PROCESAMIENTO DE IMÁGENES DIGITALES

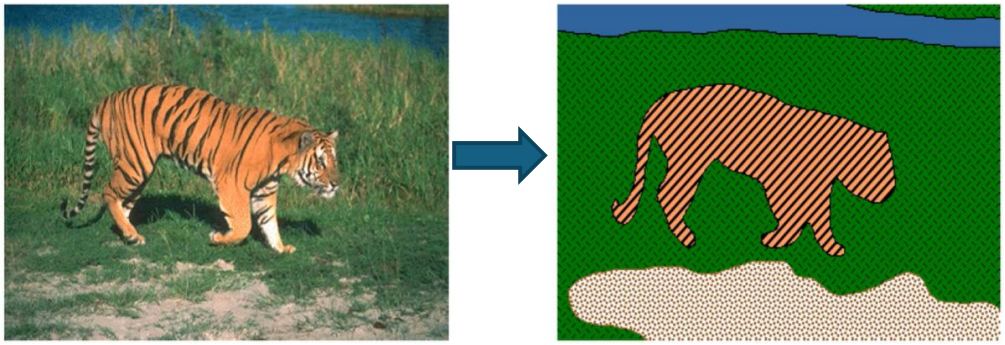
MATEMÁTICA APLICADA I

Tema 6: SEGMENTACIÓN

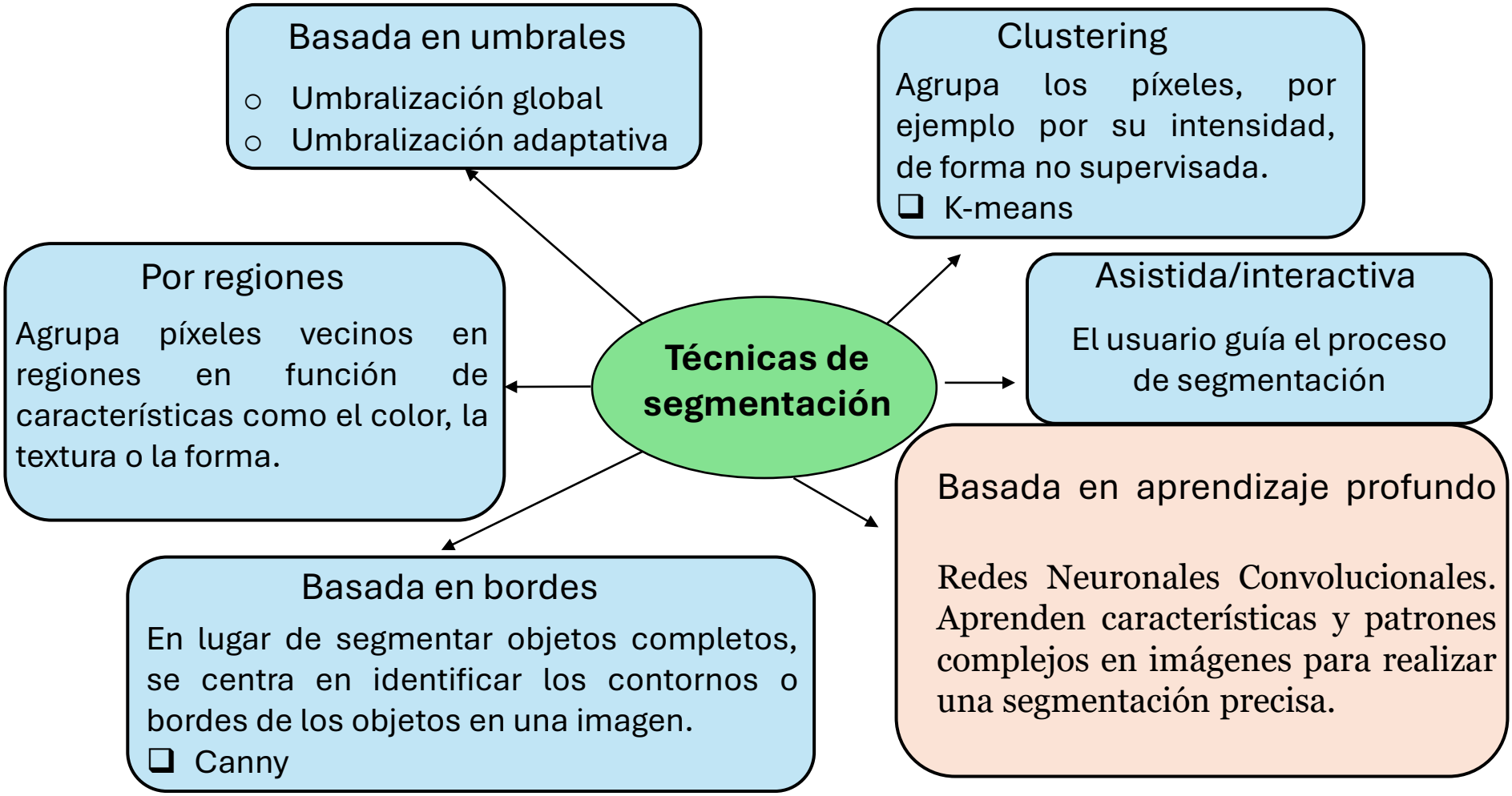
Segmentación

•El propósito de la segmentación de imágenes consiste en **dividir una imagen en regiones significativas** con respecto a una aplicación particular.

• Su resultado es una representación simplificada de la imagen, mostrando las partes significativas que la componen.



Segmentación



Segmentación

La elección de la técnica de segmentación adecuada depende de la naturaleza de los datos y los requisitos específicos de la aplicación.

A menudo, se utilizan enfoques combinados o técnicas avanzadas de aprendizaje profundo para abordar desafíos complejos de segmentación de imágenes en aplicaciones como la visión artificial, la medicina, la robótica y muchas otras áreas.

Segmentación

• Los *algoritmos clásicos* de segmentación se basan en propiedades básicas de los valores del nivel de gris:

-**Similitud**: Se divide la imagen basándose en la búsqueda de zonas que tengan valores similares, conforme a unos criterios prefijados. Hay que destacar los métodos de **umbralización** con los que se busca diferenciar un objeto del fondo de la imagen mediante una simple binarización de los niveles de gris.

- **Discontinuidad:** Los bordes de las regiones son suficientemente diferentes del fondo lo que permite la detección de los mismos basándonos en cambios bruscos de nivel de intensidad.

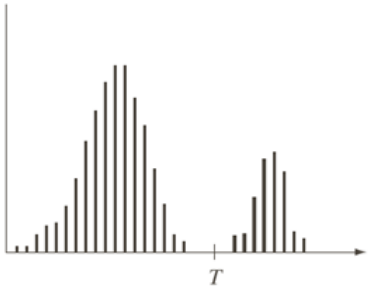
Segmentación

- **Umbralización global:**

  Es uno de los algoritmos más simples para diferenciar un objeto del fondo.
  Es eficaz cuando el objeto de interés y el fondo tienen valores de intensidad bien diferenciados en (aproximadamente) dos picos en el histograma, en cuyo caso el umbral se buscaría en el valle.

  Normalmente, binarizan la imagen inicial → dos categorías: el fondo de la imagen y los objetos buscados.

