto superior esquerdo da figura.

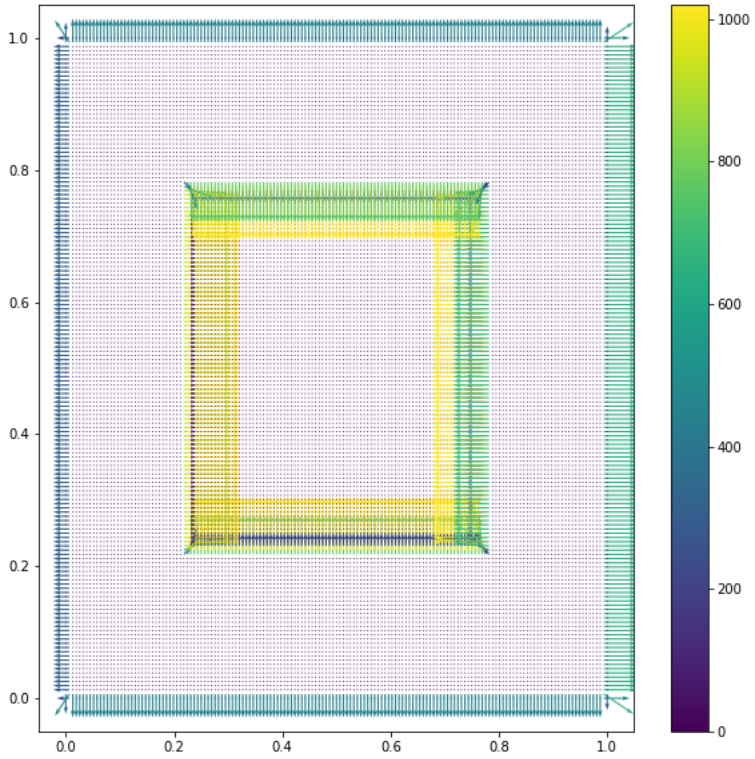
Question 4.

Utilizando-se da linguagem Python, importa-se a imagem que foi recortada da atividade. A saber:

Aplica-se os filtros de Sobel em X e Y, o que aproxima os valores do campo vetorial nessas direções. Também gera-se os valores de domínio. No caso, a figura recortada possui tamanho de 157 pixels por 157 pixels, o qual será o domínio no qual o campo vetorial está definido.

```
sobel_y = cv2.Sobel(imagem_q4,cv2.CV_64F,1,0,ksize=3)
sobel_x = cv2.Sobel(imagem_q4,cv2.CV_64F,0,1,ksize=3)
X,Y = np.meshgrid(np.linspace(0,1,157),np.linspace(0,1,157))
```

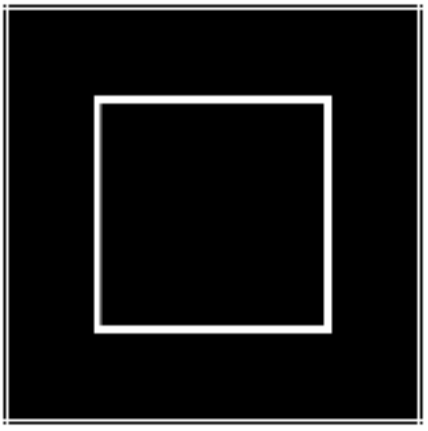
Por fim, exibe-se a figura do campo vetorial:



Nota-se como, na região em que o tom varia de preto pra branco, as setas apontam para o centro da área esbranquiçada. Isso ocorre pois o campo vetorial indica o caminho em que a função original tem o valor mais incrementado possível.

Para vermos o campo magnitude do gradiente, podemos exibir a soma elemento-a-elemento dos valores absolutos das matrizes resultantes da aplicação dos operadores de sobel.

```
cv2_imshow(np.abs(sobel_x)+np.abs(sobel_y))
```



Question 5.

