



UAI

UNIVERSIDAD ADOLFO IBÁÑEZ
FACULTAD DE INGENIERÍA Y CIENCIAS



iUAI
UNIVERSIDAD ADOLFO IBÁÑEZ
FACULTAD DE INGENIERIA Y CIENCIAS

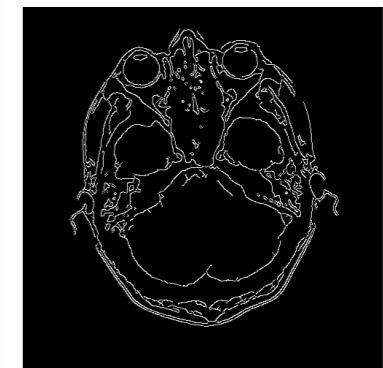
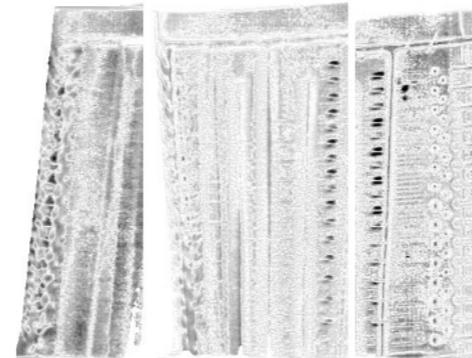
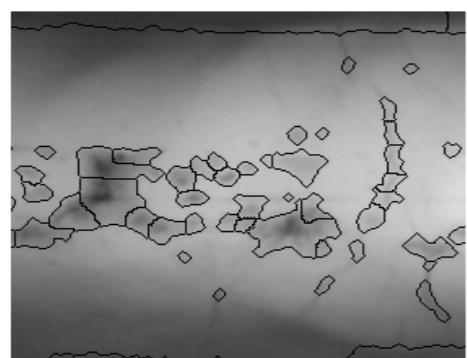
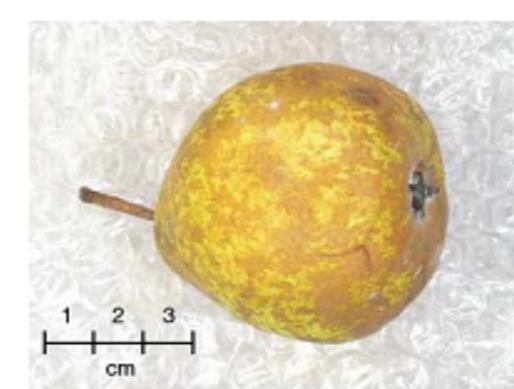
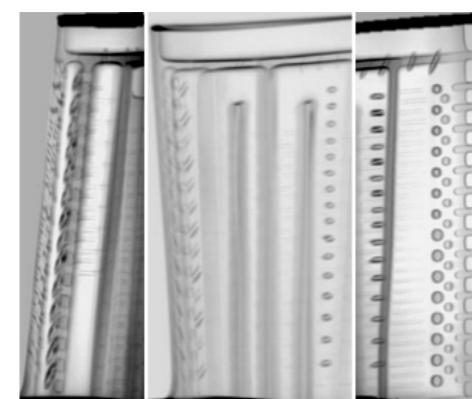
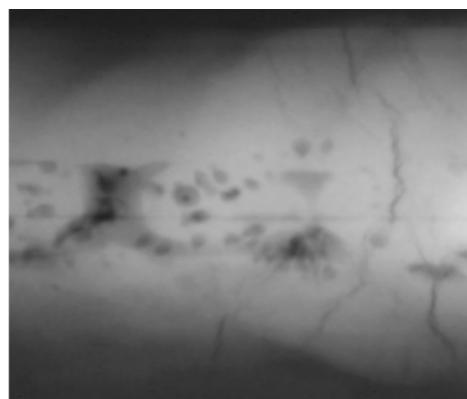
MDS²⁰¹⁹ PROCESAMIENTO DIGITAL DE IMÁGENES ALGORITMOS DE SEGMENTACIÓN

Miguel Carrasco
miguel.carrasco@uai.cl
1er Semestre 2020

- Segmentación
 - Conceptos
 - Clasificación

▶ Concepto

- La segmentación consiste en la partición/división de regiones es múltiples partes en zonas homogéneas, normalmente inconexas. El objetivo es generar una nueva representación que permita un análisis o inferencia de las regiones contenidas.
- Normalmente, a través de la segmentación sepáramos objetos de interés encontrando los bordes. Una vez segmentados podemos efectuar un análisis de estos a través de sus propiedades (area, forma, perímetro, etc).



Industria

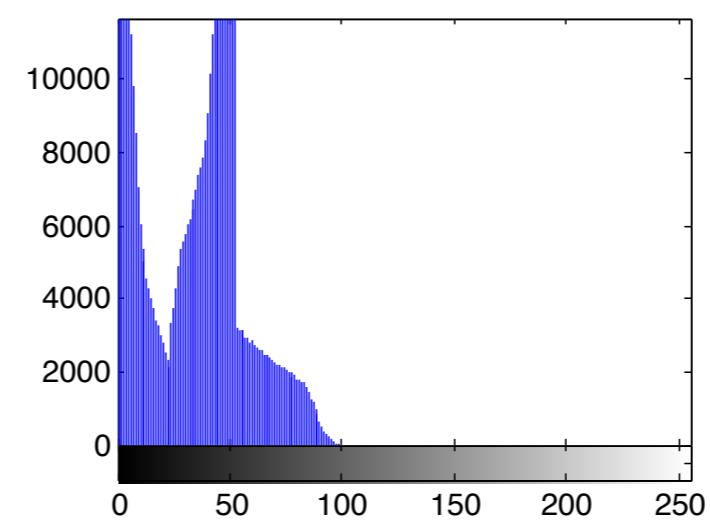
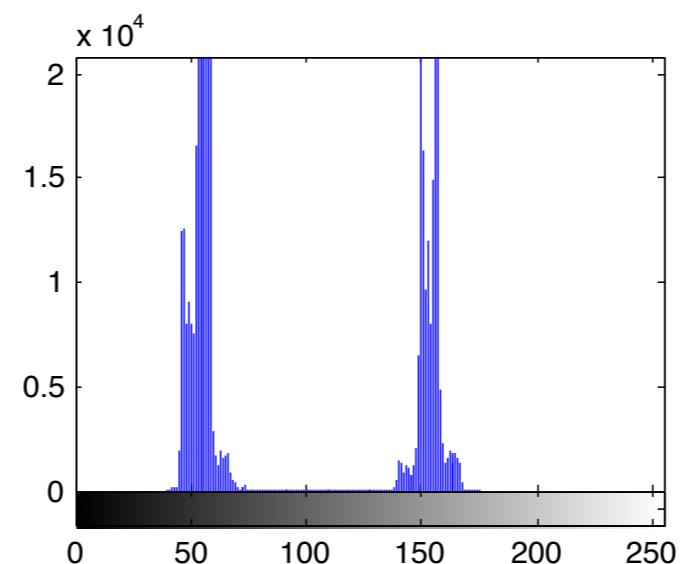
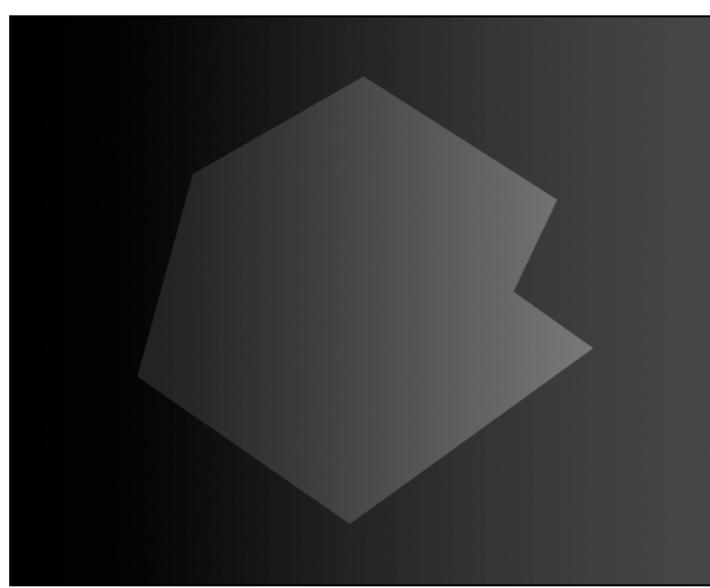
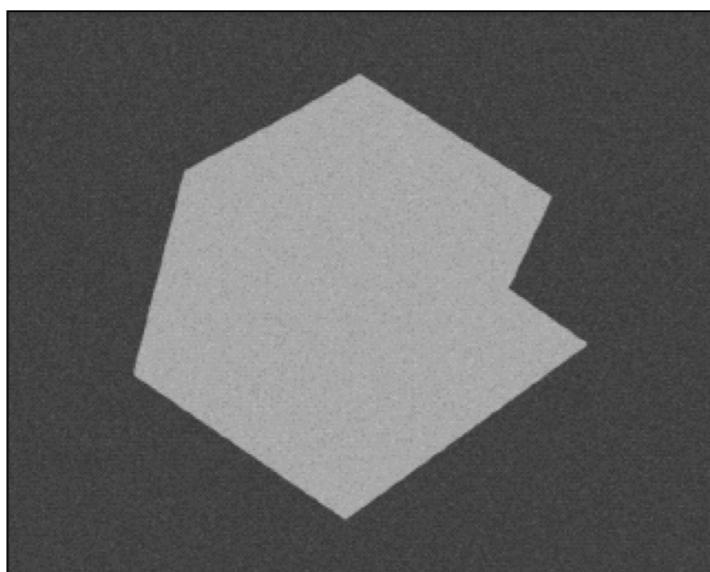
Turbinas

Alimentos

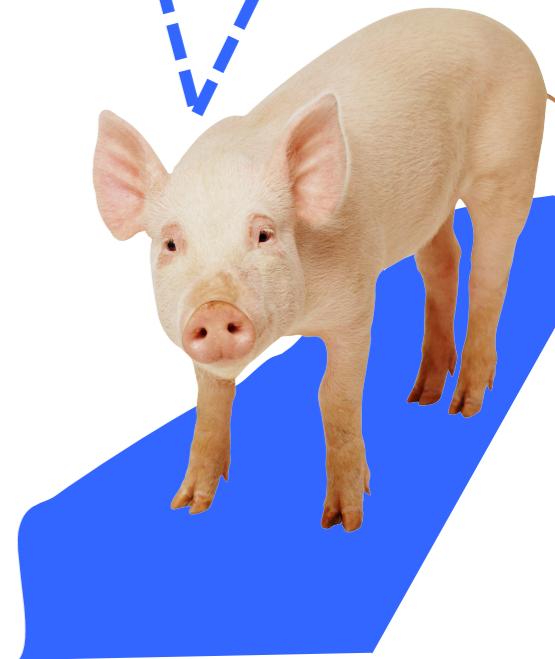
Medicina

▶ Iluminación

- La **iluminación** tiene un rol fundamental en la calidad de la segmentación facilitando o perjudicando la calidad de la separabilidad entre regiones.



¿Cuál histograma es más separable?
oink



- Segmentación
 - Conceptos
 - Clasificación

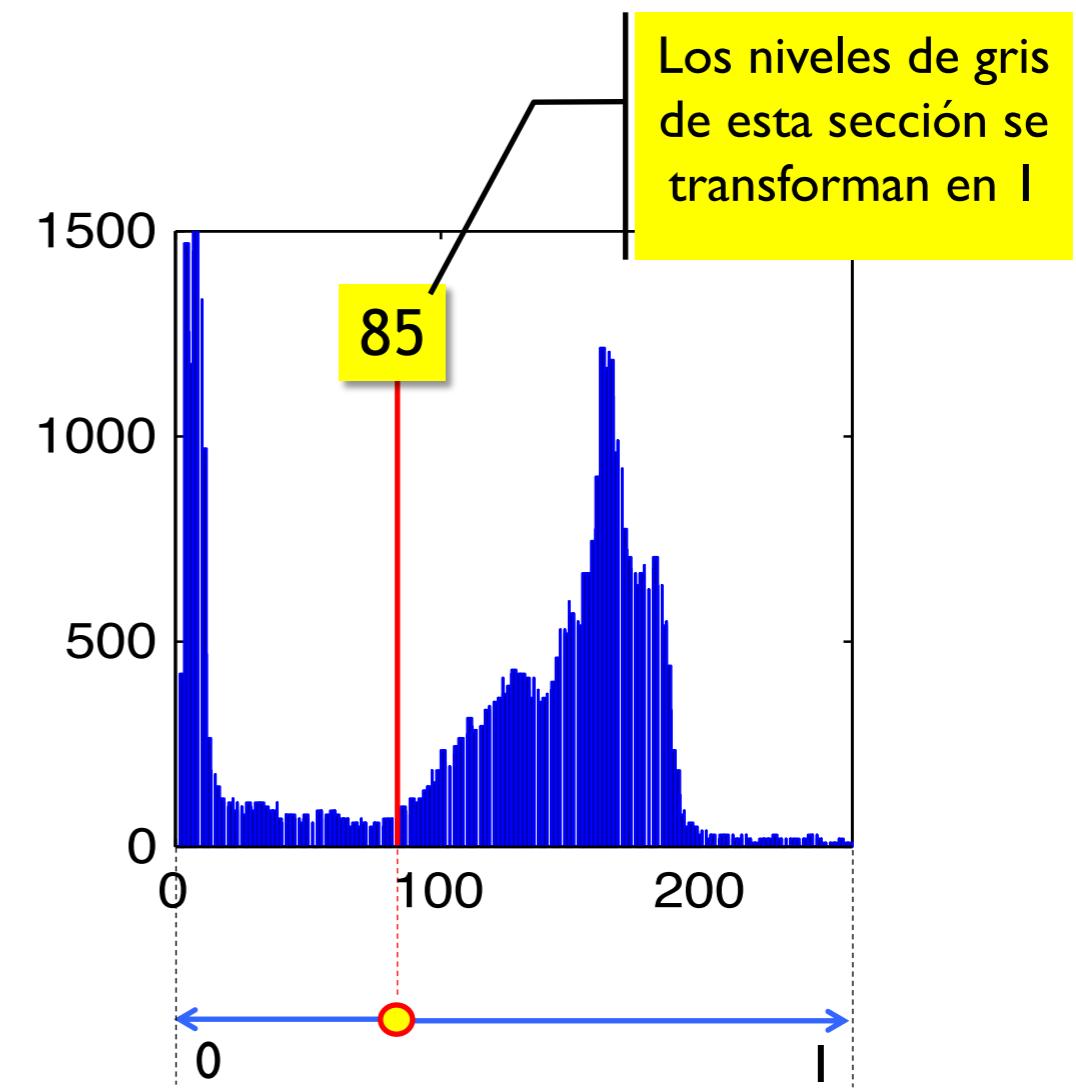
▶ Clasificación

Umbrales

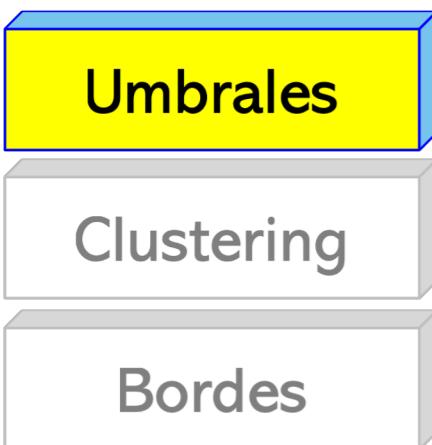
Clustering

Bordes

- Es la técnica más simple de segmentación. La idea consiste en determinar un nivel o umbral que separe los niveles de gris de la imagen en 1 o 0.



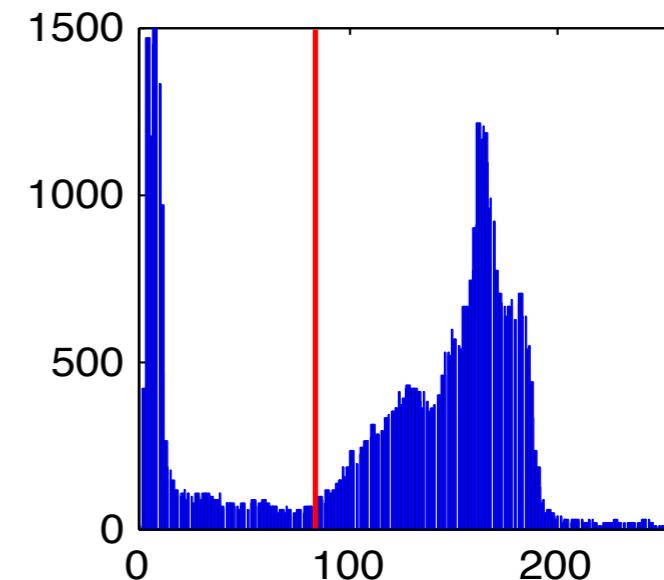
▶ Clasificación



- Es la técnica más simple de segmentación. La idea consiste en determinar un nivel o umbral que separe los niveles de gris de la imagen en 1 o 0.

Problemas

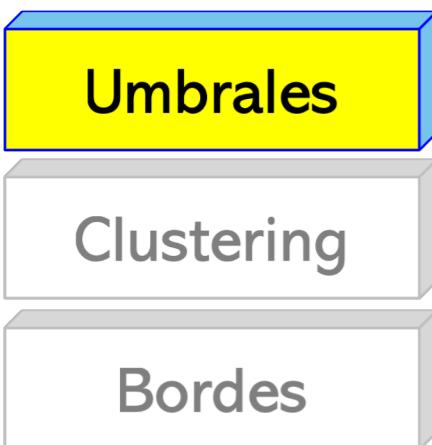
- ¿Cómo elegir un umbral óptimo?
- ¿Cómo evitar el ruido?
- ¿Cómo unir regiones que han sido erróneamente separadas?
- ¿Cómo separar regiones que han sido erróneamente unidas?



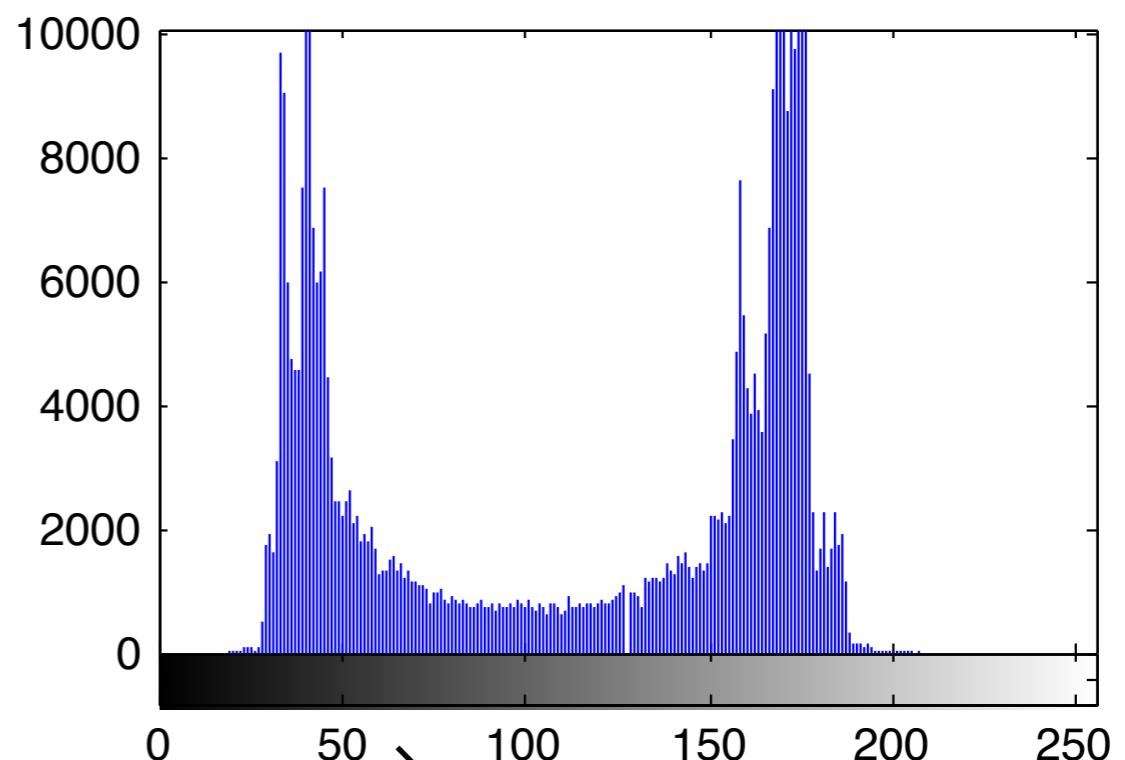
OBJETIVO



▶ Clasificación



- **Binarización Global.** Es una técnica de segmentación automática que permite determinar el valor de intensidad óptimo cuando existen dos niveles de distribución separables



¿Qué nivel de gris debemos emplear?

