MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE CIÊNCIAS DA TERRA
Departamento de Geomática

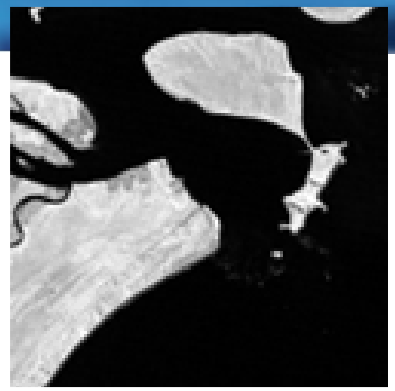
Disciplina: PROCESSAMENTO DIGITAL DE IMAGENS II
Código: GA144

CH Total:45 h

CH Semanal 03 h

Limiarizacao

Problema

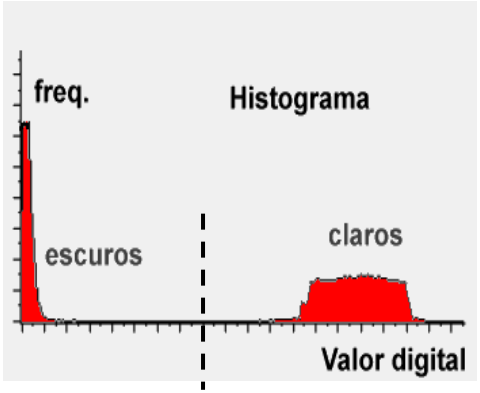


Dada uma imagem onde dois grandes grupos de pixels são visíveis (claro e escuros), achar o melhor limiar para separar estes dois grupos e binarizar a imaegm

Parte-se da hipótese de que existem dois grupos de pixels na imagem:

claros e escuros

FUNDO e OBJETO



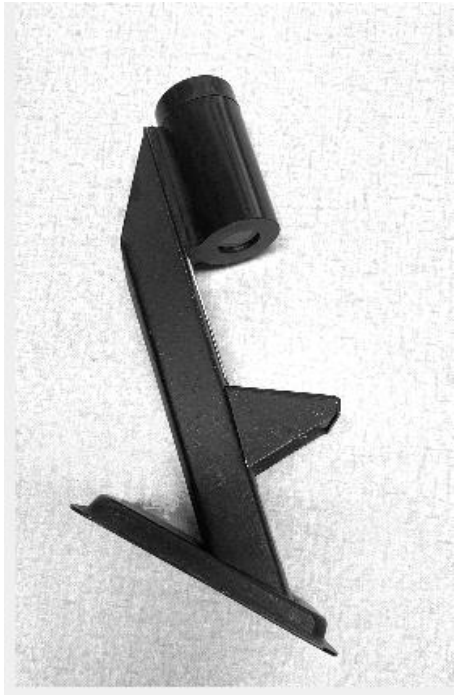
Para separar estes grupos é analisado o histograma da imagem. É assumido que o histograma é bimodal.

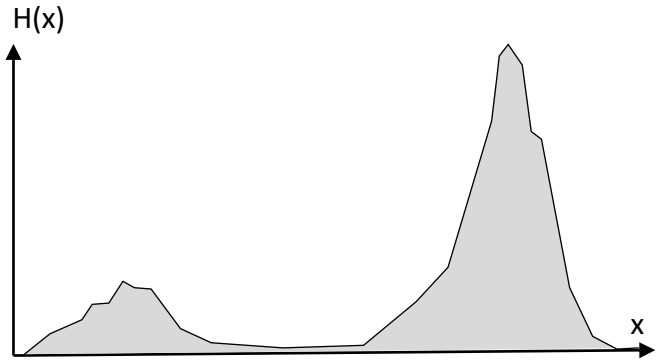
O problema é identificar automaticamente o valor ótimo para separar estes dois grupos.

Detectar limiar no Histograma

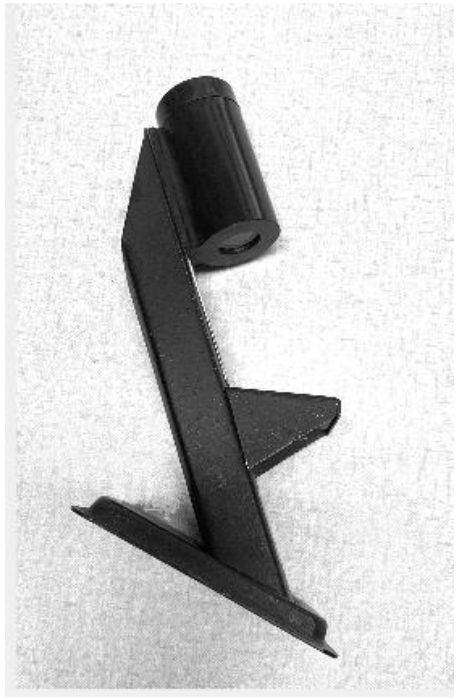
O Histograma H(x) representa a variação dos valores digitais na imagem. Se na imagem ocorrem apenas dois tipos de superfícies, claros e escurõs, o histograma será bimodal.