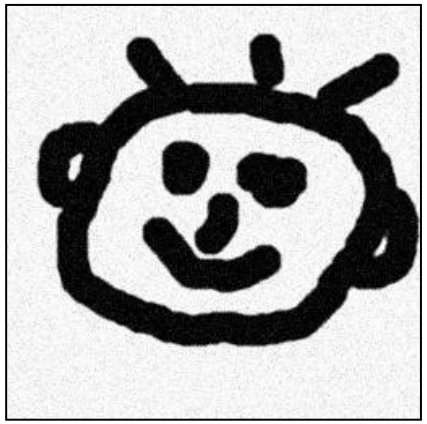
  Un único valor umbral T para toda la imagen $f(x,y) \rightarrow g(x,y)$:

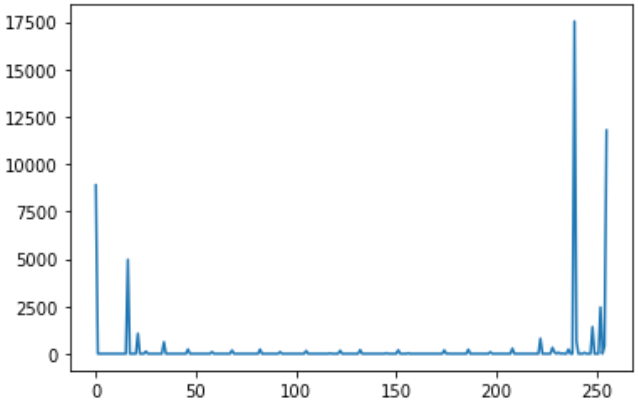


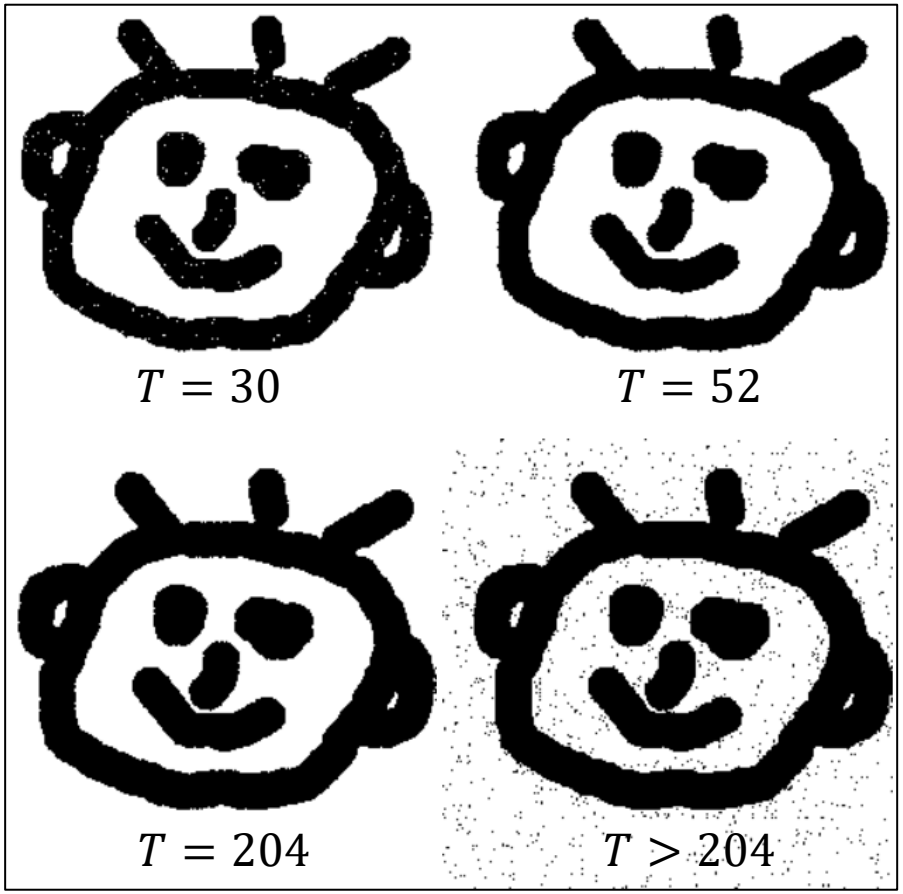
$$g(x,y) = \begin{cases} 1 & \text{si } f(x,y) > T \\ 0 & \text{si } f(x,y) \leq T \end{cases}$$

Segmentación

Umbralización global:







Segmentación

Umbralización global:

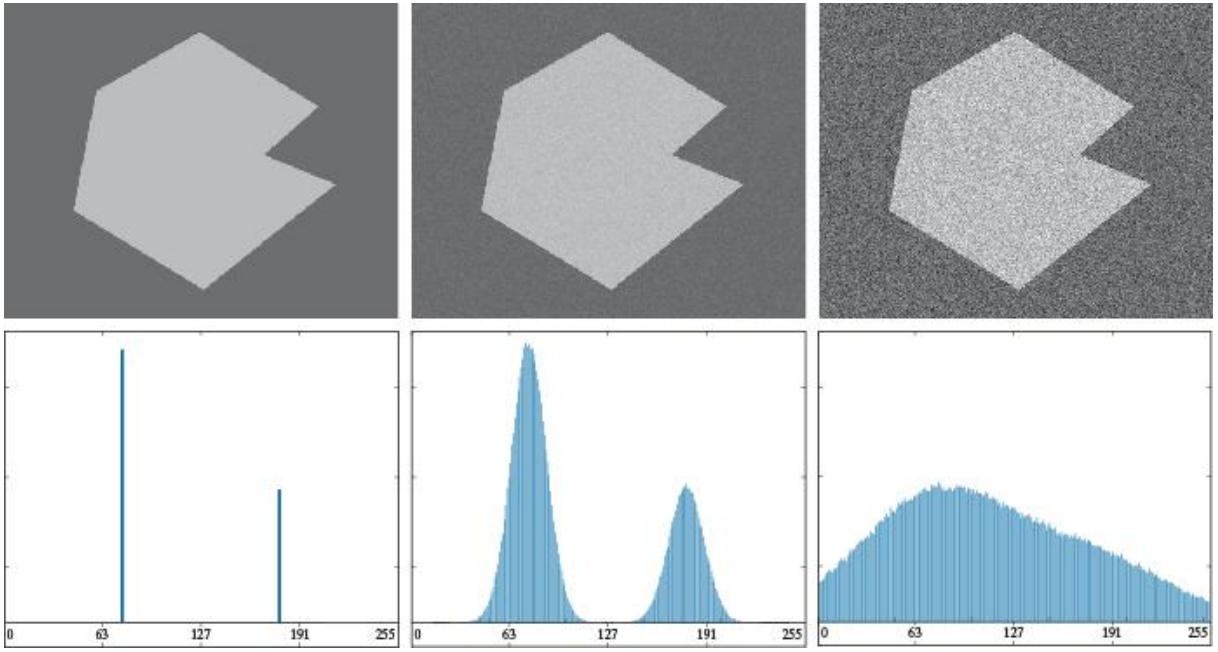
$T = 52$

Practica:

- Usa la función de OpenCV threshold para hacer binarización simple de una imagen en escala de grises. Habrá que usar el argumento apropiado para ello.
- Averigua qué otros tipos de binarizaciones se pueden hacer.
- Consulta el tutorial: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

Umbralización global:

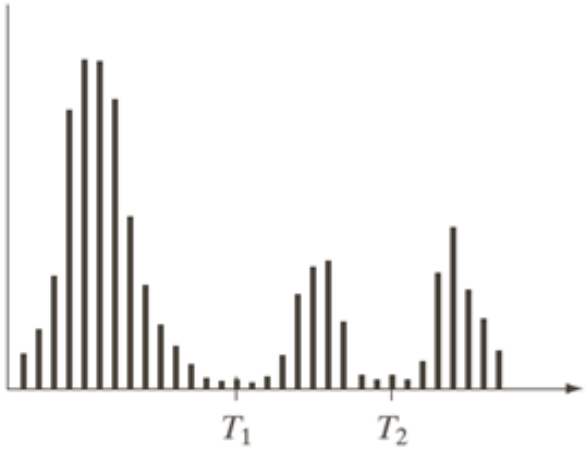
NO siempre en el histograma se distinguen mediante picos los objetos del fondo.



Segmentación

Umbralización múltiple:

P. ej., histograma de una imagen f(x,y) con dos objetos claros sobre un fondo oscuro. Se puede modificar el método de forma que tengamos tres categorías o grupos de píxeles.



$$g(x,y) = \begin{cases} a & \text{si } f(x,y) > T_2 \\ b & \text{si } T_1 < f(x,y) \leq T_2 \\ c & \text{si } f(x,y) \leq T_1 \end{cases}$$

Necesitamos dos umbrales.

Umbralización global: Método de Otsu

$f(x, y)$ = imagen $M \times N$ con L niveles de gris $\{r_1, \ldots, r_L\}$
n_i = número de píxeles con intensidad r_i.
Para cada $\boldsymbol{T = r_i}$:

1. Define dos clases:

   $C1$ = pixeles con valor de intensidad en $[r_1, r_i - 1]$

   $C2$ = pixeles con valor de intensidad en $[r_i, r_L]$

Umbralización global: Método de Otsu

$f(x, y)$ = imagen $M \times N$ con L niveles de gris $\{r_1, \ldots, r_L\}$
n_i = número de píxeles con intensidad r_i.
Para cada posible umbral $\boldsymbol{T = r_i}$:

2. Para cada clase, calcula las variables estadísticas:

   - ✓ Peso
   - ✓ Media

Umbralización global: Método de Otsu

$f(x, y)$ = imagen $M \times N$ con L niveles de gris $\{r_1, \dots, r_L\}$
n_i = número de píxeles con intensidad r_i.
Para cada $\boldsymbol{T = r_i}$:

2. Para cada clase, calcula las variables estadísticas:

- ✓ Peso
- ✓ Media

→ Proporción de píxeles en la clase respeto al total de píxeles de la imagen.

$$w_C = \frac{N_C}{M \cdot N}$$

N_C = número de píxeles en la clase C.