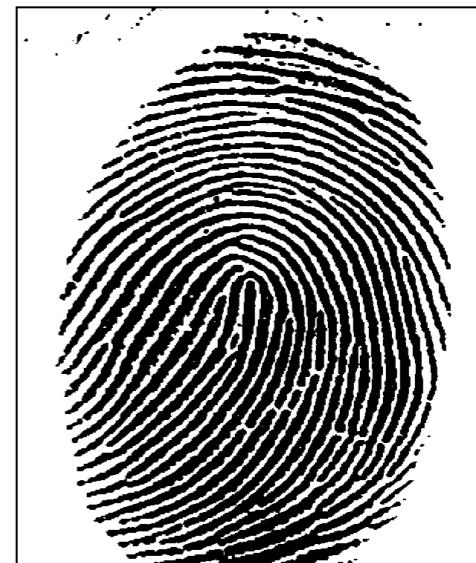
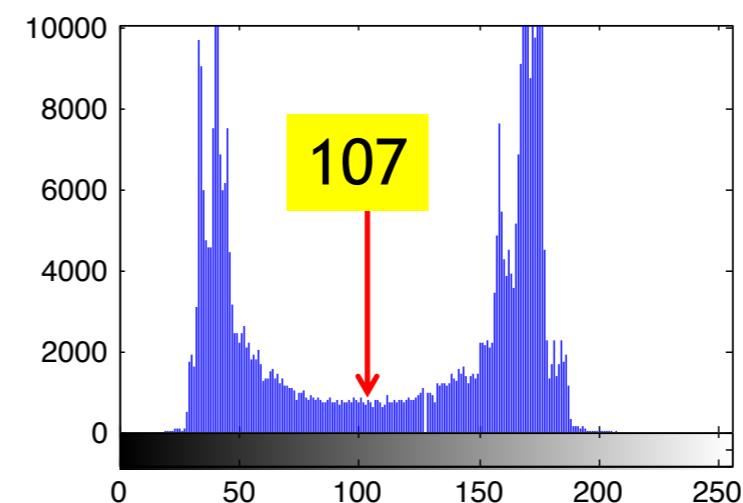
▶ Clasificación

Umbráles

Clustering

Bordes

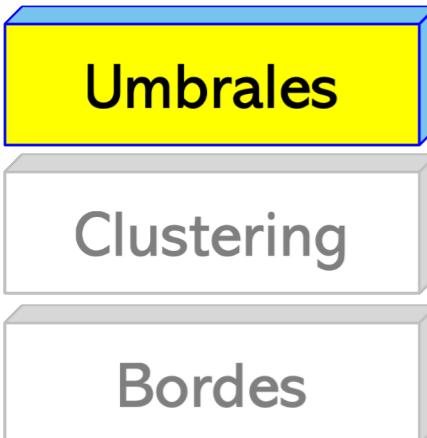
- **Binarización Global.** Es una técnica de segmentación automática que permite determinar el valor de intensidad óptimo cuando existen dos niveles de distribución separables



- **Algoritmo:**

- 1) Sea N el valor de gris inicial del algoritmo (Suponga el valor medio)
- 2) Repetir
 - (a) Sea G_1 píxeles mayores a N y G_2 píxeles menores al valor N .
 - (b) Calcular μ_1 y μ_2 los promedios de G_1 y G_2 respectivamente
 - (c) Determinar N como: $N = (\mu_1 + \mu_2)/2$

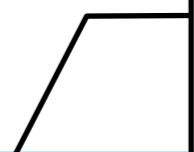
▶ Clasificación



- **Binarización Global.** Es una técnica de segmentación automática que permite determinar el valor de intensidad óptimo cuando existen dos niveles de distribución separables
- **Algoritmo:**

```
def umbral_global(A, level, iter):  
  
    for i in range(0,iter):  
        id1 = A<level  
        id2 = A>=level  
        mu1=np.mean(A[id1])  
        mu2=np.mean(A[id2])  
        level= np.round((mu1+mu2)/2)  
  
    return level
```

Esta función recibe la imagen, un nivel inicial y el número de iteraciones



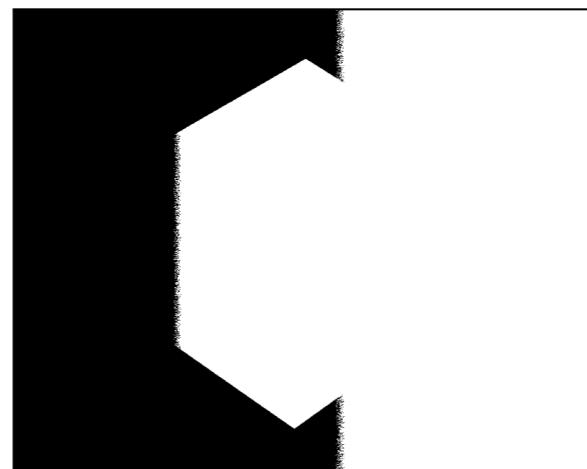
▶ Clasificación

Umbrales

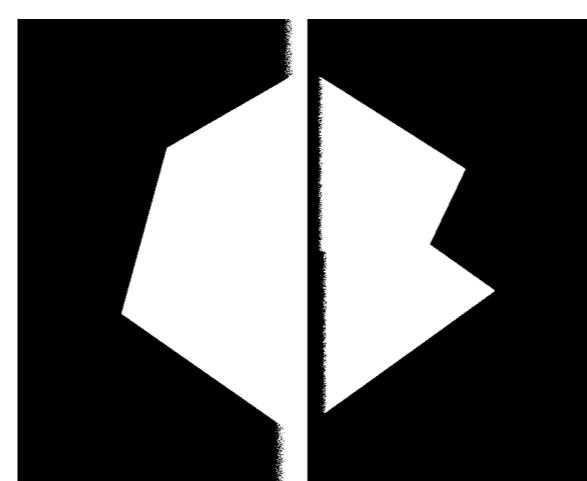
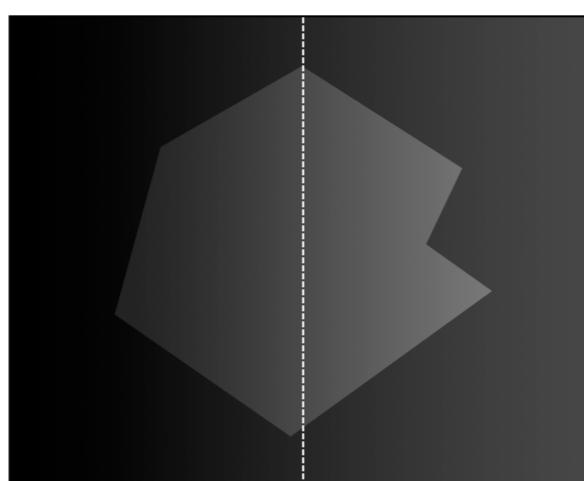
Clustering

Bordes

- **Binarización Adaptiva.** Esta técnica consiste en **dividir la imagen en subcuadrantes**. De esta forma los niveles de los cuadrantes que tengan un menor nivel de varianza son tomados como fondo.



Usando
umbral global



Usando
umbral
adaptivo

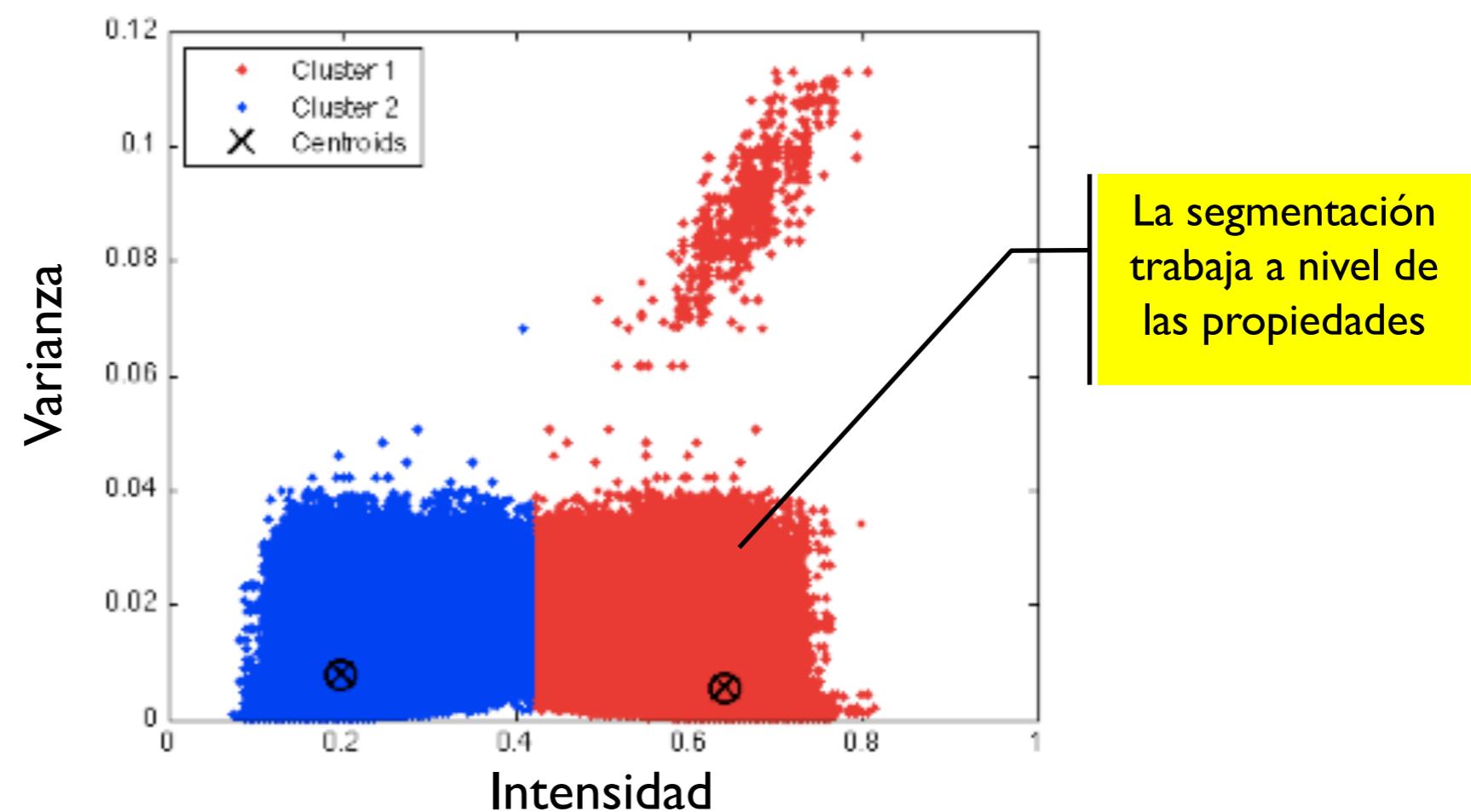
▶ Clasificación

Umbral

Clustering

Bordes

- Esta técnica busca datos que puedan agruparse según alguna propiedad. Por ejemplo, el color, intensidad, textura, o una combinación de ellos. Un algoritmo normalmente empleado para esto es K-Means



▶ Clasificación

Umbral

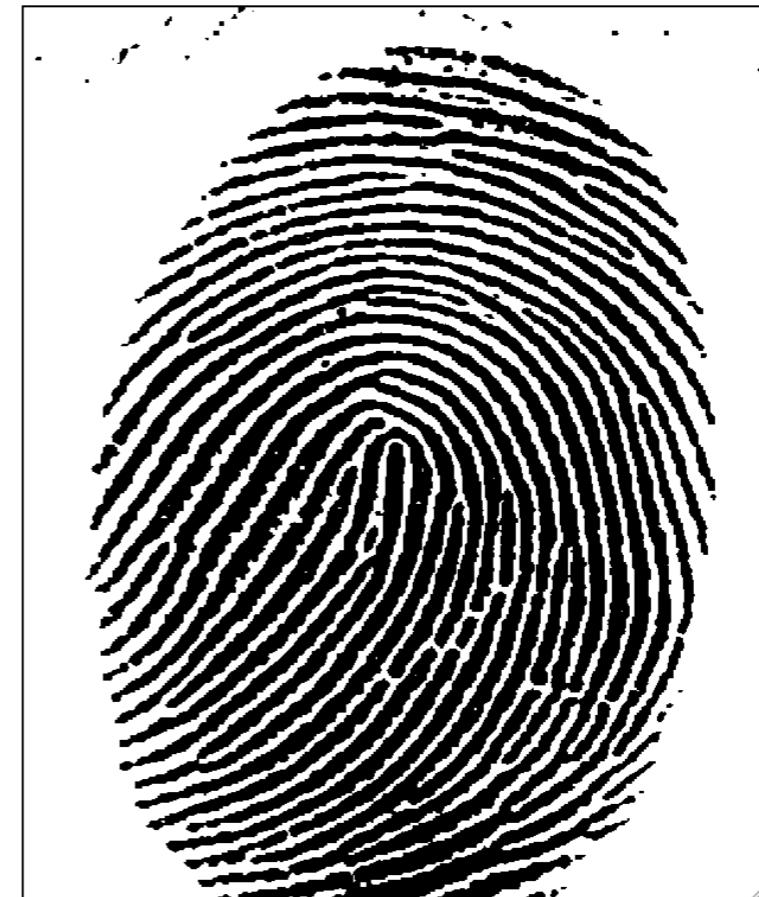
Clustering

Bordes

- Esta técnica busca datos que puedan agruparse según alguna propiedad. Por ejemplo, el color, intensidad, textura, o una combinación de ellos.
- Un algoritmo normalmente empleado para esto es K-Means

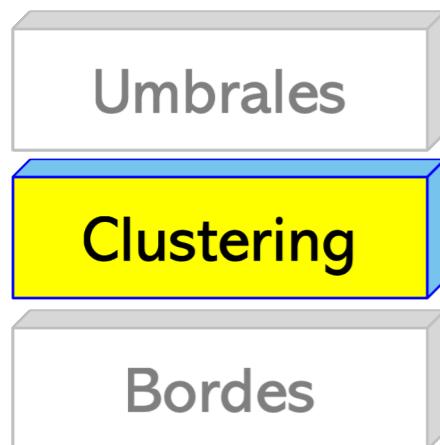


Original



Segmentada con Kmeans

▶ Clasificación



- Esta técnica busca datos que puedan agruparse según alguna propiedad. Por ejemplo, el color, intensidad, textura, o una combinación de ellos.
- Un algoritmo normalmente empleado para esto es K-Means
- **Algoritmo:**
 - ① Asignar una ubicación arbitraria para los centros de gravedad de cada cluster.
 - ② Repetir hasta convergencia
 - ① Calcular la distancia Euclídea entre cada uno de los puntos y los centros de gravedad
 - ② Asignar a cada uno de los puntos del espacio la clase cuya distancia euclídea es mínima
 - ③ Recalcular la ubicación de los centros de gravedad



▶ Clasificación

Umbral

Clustering

Bordes

- Esta técnica busca datos que puedan agruparse según alguna propiedad. Por ejemplo, el color, intensidad, textura, o una combinación de ellos.
- Un algoritmo normalmente empleado para esto es K-Means

```
img = cv2.imread('rombo.png')
gris = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
rbo = filters.roberts(gris)

std_f= ndi.generic_filter(rbo,filtro_std, [3,3])
std_m= ndi.generic_filter(rbo,filtro_mean, [3,3])

subdata = np.matrix([std_f.flatten() , std_m.flatten()])
subdata = np.transpose(subdata)

km = KMeans(n_clusters=2)
km = km.fit(subdata)
```

La función kmeans se encuentra implementada en sklearn, y posee muchas opciones.

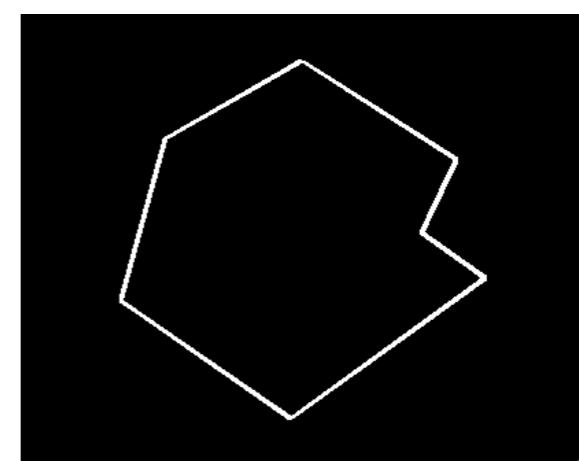
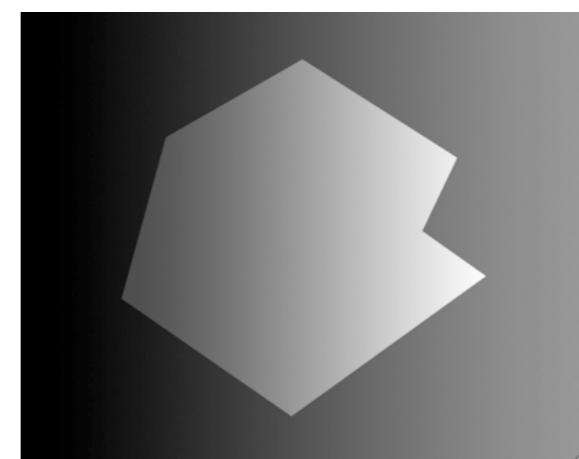
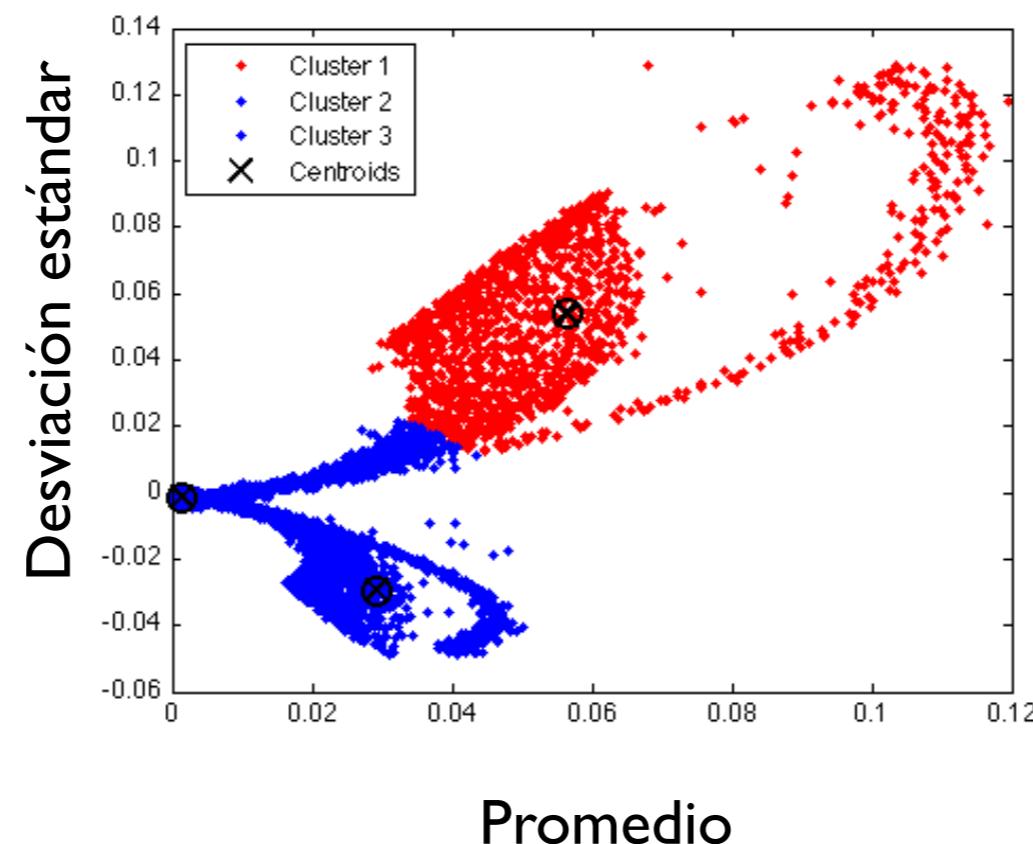
▶ Clasificación

Umbral

Clustering

Bordes

- Una de las ventajas principales es que podemos emplear más centroides. De esta forma es posible segmentar regiones más complejas.