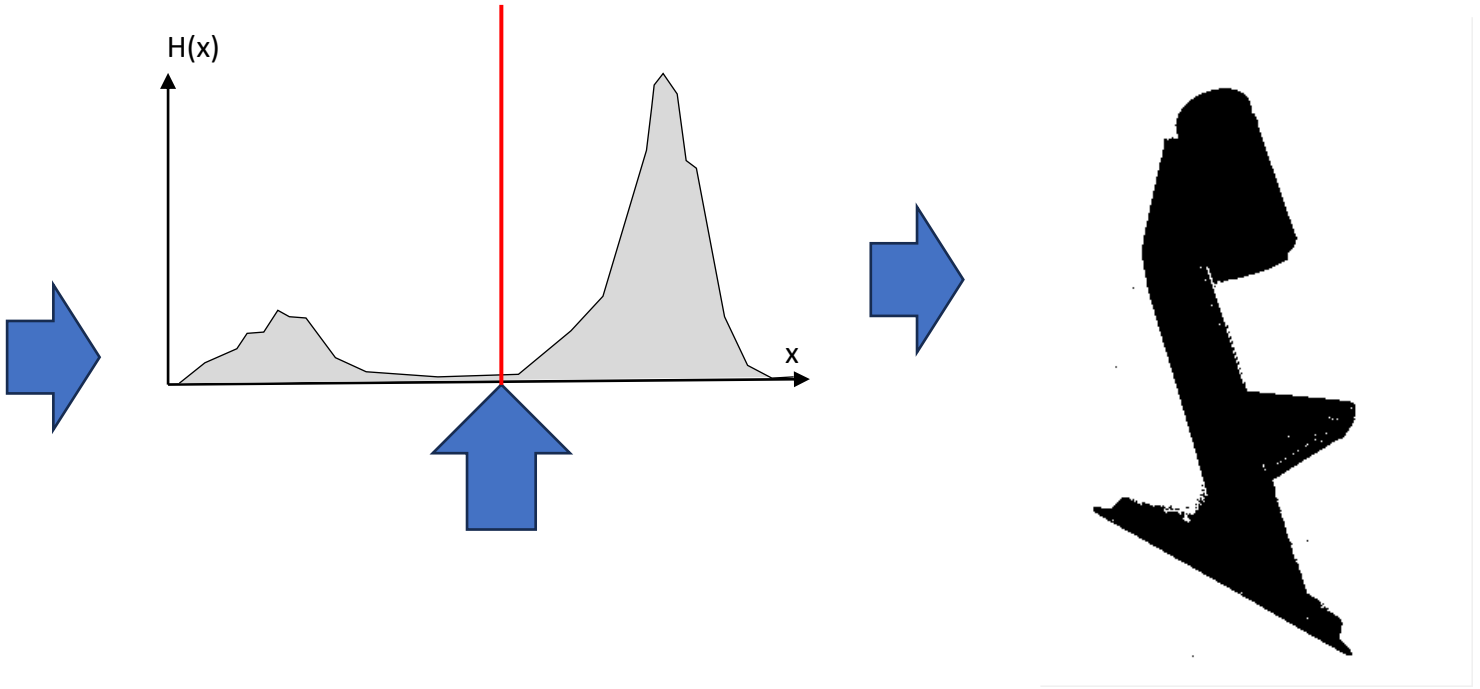
O histograma é uma função positiva definida em um domínio finito.





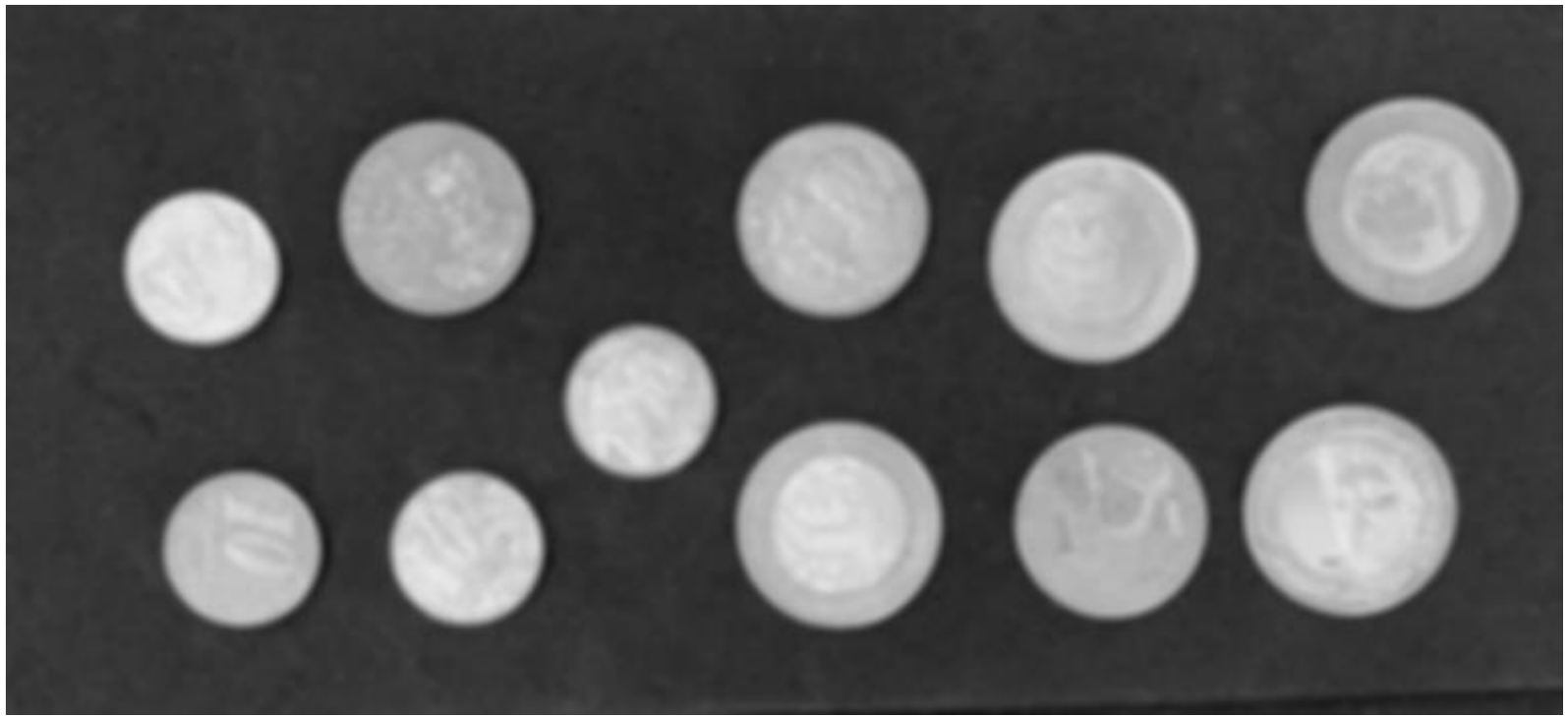
- Problema: determinar o limiar ótimo para qualquer imagem contendo claros e escuros