Umbralización global: Método de Otsu

$f(x, y)$ = imagen $M \times N$ con L niveles de gris $\{r_1, \dots, r_L\}$
n_i = número de píxeles con intensidad r_i.
Para cada $\boldsymbol{T = r_i}$:

2. Para cada clase, calcula las variables estadísticas:

- ✓ Peso
- ✓ Media ⟶ Intensidad media dentro de C.

$$\mu_C = \frac{r_1 n_1 + \dots + r_{k-1} n_{k-1}}{N_C}$$

Umbralización global: Método de Otsu

$f(x, y) =$ imagen $M \times N$ con L niveles de gris $\{r_1, \ldots, r_L\}$
n_i = número de píxeles con intensidad r_i.
Para cada $\boldsymbol{T = r_i}$:

3. Calcula una nueva variable estadística, llamada **varianza entre las clases**, dada por:

$$\sigma^2{}_{C1,C2}[T = r_i] = w_{C1} w_{C2} (\mu_{C1} - \mu_{C2})^2$$

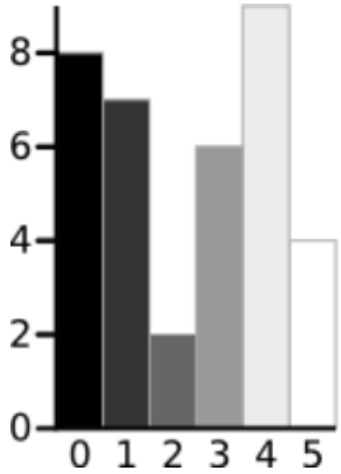
Umbralización global: Método de Otsu

$$T = r_2 \rightarrow {\sigma^2}_{C1,C2}[T = r_2]$$
$$T = r_3 \rightarrow {\sigma^2}_{C1,C2}[T = r_3]$$
$$\vdots$$
$$T = r_{L-1} \rightarrow {\sigma^2}_{C1,C2}[T = r_{L-1}]$$

$$\boldsymbol{T = \max_{i=2,\ldots,L-1} \{{\sigma^2}_{C1,C2}[T = r_i]\}}$$

Es decir, se toma como valor umbral final aquel nivel de gris de la imagen que haga que la dispersión entre las intensidades de las dos clases que se definen (valores menores y mayores que dicho nivel, respectivamente) sea la mayor posible.

Umbralización global: Método de Otsu. Ejemplo

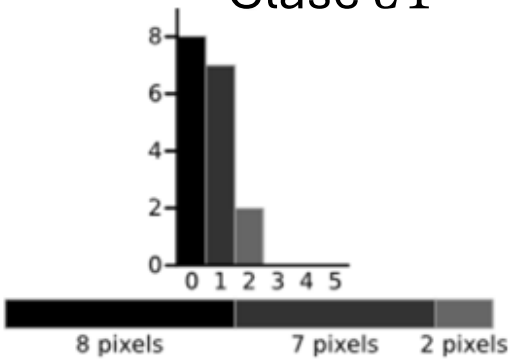


$f(x, y)$ imagen de tamaño 6×6 con 6 niveles de gris $\{0,1,2,3,4,5\}$.

Calculamos la varianza entre clases para T= todos los niveles de gris

Umbralización global: Método de Otsu. Ejemplo

Clase $C1$

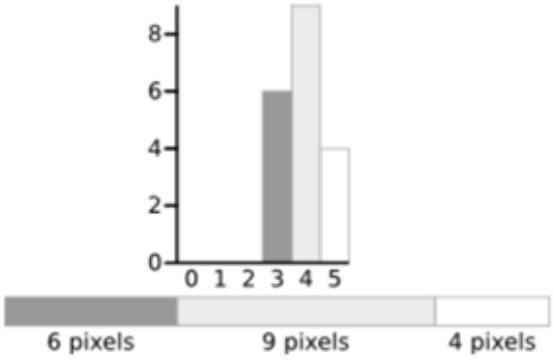


$$\text{Weight} \quad W_b = \frac{8+7+2}{36} = 0.4722$$

$$\text{Mean} \quad \mu_b = \frac{(0\times 8)+(1\times 7)+(2\times 2)}{17} = 0.6471$$

$$\boldsymbol{T = 3}$$

Clase $C2$

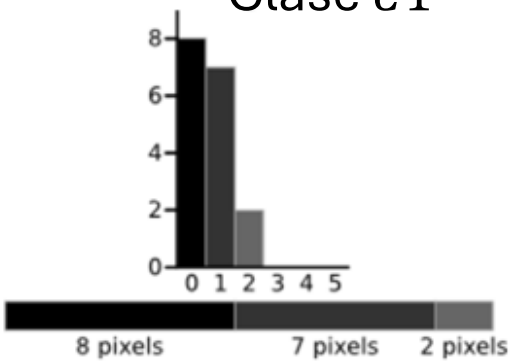


$$\text{Weight} \quad W_f = \frac{6+9+4}{36} = 0.5278$$

$$\text{Mean} \quad \mu_f = \frac{(3\times 6)+(4\times 9)+(5\times 4)}{19} = 3.8947$$

Umbralización global: Método de Otsu. Ejemplo

Clase $C1$



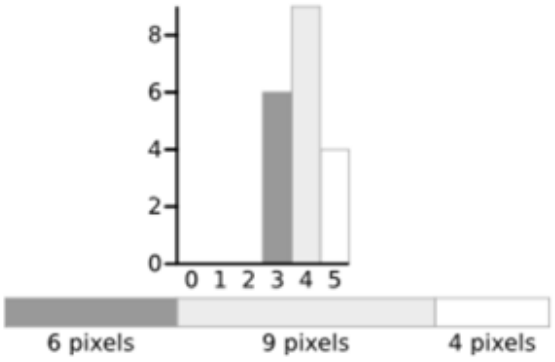
$$\text{Weight} \quad W_b = \frac{8+7+2}{36} = 0.4722$$

$$\text{Mean} \quad \mu_b = \frac{(0\times 8)+(1\times 7)+(2\times 2)}{17} = 0.6471$$

$T = 3$

Varianza entre las clases: $\sigma^2_B = 2.6287$

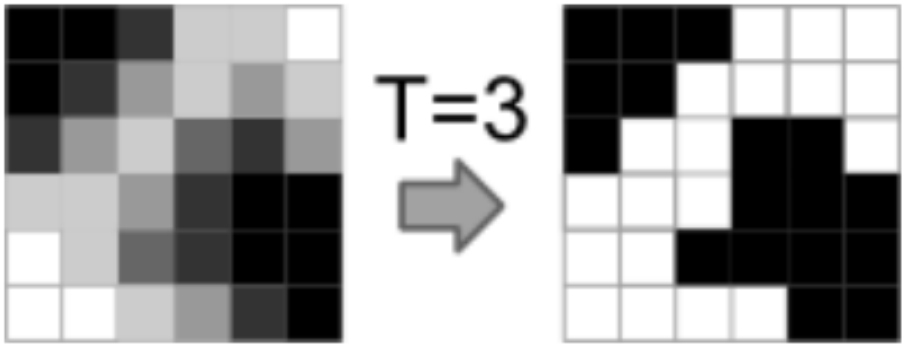
Clase $C2$



$$\text{Weight} \quad W_f = \frac{6+9+4}{36} = 0.5278$$

$$\text{Mean} \quad \mu_f = \frac{(3\times 6)+(4\times 9)+(5\times 4)}{19} = 3.8947$$