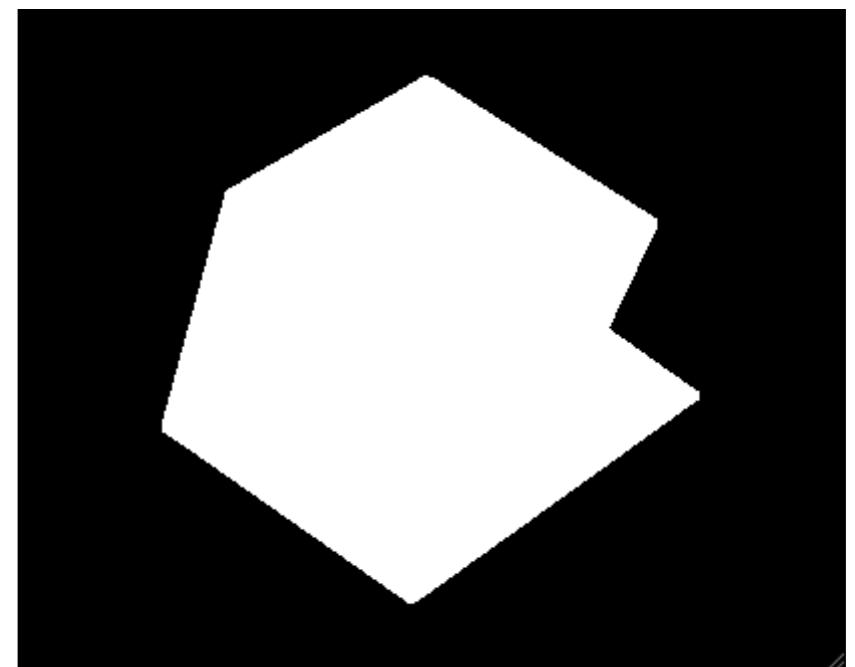
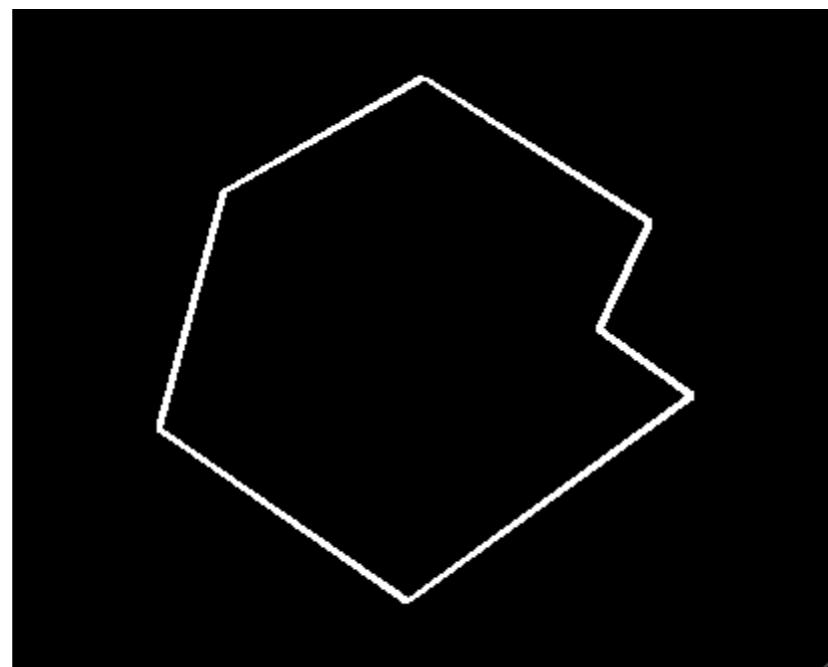
▶ Clasificación

Umbral

Clustering

Bordes

- Una de las ventajas principales es que podemos emplear más centroides. De esta forma es posible segmentar regiones más complejas.
- Posteriormente con el comando `imfill` es posible llenar las regiones internas



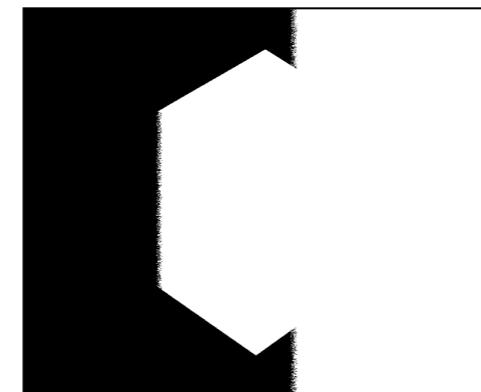
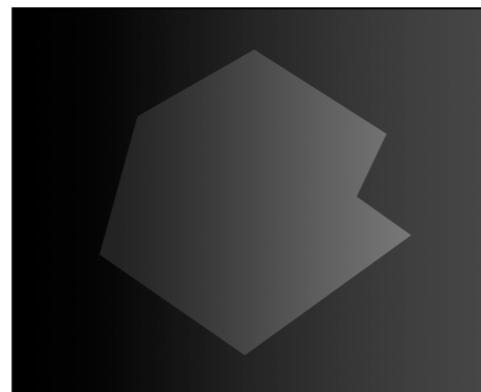
▶ Clasificación

Umbrales

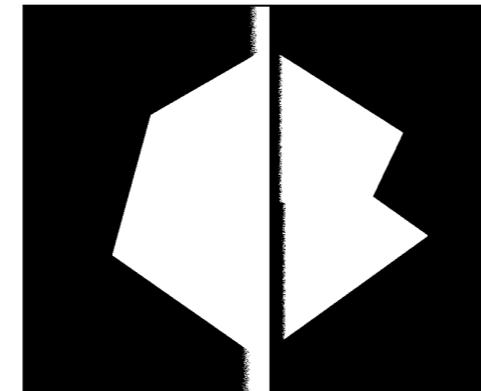
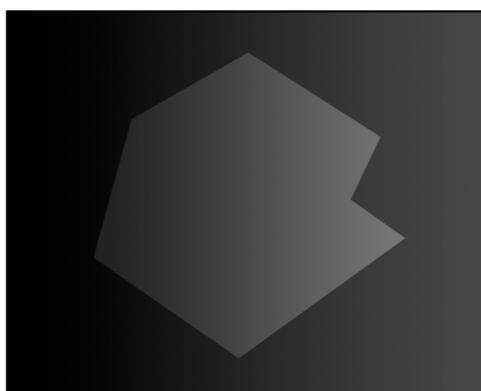
Clustering

Bordes

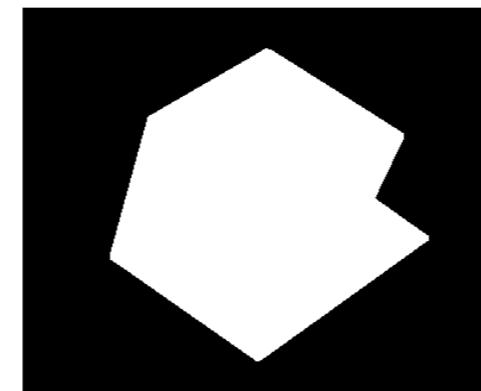
- En resumen, vemos que esta técnica super largamente las anteriores al permitir un mayor control sobre las regiones



Usando
umbral global

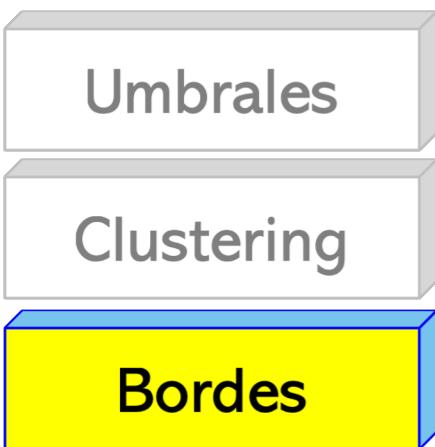


Usando
umbral
adaptivo



Usando
clustering más
relleno

▶ Clasificación



- Los filtros de detección de bordes realizan operaciones diferenciales (con máscaras) o operaciones más complejas sobre los niveles de gris. Dentro de los principales algoritmos de detección de bordes se encuentran:

- ✓ Sobel
- ✓ Prewitt
- ✓ Roberts
- ✓ LoG
- ✓ Canny
- ✓ Watershed

Implementados
por OpenCV



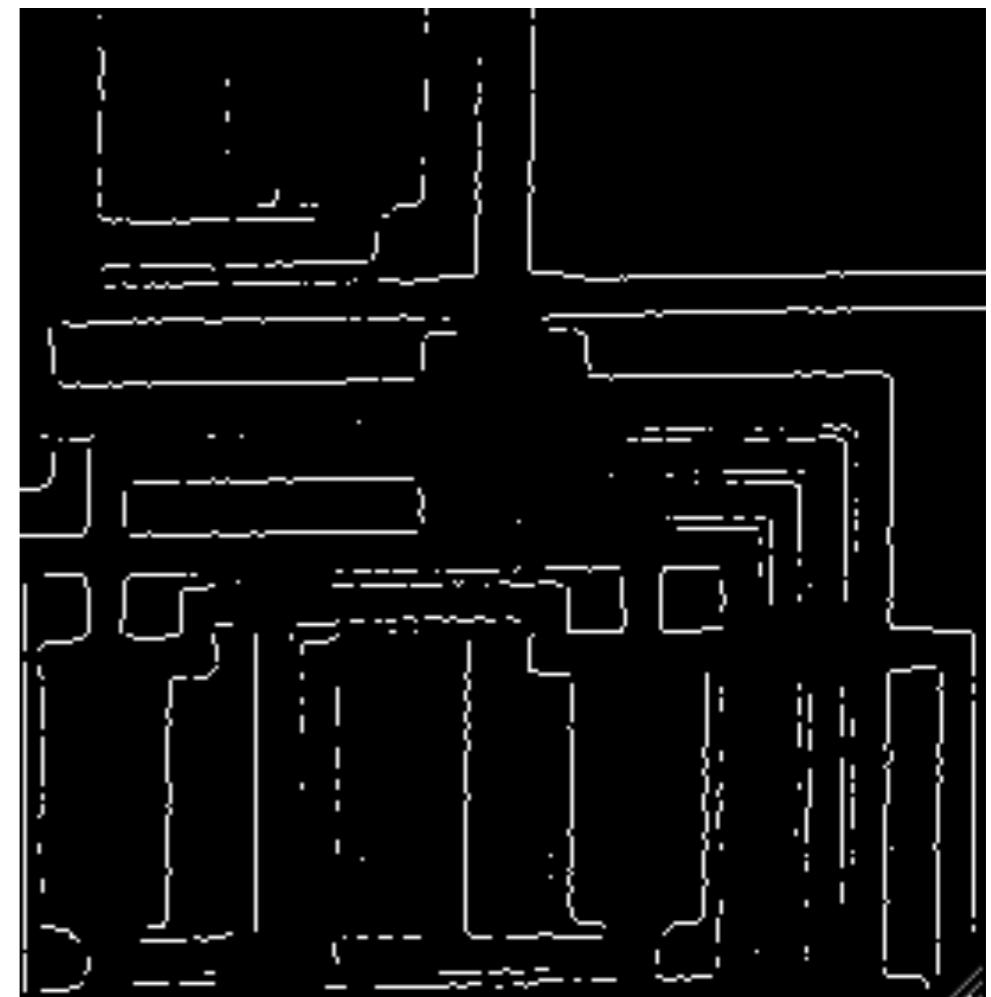
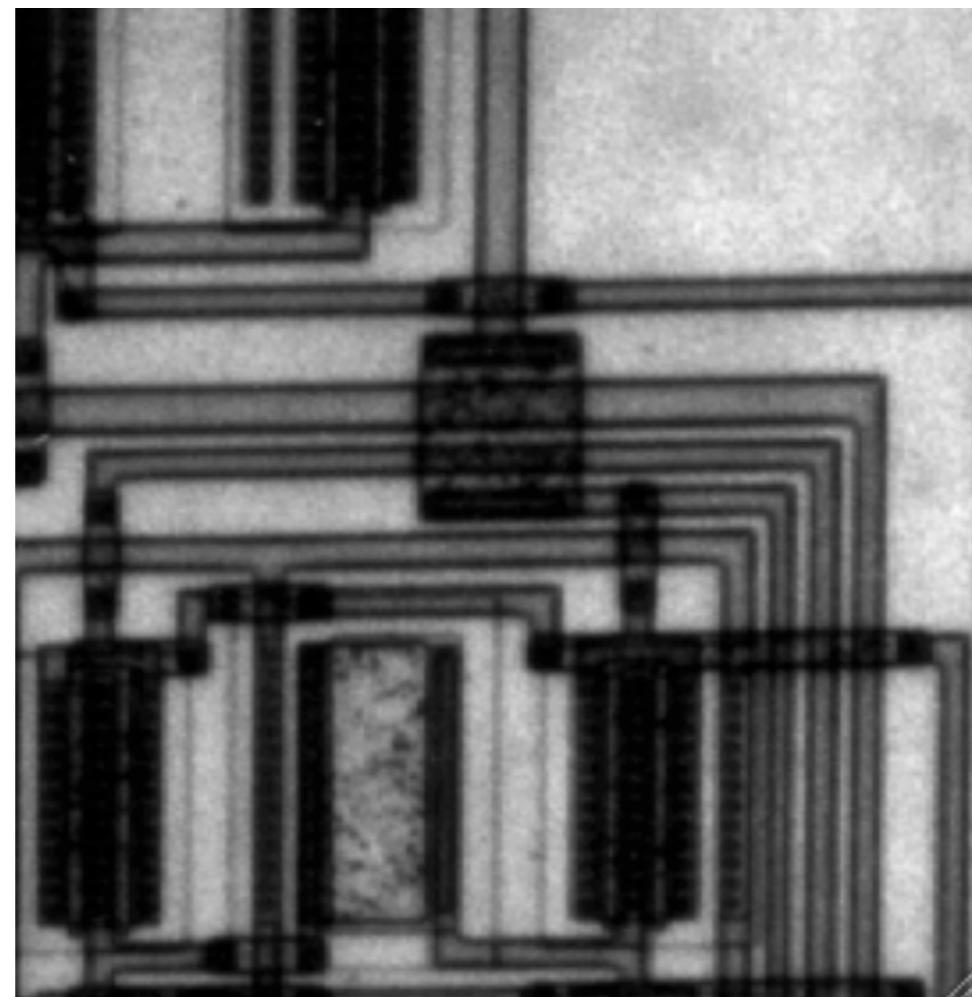
▶ Clasificación

- **Sobel.** El operador de sobel utiliza dos máscaras direccionales de 3x3 que recorre la imagen a través de una operación en bloques.

Umbralas

Clustering

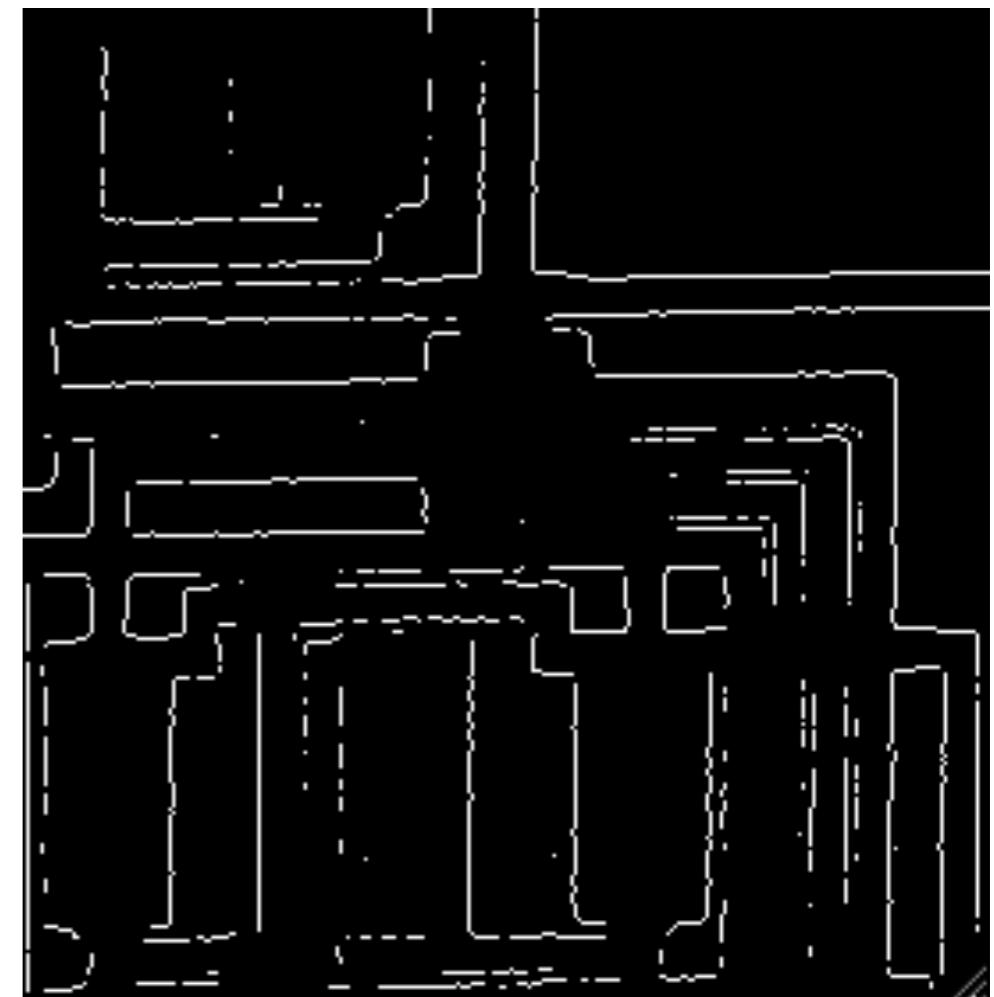
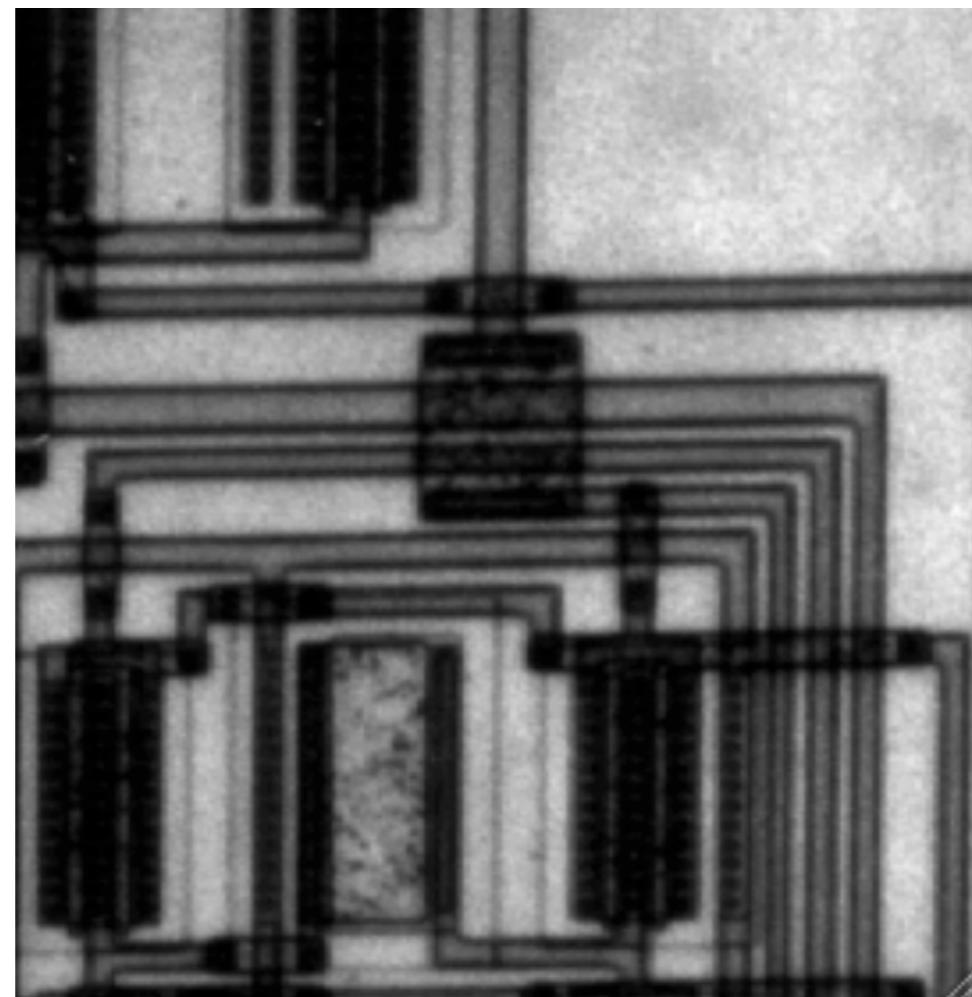
Bordes



▶ Clasificación

- **Prewitt.** El operador prewitt utiliza dos máscaras direccionales de 3x3 que recorre la imagen a través de una operación en bloques.