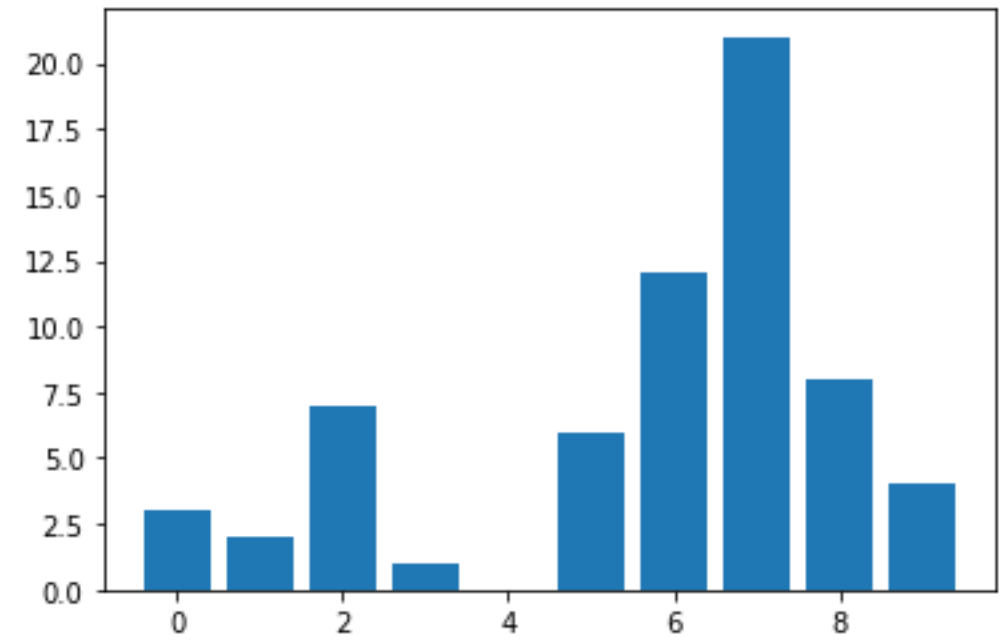
A primeira etapa é definir a matriz usando numpy:

```
matriz = np.array([[0,5,7,7,5,8,7,8],
                   [7,2,6,2,6,5,6,8],
                   [6,9,7,7,0,7,2,7],
                   [6,6,1,7,6,7,7,5],
                   [9,6,0,7,8,2,6,7],
                   [2,8,8,2,7,6,7,8],
                   [7,3,2,6,1,7,5,8],
                   [9,9,5,6,7,7,7,7]])
```

Podemos exibir o histograma utilizando uma função predefinida em Python

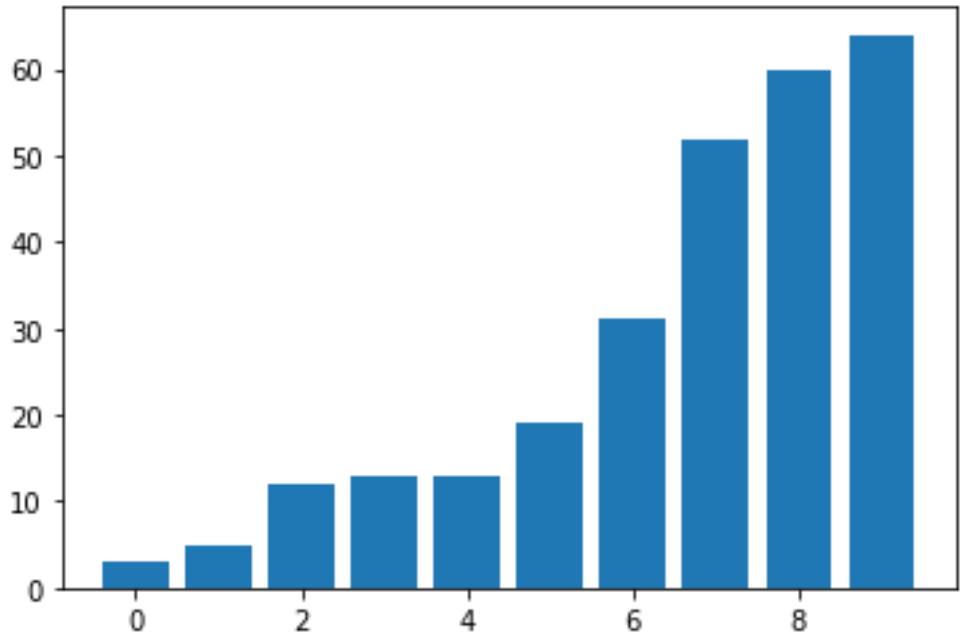
```
histogram_vector,bin_edges = np.histogram(matriz)
plt.bar(np.arange(0,10),histogram_vector)
```

```
<BarContainer object of 10 artists>
```