exemplo

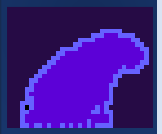
- Dada a imagem com diferentes objetos, separe os objetos do fundo escuro. Vale a pena analisar o histograma para identificar a fronteira entre grupos (escuro e claro)



Histograma

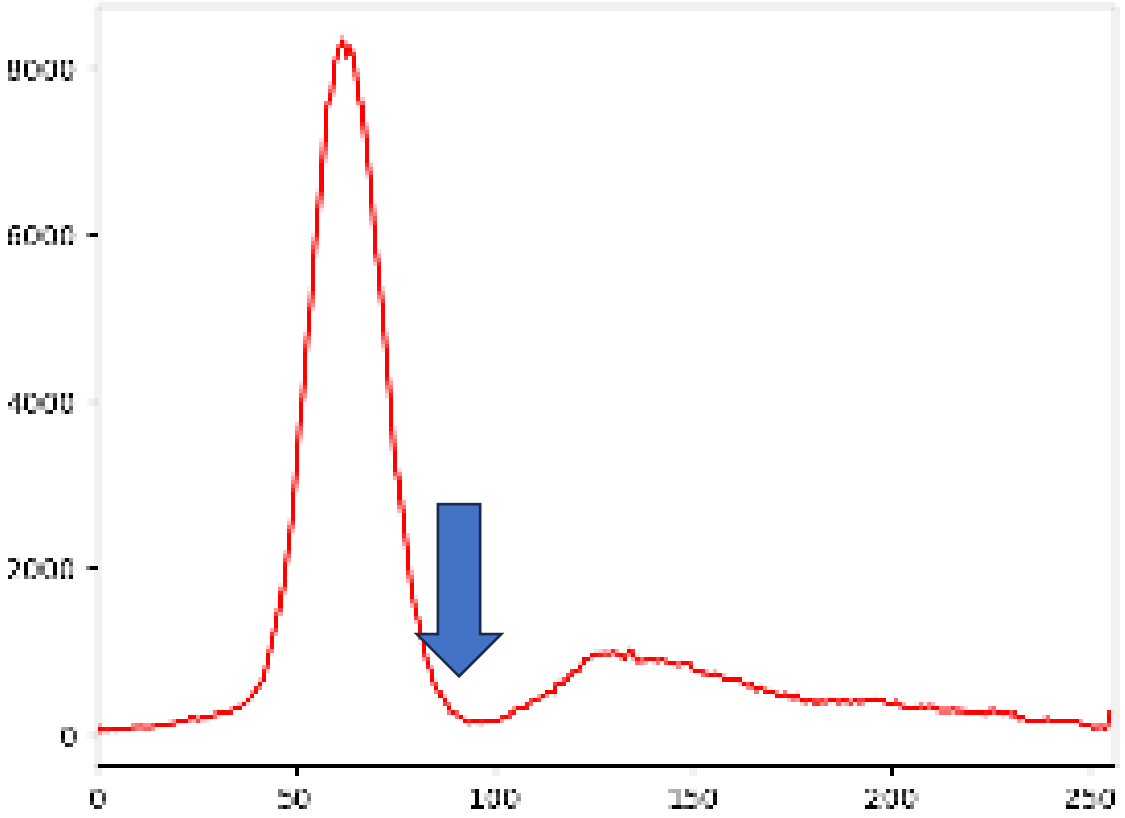
cv.calcHist(imagem, banda, mask, histSize, faixa)

- image : entrada uint8 ou float32. Deve ser escrito em colchetes, "[I]".
- banda : No caso de imagens coloridas, especifica qual banda será processada. Em imagens em nível de cinza, deve-se usar [0], sempre entre colchetes.
- mask : opção de processor apenas uma parte da imagem (mascara). Por default se processa toda a imagem com a opção "None".
- histSize : número de elementos do histograma. Em imagens de 8 bits o correto é usar [256].
- faixa : a faixa a ser representada, em colchetes. Geralmente [0,256]

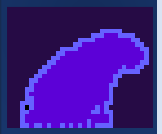


```
hist = cv2.calcHist([I],[0],None,[256],[0,256])

from matplotlib import pyplot as plt

plt.plot(hist,color = 'red')
plt.xlim([0,256])
plt.show()
```

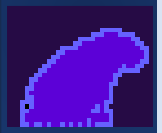
- Analisar o histograma e detectar o melhor valor LIMAR



Limiar =90, ou 100?