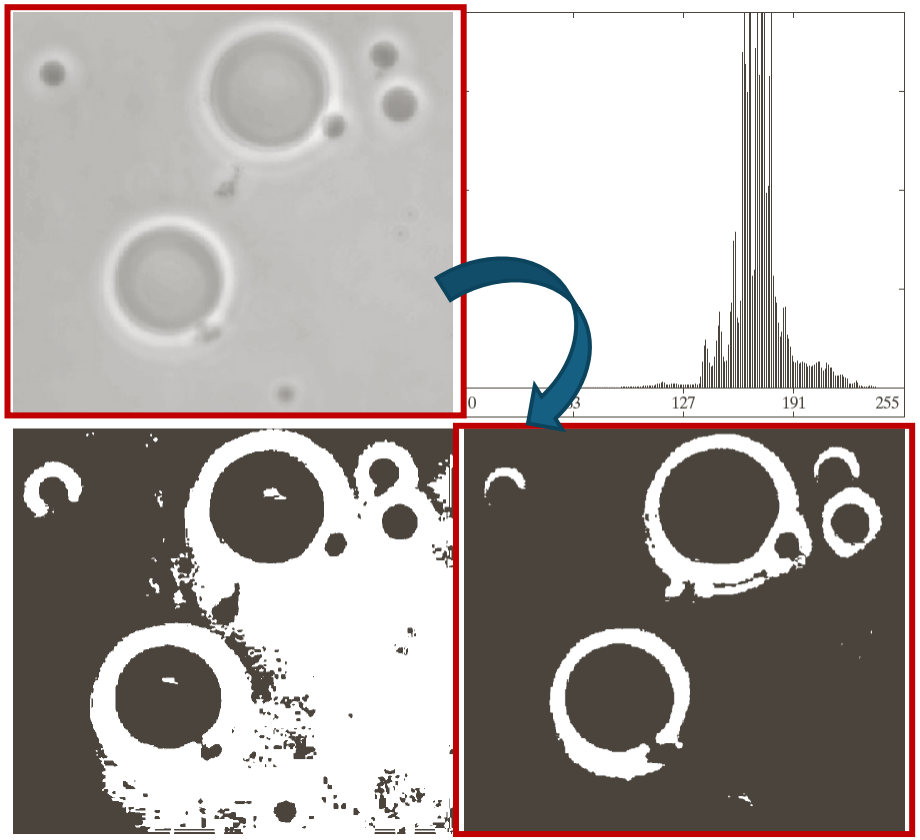
Umbralización global: Método de Otsu. Ejemplo



Practica:

- Averigua qué argumento hay que añadir a la función threshold para calcular la umbralización de Otsu. Compáralo con una binarización simple.
- Consulta el tutorial: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

Umbralización global: Método de Otsu. Ejemplo



a b
c d

FIGURE 10.39
(a) Original image.
(b) Histogram (high peaks were clipped to highlight details in the lower values).
(c) Segmentation result using the basic global algorithm from Section 10.3.2.
(d) Result obtained using Otsu's method.
(Original image courtesy of Professor Daniel A. Hammer, the University of Pennsylvania.)

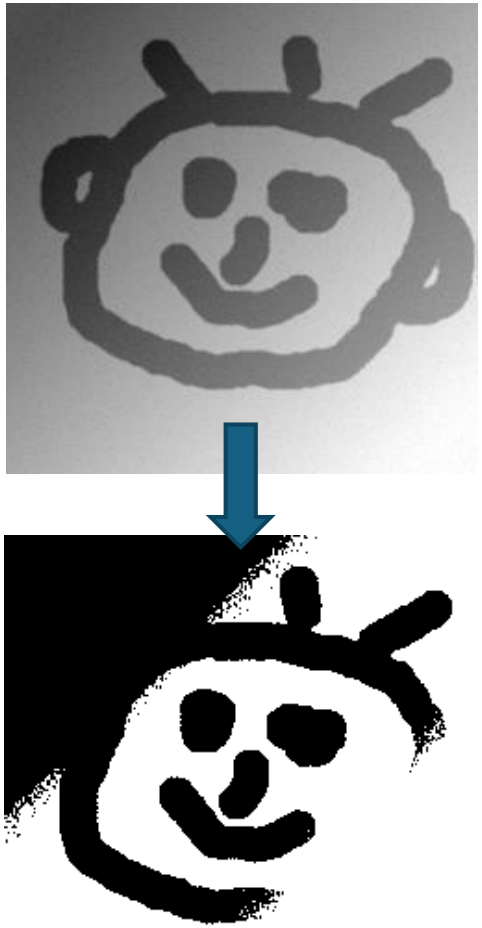
Segmentación

Umbralización local o adaptativa:

El ruido o una deficiente iluminación en la imagen hacen que no sea conveniente el uso de umbrales globales para toda la imagen.

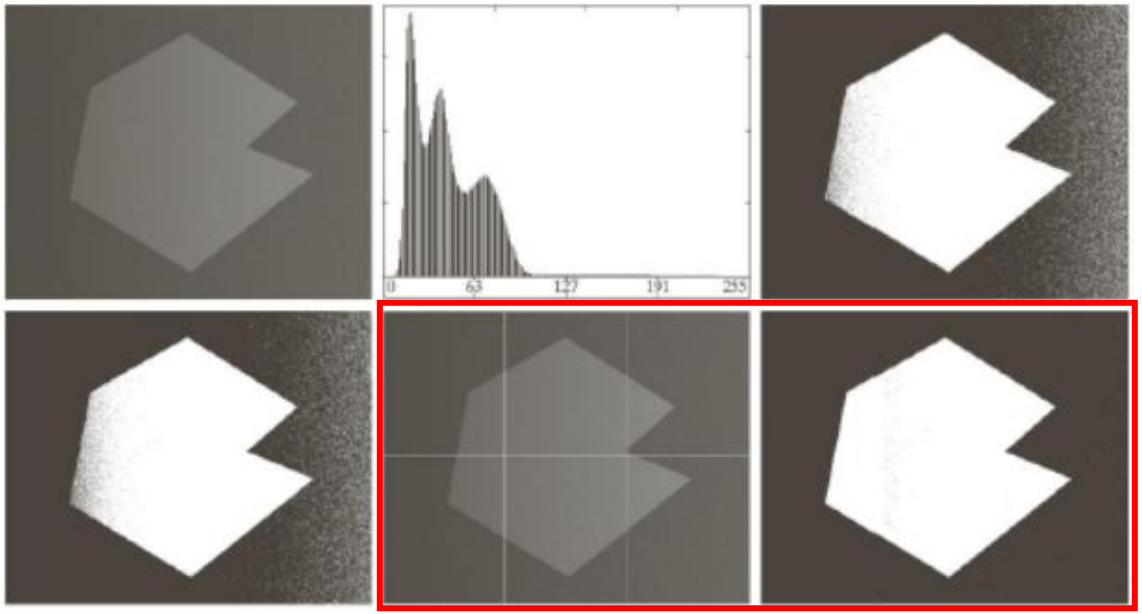
Dos técnicas para elegir umbrales variables:

- Partición de la imagen.
- Propiedades locales de la imagen.



$T = 127$

Umbralización local o adaptativa: Partición de la imagen.



a b c
d e f

FIGURE 10.46 (a) Noisy, shaded image and (b) its histogram. (c) Segmentation of (a) using the iterative global algorithm from Section 10.3.2. (d) Result obtained using Otsu's method. (e) Image subdivided into six subimages. (f) Result of applying Otsu's method to each subimage individually.

Segmentación

Umbralización local o adaptativa: Propiedades locales de la imagen.

- Calcular un umbral T_{xy} para cada pixel (x, y) de la imagen basándose en una o más propiedades específicas computadas en un entorno de vecindad S_{xy} del píxel.

  Ejemplo: σ_{xy} = desviación típica y m_{xy} = media en S_{xy}

$$T_{xy} = a\,\sigma_{xy} + b\,m_{xy},\ \ a, b > 0$$

Luego:

$$g(x,y) = \begin{cases} 1 & \text{si } f(x,y) > T_{xy} \\ 0 & \text{si } f(x,y) \leq T_{xy} \end{cases}$$

Umbralización local o adaptativa: Propiedades locales de la imagen.

- Calcular un umbral T_{xy} para cada pixel (x, y) de la imagen basándose en una o más propiedades específicas computadas en un entorno de vecindad S_{xy} del píxel.

- Método muy estable frente a cambios de luminosidad localizados, pero la potencia de cálculo necesaria se incrementa mucho, ya que para cada pixel se debe calcular un nuevo valor umbral.