Pode-se calcular o histograma acumulado se utilizando de:

```
cumulative_histogram = np.cumsum(histogram_vector)
plt.bar(np.arange(0,10),cumulative_histogram)
```



Eu defini uma função de mapeamento para levar a matriz original a uma equalizada

```
def mapeamento_cum_hist(matriz_recebida,histograma_acumulado):
  linhas,colunas = np.shape(matriz_recebida)
  matriz_cum_hist = np.zeros((linhas,colunas))
  for i in range(0,linhas):
    for j in range(0,colunas):
      matriz_cum_hist[i,j]=histograma_acumulado[matriz_recebida[i,j]]
  return np.amax(matriz_recebida)*np.round(matriz_cum_hist/(linhas*colunas),1)
```

```
mapeamento_cum_hist(matriz, cumulative_histogram)
```

```
array([[0. , 2.7, 7.2, 7.2, 2.7, 8.1, 7.2, 8.1],
       [7.2, 1.8, 4.5, 1.8, 4.5, 2.7, 4.5, 8.1],
       [4.5, 9. , 7.2, 7.2, 0. , 7.2, 1.8, 7.2],
       [4.5, 4.5, 0.9, 7.2, 4.5, 7.2, 7.2, 2.7],
       [9. , 4.5, 0. , 7.2, 8.1, 1.8, 4.5, 7.2],
       [1.8, 8.1, 8.1, 1.8, 7.2, 4.5, 7.2, 8.1],
       [7.2, 1.8, 1.8, 4.5, 0.9, 7.2, 2.7, 8.1],
       [9. , 9. , 2.7, 4.5, 7.2, 7.2, 7.2, 7.2]])
```

Vale saber que a equalização de histogramas é adequada para facilitar a compreensão de imagens com brilho saturado (muito escuras ou muito claras). Caso esteja se tratando desse caso, o resultado será bastante útil; caso não seja esse o problema, o processamento pode ser inútil.

Question 6.

a. A cor é percebida pelo sistema visual humano através da excitação de células especializadas conheci-

das como cones. Existe três tipos de cones, cada um responsável por responder melhor aos estímulos das frequências luminosas que interpretamos como verde, vermelho e azul.

b. Partindo da premissa de que o que entendemos como cor é a composição de três estímulos, se define o conceito da teoria dos tristímulos, em que as cores são a composição de diferentes intensidades de verde, vermelho e azul.

c. Dado que a teoria dos tristímulos define o conceito de composição de cores, a captura de imagens e a posterior exibição em telas é feita através da adição de luz, enquanto se imprime adicionando pigmentos (o que causará a subtração da capacidade reflexiva por parte da tela a ser tingida). Ao primeiro método é dado o nome "sistema aditivo", enquanto que o último é denominado "sistema subtrativo".

d. Considera-se como atributos da cor as seguintes entidades: radiância (energia emitida pela fonte de luz); luminância (energia percebida pelo observador); e brilho (noção acromática de intensidade da luz)

Question 7.

Suponha que a faixa é dada pelos seus limites máximo e mínimo.

Aplica-se os operadores de sobel em ambos os eixos e têm-se esses valores como as coordenadas dos vetores do campo gradiente.

Calcula-se o ângulo do vetor gradiente em cada ponto da imagem. Caso esse ângulo se encontre dentro da faixa definida pelo usuário, o ponto em questão participa da reta.

Ora, para se construir as retas, basta seguir o seguinte protocolo ao encontrar um ponto válido (dentro da margem definida): se mover na direção da linha (note que a equação da linha é y=mx+b, onde m é a tangente do ângulo definido e b é a altura do pixel),em ambos os sentidos, e procurar pontos com ângulo válido.

Question 8.

Tal sistema de captura invariavelmente necessitará de dois processamentos principais: suavização de ruído a partir de média de imagens e aplicação de pseudo-cor para realçar detalhes desejados.

Em primeiro lugar, sana-se o ruído: captura-se uma quantidade de imagens e realiza-se a média pixel-a-pixel. O número de matrizes utilizadas para atenuar o ruído deve ser determinado empiricamente. Vale saber que tal processo só seria efetivo se o ruído seguir uma distribuição gaussiana. Caso seja notado que o ruído possui outra distribuição, deve-se utilizar outra medida estatística: mediana, por exemplo.

Por fim, escolhe-se um mapeamento de pseudo-cor empiricamente, baseado nas características que se deseja realçar.