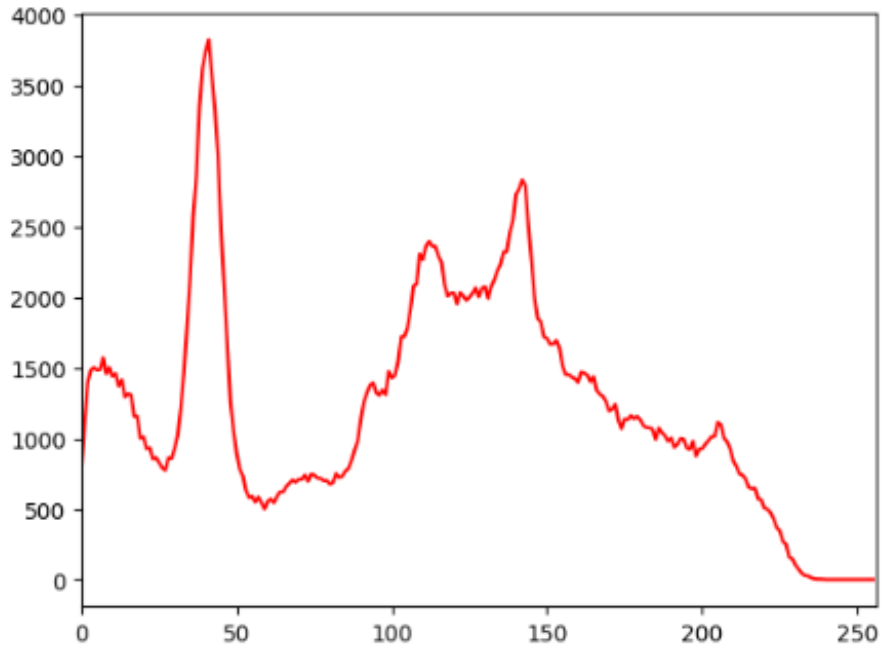
```
Limiar=100
for i in range(nlin):
  for j in range(ncol):
     if I[i,j] > Limiar:
          J[i,j] = 255
     else:
          J[I,j] = 0
```



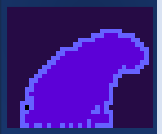
- Limiarizacao MANUAL

OpenCV

- histograma



especificar



opções

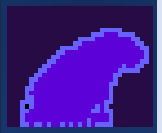
th, K = cv2.threshold(IMA, limiar, maximo, MODO)

- th: limiar (redundante)
- K: imagem de saída
- **IMA:** imagem de entrada
- **Limiar:** Limiar especificado pelo usuário. Deve estar dentro da faixa de valores da imagem. Vale a pena visualizar o histograma para escolher um valor
- **Maximo:** o valor que será atribuído aos pixels que superem o limiar
- **MODO:** opção de binarização, pode ser simples ou preservando valores originais. veja a lista …

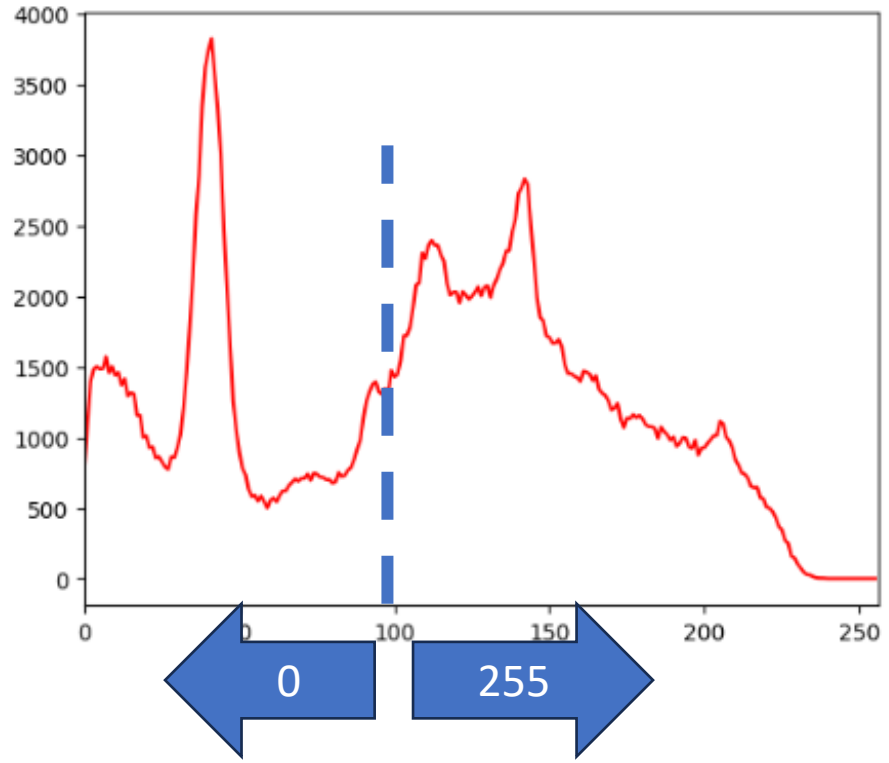
Exemplo

cv2.THRESH_BINARY: opção de binarização simples

```
maximo=255
Limiar=100
if I[x,y] > Limiar:
  J[x,y] = maximo
else:
  J[x,y] = 0
```



```
limiar=100
maximo=255
th, K = cv2.threshold(I, limiar, maximo, cv2.THRESH_BINARY)

cv2_imshow(K)
print(th)
```



Limiar=100

Qual o melhor Limiar para separar as áreas escuras desta imagem?

- MODO: THRESH_BINARY_INV

Neste caso, o resultado é o negative do anterior. Ou seja, os valores que superam o limiar são marcados como zero.

