



# Processamento digital de imagens

Segmentação  
Limiarização



# Problema

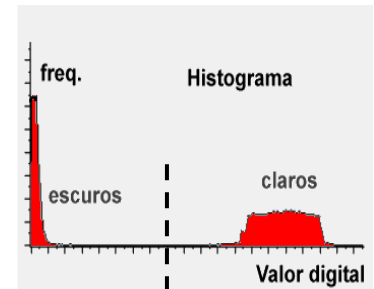
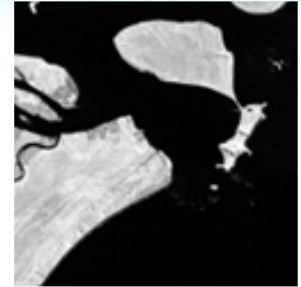
Dada uma imagem onde dois grandes grupos de pixels são visíveis (claro e escuros), achar o melhor limiar para separar estes dois grupos e binarizar a imagem

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.

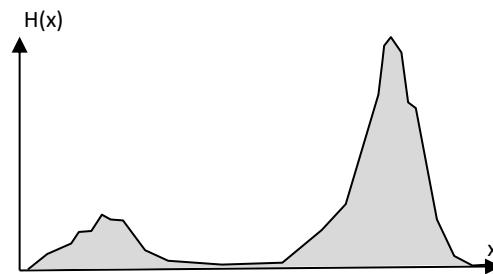
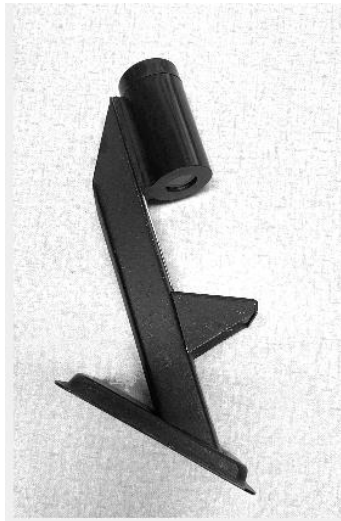
O problema é identificar automaticamente o valor ótimo para separar estes dois grupos.



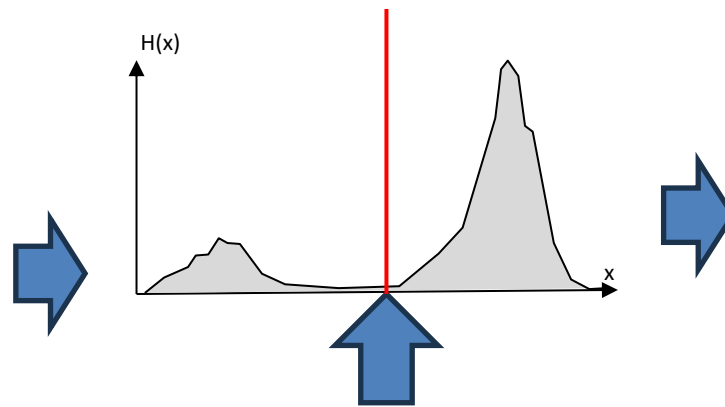
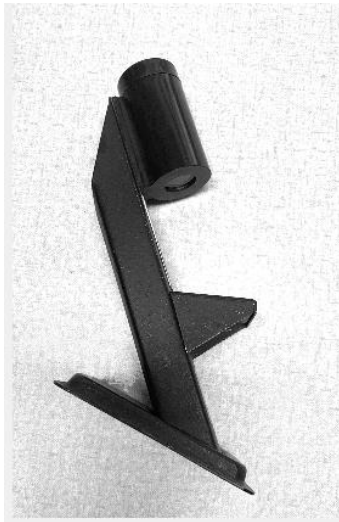
# Detectar limiar no Histograma

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

O Histograma representa a variação dos valores digitais na imagem. Se na imagem ocorrem apenas dois tipos de pixels, claros e escuros, o histograma será bimodal.

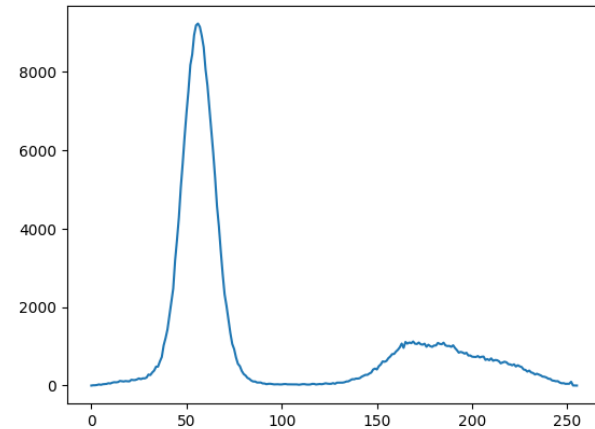


- Problema: determinar o limiar ótimo para qualquer imagem contendo claros e escuros



# exemplo

Dada a imagem com diferentes objetos, para separar os objetos (moedas) do fundo escuro, analisar o histograma para identificar o melhor limiar para definir os grupos (escuro e claro)