Question 9.

Em primeiro lugar, remove-se as bolhas que estão tocando a bordada imagem ao varrer o perímetro da imagem e aplicar o algoritmo floodfill com semente igual a itensidade do tom de cinza do fundo da cena.

Para concluir o processo que irá produzir a imagem que só terá bolhas isoladas, devemos estimar a área de uma bolha isolada. Pode-se recortar um trecho retangular da imagem que possui uma bolha isolada e executar uma estimativa de área baseada na quantidade de pixels que compõem a figura, para realizar-se tal estimativa.

Por fim, itera-se a imagem cujas bolhas marginais já foram removidas, partindo do topo esquerdo até atingir o canto inferior direito. Caso se encontre um pixel que compõe um objeto, mede-se a área da figura em

foreground, e caso ela não seja igual a estimativa da área da bolha isolada, aplica-se floodfill cuja semente é o valor do tom de cinza que compõe o fundo.

Ao fim desse processo haverá a imagem cujas bolhas estão isoladas e não tocam a margem.

Question 10.

(a) Ao se zerar os bits menos significativos, todas as colunas do histograma relativas a valores ímpares seriam zeradas, e os valores que lá foram registradas seriam adicionadas nas colunas imediatamente a esquerda (isto é, seriam adicionadas aos valores pares imediatamente menores).

(b) Ao se zerar os bits mais significativos, as colunas do histograma relativas à metade maior de intensidades (itensidade 2 elevado a 4 até itensidade 2 elevado a 7) seriam zeradas e seus valores seriam redistribuídos na metade menor de itensidades (itensidade 2 elevado a 0 até itensidade 2 elevado a 3). Por exemplo, se houveram, inicialmente, 11 pixels com itensidade 10000001 (itensidade 129) e 7 pixels com itensidade 00000001 (itensidade 1), haverão, ao fim do procedimento, 0 pixels com itensidade 129 e 11+7=18 pixels com itensidade 01.

Question 11.

1. Máximo Acima de Limiar 2. Detector de Bordas de Sobel 3. Filtro de Aguçamento 4. Filtro da Mediana 5. Equalização 6. Filtro da Média 7. Detector de Bordas de Sobel 8. Transformação de Brilho

Question 12.

Existem 3 intensidade: r1, r2 e r3. Note que o contagem de cada intensidade é igual à área do quadrado com cada itensidade. Definamos um vetor que determinará o histograma:

```
histogram_q12 = np.zeros([3])
histogram_q12
```

```
array([0., 0., 0.])
```

A intensidade r2 possui como histograma:

```
histogram_q12[1] = np.power(175-144,2)
```

A intensidade r3 possui como histograma:

```
histogram_q12[2] = np.power(191-128,2) - histogram_q12[1]
```

A intensidade r1 possui como intensidade:

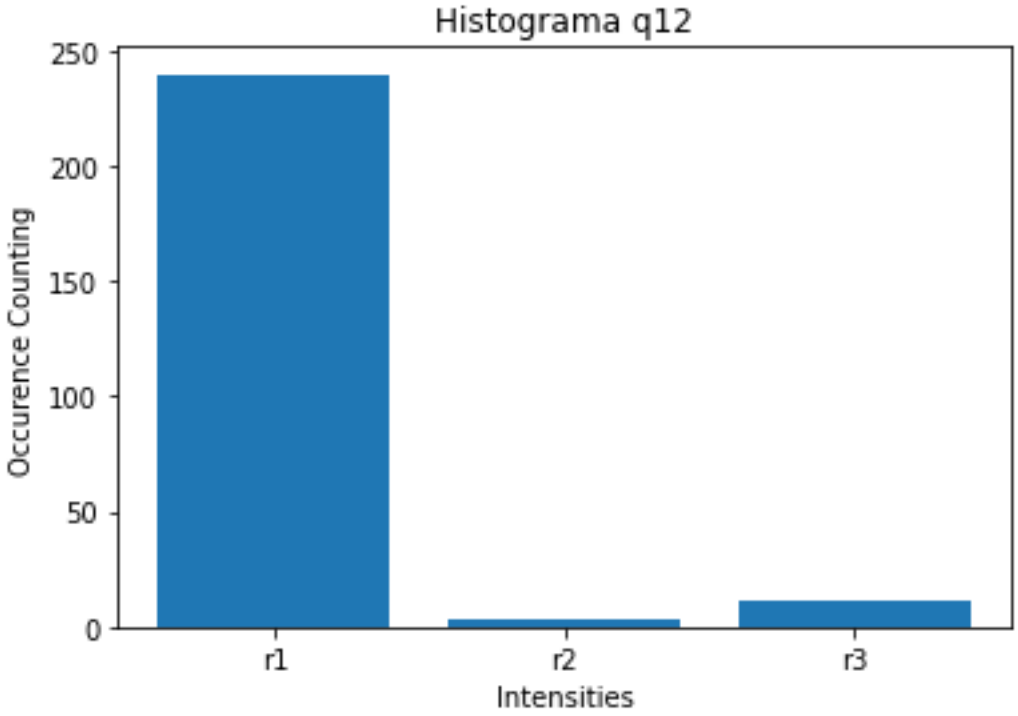
```
histogram_q12[0] = np.power(256,2) - histogram_q12[1] - histogram_q12[2]
```