Umbral
Clustering
Bordes



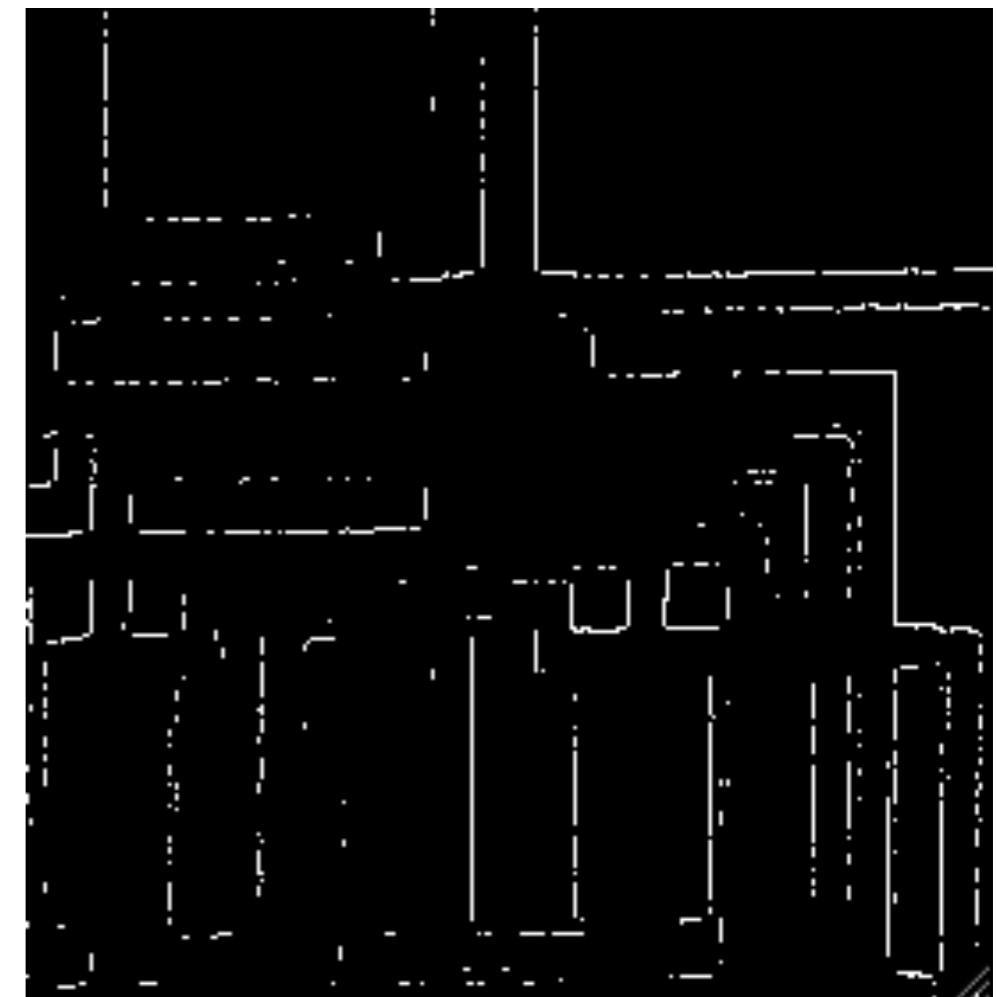
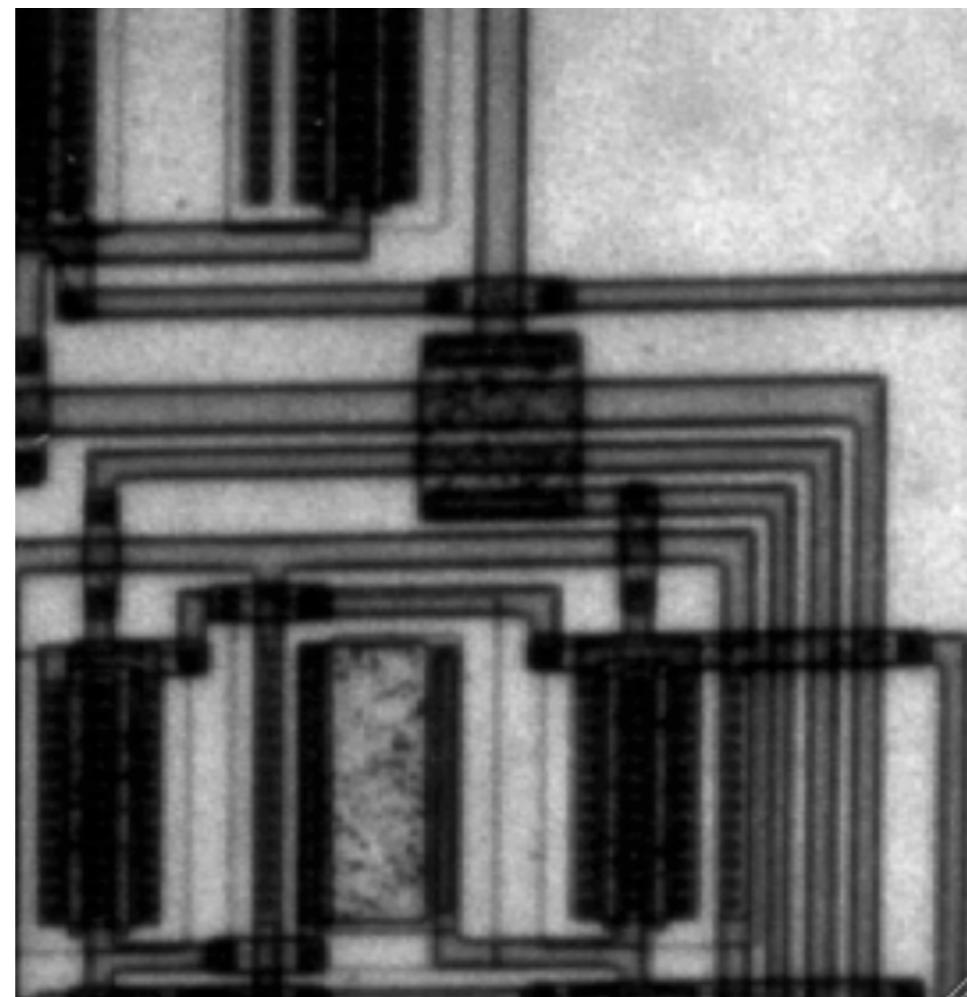
▶ Clasificación

- **Roberts.** El operador de sobel utiliza dos máscaras direccionales de 2x2 que recorre la imagen a través de una operación en bloques.

Umbralas

Clustering

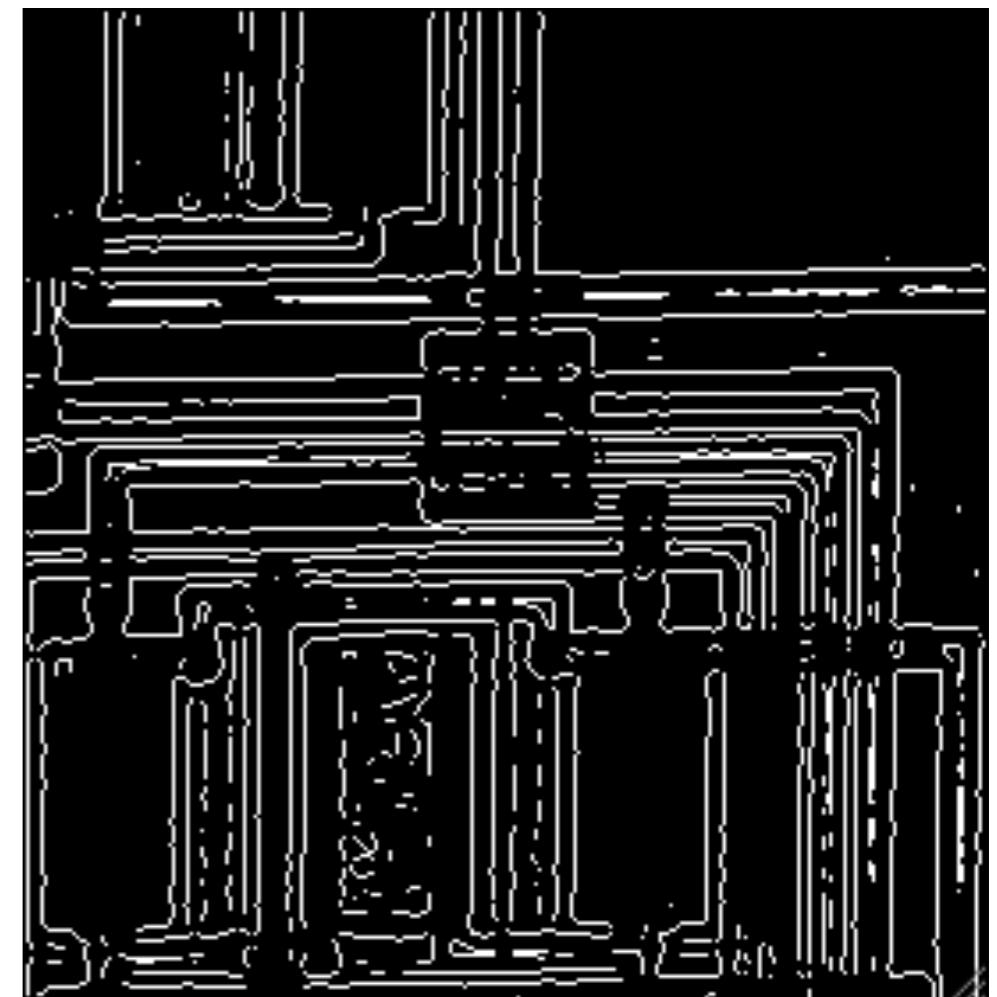
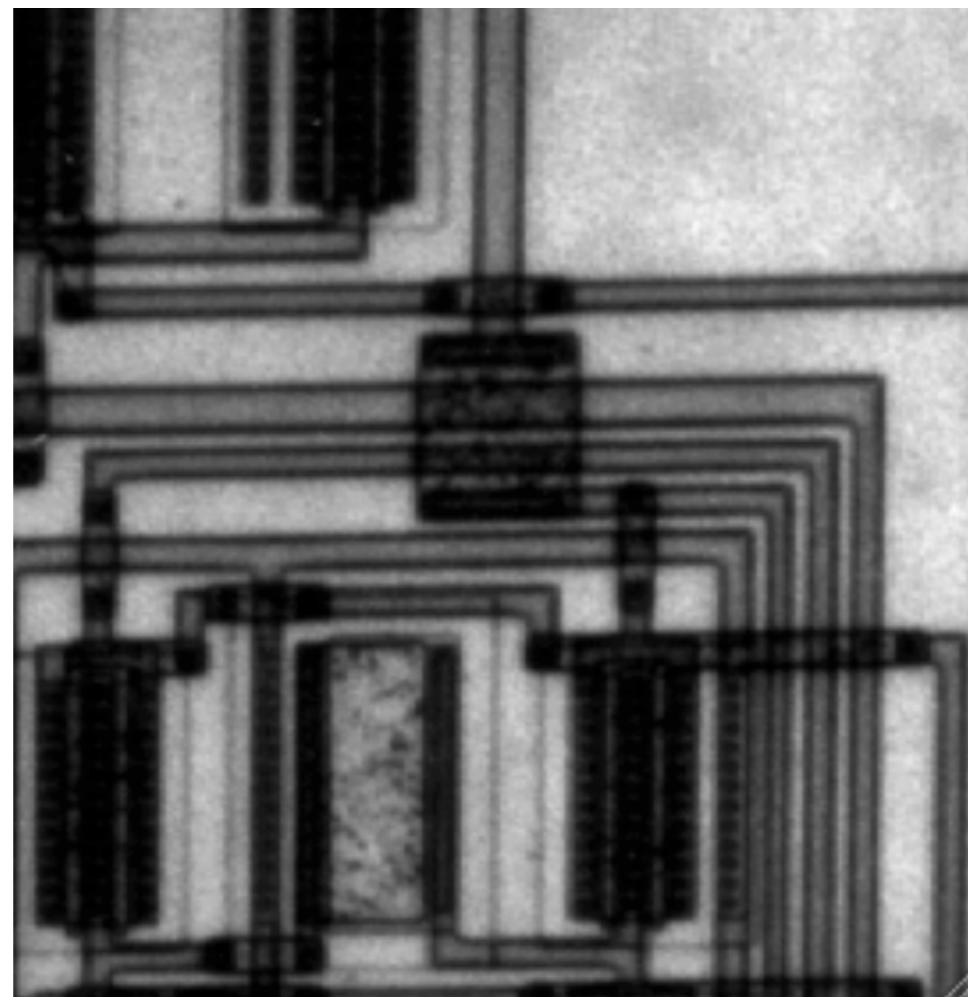
Bordes



▶ Clasificación

- Umbrales
- Clustering
- **Bordes**

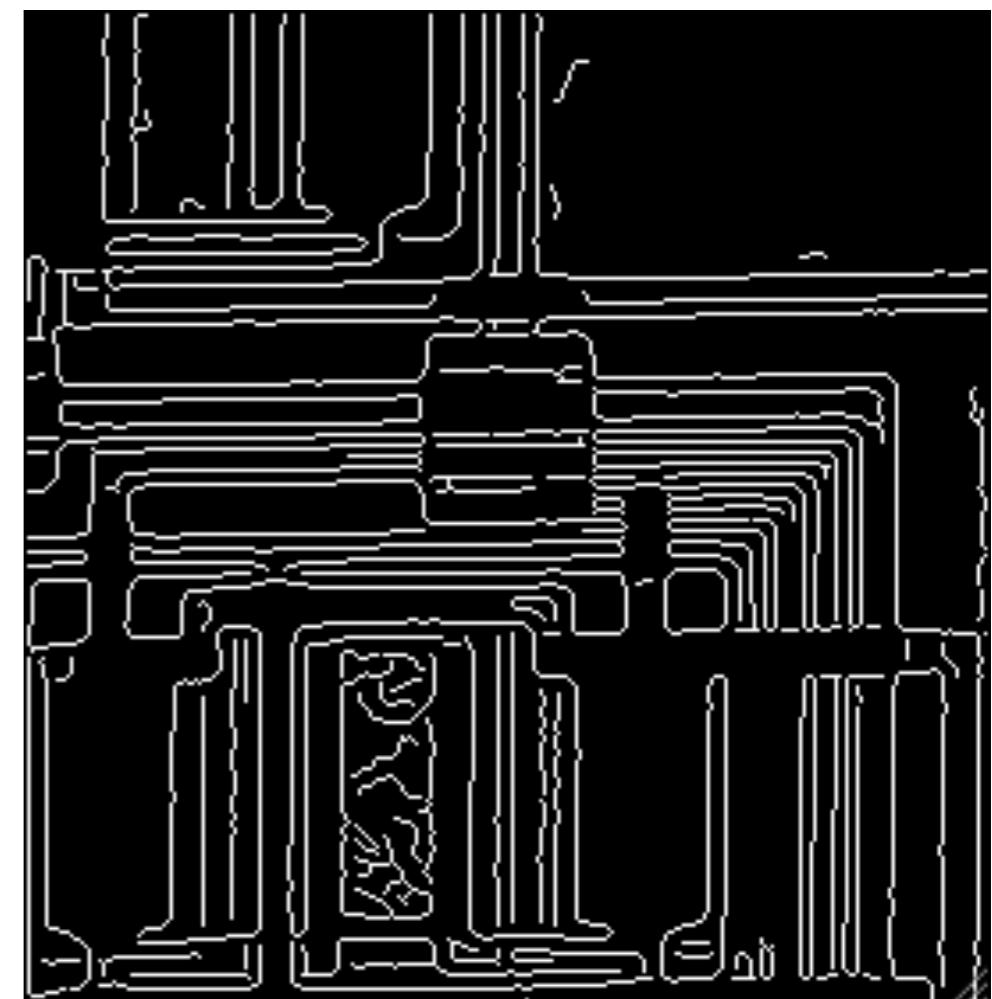
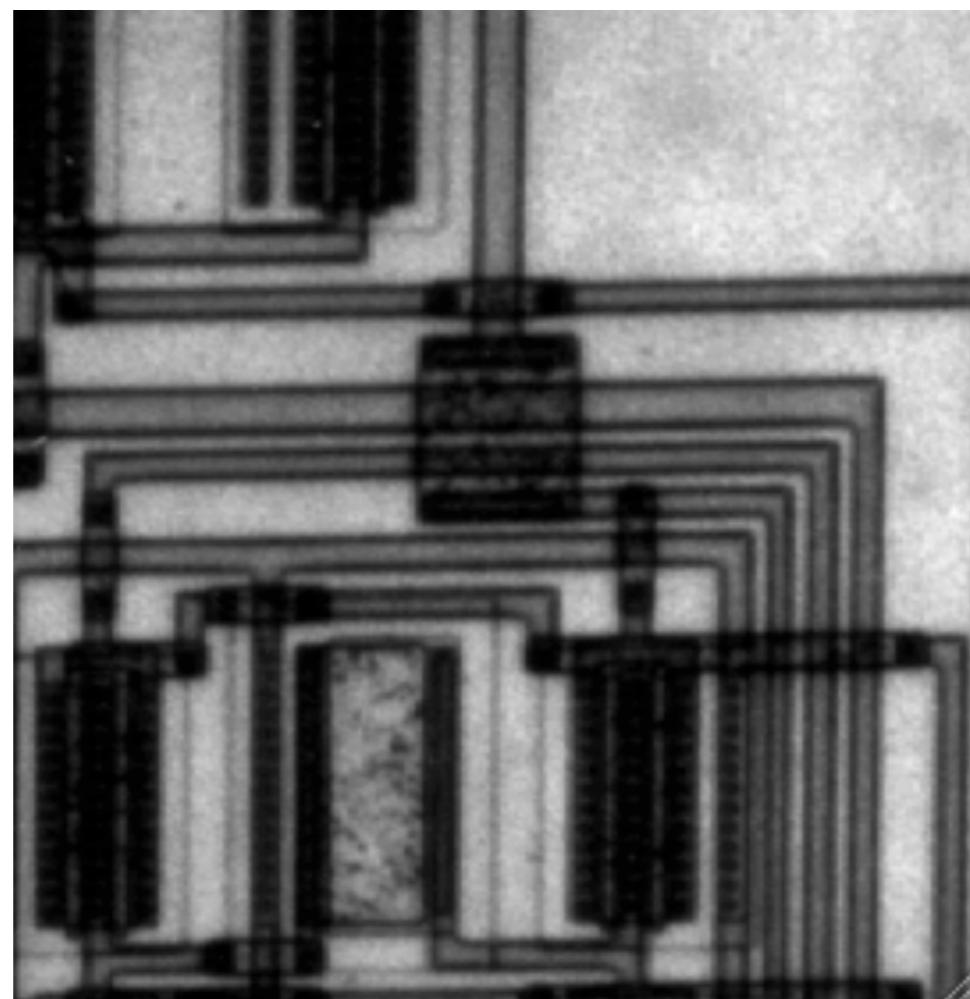
- **LoG.** Significa Laplacian of Gaussian es un operador que primero aplica una gaussiana y luego aplica el filtro laplaciano con una máscara de 3x3