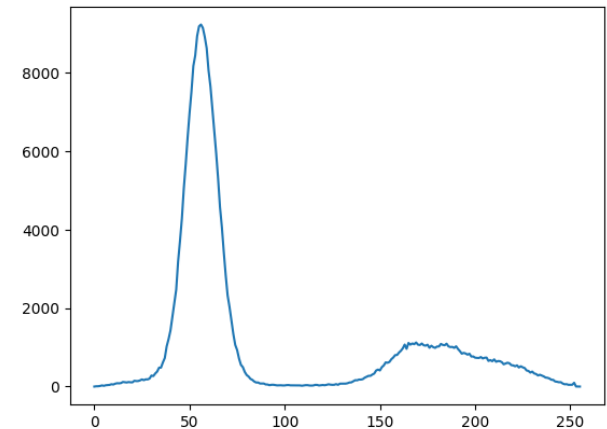
# Thresholding - (limiarização)

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

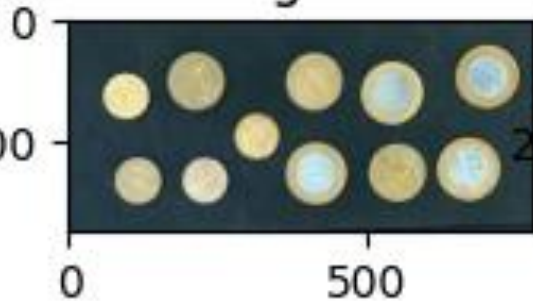
claros e escuros (FUNDO e OBJETO)

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

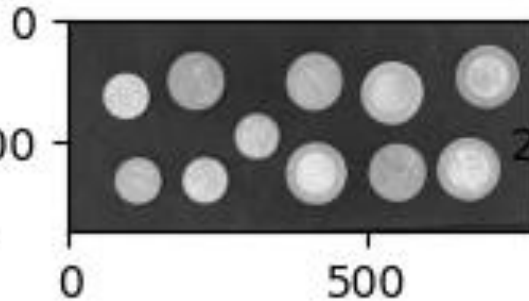
O resultado é uma imagem binária.



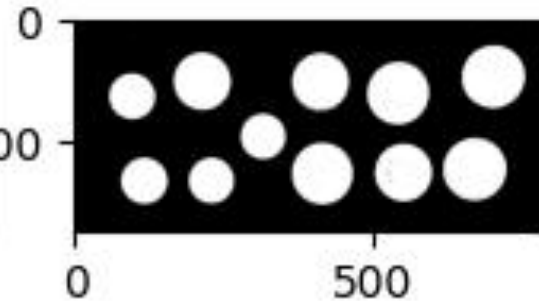
original




cinza



binaria





Disciplina: PROCESSAMENTO DIGITAL DE IMAGENS I  
Código: GA107

# Método de OTSU



# OTSU

## Método de OTSU

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

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

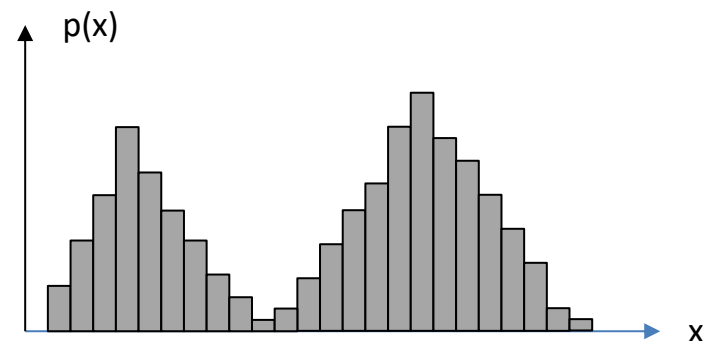
Onde:

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

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

$N$  = número total de pixels na imagem

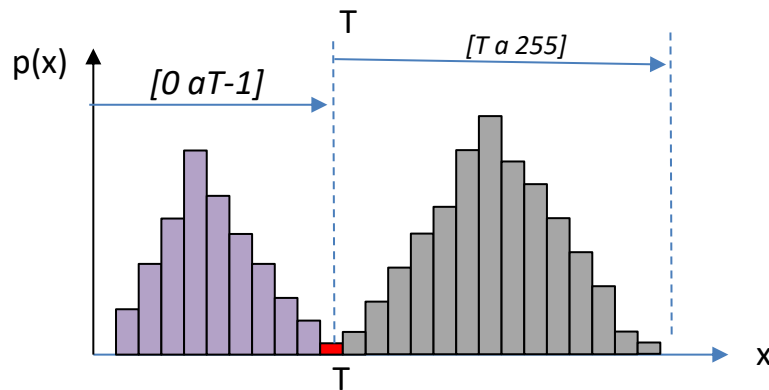
Qual valor é mais provável?  
Qual é mais frequentemente!