Eis o nosso vetor:

```
255*histogram_q12/(256*256)
```

```
> array([239.55665588,   3.73924255,  11.70410156])
```

Visualmente:

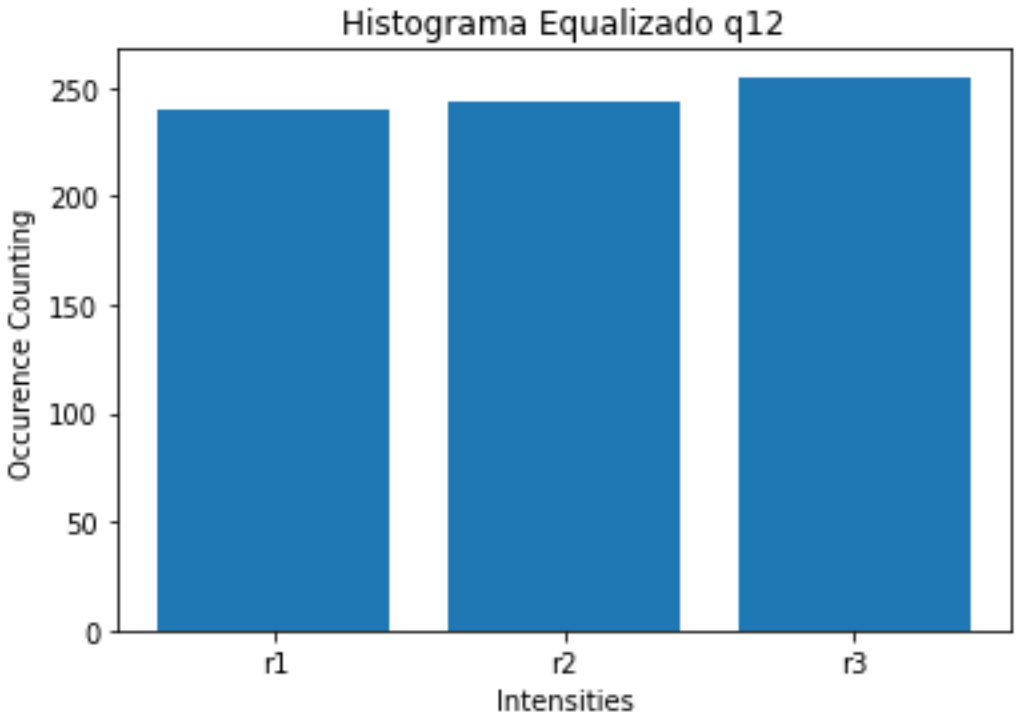


Agora vamos equalizar:

```
cum_hist = np.cumsum(histogram_q12)
255*cum_hist/(256*256)
```

```
> array([239.55665588, 243.29589844, 255.        ])
```