Practica:

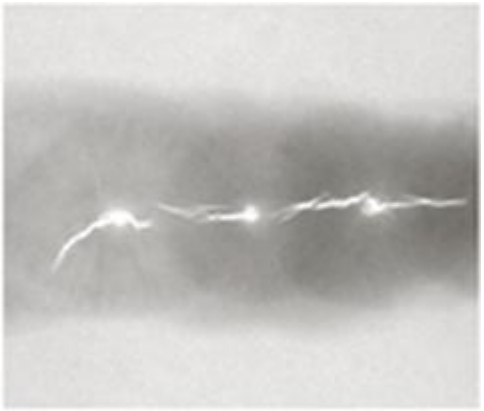
- Averigua qué opciones ofrece OpenCV para hacer umbralización adaptativa. ¿Qué tipos de condiciones locales implementa en cada caso?
- Busca imágenes en las que funcione mejor una umbralización adaptativa.
- Consulta el tutorial:

  https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

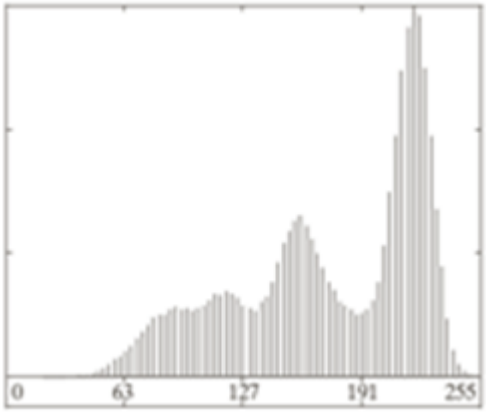
Crecimiento de regiones

- Es un procedimiento que agrupa los píxeles o subregiones de la imagen en regiones mayores basándose en un criterio prefijado.
- Se empieza con unos *puntos "semillas"* para formar una determinada región.
- Se van anexando píxeles vecinos (8-adyacentes o 4-adyacentes) que cumplan la propiedad especificada (por ejemplo, que estén en un rango de nivel de gris determinado).
- Se establece una *condición de parada*.

Segmentación



Imagen



Histograma

Puntos semilla: puntos con valor de intensidad 255

Resultado

Criterios para aumentar una región:

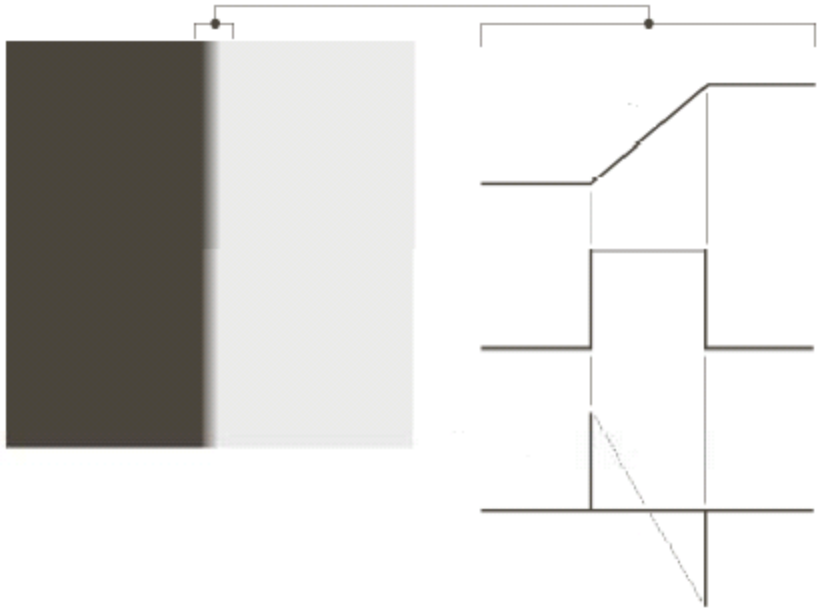
1. Puntos tal que la diferencia en valor absoluto con un píxel semilla sea menor que 65.

2. 8-adyacencia con algún píxel de la región.

Segmentación

Detección de bordes

Los métodos de segmentación basados en la detección de cambios bruscos de intensidad (nivel de gris) usan técnicas de derivación.



Detección de bordes

Los métodos de segmentación basados en la detección de cambios bruscos de intensidad (nivel de gris) usan técnicas de derivación.

- **Pasos fundamentales** en la detección de bordes:

  - **Paso 1**: Realizar un suavizado de la imagen para eliminar el ruido.

  - **Paso 2**: Detectar los posibles candidatos a ser puntos del borde (derivación).

  - **Paso 3**: Seleccionar, de entre los candidatos, aquellos que pertenecen realmente al borde.

Detector Canny:

- Es el operador borde más popular. Se caracteriza por evitar la ruptura de los bordes de los objetos.

- Su fundamento se basa en un proceso de optimización, teniendo en cuenta los siguientes dos objetivos:

1. Disminuir todo lo posible la distancia entre el borde detectado y el borde real.

2. No identificar un borde por un único pixel, sino por un conjunto de píxeles que tengan cierta conectividad.

Detector Canny:

Los pasos principales del algoritmo son:

- **Paso 1**: Se realiza una convolución con un filtro gaussiano para suavizar la imagen (eliminar ruidos).

Detector Canny:

Los pasos principales del algoritmo son:

- **Paso 1**: Se realiza una convolución con un filtro gaussiano para suavizar la imagen (eliminar ruidos).

- **Paso 2**: Se calcula la imagen gradiente de la imagen suavizada. También se determina la dirección del vector gradiente.

$$M(x,y) \approx |g_x| + |g_y|$$

$$\alpha(x,y) = \tan^{-1}(g_y/g_x)$$

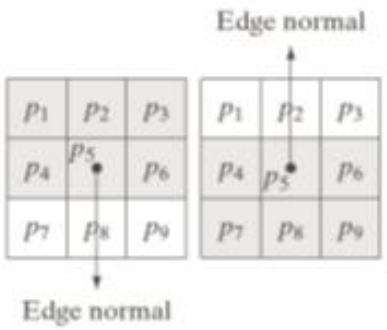
−1	−2	−1	−1	0	1
0	0	0	−2	0	2
1	2	1	−1	0	1

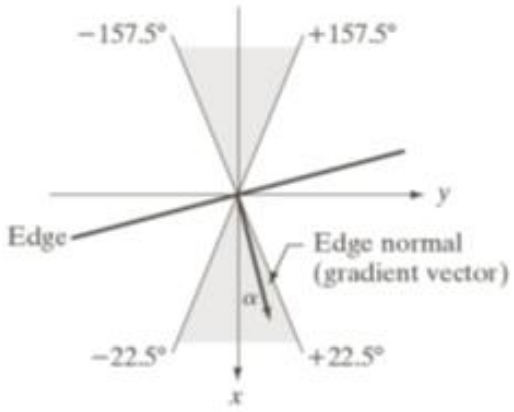
g_x Sobel g_y

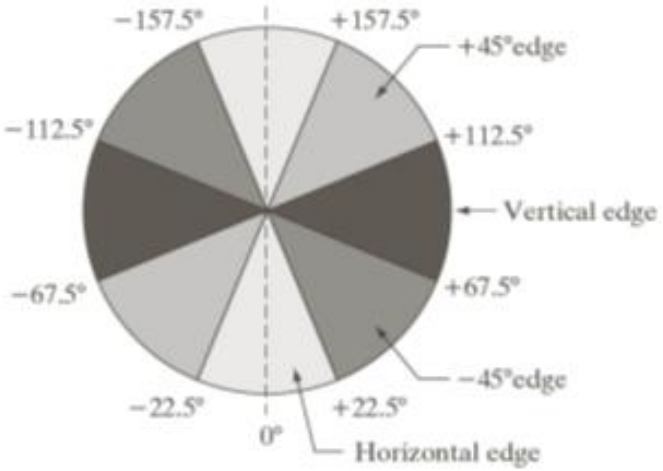
Segmentación

Detector Canny:

- **Paso 3**: Se aplica un procedimiento para eliminar (igualar a cero) aquellos píxeles que no son máximos locales en la dirección del gradiente (que es perpendicular al borde), considerando un entorno 3 x 3.







Las direcciones se discretizan en 8 posibles valores.

Detector Canny:

- **Paso 3**: Se aplica un procedimiento para eliminar (igualar a cero) aquellos píxeles que no son máximos locales en la dirección del gradiente (que es perpendicular al borde), considerando un entorno 3 x 3.

Sean d_1, d_2, d_3 y d_4 las cuatro direcciones básicas (horizontal, -45º, vertical y +45º). Entonces, para cada pixel (x,y):

1. Encuentra la dirección d_k más cercana a $\alpha(x,y)$.

2. Si M(x,y) es menor que al menos uno de sus dos vecinos a lo largo de la dirección d_k, sea $g_N(x,y)=0$; en caso contrario $g_N(x,y)=M(x,y)$.