# 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  $[T, 255]$

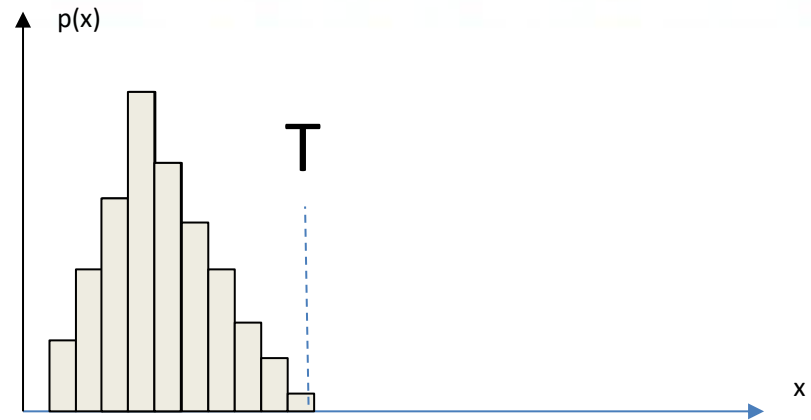


```
Limiar=100
if I[x,y] < Limiar:
    J[x,y] = 0
else:
    J[x,y] = 255
```

# Grupo 1: 0 até "T"

A área (w):

$$w_1 = \sum_0^{T-1} p(x)$$



A média (m):

$$m_1 = \frac{\sum_0^{T-1} x * p(x)}{\sum_0^{T-1} p(x)}$$

$$m_1 = \frac{\sum_0^{T-1} x * p(x)}{w_1}$$

A variância (S):

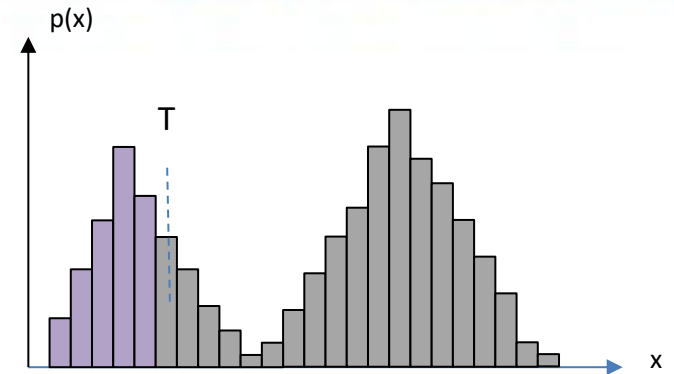
$$S_1 = \frac{\sum_0^{T-1} (x - m_1)^2 * p(x)}{\sum_0^{T-1} p(x)}$$

$$S_1 = \frac{\sum_0^{T-1} (x - m_1)^2 * p(x)}{w_1}$$

# Dois Grupos

Cada grupo é descrito por:  
a área de cada grupo ( $w$ )

- valor  $x$  médio ( $m$ )
- variância ( $S$ )



$$w_1 = \sum_0^{T-1} p(x)$$

$$m_1 = \sum_0^{T-1} x * p(x) / w_1$$

$$S_1 = \sum_0^{T-1} (x - m_1)^2 * p(x) / w_1$$

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

$$m_2 = \sum_T^{255} x * p(x) / w_2$$

$$S_2 = \sum_T^{255} (x - m_2)^2 * p(x) / w_2$$

# Variância (S) mínima

Se o Limiar ótimo separa os pixels em dois grupos uniformes ( com baixa variância).

Logo, a variância combinada dos dois grupos deve ser baixa.

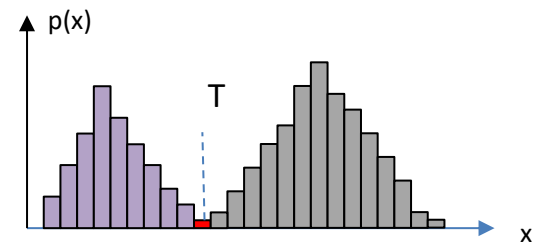
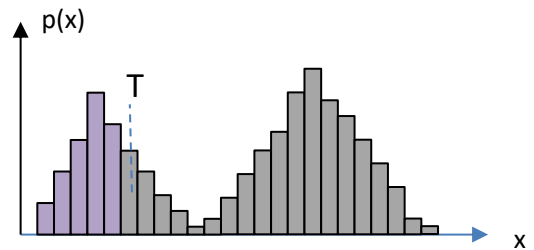
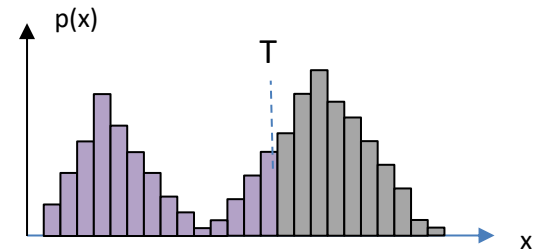
$$S^2 = S_1^2 + S_2^2 \dots \text{baixa}$$

Mas, como os grupos tem áreas diferentes, devemos dar pesos diferentes a cada variância.

$$S^2 = a_1 S_1^2 + a_2 S_2^2$$

a= fator de ponderação.

Como a área de cada grupo depende do limiar, este fator depende do limiar adotado "T" ... a(T)



# Variância (S) mínima

Busca-se o Limiar “T” que minimiza a variância combinada (dentro dos grupos).

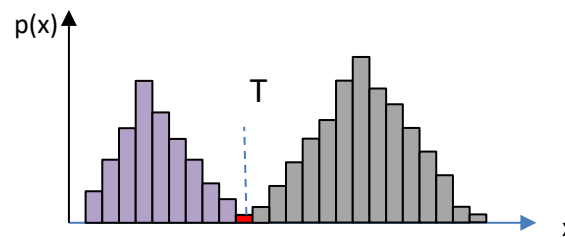
Como “peso”, usa-se a soma das probabilidades das classes (a área).