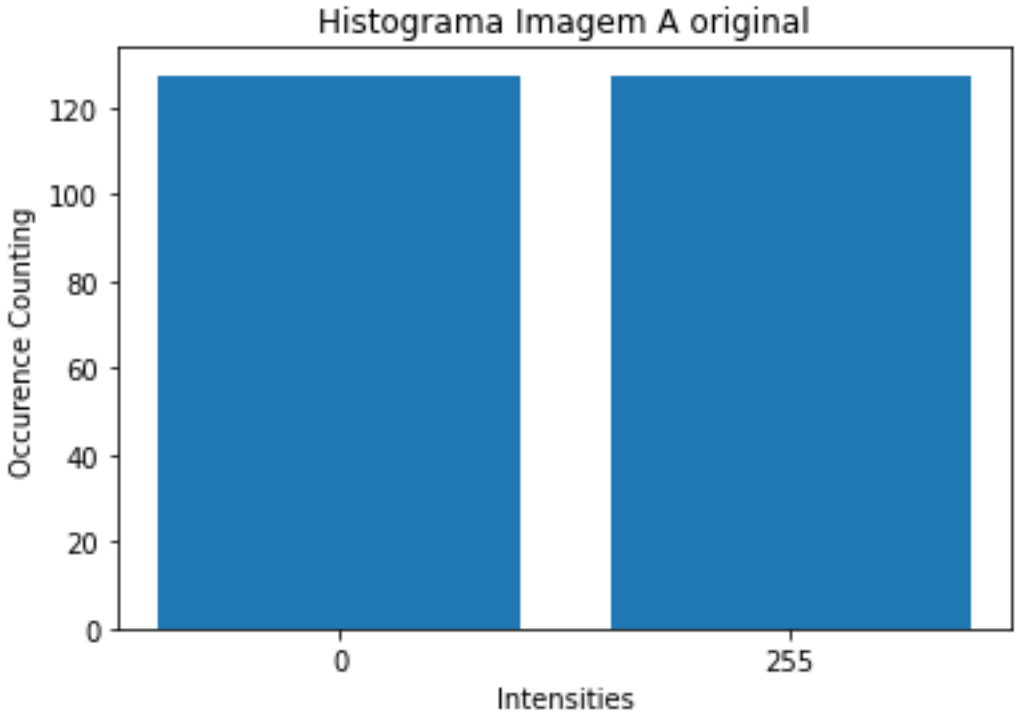
Visualmente:

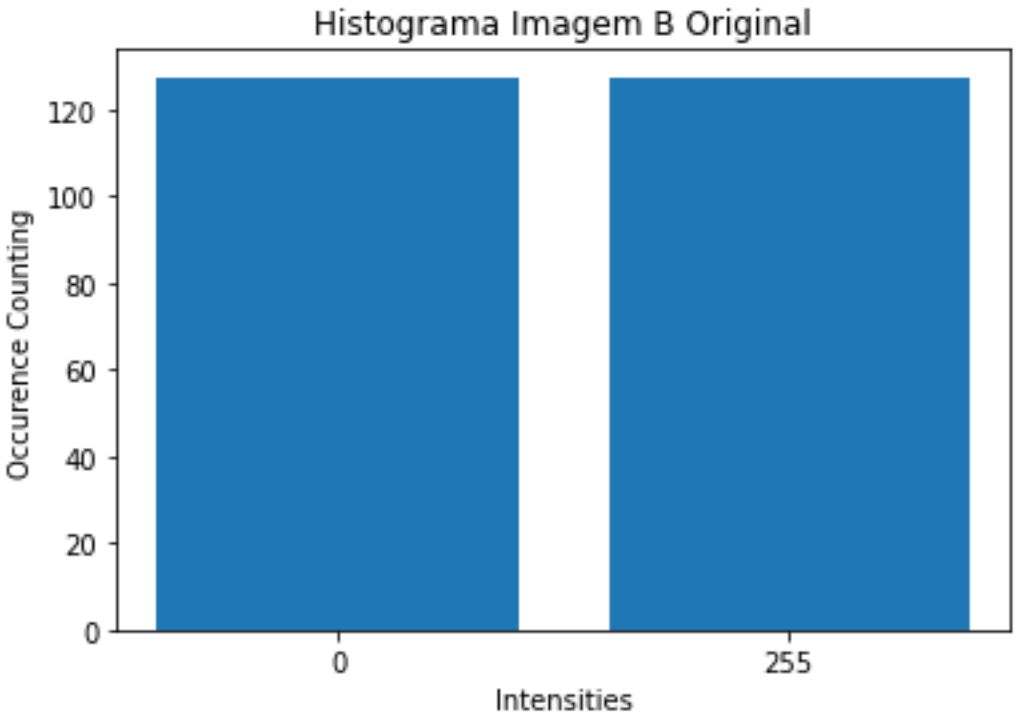


Note que, após equalizado, os valores do histograma não possuem uma diferença maior que 8 tons. Logo, não seria possível distinguir as regiões.