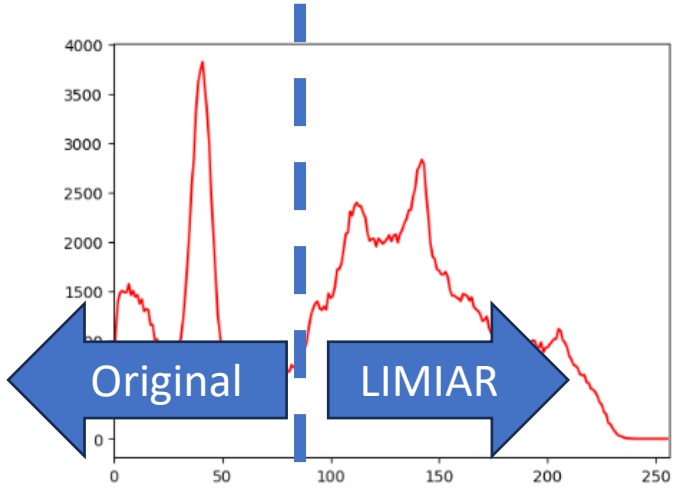
```
maximo=255
Limiar=100
if I[x,y] > Limiar:
  J[x,y] =0
else:
  J[x,y] = maximo
```

- MODO THRESH_TRUNC

É usado para salientar os valores acima do limiar e preservar aqueles abaixo do limiar. Neste caso, os valores que seriam anulados(abaixo do limiar) preservam seu valor original e aqueles acima do limiar assumem o valor do limiar. Equivale a truncar a imagem com o limiar.

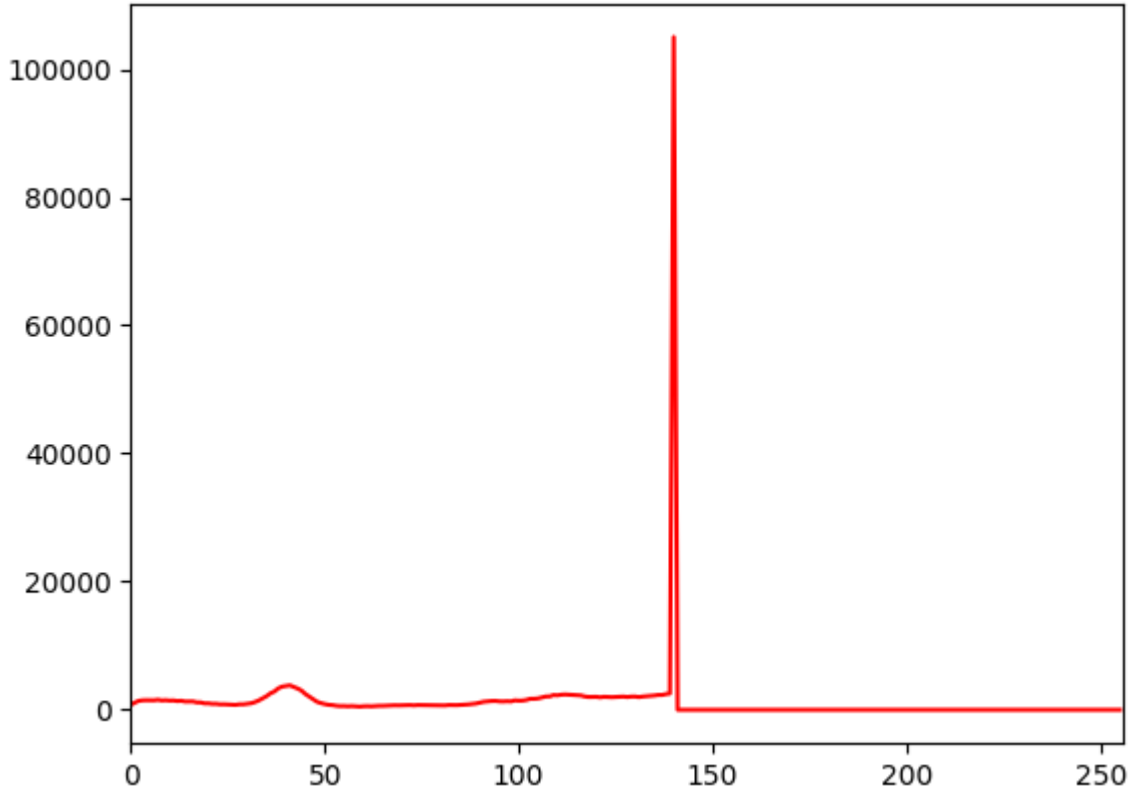
```
th, K = cv2.threshold(I, limiar, maximo, cv2.THRESH_TRUNC)
```

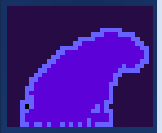
Ignora-se máximo




```
maximo=255
Limiar=100
if I[x,y] > Limiar:
  J[x,y] = maximo
else:
  J[x,y] =I[i,j]
```

```
limiar=140
th, K = cv2.threshold(I, limiar, maximo, cv2.THRESH_TRUNC)
cv2_imshow(K)
```



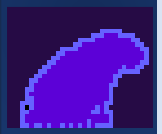


THRESH_TOZERO

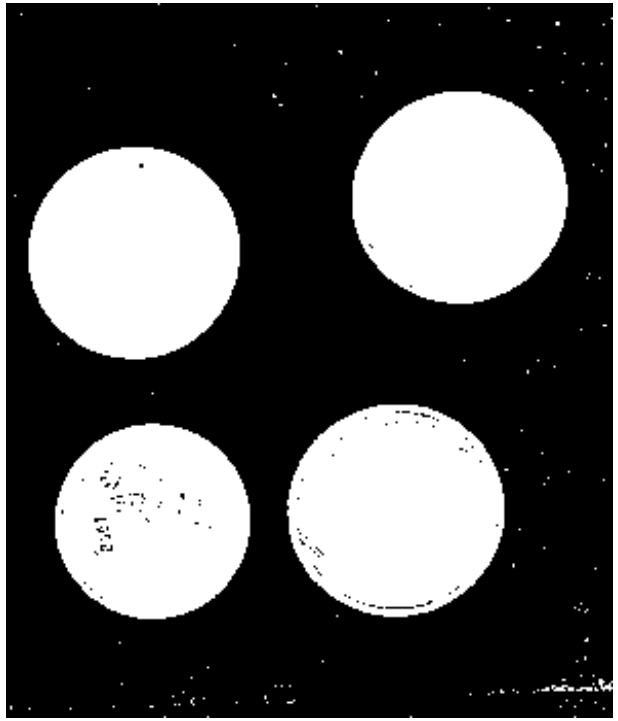
- Similar ao anterior, mas neste caso o valor original é representado se o limiar for superado. Os pixels abaixo do limiar são anulados (zero)

THRESH_TOZERO_INV

- Inverted Threshold to Zero,
- Se o valor do limiar for superado, o pixel recebe zero. Caso contrário o valor original é preservado.