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:

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

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

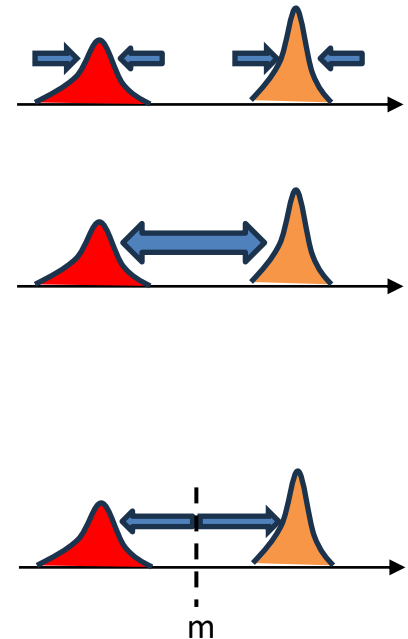
média da imagem (ponderada pela área)

$$m = w_1 m_1 + w_2 m_2$$

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

Ou, desenvolvendo e agrupando...

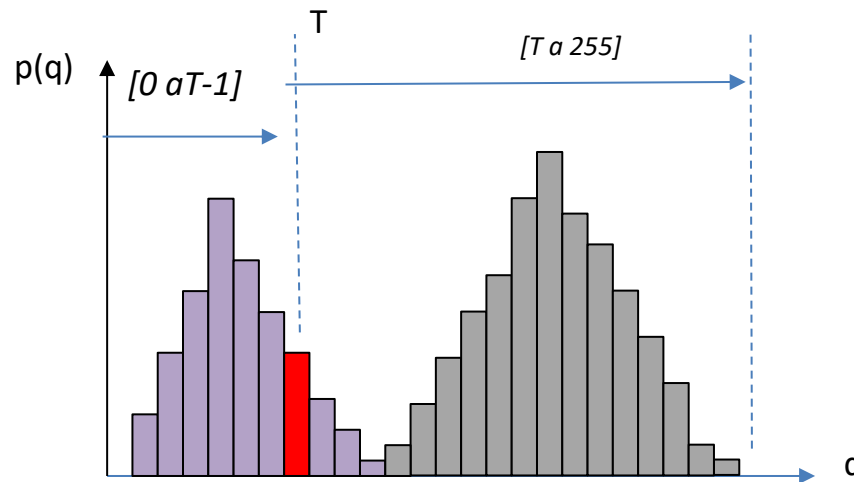
$$S_{entre}^2 = w_1 w_2 (m_1 - m_2)^2$$



# OTSU

Como fazer?

Varrer todos os possíveis valores ( $T$ ) e calcular a variância combinada para cada caso e escolher o limiar associado à maior variância entre classes



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.



# 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])
```

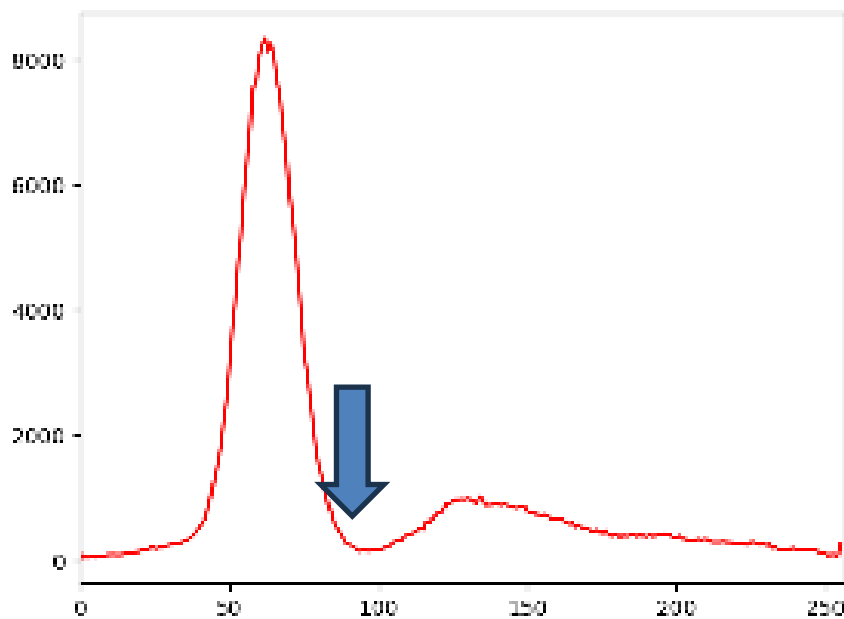
```
from matplotlib import pyplot as plt
```

```
plt.plot(hist,color = 'red')
```

```
plt.xlim([0,256])
```

```
plt.show()
```

- Analisar o histograma e detectar o melhor valor LIMAR



Limiar =90, ou 100?

# PROGRAMA SIMPLES

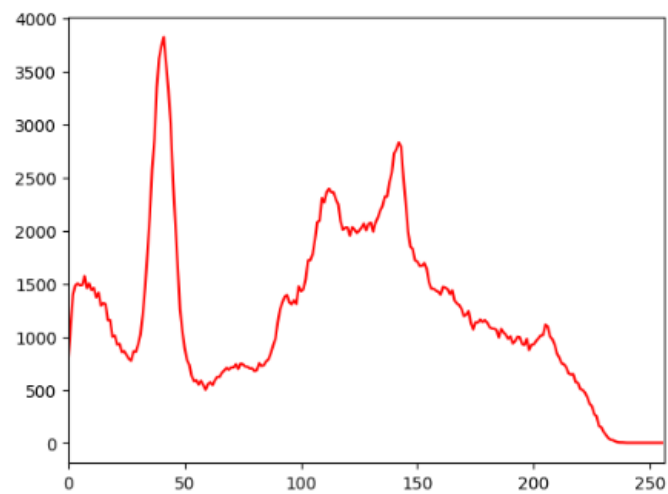
## MANUAL