▶ Clasificación

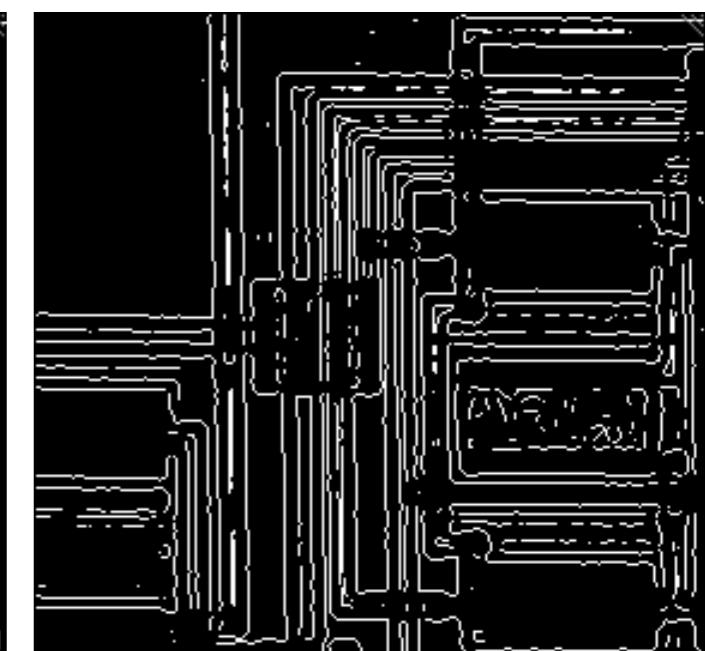
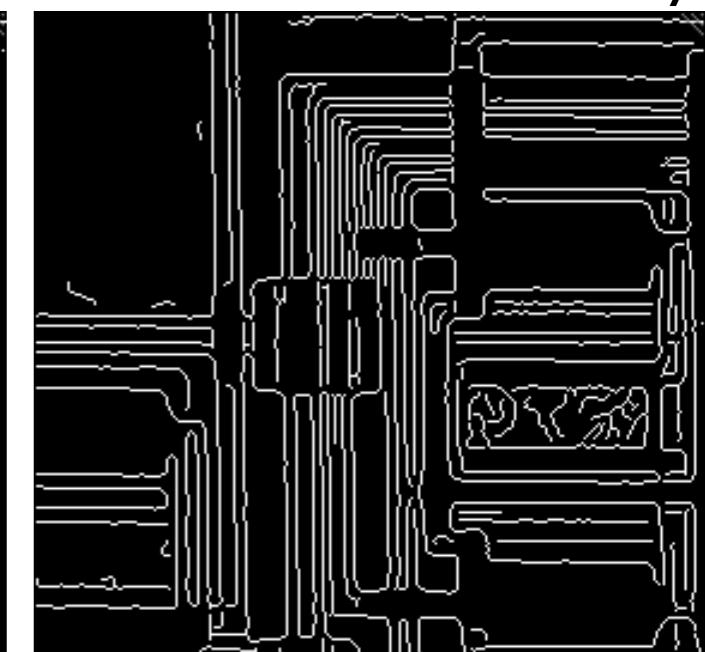
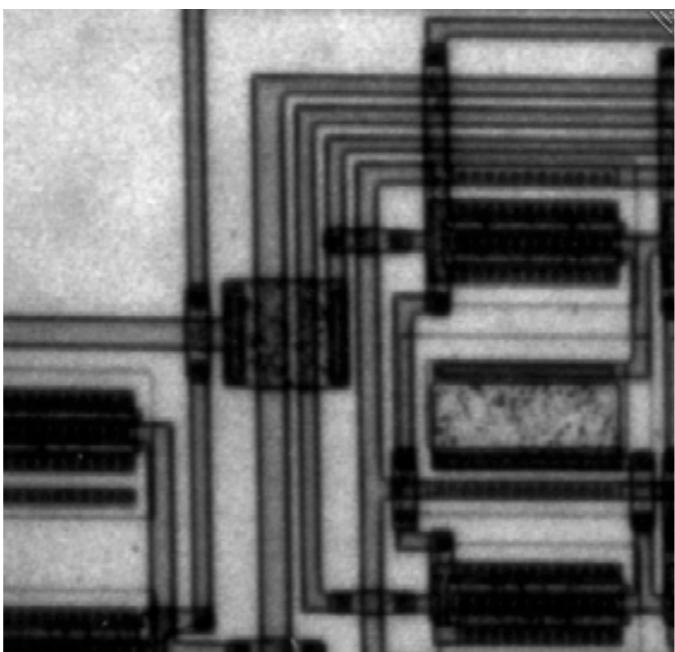
- Umbralas
- Clustering
- Bordes

- **Canny.** El operador de Canny es el más complejo ya que encuentra los bordas buscando el máximo local del gradiente en un proceso denominado Histérisis.



▶ Clasificación

Umbrales
Clustering
Bordes



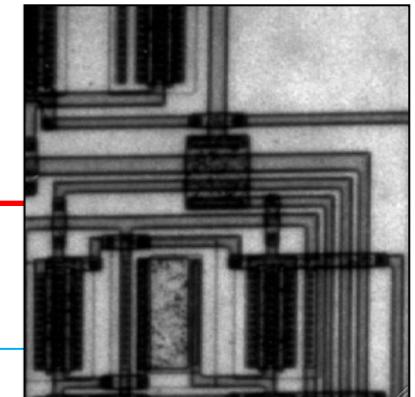
▶ Clasificación

Umbral

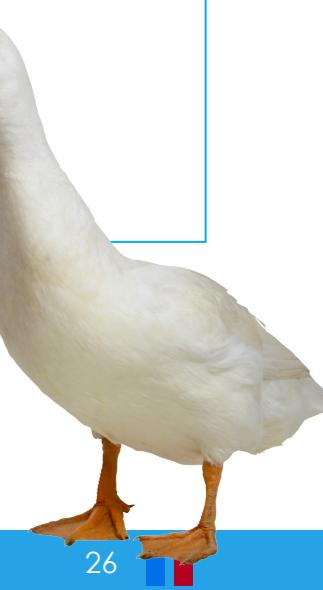
Clustering

Bordes

```
img = cv2.imread('circuits.png')  
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  
  
#binarizacion  
ret, thresh = cv2.threshold(gray,0,255, cv2.THRESH_OTSU)  
  
# Segmentacion con watershed  
ret, markers = cv2.connectedComponents(thresh)  
markers = cv2.watershed(img,markers)  
watershed = np.zeros(gray.shape)  
watershed[markers == -1] = [255]  
  
roberts = filters.roberts(gray)  
prewitt = filters.prewitt(gray)  
sobel = filters.sobel(gray)  
sato = filters.sato(gray)  
canny = feature.canny(gray, sigma=2)
```



Código para
los amigos



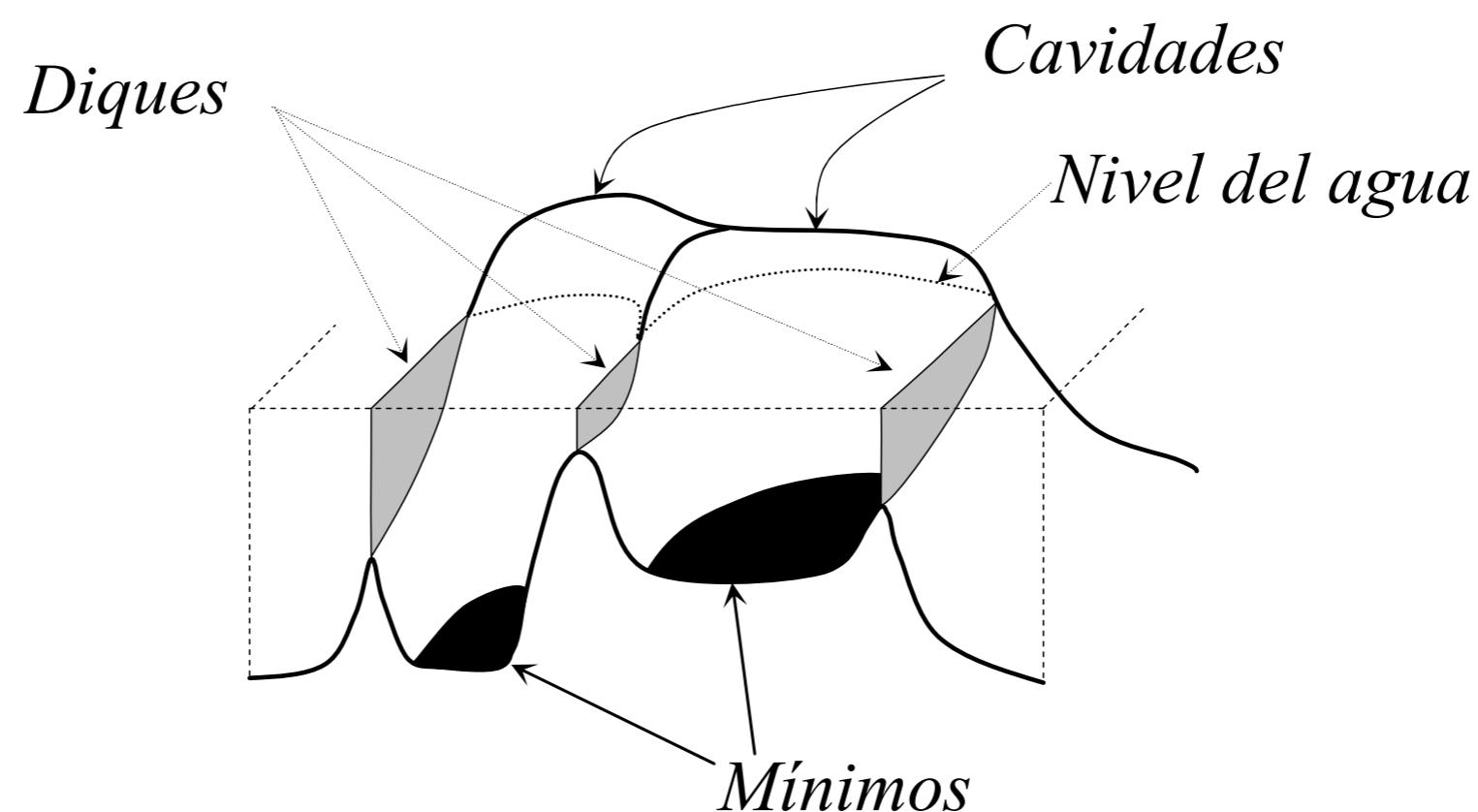
▶ Clasificación

Umbráles

Clustering

Bordes

- **Watershed:** Este método es aún más complejo ya que simula que la imagen está compuesta por valles y los bordes corresponden a diques.
- Estos valles son creados por los niveles de intensidad de cada región. A medida que los valles se van llenando de agua, se crean diques cuando dos valles se unen



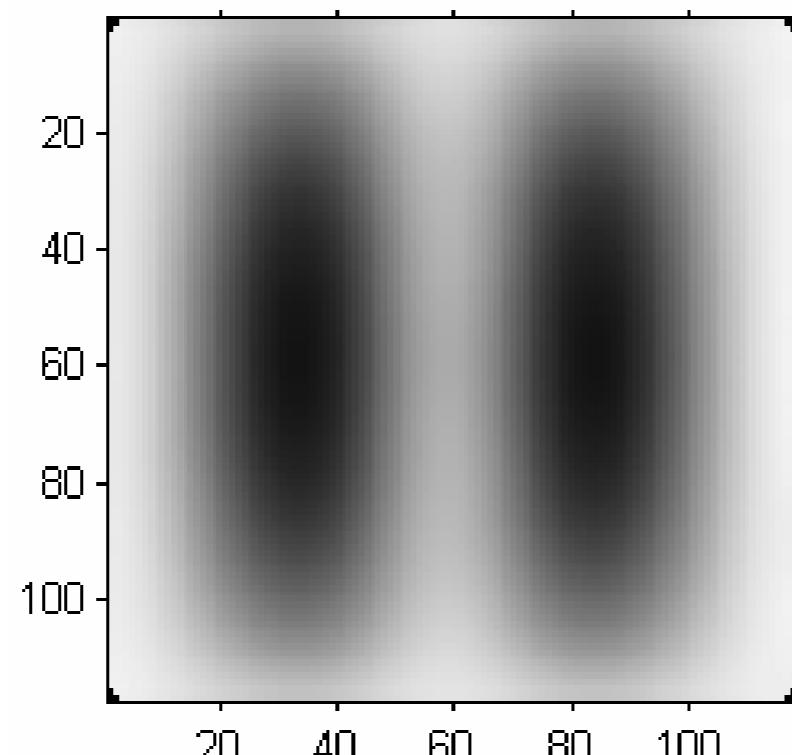
▶ Clasificación

Umbral

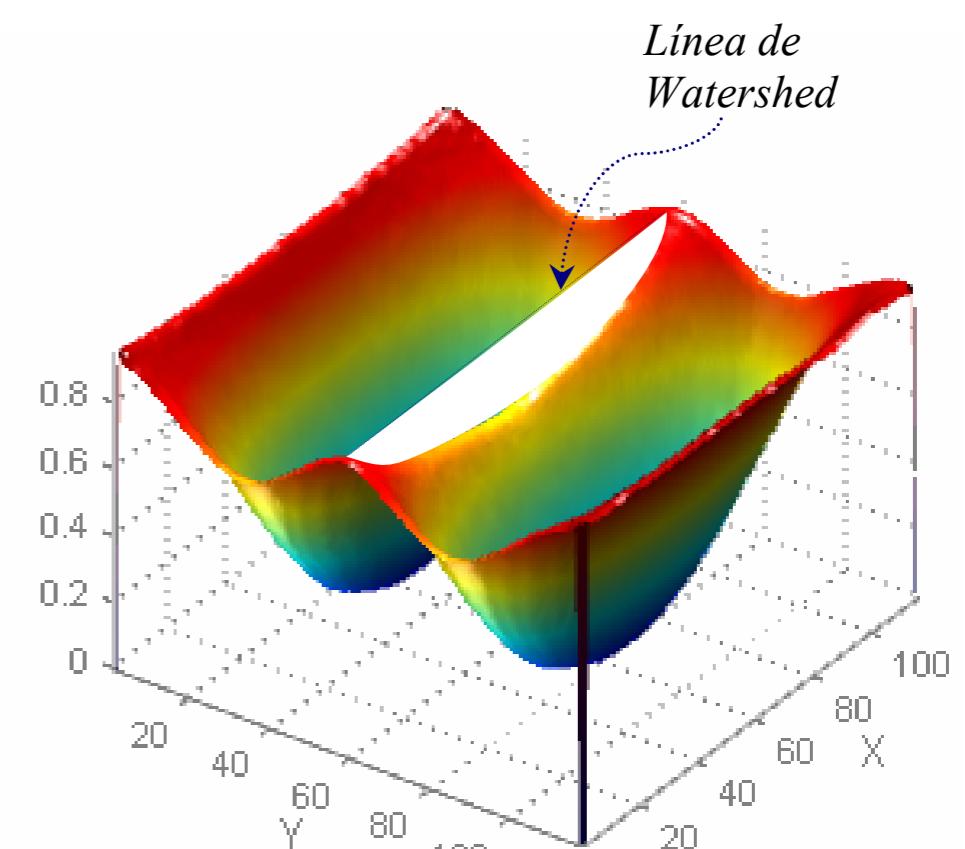
Clustering

Bordes

- **Watershed:** Este método es aún más complejo ya que simula que la imagen está compuesta por valles y los bordes corresponden a diques.
- Estos valles son creados por los niveles de intensidad de cada región. A medida que los valles se van llenando de agua, se crean diques cuando dos valles se unen



(a)



(b)

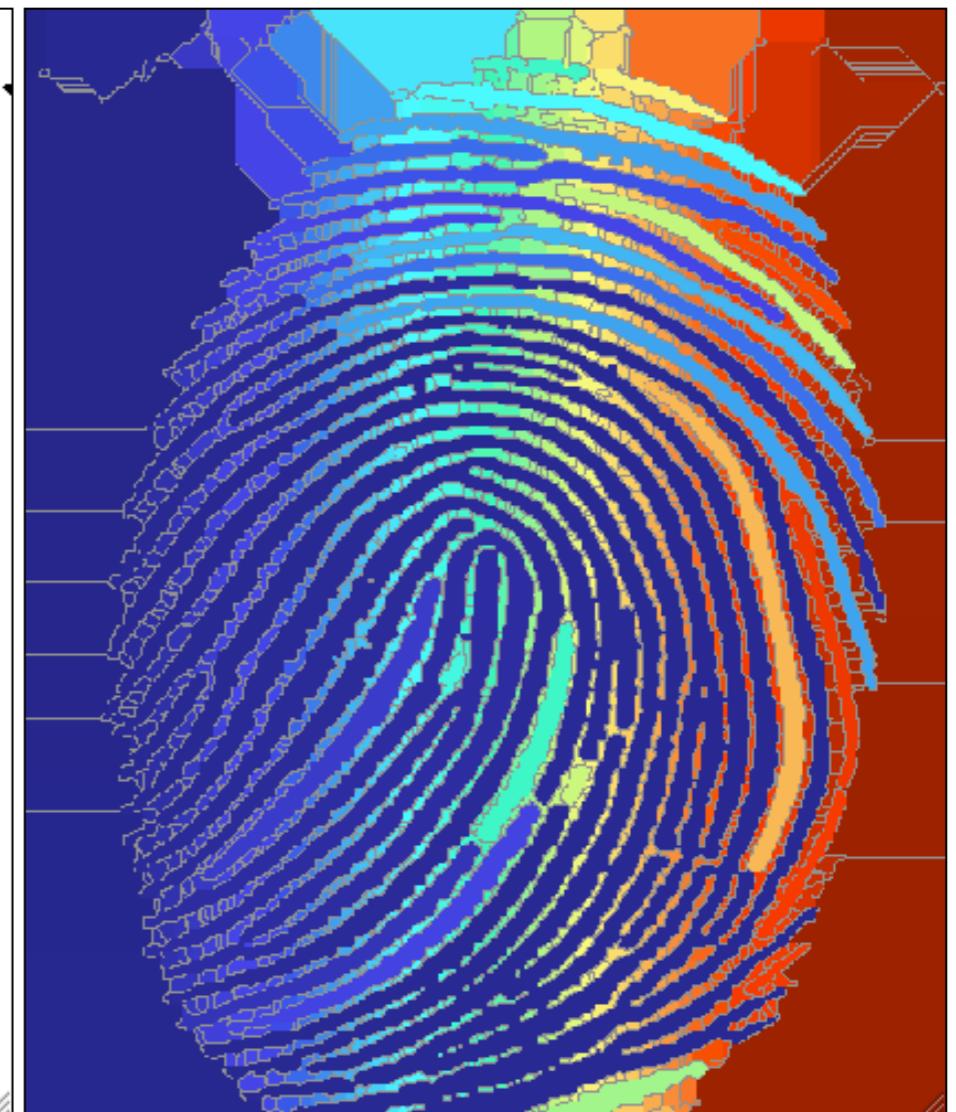
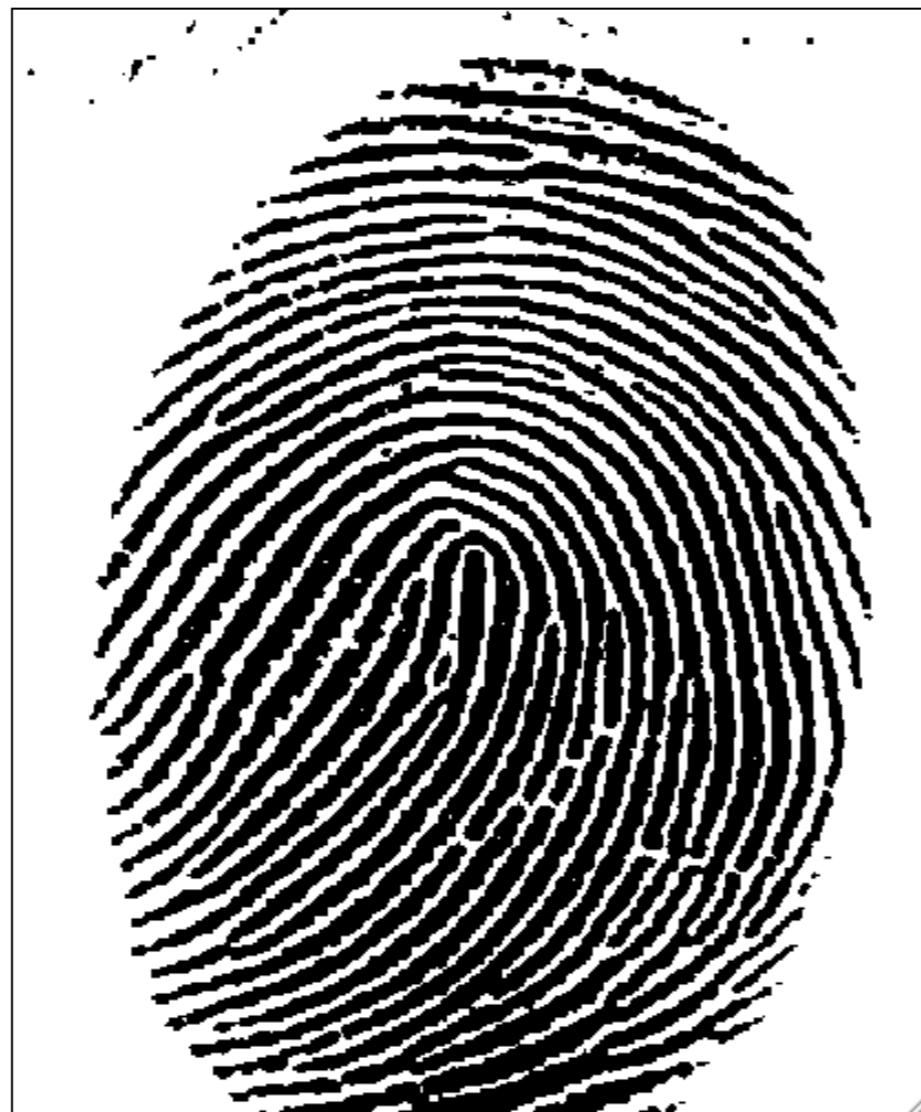
▶ Clasificación

Umbral

Clustering

Bordes

- **Watershed:** Este método es aún más complejo ya que simula que la imagen está compuesta por valles y los bordes corresponden a diques.