Detector Canny:

- **Paso 4**: Se realiza un proceso de doble umbralización para determinar los píxeles del borde:

  - se marcan los píxeles con valor por encima de un umbral T_1 (se localizan las semillas borde);

  - se marcan aquellos píxeles conectados a los primeros cuyo valor esté por encima de un segundo umbral T_2 ($T_2 < T_1$).

Esto eliminará falsos bordes al mismo tiempo que permite un resultado conexo.

Segmentación

Detector Canny:

El filtro Gaussiano se ha realizado para σ=4 y una máscara de tamaño 25x25. Los umbrales considerados han sido T_1=0.1 y T_2=0.04 (porque los valores están normalizados entre 0 y 1).

Practica:

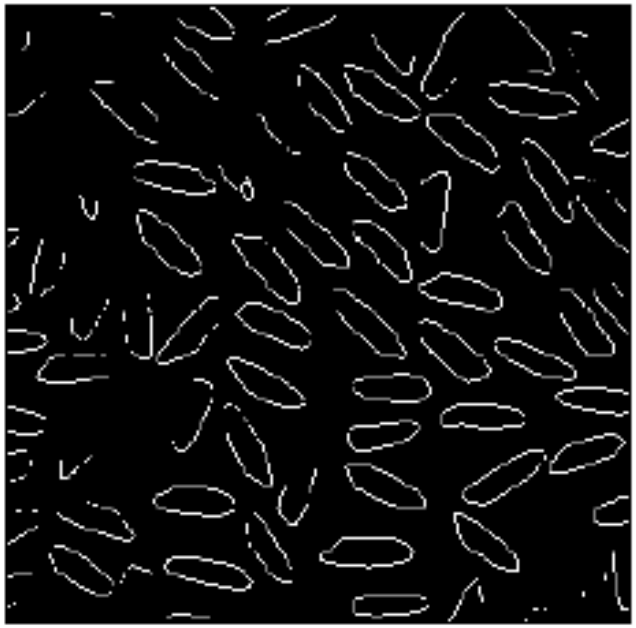
- Usa la función de OpenCV Canny para obtener los bordes de una imagen.

  Relaciona los parámetros de la función con lo explicado en teoría.

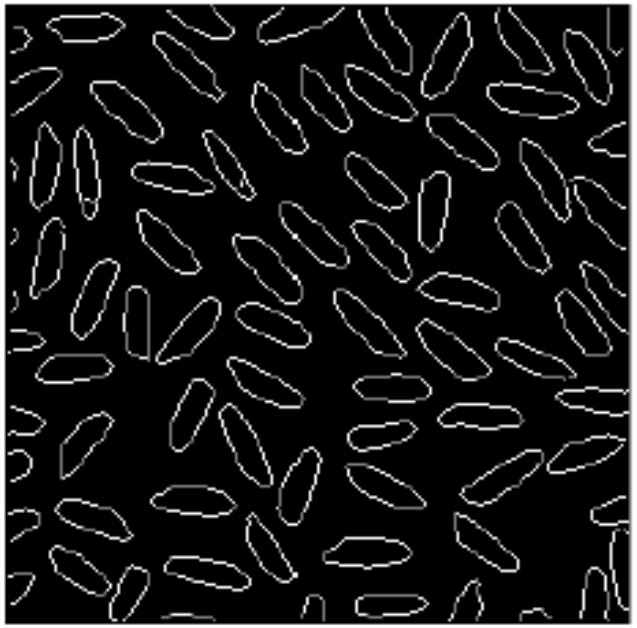
- https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html#autotoc_md1300

Segmentación

- **Comparando Sobel y Canny:**



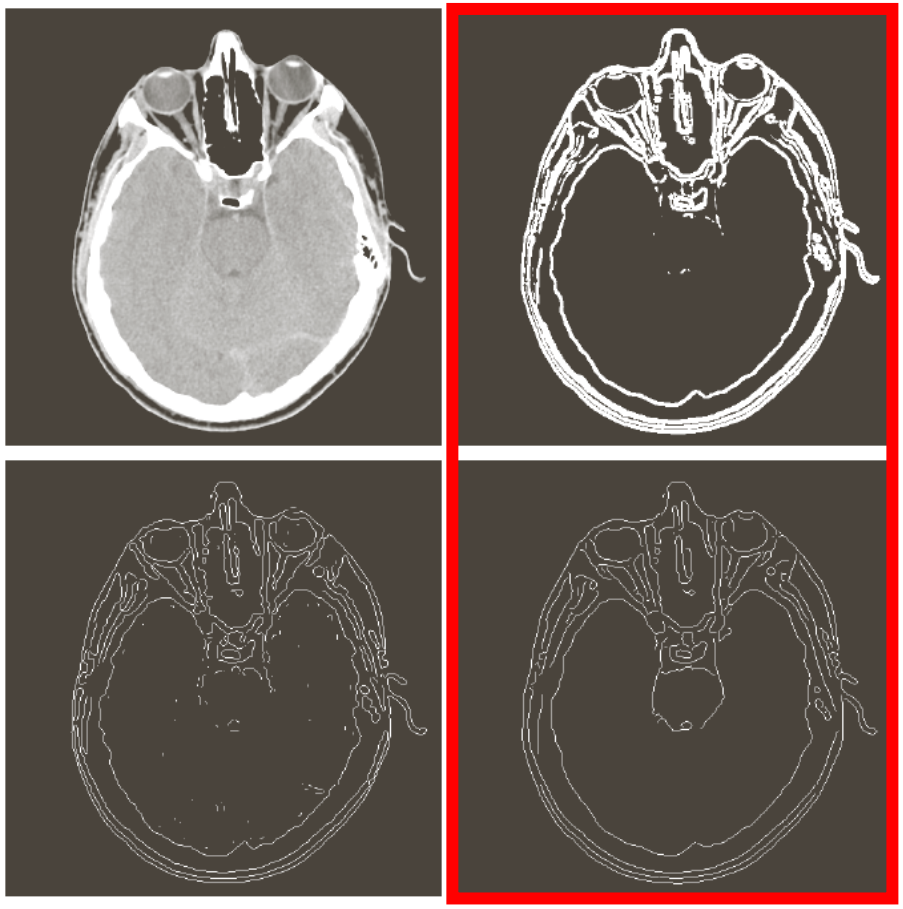
Sobel Filter



Canny Filter

Segmentación

- **Comparando Sobel y Canny:**



a b
c d

FIGURE 10.26
(a) Original head CT image of size 512 × 512 pixels, with intensity values scaled to the range [0, 1]. (b) Thresholded gradient of smoothed image. (c) Image obtained using the Marr-Hildreth algorithm. (d) Image obtained using the Canny algorithm. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

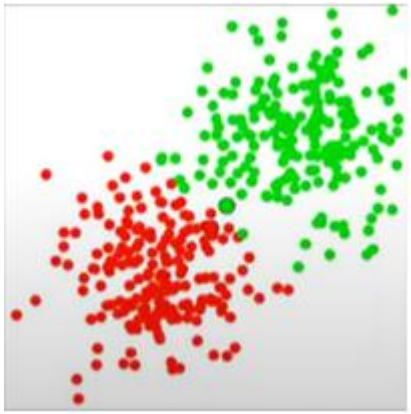
Segmentación

- **Agrupamiento (clustering):**

  Los algoritmos de agrupamiento se utilizan para agrupar píxeles en diferentes categorías o clases según su similitud.

  Los píxeles se agrupan en función de características como la proximidad espacial o los valores de intensidad.