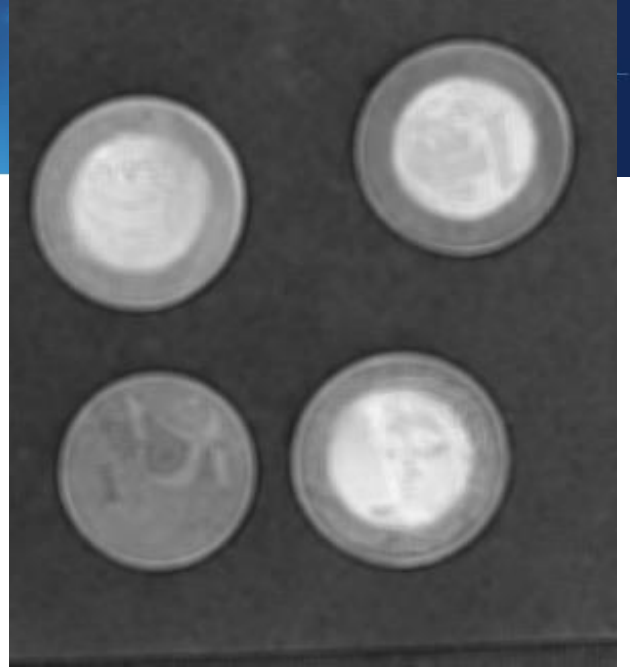
- Cuidado! Devido às sombras, as regiões podem conter “buracos”, e pontos claros podem ocorrer no “fundo”. Como contornar este problema?

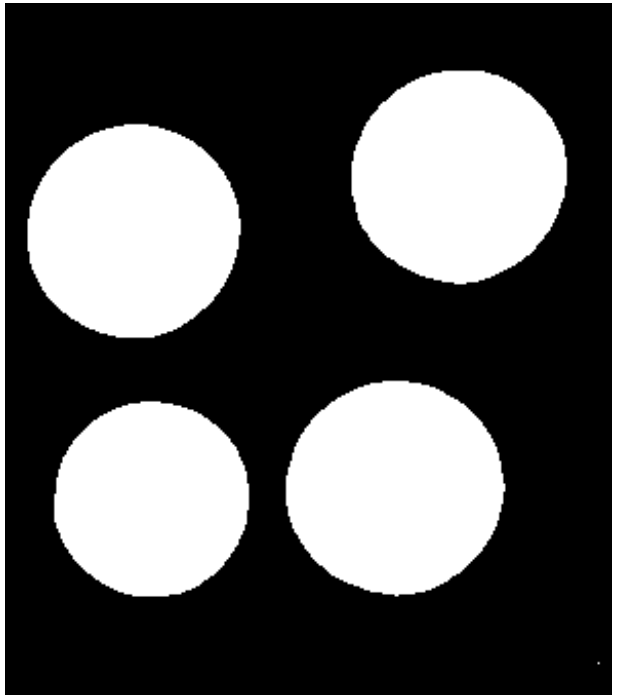


Aplicar suavização para uniformizar as regiões

E depois, bnarizar

```
im = cv2.blur(src=I, ksize=(5,5))
 cv2_imshow(im)

limiar=90
maximo=255
th, K = cv2.threshold(im, limiar, maximo, cv2.THRESH_BINARY)
cv2_imshow(K)
```





MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE CIÊNCIAS DA TERRA
Departamento de Geomática

Disciplina: PROCESSAMENTO DIGITAL DE IMAGENS II
Código: GA144

CH Total:45 h

CH Semanal 03 h

Método de OTSU

OTSU

Método de OTSU

Trata o Histograma da imagem como uma Função Densidade de Probabilidade Discreta:

$$p(x)=H(x)/N$$

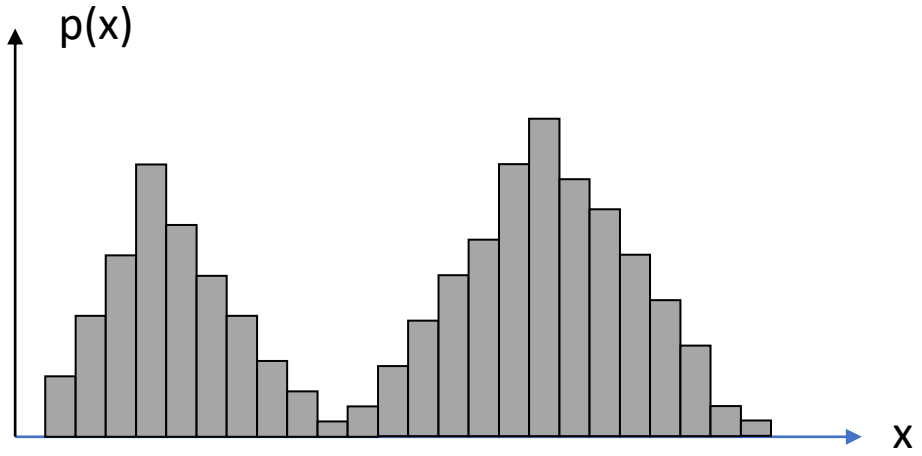
Onde:

x= valor digital, com q= 0,1,2,……255 (*pode ser outro valor máximo)