- Segmentación
 - Clasificación

Umbrales

Bordes

- **Umbral de Otsu:** Es una algoritmo que busca el umbral de segmentación que minimiza la varianza entre dos clases. El algoritmo supone que el histograma es bimodal, es decir, que hay dos clases separables

① **Objetivo:** buscar el valor mínimo de la varianza entre clases

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

$\forall t = 1, \dots, 256$

Para todos los niveles de intensidad

② Donde las probabilidades de cada clases son:

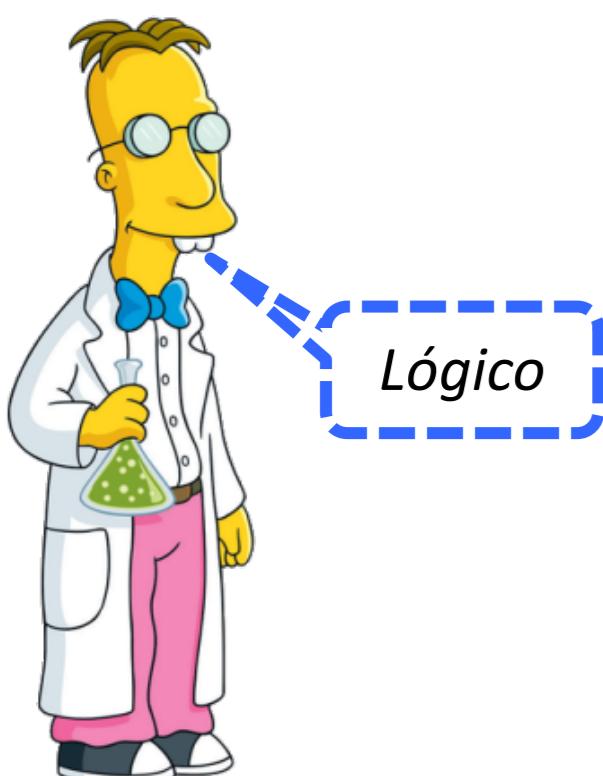
$$q_1(t) = \sum_{i=1}^t P(i)$$

$$q_2(t) = \sum_{i=t+1}^{256} P(i)$$

$P(i)$

Probabilidad del i-ésimo píxel

Lógico



Umbrales

Bordes

- **Umbral de Otsu:** Es una algoritmo que busca el umbral de segmentación que minimiza la varianza entre dos clases. El algoritmo supone que el histograma es bimodal, es decir, que hay dos clases separables

③ Y las medias de las clases son

$$\mu_1(t) = \sum_{i=1}^t \frac{i \times P(i)}{q_1(t)}$$

$$\mu_2(t) = \sum_{i=t+1}^{256} \frac{i \times P(i)}{q_2(t)}$$

④ Finalmente las variancias individuales son

$$\sigma_1^2(t) = \sum_{i=1}^t [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{256} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$



Umbrales

Bordes

- **Umbral de Otsu:** Es una algoritmo que busca el umbral de segmentación que minimiza la varianza entre dos clases.

1

$$q_1(t) = \sum_{i=1}^t P(i)$$

2

$$\mu_1(t) = \sum_{i=1}^t \frac{i \times P(i)}{q_1(t)}$$

3

$$\sigma_1^2(t) = \sum_{i=1}^t [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)}$$

4

$$\sigma_w^2(t) = q_1(t) \sigma_1^2(t) + q_2(t) \sigma_2^2(t)$$

$$q_2(t) = \sum_{i=t+1}^{256} P(i)$$

$$\mu_2(t) = \sum_{i=t+1}^{256} \frac{i \times P(i)}{q_2(t)}$$

Objetivo:
Buscar el
valor mínimo

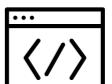
Umbrales

Bordes

- **Umbral de Otsu:** Es una algoritmo que busca el umbral de segmentación que minimiza la varianza entre dos clases.

0

$P(i)$ Probabilidad del i-ésimo píxel



```
def otsu(img):  
    bns= np.linspace(0,255,256,dtype='uint8')  
    h, bins = np.histogram(img, bins=bns)  
    P = h/np.sum(h)
```

Acá
obtenemos P
normalizado

Umbrales

Bordes

- **Umbral de Otsu:** Es una algoritmo que busca el umbral de segmentación que minimiza la varianza entre dos clases.

1

$$q_1(t) = \sum_{i=1}^t P(i)$$

$$q_2(t) = \sum_{i=t+1}^{256} P(i)$$

...

```
def otsu(img):
    bns= np.linspace(0,255,256,dtype='uint8')
    h, bins = np.histogram(img, bins=bns)
    P = h/np.sum(h)

    for t in range(1,255):
        iL= np.linspace(0,t-1,t,dtype='uint8')
        iR= np.linspace(t,254,255-t,dtype='uint8')
```

valores de 1 a t

Umbrales

Bordes

- **Umbral de Otsu:** Es una algoritmo que busca el umbral de segmentación que minimiza la varianza entre dos clases.

1

$$q_1(t) = \sum_{i=1}^t P(i)$$

$$q_2(t) = \sum_{i=t+1}^{256} P(i)$$

...

```
def otsu(img):
    bns= np.linspace(0,255,256,dtype='uint8')
    h, bins = np.histogram(img, bins=bns)
    P = h/np.sum(h)

    for t in range(1,255):
        iL= np.linspace(0,t-1,t,dtype='uint8')
        iR= np.linspace(t,254,255-t,dtype='uint8')
        q1[t]=np.sum(P[iL])+0.00001
        q2[t]=np.sum(P[iR])+0.00001
```

sumamos un épsilon
para que la suma no
sea cero

Umbrales

Bordes

- **Umbral de Otsu:** Es una algoritmo que busca el umbral de segmentación que minimiza la varianza entre dos clases.

2

$$\mu_1(t) = \sum_{i=1}^t \frac{i \times P(i)}{q_1(t)}$$

$$\mu_2(t) = \sum_{i=t+1}^{256} \frac{i \times P(i)}{q_2(t)}$$



```
def otsu(img):
    bns= np.linspace(0,255,256,dtype='uint8')
    h, bins = np.histogram(img, bins=bns)
    P = h/np.sum(h)

    for t in range(1,255):
        iL= np.linspace(0,t-1,t,dtype='uint8')
        iR= np.linspace(t,254,255-t,dtype='uint8')
        q1[t]=np.sum(P[iL])+0.00001
        q2[t]=np.sum(P[iR])+0.00001
        mu1[t]= np.sum(iL*P[iL]/q1[t])
        mu2[t]= np.sum(iR*P[iR]/q2[t])
```

Umbrales

Bordes

- **Umbral de Otsu:** Es una algoritmo que busca el umbral de segmentación que minimiza la varianza entre dos clases.

3

$$\sigma_1^2(t) = \sum_{i=1}^t [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{256} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$

```
def otsu(img):
    bns= np.linspace(0,255,256,dtype='uint8')
    h, bins = np.histogram(img, bins=bns)
    P = h/np.sum(h)

    for t in range(1,255):
        iL= np.linspace(0,t-1,t,dtype='uint8')
        iR= np.linspace(t,254,255-t,dtype='uint8')
        q1[t]=np.sum(P[iL])+0.00001
        q2[t]=np.sum(P[iR])+0.00001
        mu1[t]= np.sum(iL*P[iL]/q1[t])
        mu2[t]= np.sum(iR*P[iR]/q2[t])
        va1[t]= np.sum((iL-mu1[t])**2* (P[iL]/ q1[t]))
        va2[t]= np.sum((iR-mu2[t])**2* (P[iR]/ q2[t]))
```

Umbrales

Bordes

- **Umbral de Otsu:** Es una algoritmo que busca el umbral de segmentación que minimiza la varianza entre dos clases.

4

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$



```
def otsu(img):
    bns= np.linspace(0,255,256,dtype='uint8')
    h, bins = np.histogram(img, bins=bns)
    P = h/np.sum(h)
    for t in range(1,255):
        iL= np.linspace(0,t-1,t,dtype='uint8')
        iR= np.linspace(t,254,255-t,dtype='uint8')
        q1[t]=np.sum(P[iL])+0.00001
        q2[t]=np.sum(P[iR])+0.00001
        mu1[t]= np.sum(iL*P[iL]/q1[t])
        mu2[t]= np.sum(iR*P[iR]/q2[t])
        va1[t]= np.sum((iL-mu1[t])**2* (P[iL]/ q1[t]))
        va2[t]= np.sum((iR-mu2[t])**2* (P[iR]/ q2[t]))

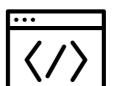
    vartotal= q1*va1 + q2*va2
    print(vartotal)
    level=np.argmin(vartotal[1:255])
    return level
```

Objetivo: Buscar el valor mínimo

Umbrales

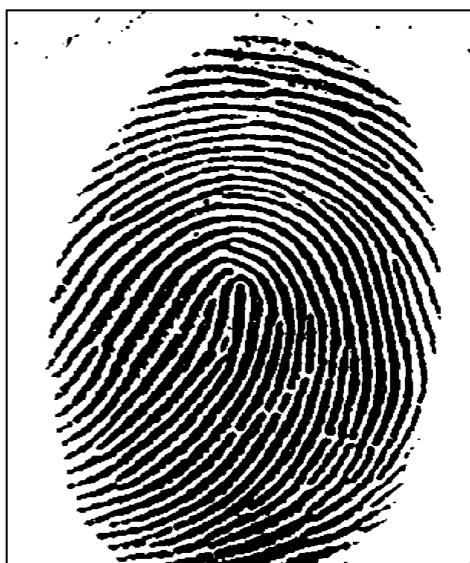
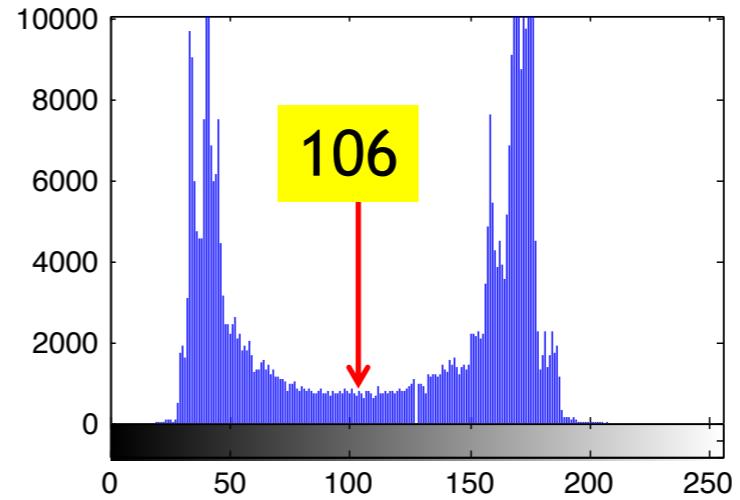
Bordes

- **Umbral de Otsu:** Es una algoritmo que busca el umbral de segmentación que minimiza la varianza entre dos clases.



```
img = cv2.imread('huella.png', cv2.IMREAD_GRAYSCALE)
level= otsu(img)
ret, bin = cv2.threshold(img, level, 255,type=cv2.THRESH_BINARY)

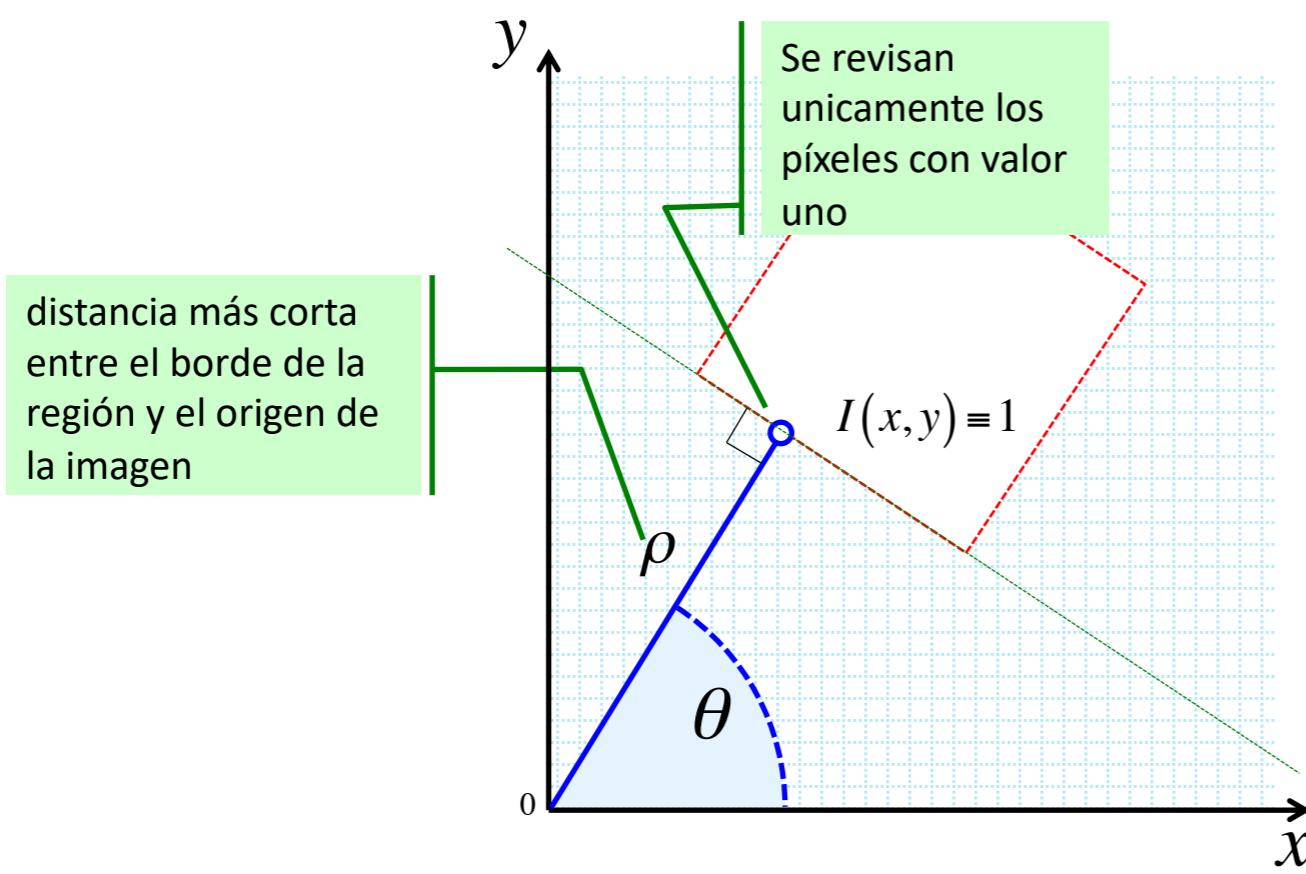
cv2.imshow('huella', bin)
cv2.waitKey(0)
```



Umbrales

Bordes

- **Transformada de Hough:** Es una técnica que permite detectar formas simples en una imagen tales como líneas, círculos y elipses. Para emplearla, requiere una imagen binaria de los píxeles que forman parte de los objetos.



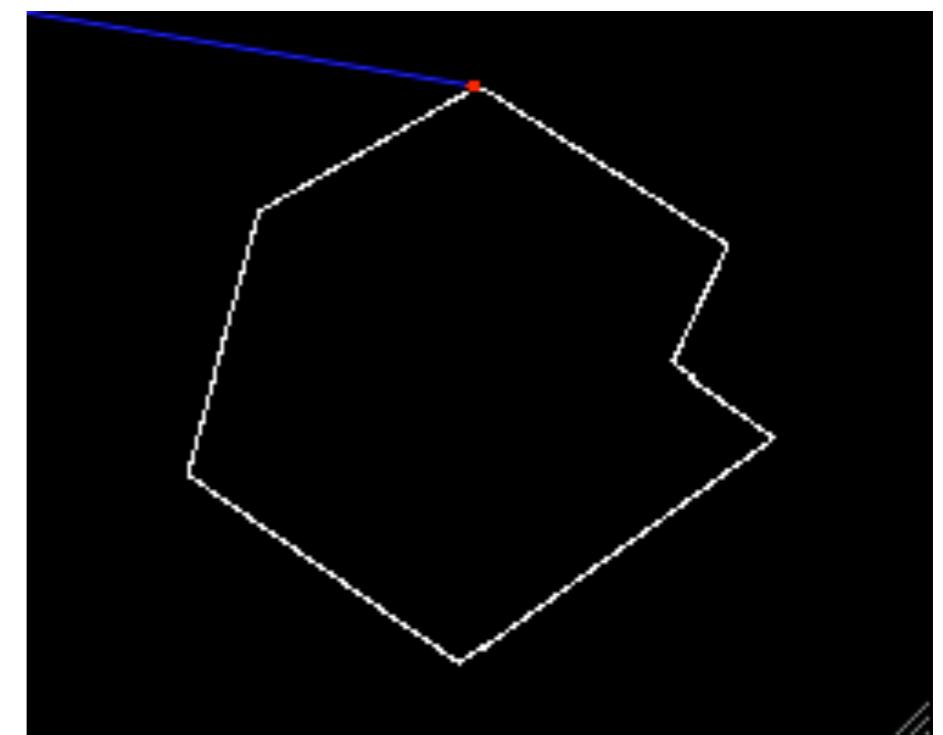
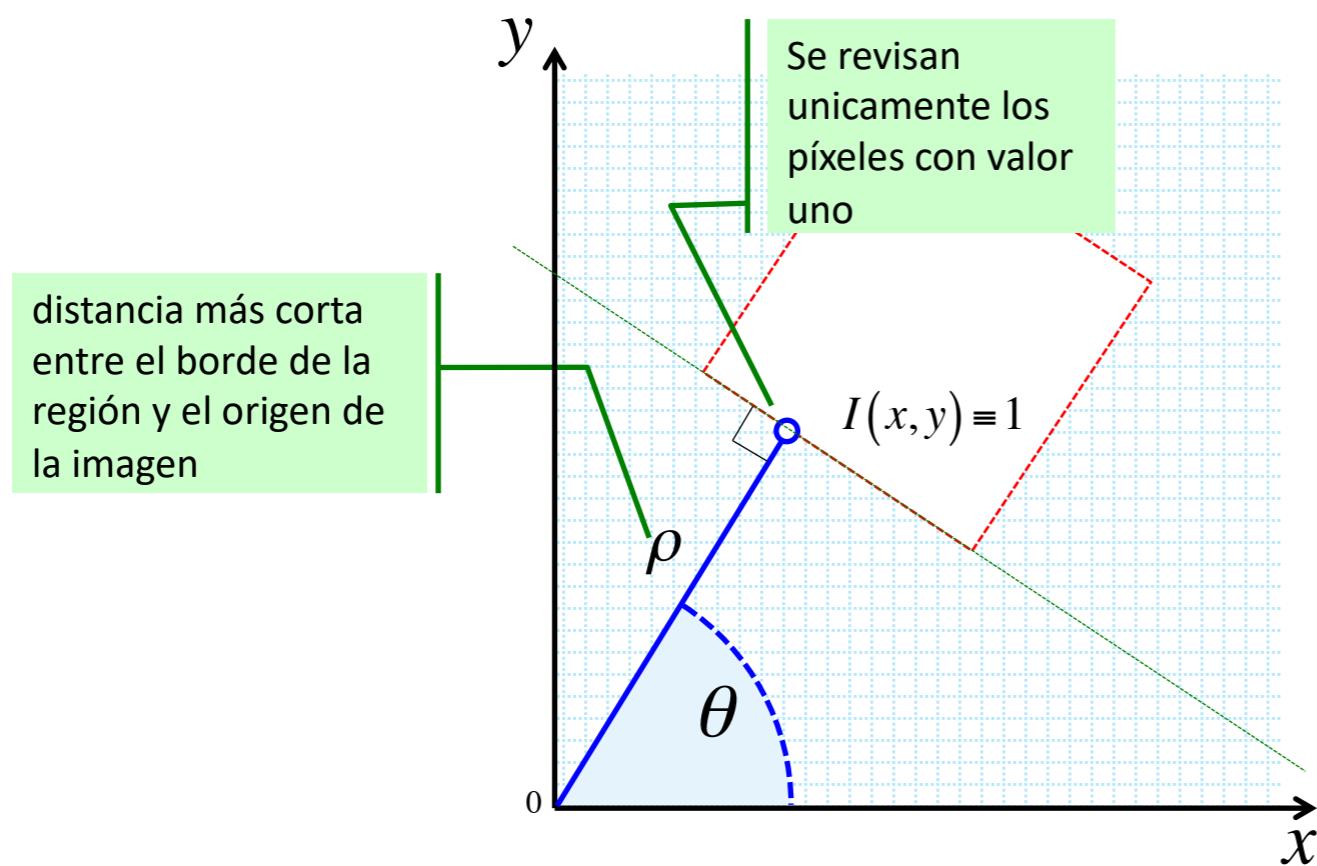
Proceso

- Binarizar la imagen para detectar los bordes de las regiones
 - Cada punto binario se representa por un punto en el sistema de coordenadas polares con coordenadas: $\{\rho, \theta\}$
 - Evaluar para cada coordenada de la imagen si se cumple la ecuación de la recta en forma polar.
- $$x \cos \theta + y \sin \theta = \rho$$
- Si se cumple la condición se incrementa un contador en la coordenada polar

Umbrales

Bordes

- **Transformada de Hough:** Es una técnica que permite detectar formas simples en una imagen tales como líneas, círculos y elipses. Para emplearla, requiere una imagen binaria de los píxeles que forman parte de los objetos.