```
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
```

- Binarização em OpenCV selecionando limiar MANUALmente



- 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

```
limiar=100
```

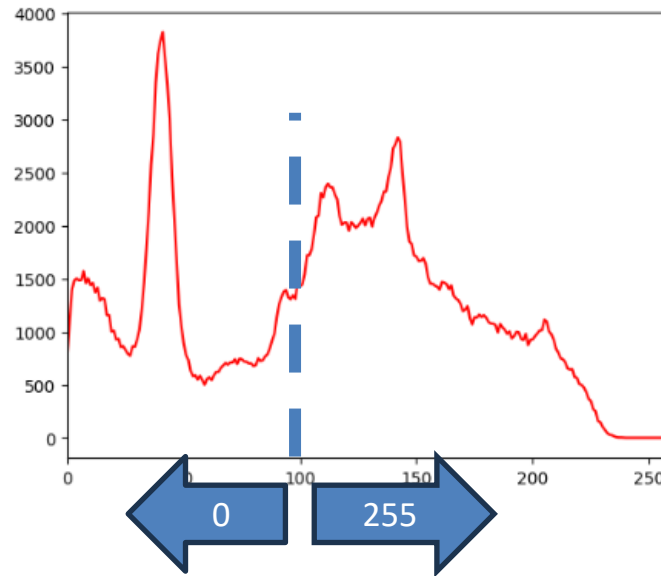
```
maximo=255
```

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

```
cv2.imshow(K)
```

```
print(th)
```

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



Limiar=100

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

- MODO: THRESH\_BINARY\_INV

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



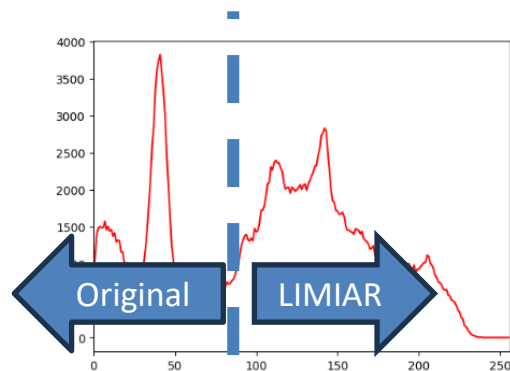
```
maximo=255
Limiar=100
if I[x,y] > Limiar:
    J[x,y] =0
else:
    J[x,y] = maximo
```

## • 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)
```

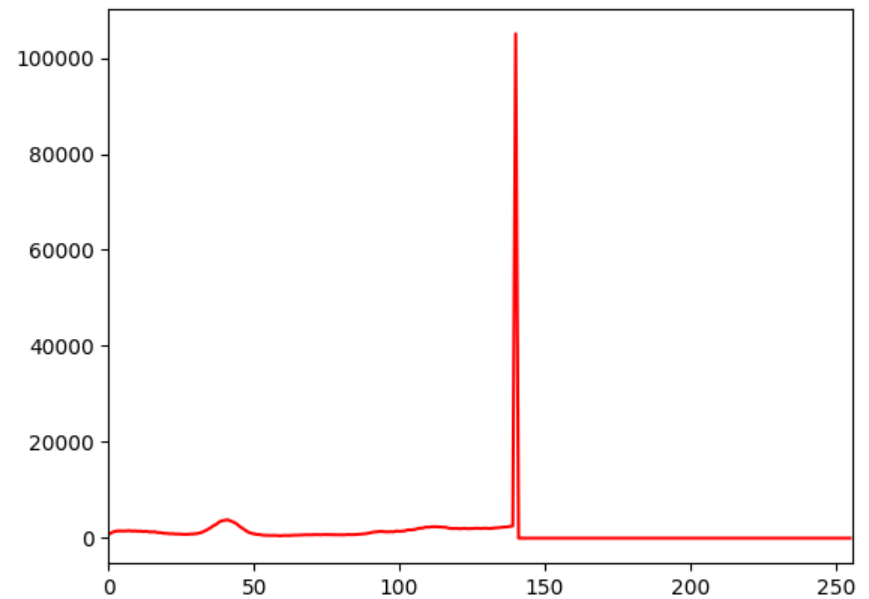
Ignora-se máximo



```
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)
```