H(x) = número de pixels com valor digital x

N = número total de pixels na imagem

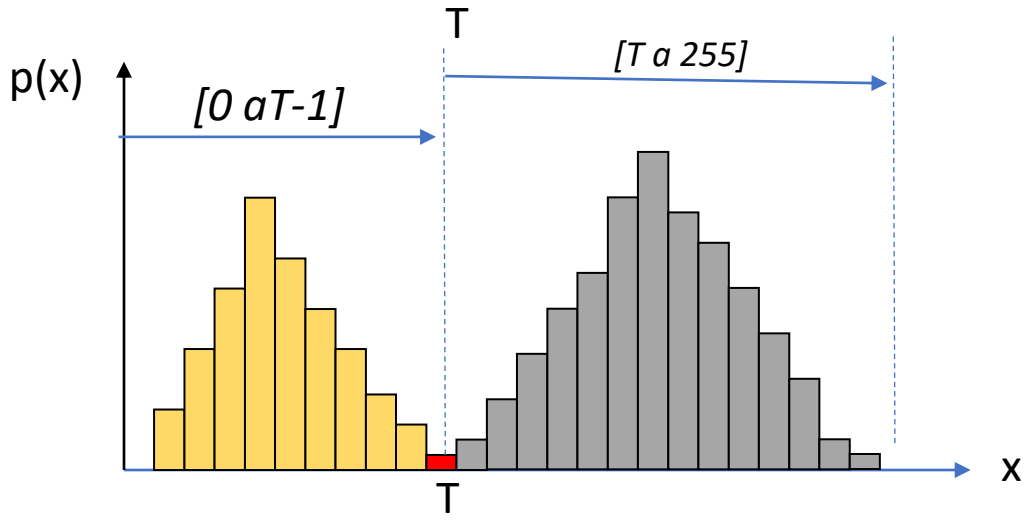
Qual valor é mais provável?
quele mais frequentemente!



OTSU

Usando o Limiar (T) pode-se separar a imagem em duas classes, dois grupos:

- A = pixels com valores entre [0, T-1] e
- B= pixels com valores entre [T, 255]

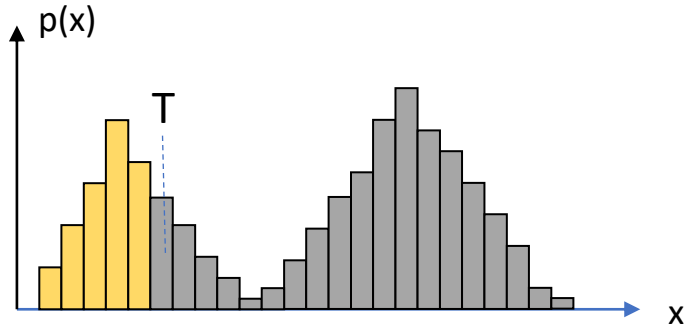



```
Limiar=100
if I[x,y] < Limiar:
  J[x,y] = 0
else:
  J[x,y] = 255
```

Cada grupo é descrito por:
soma das probabilidades das classes ,

a área de cada grupo (w)

- valor x médio (m)
- variância (S)



$$w_1 = \sum_{o}^{T-1} p(x) \qquad w_2 = \sum_{T}^{255} p(x)$$

$$m_1 = \sum_{0}^{T-1} x * p\,(x)/w_1 \qquad m_2 = \sum_{T}^{255} x * p\,(x)/w_1$$

$$S_1 = \sum_{0}^{T-1} (x - m_1)^2 * p\,(x)/w_1 \qquad S_2 = \sum_{T}^{255} (x - m_1)^2 * p\,(x)/w_2$$

Variância (S) mínima

O Limiar ótimo separa os pixels em dois grupos uniformes.

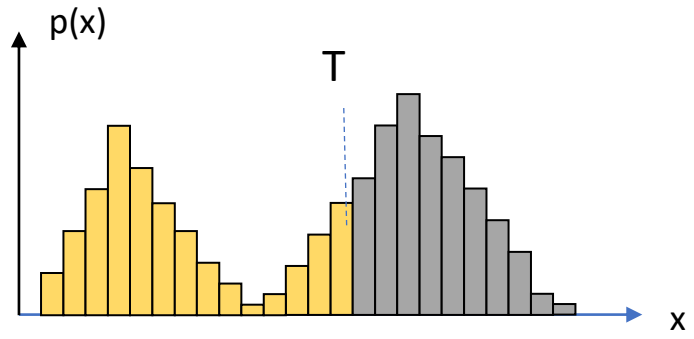
Grupos uniformes tem variância baixa.

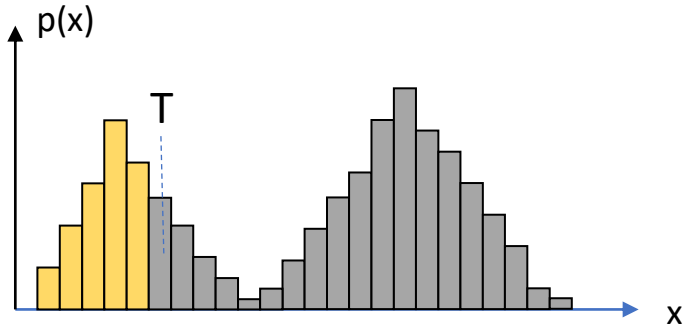
Logo, a variância combinada dos dois grupos deve ser baixa