Animación del proceso de Hough

Umbral

Bordes

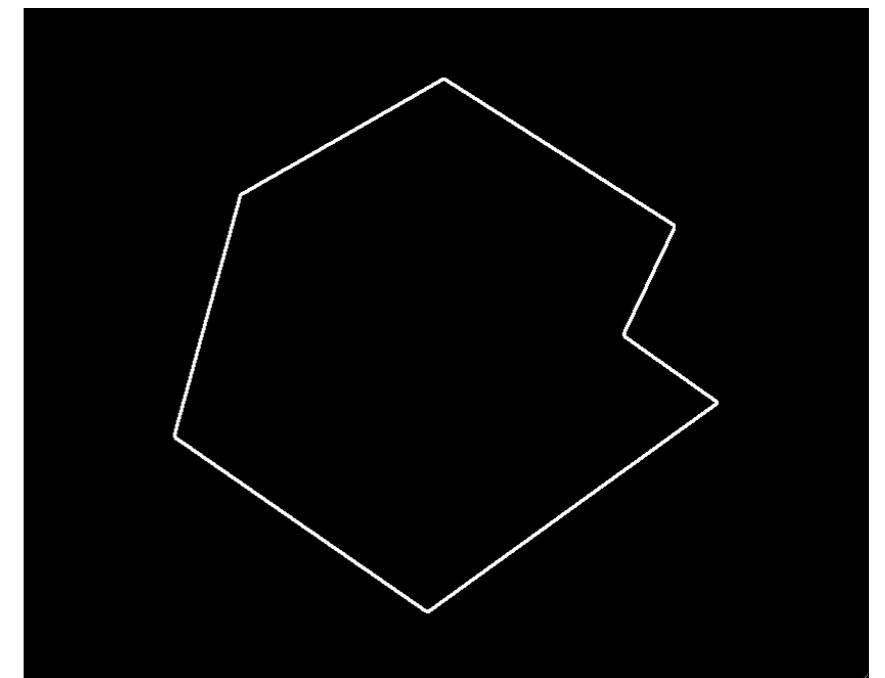
- Transformada de Hough: Primero, determinamos los bordes del objeto

```
from skimage.transform import hough_line, hough_line_peaks
import matplotlib.pyplot as plt
from matplotlib import cm

I = cv2.imread('rombo.png', cv2.IMREAD_GRAYSCALE)
BW = cv2.Canny(I,50,150,apertureSize = 3)
```



I



BW

Umbrales

Bordes

- **Transformada de Hough:** Segundo, empleamos la función integrada de Python para calcular la transformada de Hough

```
from skimage.transform import hough_line, hough_line_peaks
import matplotlib.pyplot as plt
from matplotlib import cm

I = cv2.imread('rombo.png', cv2.IMREAD_GRAYSCALE)
BW = cv2.Canny(I,50,150,apertureSize = 3)

angulos = np.linspace(-np.pi/2, np.pi/2, 360)

h, theta, d = hough_line(BW, theta=angulos)
```

Determina los valores Theta y Rho acumulados (en H)

Angulos donde recorrer el espacio

Umbral

Bordes

- Transformada de Hough: Segundo, empleamos la función integrada de Python para calcular la transformada de Hough

```
from skimage.transform import hough_line, hough_line_peaks
import matplotlib.pyplot as plt
from matplotlib import cm

I = cv2.imread('rombo.png', cv2.IMREAD_GRAYSCALE)
BW = cv2.Canny(I, 50, 150, apertureSize = 3)

angulos = np.linspace(-np.pi/2, np.pi/2, 360)

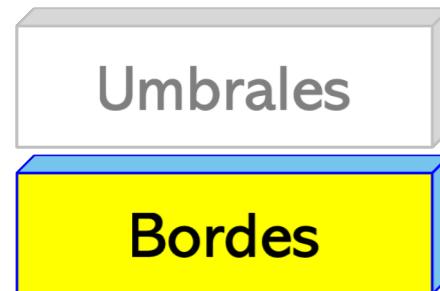
h, theta, d = hough_line(BW, theta=angulos)
```

Mapa de acumulación

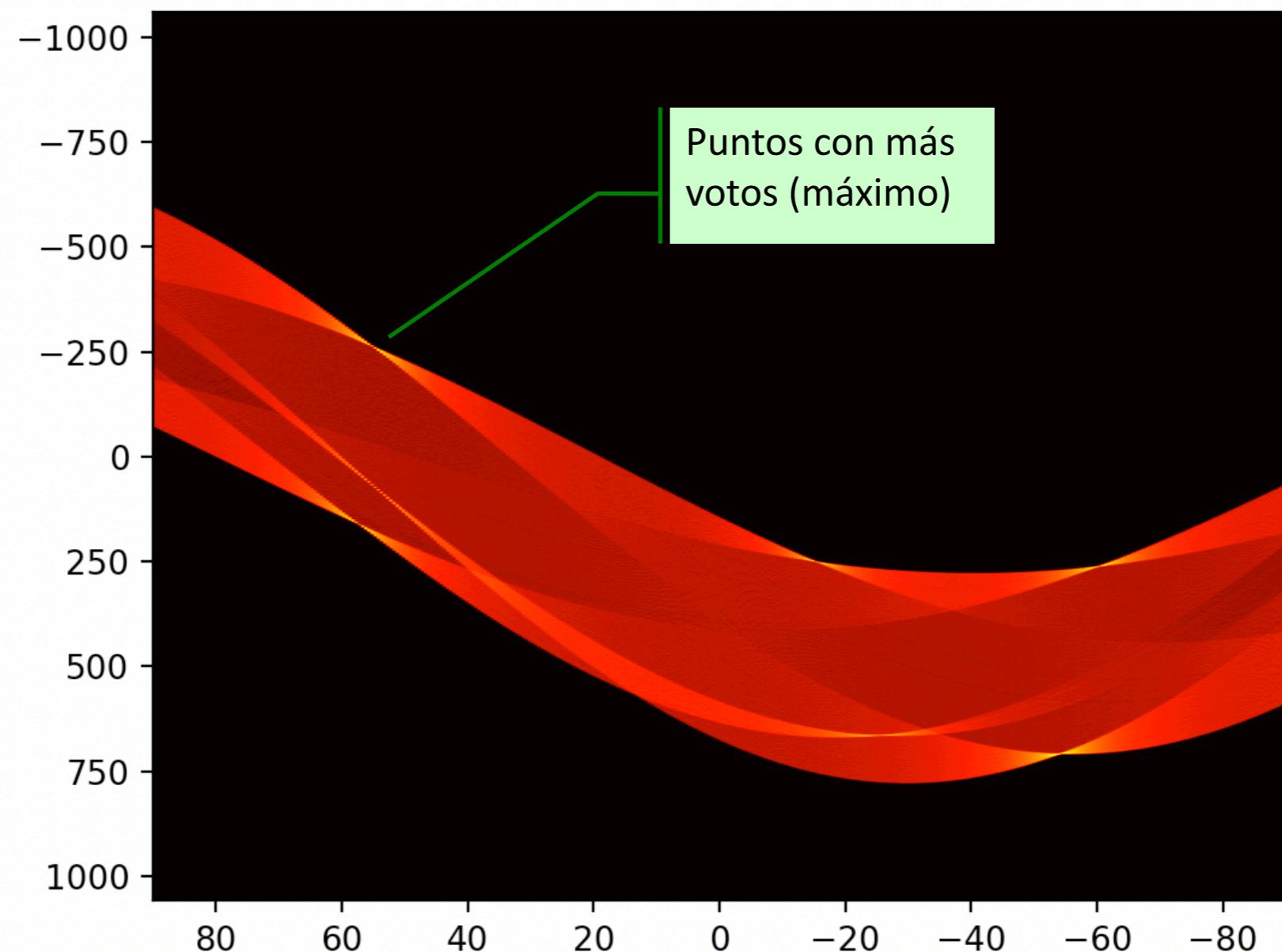
```
fig = plt.figure()
plt.subplot(111)
plt.imshow(np.log(1 + h),
           extent=[np.rad2deg(theta[-1]), np.rad2deg(theta[0]),
                    d[-1], d[0]], cmap=cm.hot, aspect=1/15)
```

extent define los límites del espacio donde plotear

Ploteamos el resultado de los acumuladores



- **Transformada de Hough:** Segundo, empleamos la función integrada de Python para calcular la transformada de Hough



Umbrales

Bordes

forma paramétrica

$$x\cos\theta + y\sin\theta = \rho$$

despejando

$$y = -\frac{\cos\theta}{\sin\theta}x + \frac{\rho}{\sin\theta}$$

- **Transformada de Hough:** Tercero, buscamos los valores máximos encontrados por la transformada de Hough, es decir, aquellos puntos con más votos.

```
fig = plt.figure()
plt.subplot(111)
plt.imshow(BW, cmap=cm.gray)
eje_x = np.array((0, BW.shape[1]))

valores_maximos= hough_line_peaks(h, theta, d)

for accum, theta, rho in zip(*valores_maximos):
```

Busca un determinado número de máximos específico



Umbrales

Bordes

forma paramétrica
 $x\cos\theta + y\sin\theta = \rho$

despejando

$$y = -\frac{\cos\theta}{\sin\theta}x + \frac{\rho}{\sin\theta}$$

$$m = -\frac{\cos\theta}{\sin\theta}$$

$$b = \frac{\rho}{\sin\theta}$$

$$y = mx + b$$

- **Transformada de Hough:** Tercero, buscamos los valores máximos encontrados por la transformada de Hough, es decir, aquellos puntos con más votos.

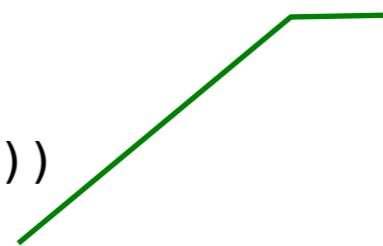
```
fig = plt.figure()
plt.subplot(111)
plt.imshow(BW, cmap=cm.gray)
eje_x = np.array((0, BW.shape[1]))

valores_maximos= hough_line_peaks(h, theta, d)

for accum, theta, rho in zip(*valores_maximos):
    y0 = (rho - eje_x[0] * np.cos(theta)) / np.sin(theta)
    y1 = (rho - eje_x[1] * np.cos(theta)) / np.sin(theta)
    plt.plot(eje_x, (y0, y1), '-r')

plt.xlim(eje_x)
plt.ylim((BW.shape[0], 0))
plt.title('Lineas de mapa de Hough')
plt.show()
```

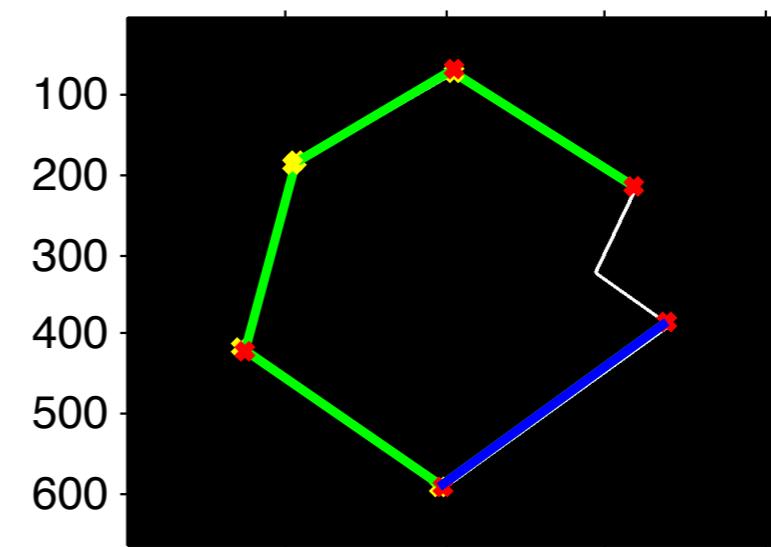
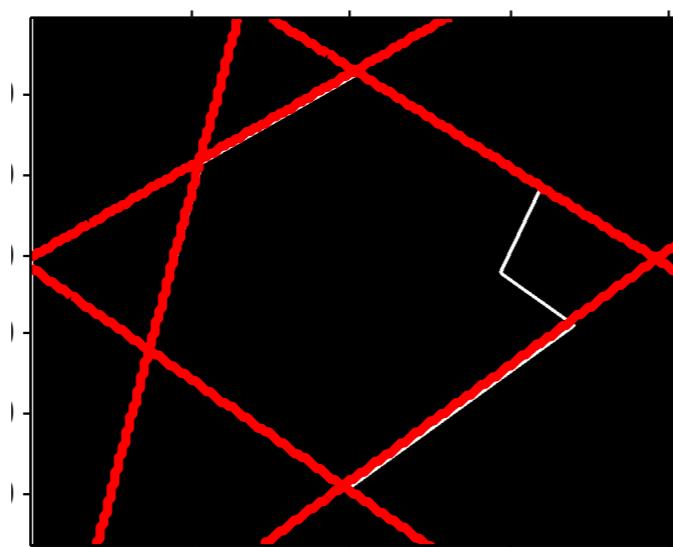
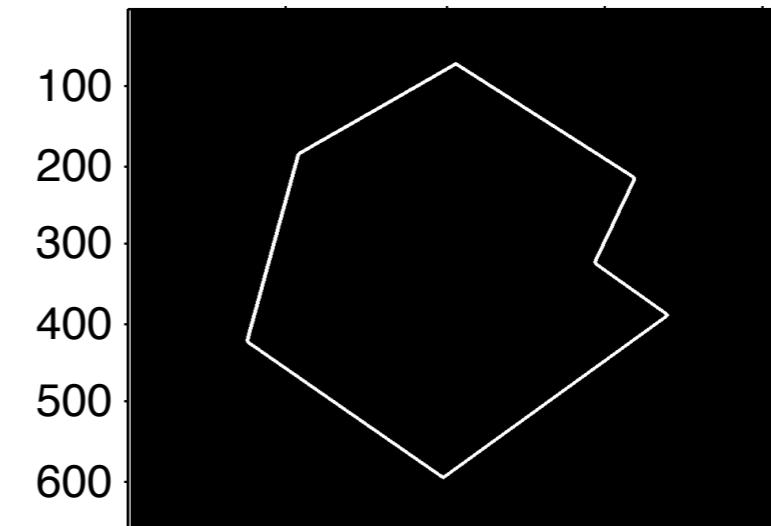
Busca un determinado número de máximos específico



Umbráles

Bordes

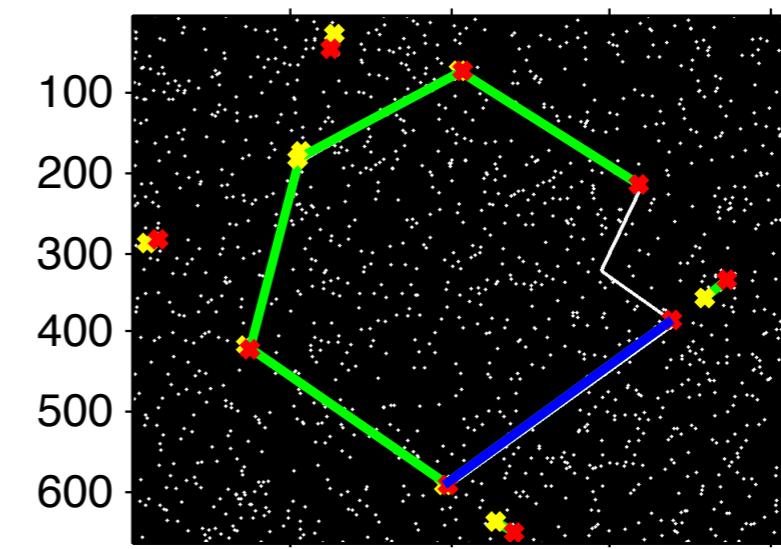
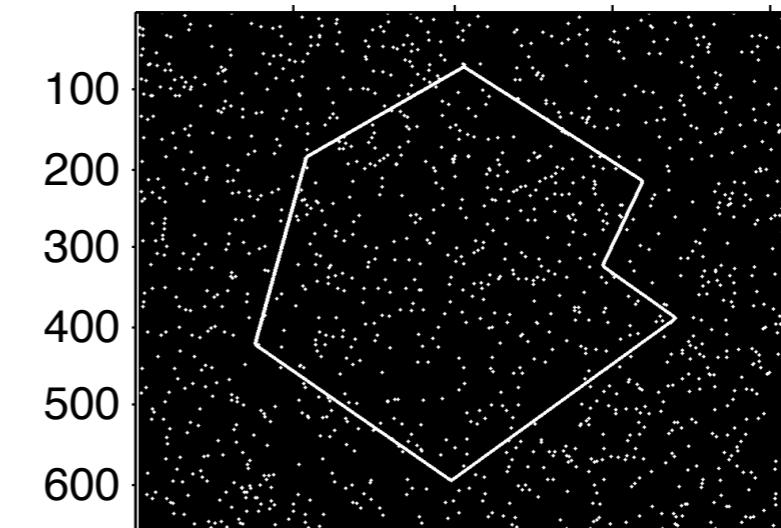
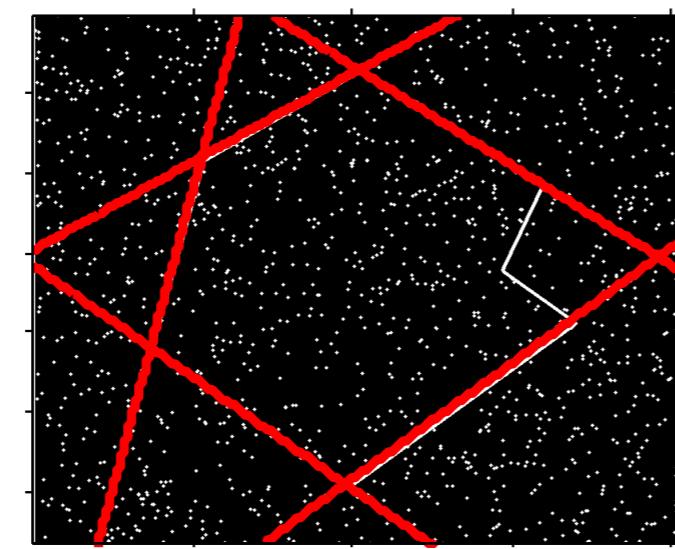
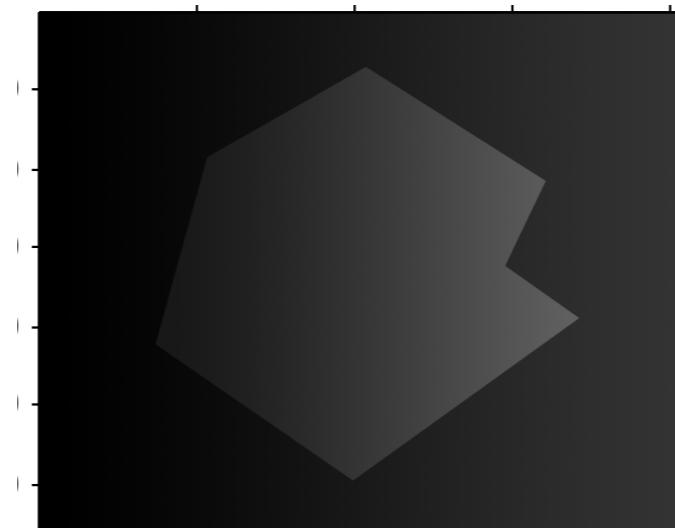
- Transformada de Hough: Resultados con imagen libre de ruido