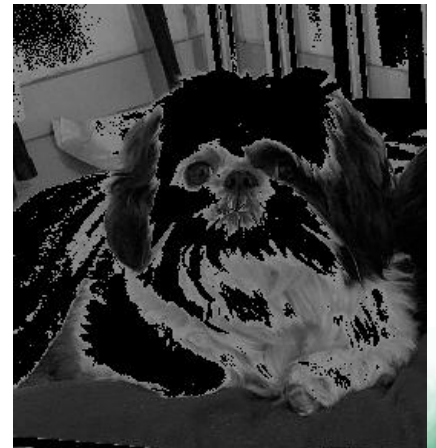
## THRESH\_TOZERO

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

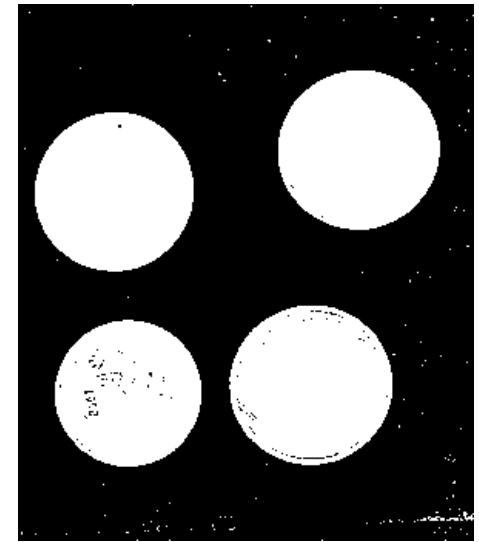


## THRESH\_TOZERO\_INV

- 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, binarizar

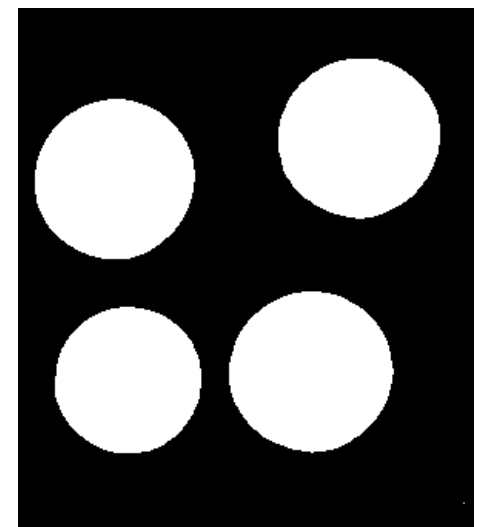
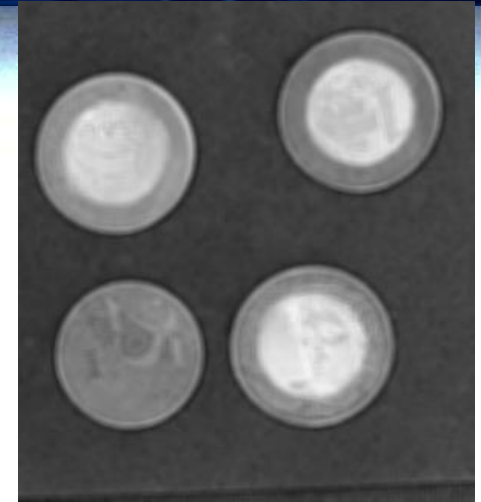
```
im = cv2.blur(src=I, ksize=(5,5))  
cv2_imshow(im)
```

```
limiar=90
```

```
maximo=255
```

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

```
cv2_imshow(K)
```



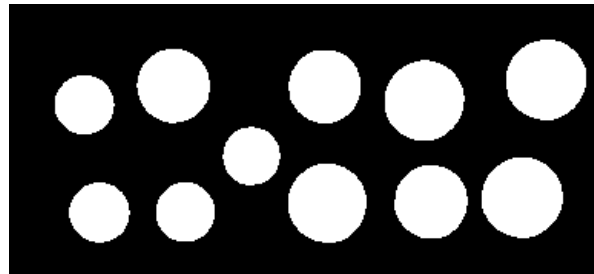
- 
- Seleção automática de LIMIAR por OTSU
- 