Question 13.

Para ambas imagens temos histogramas com duas colunas: uma para o tom escuro (intensidade 0) e outra para o tom claro (intensidade 255). Note, também, que o histograma de ambas as colunas se iguala a área de cada cor. Isto é, a área "branca" (definida como contagem de pixels brancos), iguala o histograma para a intensidade 255. O mesmo se repete para a área preta.





Considere que vai ser aplicado um filtro suavizante 3x3, dado por :

$$\frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Considere, também, que a imagem se repete infinitamente, o que nos permitirá definir:

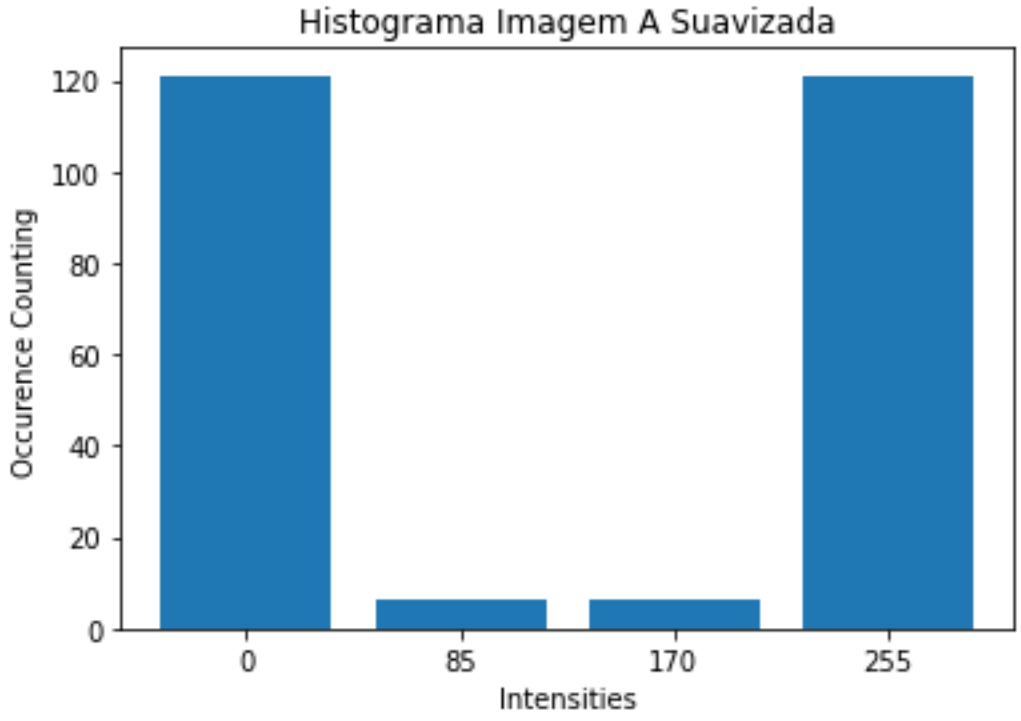
- acima da borda superior existe a cópia da imagem
- abaixo da borda inferior existe a cópia da imagem
- ao lado da borda direita existe a cópia da imagem
- ao lado da borda esquerda existe a cópia da imagem

Aplicado o filtro suavizante descrito acima na imagem a, teremos uma imagem tal que:

- A primeira e quadragésima coluna é preenchida com elementos de intensidade 170;
- Da segunda a trigésima nona coluna todos os elementos são preenchidos com intensidade 255;
- A quadragésima primeira e última coluna é preenchida com intensidade 85;
- Da quadragésima segunda até a penúltima coluna, todos os elementos são nulos.