Umbráles

Bordes

- Transformada de Hough: Resultado con imagen con ruido salt&pepper



Umbras

Bordes

- **Hough en círculos:** Analicemos el caso de los círculos. El proceso es muy similar, solo que ahora buscamos los parámetros del círculo. Veamos en detalles este proceso.
- Sea un círculo con radio R y centro (a, b) . Descrito en ecuaciones paramétricas corresponde a

$$x = a + R \cos(\theta)$$

$$y = b + R \sin(\theta)$$

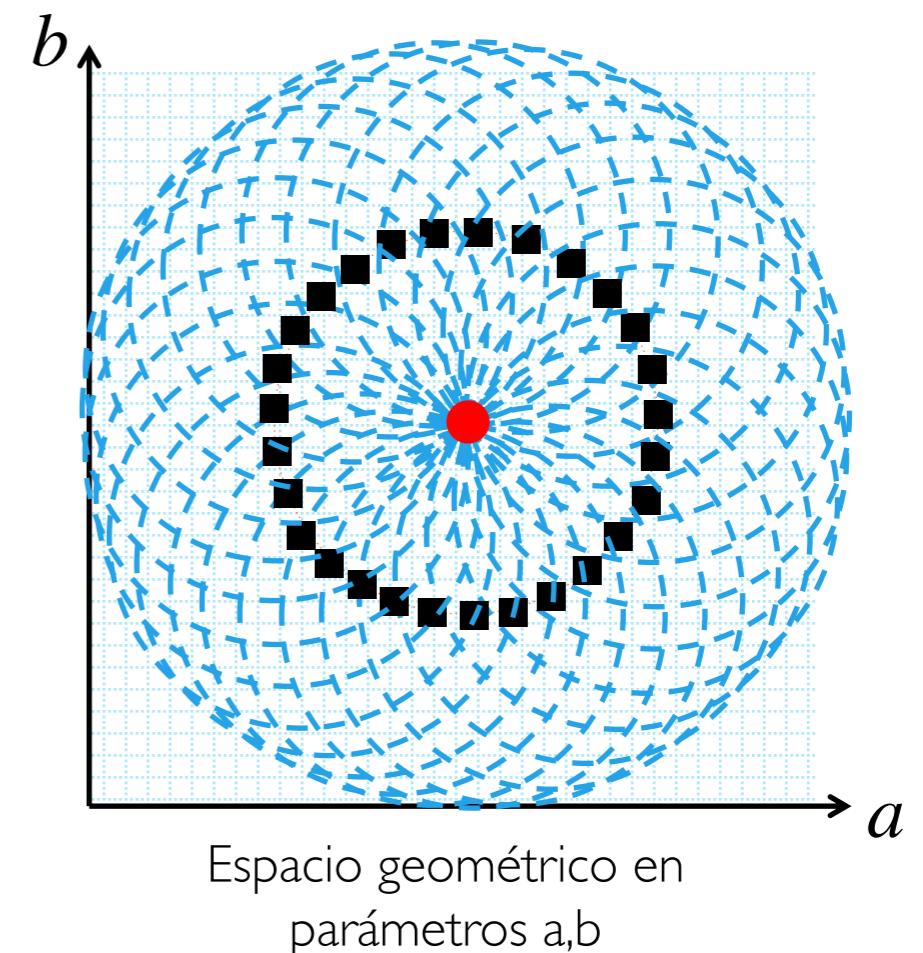
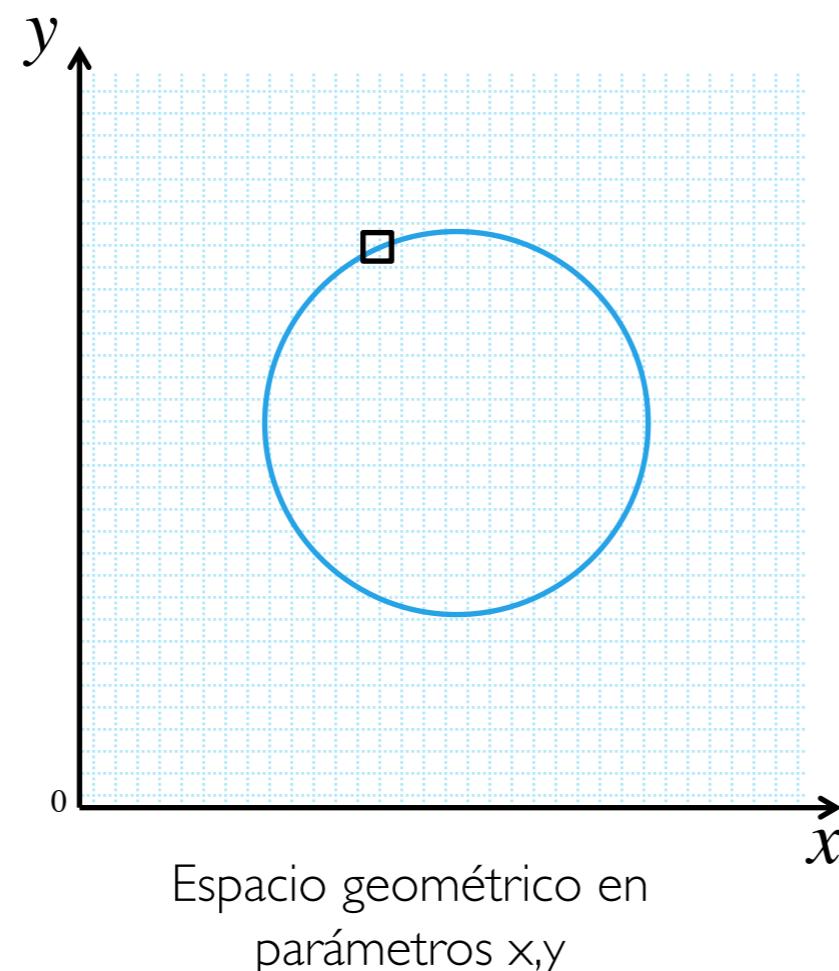
- Cuando el ángulo θ completa los 360 grados, trazamos el perímetro del círculo. Imagine ahora que tiene una imagen donde se encuentran potenciales perímetros de un círculo. El objetivo es buscar los parámetros $\{R, a, b\}$ que describan dicho círculo.



Umbras

Bordes

- **Hough en círculos:** Si los círculos de la imagen tienen un radio conocido R , entonces la búsqueda se reduce a encontrar los parámetros (a, b) .

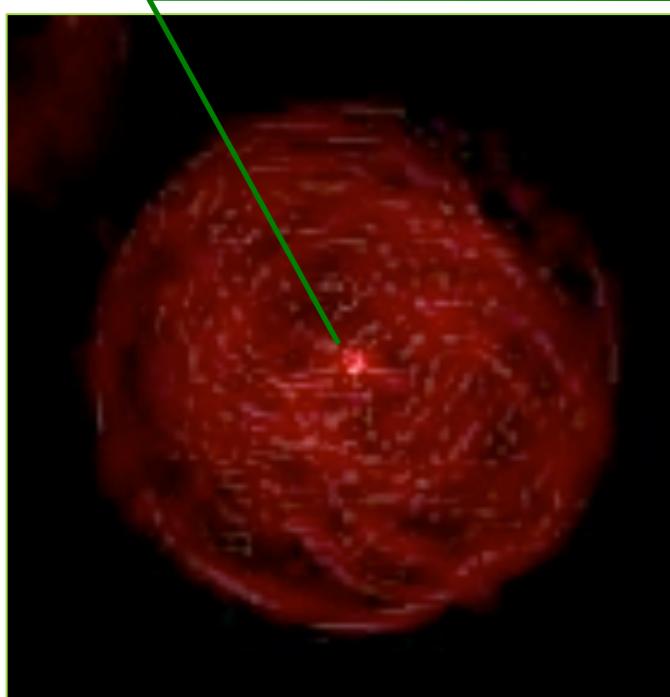


Umbrales

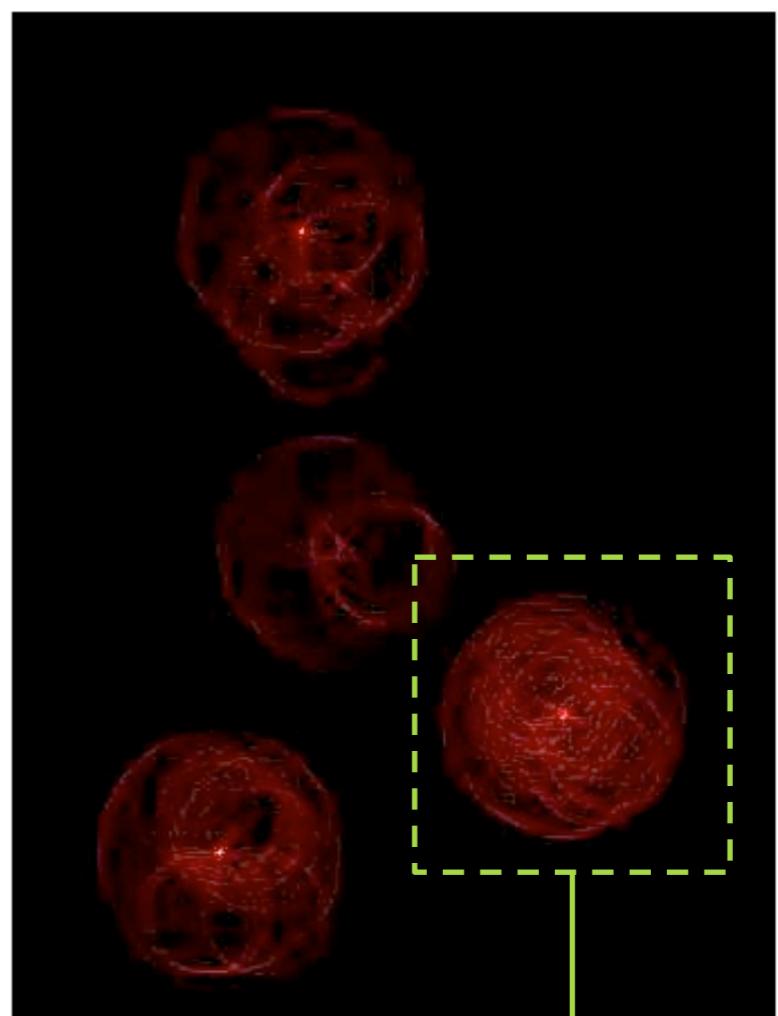
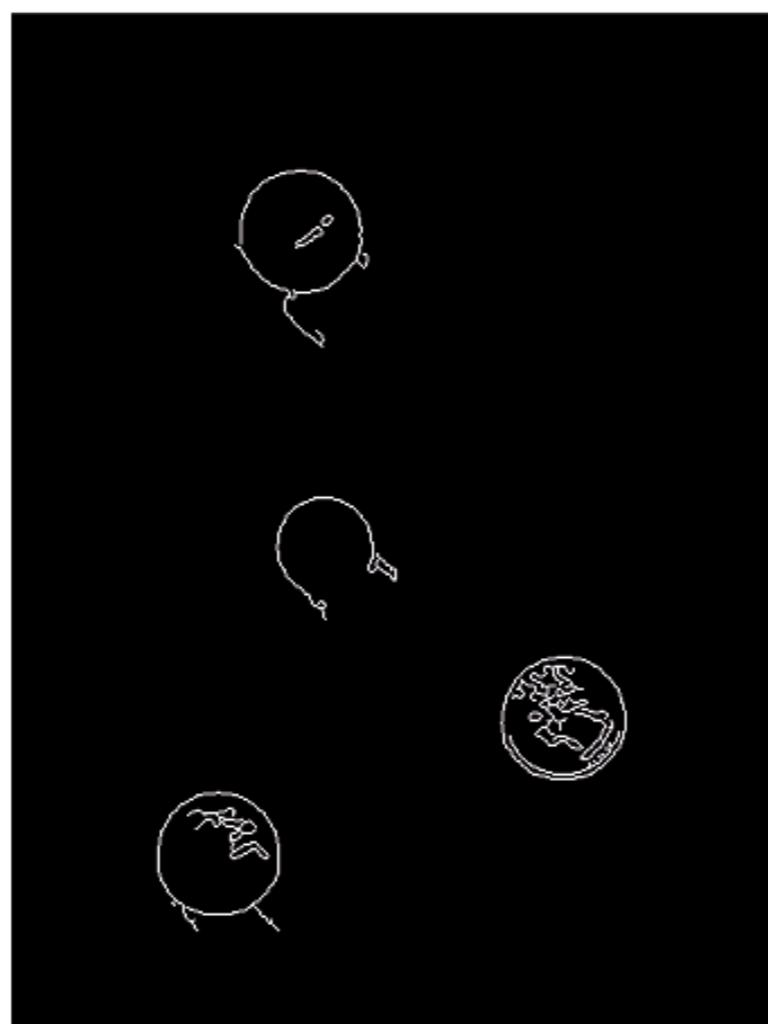
Bordes

- **Hough en círculos:** El objetivo es encontrar en el espacio (a,b) la mayor cantidad de votos y seleccionar aquellos máximos como los parámetros del círculo

Posición en el espacio A-B donde mayor acumulación está presente.



Bordes de la imagen

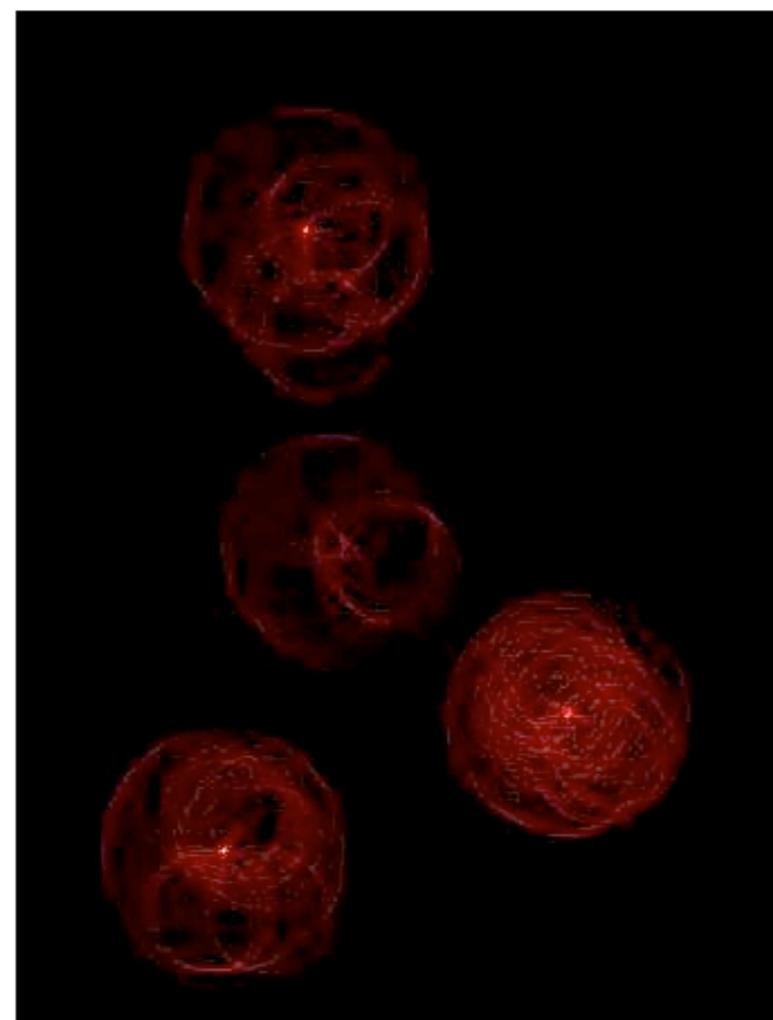


Acumulación

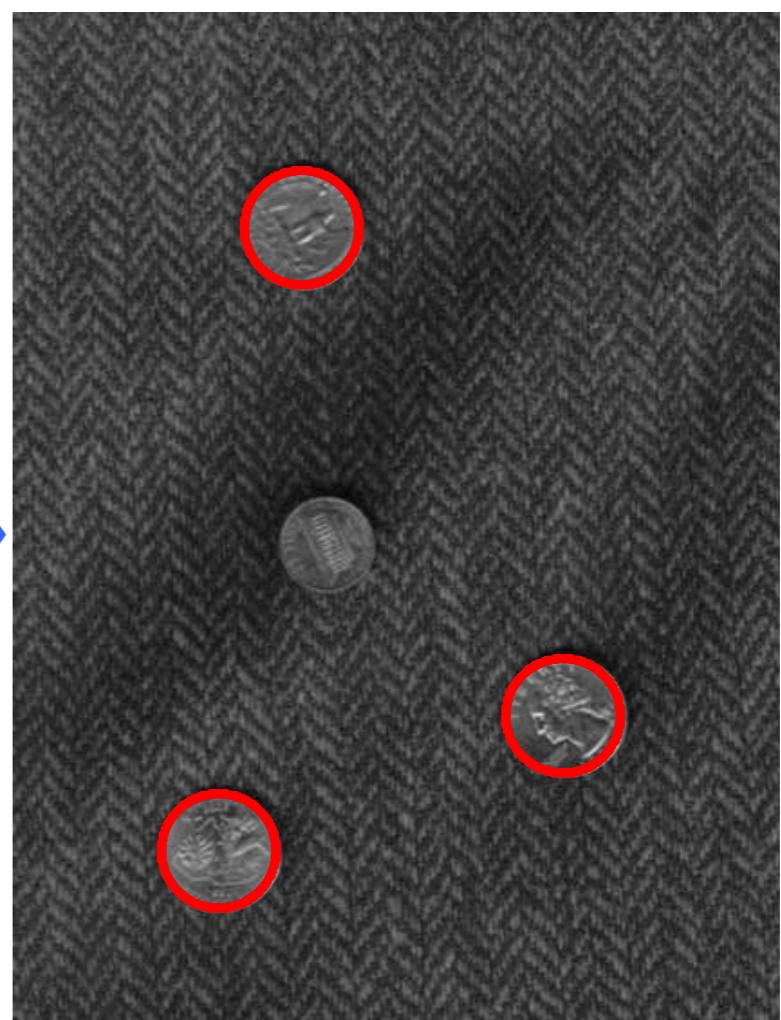
Umbras

Bordes

- **Hough en círculos:** El objetivo es encontrar en el espacio (a, b) la mayor cantidad de votos y seleccionar aquellos máximos como los parámetros del círculo



Buscamos los parámetros
máximos



Acumulación