# OTSU



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

# threshold com OTSU
th, K = 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
```





# Processamento digital de imagens

Componentes Conexos

CPGCG/UFPR

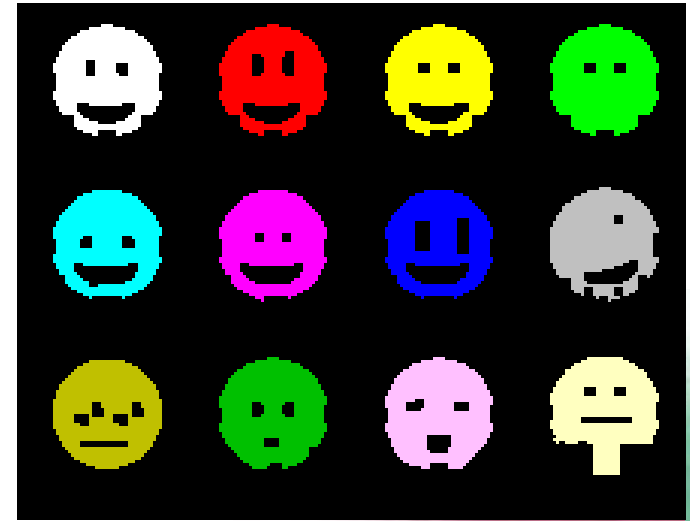
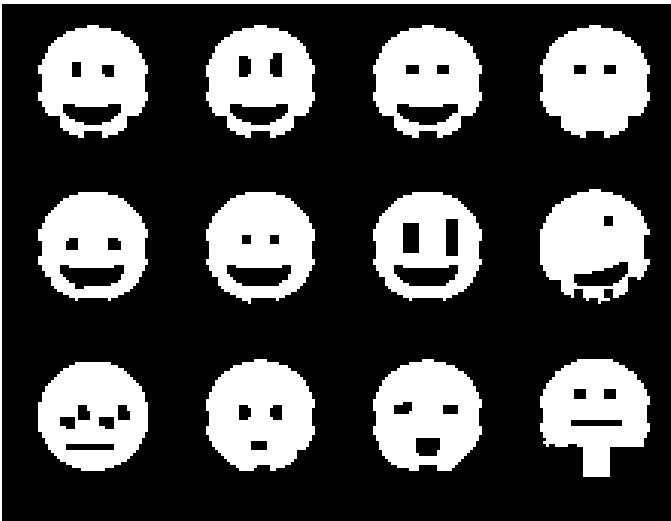
Prof. Dr. Jorge Centeno



# Componentes conexos

- Objetivo

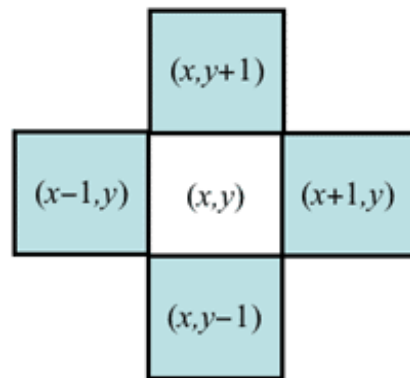
Dada uma imagem contendo regiões conexas de mesma cor, identificar cada grupo de pixels (conexos) como uma região única.



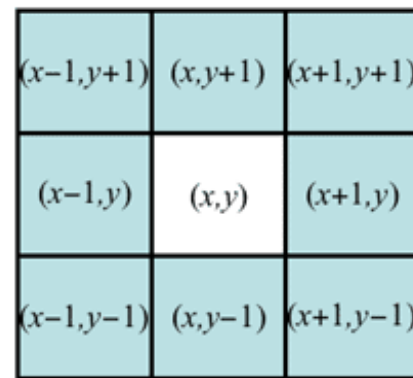


# Conectividade

- Definir Conceito de conectividade
- Quando podemos considerar que dois pixels são “vizinhos”?
- Verificar a conectividade 4 ou 8 vizinhos



Vizinhança 4



Vizinhança 8

Dois pixels estão conectados se eles satisfazem uma relação de adjacência

0	0	0	0	0	0	0	0	0
0		D	0	0				0
0			0	0	0	0		0
0	A	B	0	0	0	0		0
0	0	0	0	C		0	0	0
0	0	0	0			0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

A e B estão conectados

B e C não

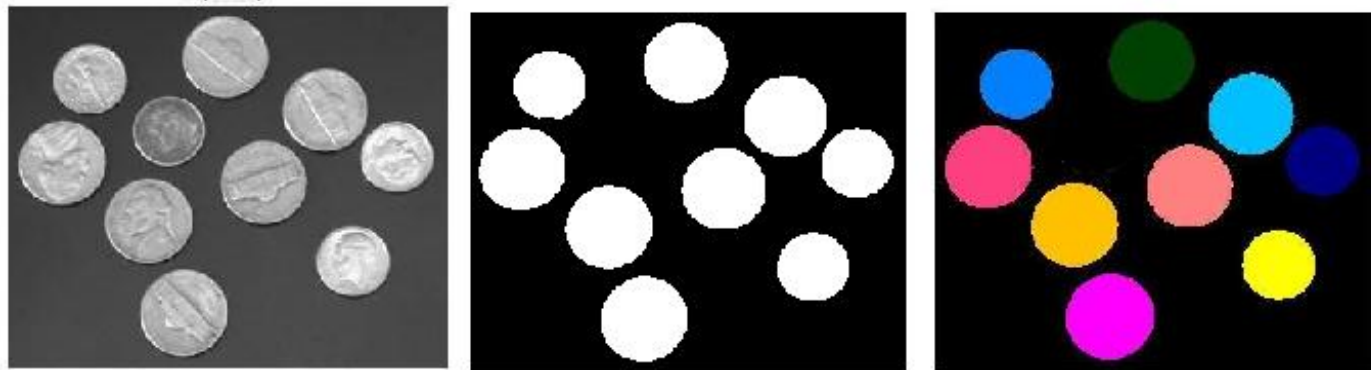
.



# Rotular pixels conexos

Problema: dada uma imagem com pixels rotulados com o mesmo valor e o fundo:

- Encontrar quantos grupos existem e
- Rotular todos os pixels de um mesmo grupo com o mesmo valor



# Algoritmo

1) Numerar todos os pixels “ativos” começando no canto superior esquerdo até o canto inferior direito.

2) Analisar a vizinhança dos pixels numerados:

- se existe um pixel “ativo” com valor menor que o valor do pixel central, o pixel central adota esse menor valor.

3) Repetir isto até não ocorrer mais mudanças

4) Rotular os grupos resultantes com base no histograma.

0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	1	0
0	1	1	1	0	1	0
0	0	1	0	0	1	0
0	0	0	0	1	1	0
0	0	0	0	0	0	0

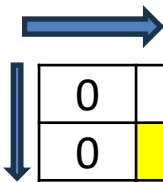
0	0	0	0	0	0	0
0	1	0	2	0	0	0
0	3	0	4	0	5	0
0	6	7	8	0	9	0
0	0	10	0	0	11	0
0	0	0	0	12	13	0
0	0	0	0	0	0	0