  Se trata de procesos no supervisados,
  con frecuencia iterativos:
  - K-means
  - Mean shift



Segmentación

- **Ejemplo: K-means**

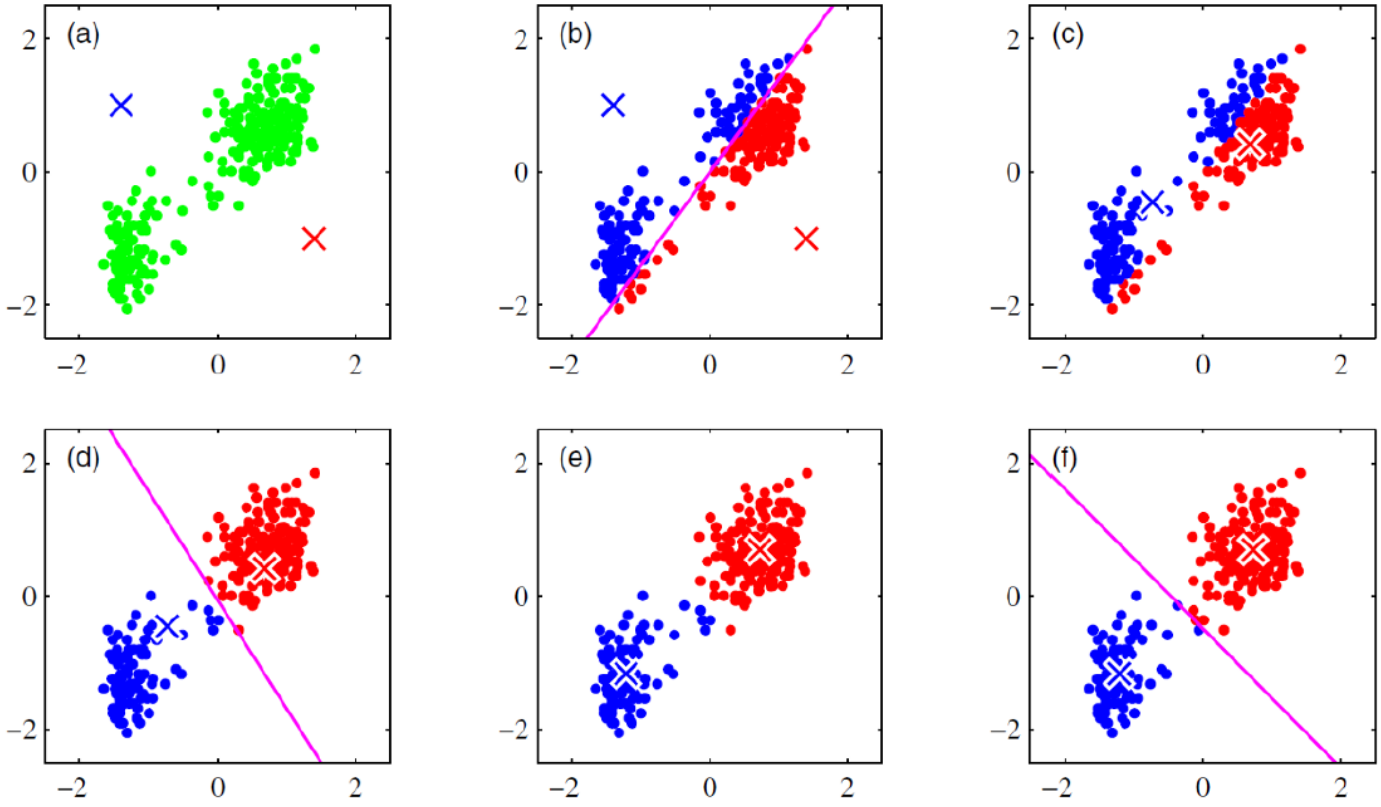


Figure 5.16 The k-means algorithm starts with a set of samples and the number of desired clusters (in this case, $k = 2$) (Bishop 2006) © 2006 Springer. It iteratively assigns samples to the nearest mean, and then re-computes the mean center until convergence.

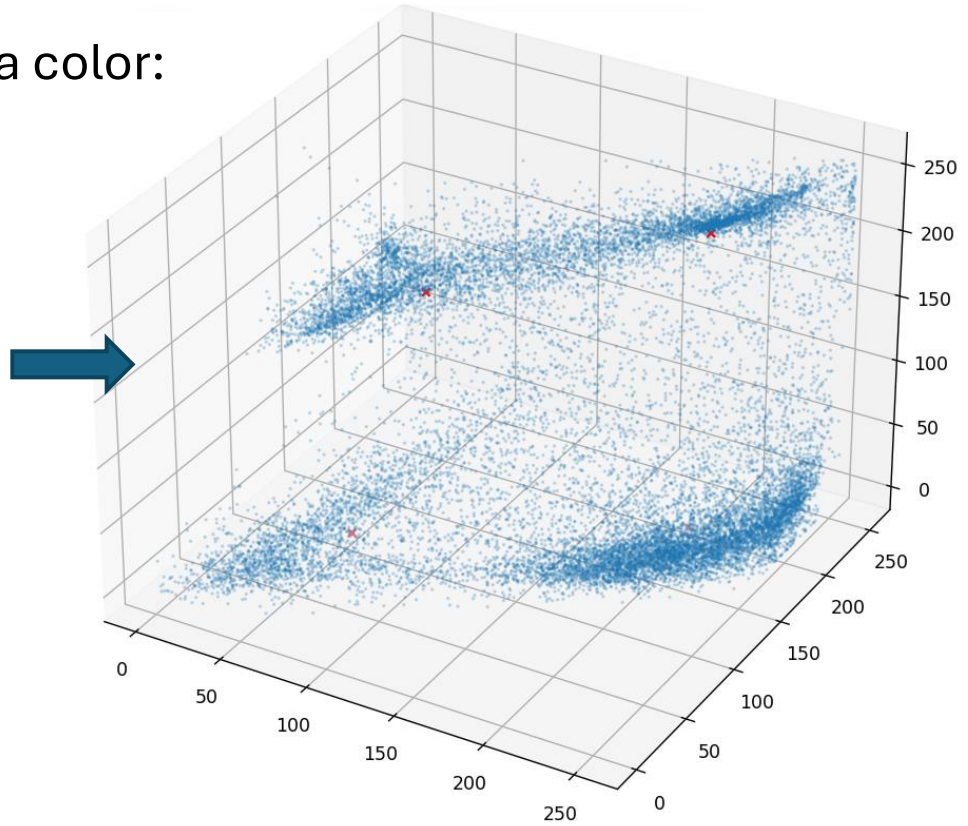
Fuente: Szeliski, 2022.

Segmentación

- **Agrupamiento (clustering): K-means**

  En imágenes a color:

Imagen RGB
$102\ x\ 180$ pixeles



Representación 3D:

$102 \cdot 180 = 18.360$ puntos

Eje x = canal R
Eje y = canal G
Eje z = canal B

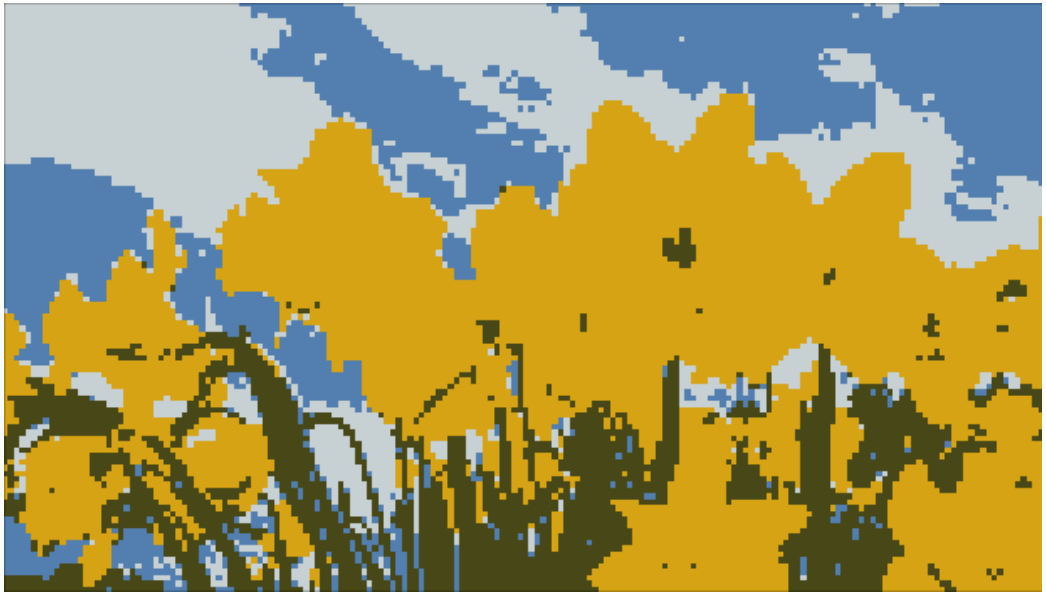
× = clusters (K=4)