Por que isso ocorre? Note que:

- A média executada pela máscara sobre a primeira e a quadragésima coluna sempre será entre 6 elementos com intensidade 255 e 3 elementos de intensidade nula;
- Da segunda a trigésima nona coluna todos os elementos multiplicados pela máscara possui mesmo valor (igual a 255);
- A média executada pela máscara sobre a quadragésima primeira e última coluna sempre será entre 6 elementos com intensidade nula e 3 elementos de intensidade igual a 255;
- Da quadragésima segunda até a penúltima coluna, todos os elementos multiplicados pela máscara possui mesmo valor (igual a 0)

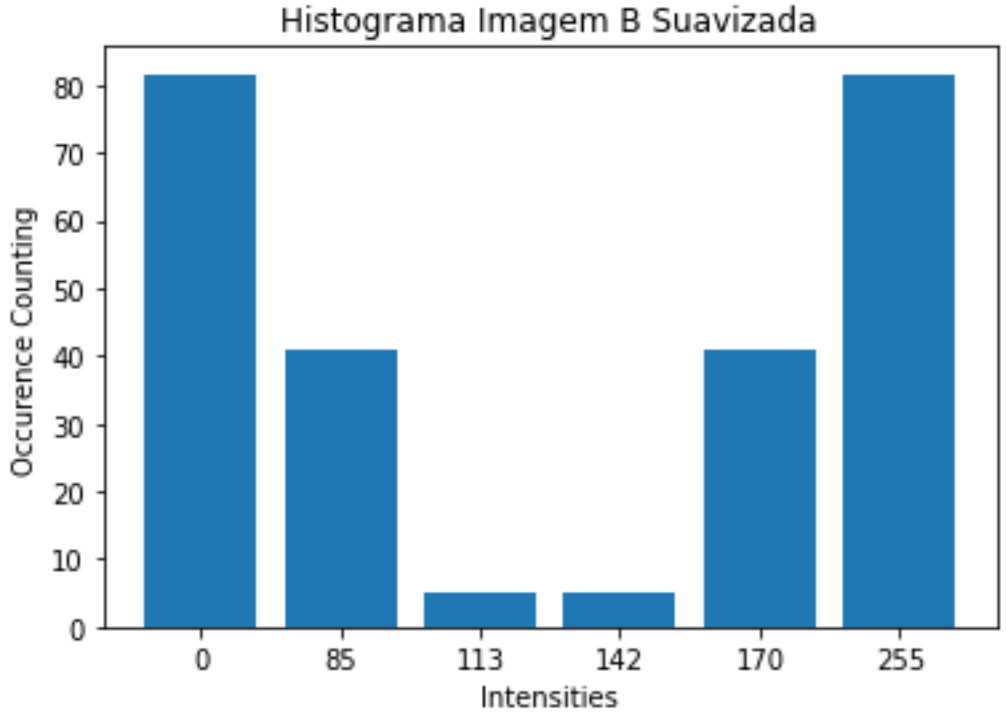


Aplicado o filtro suavizante descrito anteriormente na figura b, temos:

- 128 pixels de intensidade 113
- 1024 pixels de intensidade 85
- 2048 pixels de intensidade 0
- 128 pixels de intensidade 142
- 1024 pixels de intensidade 170
- 2048 pixels de intensidade 255

Por que isso ocorre? Bem, note que existem 32 quadrados pretos e 32 quadrados brancos. Cada um será rodeado por 4 quadrados de cor oposta e 4 quadrados de cor semelhante.

- Note que os pixels dos vértices sempre estarão rodeados de 4 pixels de cor oposta e 4 pixels de cor semelhante. Assim temos 4x32=128 pixels (e intensidades (4/9)x255 aprox. 113 e (5/9)x255 aprox. 142).
- Note que os pixels das bordas dos quadrados (excluindo-se os vértices, que já foram computados), sempre estarão rodeados por exatos 3 pixels de cores opostas e 5 de cores semelhantes. Assim temos 8 pixels para cada uma das 4 bordas para cada um dos 32 quadrados (8x4x32=1024) e intensidades(3/9)x255 e (6/9)x255.
- Note que os pixels centrais estarão rodeados por semelhantes. Disso deduz que haverão 8x8x32=2048 pixels com as intensidades da imagem original.



Question 14.

Para que o procedimento seja realizado corretamente, é necessário que a iluminação não tenha variado entre a aquisição da figura padrão e a figura das peças analizadas.

Também é necessário que a posição (cartesiana e angular) das peças não haja variado.

Question 15.

a. Aplicou-se as máscaras de filtro de sobel em x e y. Por fim, somou-se o valor absoluto dos resultados.

b. Aplicou-se as máscaras de filtro de sobel em x e y. Por fim, somou-se os resultados.

c. Aplicou-se as máscaras de filtro de sobel em x e y. Por fim, subtraiu-se o resultado do filtro de sobel em y do resultado do filtro de sobel em x.

d. Aplicou-se a máscara do filtro de sobel em x (direção vertical).