# Algoritmo

2) Analisar a vizinhança dos pixels numerados:

se existe um pixel “ativo” com valor menor que o valor do pixel central, o pixel central adota esse menor valor.

0	0	0	0	0	0	0
0	1	0	2	0	0	0
0	3	0	4	0	5	0
0	6	7	8	0	9	0
0	0	10	0	0	11	0
0	0	0	0	12	13	0
0	0	0	0	0	0	0



0	0	0	0	0	0	0
0	1	0	2	0	0	0
0	1	0	2	0	5	0
0	6	7	8	0	9	0
0	0	10	0	0	11	0
0	0	0	0	12	13	0
0	0	0	0	0	0	0

0	0	0	0	0	0	0
0	1	0	2	0	0	0
0	1	0	2	0	5	0
0	1	1	1	0	5	0
0	0	1	0	0	5	0
0	0	0	0	5	5	0
0	0	0	0	0	0	0

# Algoritmo

- 1) Numerar todos os pixels “ativos” começando no canto superior esquerdo até o canto inferior direito.
- 2) Analisar a vizinhança dos pixels numerados:  
se existe um pixel “ativo” com valor menor que o valor do pixel central, o pixel central adota esse menor valor.
- 3) Repetir isto até não ocorrer mais mudanças

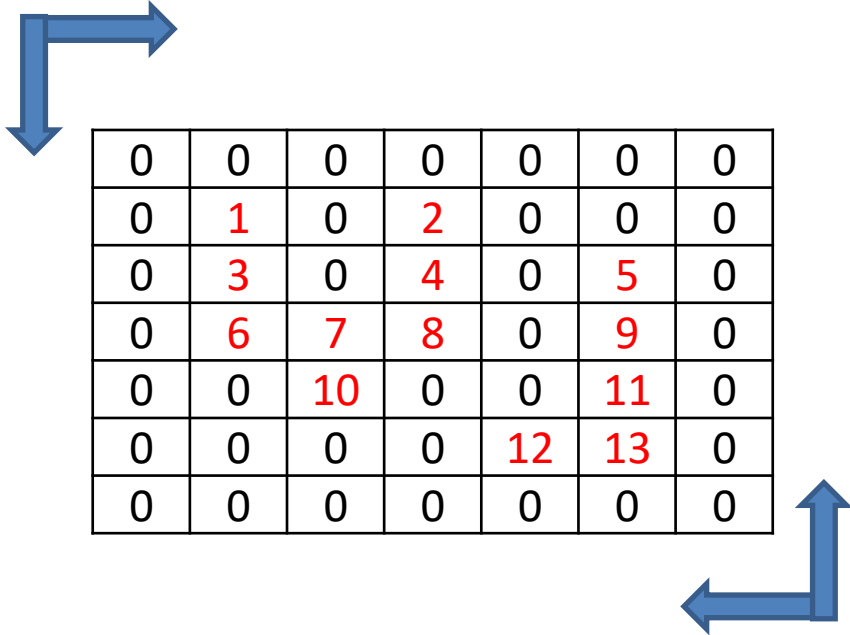
0	0	0	0	0	0	0
0	1	0	2	0	0	0
0	1	0	2	0	5	0
0	1	1	1	0	5	0
0	0	1	0	0	5	0
0	0	0	0	5	5	0
0	0	0	0	0	0	0

0	0	0	0	0	0	0
0	1	0	2	0	0	0
0	1	0	1	0	5	0
0	1	1	1	0	5	0
0	0	1	0	0	5	0
0	0	0	0	5	5	0
0	0	0	0	0	0	0

0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	5	0
0	1	1	1	0	5	0
0	0	1	0	0	5	0
0	0	0	0	5	5	0
0	0	0	0	0	0	0

# acelerar

- Alternando varreduras de :  
cima para baixo, esquerda para  
direita
  - Baixo para cima, direita para  
esquerda
- O processo pode ser mais rápido

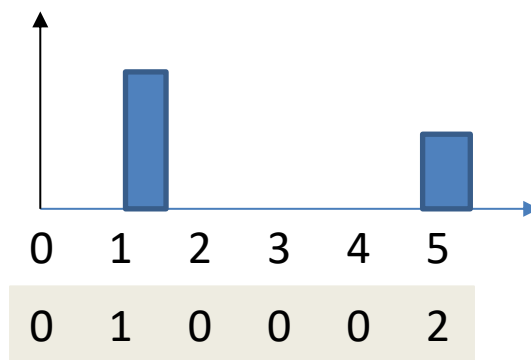


0	0	0	0	0	0	0
0	1	0	2	0	0	0
0	3	0	4	0	5	0
0	6	7	8	0	9	0
0	0	10	0	0	11	0
0	0	0	0	12	13	0
0	0	0	0	0	0	0

#### 4) Rotular os grupos resultantes com base no histograma.

- Calcular o Histograma
- Numerar apenas as posições do histograma que possuem valores acima de zero.
- Aplicar esta nova tabela para mudar os números na imagem

0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	5	0
0	1	1	1	0	5	0
0	0	1	0	0	5	0
0	0	0	0	5	5	0
0	0	0	0	0	0	0



0	0	0	0	0	0	0
0	1	0	1	0	0	0
0	1	0	1	0	2	0
0	1	1	1	0	2	0
0	0	1	0	0	2	0
0	0	0	0	2	2	0
0	0	0	0	0	0	0