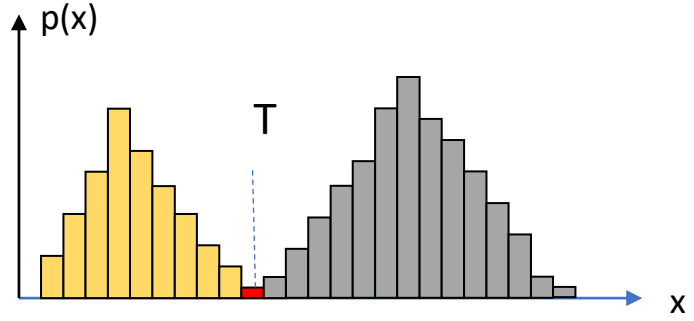
A variância combinada é dada pela soma (ponderada) das variâncias

$$S^2(T) = a_1(T)\ S_1^2(T) + a_2(T)\ S_2^2(T)$$

a= fator de ponderação







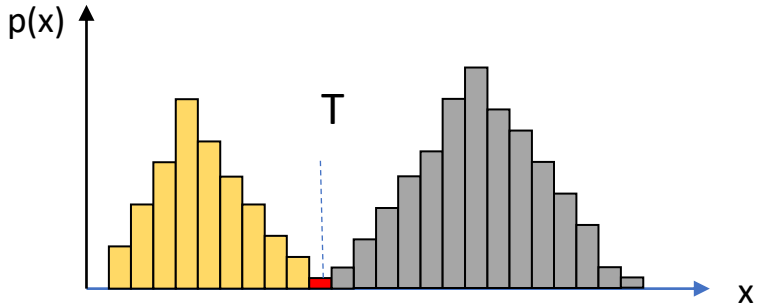
Variância (S) mínima

Busca-se o Limiar “T” que minimiza a variância combinada (dentro dos grupos).
Como “peso”, usa-se a soma das probabilidades das classes (a área).
Classe mais frequente “pesa” mais.

$$S^2(T) = w_1(T)\ S_1^2(T) + w_2(T)\ S_2^2(T)$$

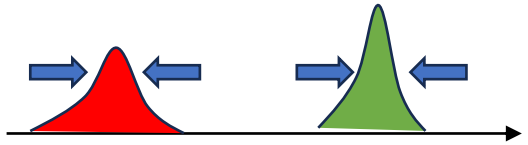
$$w_1(T) = \sum_{0}^{T-1} p(x)$$

$$w_2(T) = \sum_{T}^{255} p(x)$$



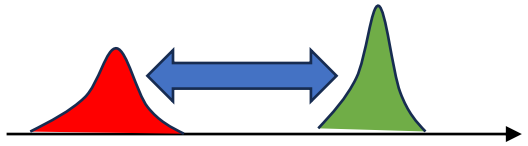
Minimizar a variância dentro das classes equivale a

Maximizar a variância entre classes:



$$S^2_{entre}(T) = S^2 - S^2_{dentro}(T)$$

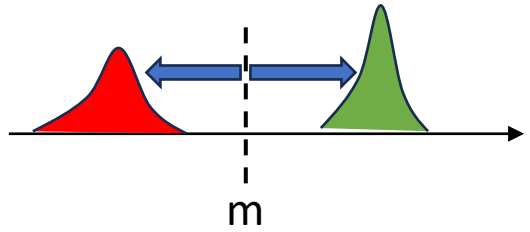
A variância entre classes pode ser calculada com ajuda da distância das médias dos grupos à média de toda a imagem



média da imagem

$$m = w_1 m_1 + w_2 m_2$$

$$S_{entre}{}^2 = w_1(m_1 - m)^2 + w_2(m_2 - m)^2$$



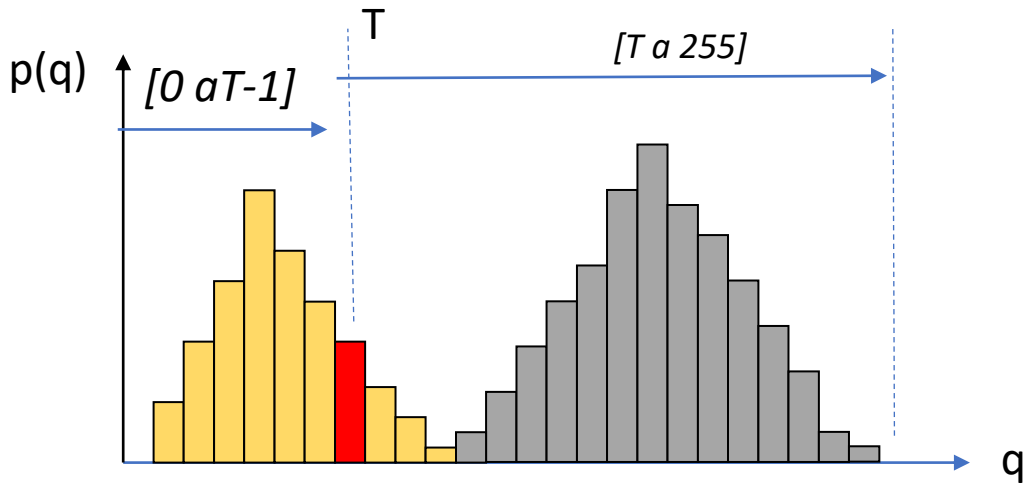
Ou, desenvolvendo e agrupando...

$$S_{entre}{}^2 = w_1 w_2 (m_1 - m_2)^2$$

OTSU

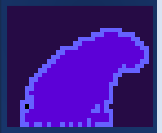
Como fazer?

Varrer todos os possíveis valores e calcular a variância para achar o máximo



Consulte:Otsu N.,"*A Threshold Selection Method from Gray-level Histograms*", IEEE Transactions on Systems, Man and Cybernetics, v. SMC 9, no 1, pp.62-66, 1979.

OTSU



```
# Gaussian Blur (7x7)
im = cv2.GaussianBlur(I, (7, 7), 0)


#   threshold com OTSU
th, threshold = cv2.threshold(im, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
cv2_imshow(K)
print(th)  # veja o valor do limiar que foi calculado automaticamente
```