Segmentación

- **Agrupamiento (clustering): K-means**

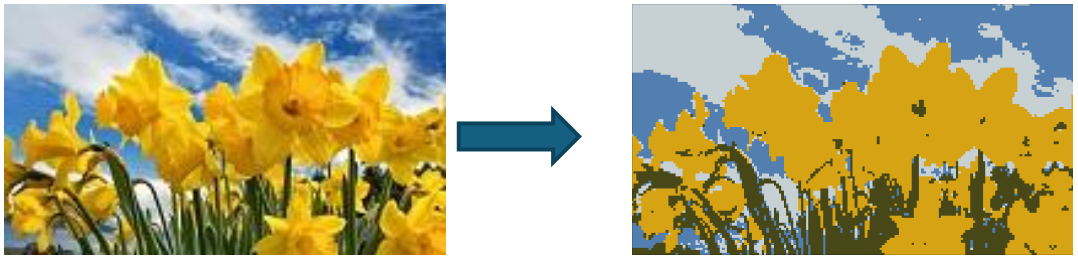
  En imágenes a color:

Imagen RGB
$102\ x\ 180$ pixeles



Segmentación

- **Agrupamiento (clustering): K-means**



Practica:

- Usa la función de OpenCV kmeans para hacer una segmentación de pastes de una imagen sencilla como la del ejemplo. En realidad estamos haciendo una cuantización del color.
- Investiga cómo son los argumentos de entrada y cómo es la salida.
- https://docs.opencv.org/4.x/d1/d5c/tutorial_py_kmeans_opencv.html

Segmentación

- **Segmentación asistida:**

  También conocida como segmentación interactiva o semiautomática, es un enfoque en el que se combina la intervención humana con algoritmos de segmentación para obtener resultados precisos y controlados.

  Esto se realiza a través de interacciones directas con la imagen, por ejemplo:

  - La selección de puntos de inicio o semillas: ej. Watershed
  - La delimitación de regiones de interés: ej. GrabCut

Segmentación

- **Segmentación asistida:**

Requiere que el usuario indique con verde algunos de los píxeles incluidos en el objeto a segmentar, y con azul algunos de los píxeles que no se deben incluir.

Requiere indicar con un recuadro rojo el objeto a segmentar.

Segmentación

- **Segmentación asistida:**

Requiere que el usuario indique con verde algunos de los píxeles incluidos en el objeto a segmentar, y con azul algunos de los píxeles que no se deben incluir.

Requiere indicar con un recuadro rojo el objeto a segmentar.

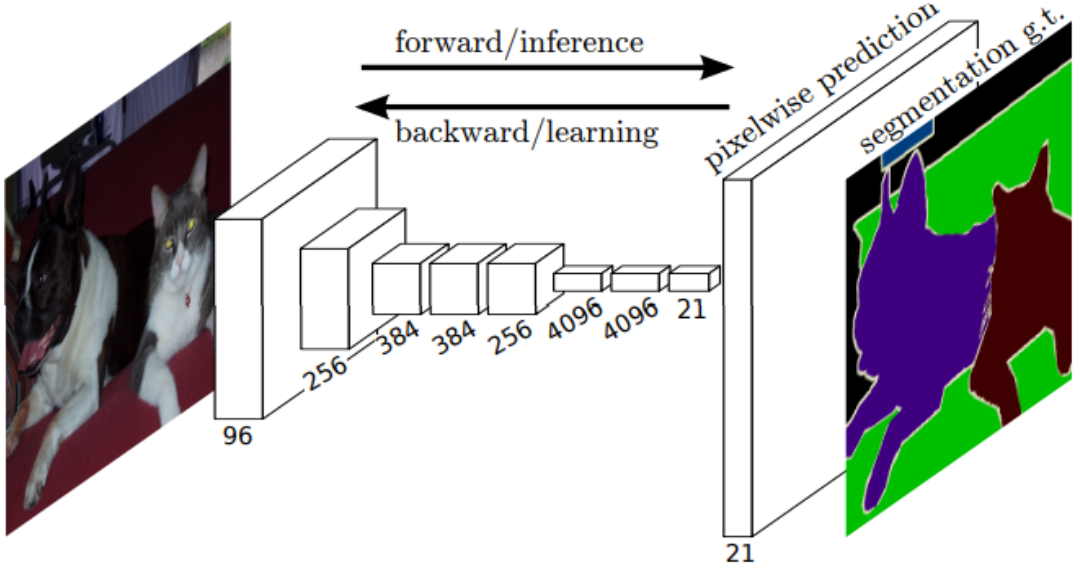
Segmentación

• Los *algoritmos de aprendizaje profundo* de segmentación se basan principalmente en Redes Neuronales de Convolución (CNN), que aprenden características y patrones complejos en imágenes para realizar la segmentación.

•Esta segmentación suele resultar de la combinación de detección + clasificación.

•Se puede distinguir:

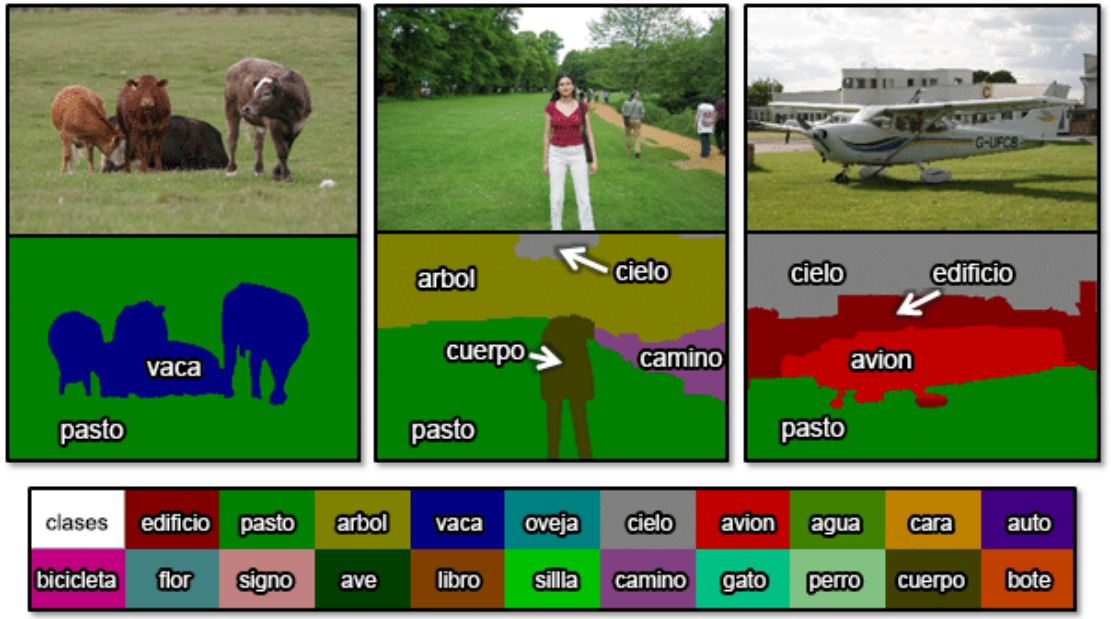
- •Segmentación semántica
- •Segmentación instanciada



Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Segmentación

- **Segmentación semántica:**
  - Asigna una etiqueta a cada píxel para indicar la clase a la que pertenece.

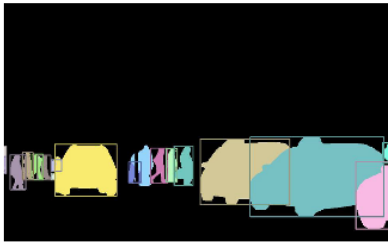


Segmentación

- **Segmentación instanciada:**

  - Además de etiquetar cada píxel con una categoría semántica, se distinguen las diferentes instancias de objetos dentro de la misma categoría. Por lo tanto, cada objeto individual se etiqueta de manera única.

  - Por ejemplo, en una imagen con varias personas y varios coches, se diferenciarían y etiquetarían cada persona y coche individualmente



Segmentación

- **Segmentación semántica e instanciada:**

  Los enfoques basados en aprendizaje profundo han demostrado ser muy efectivos en una amplia gama de aplicaciones de segmentación semántica e instanciada.

  - Mask R-CNN
  - FCN (Fully Convolutional Network)
  - U-Net
  - Etc

Segmentación

Bibliografía y recursos:

- Szeliski, Richard. *Computer vision: algorithms and applications*. Springer Nature, 2022.

- Segmenting an image. https://ipython-books.github.io/113-segmenting-an-image/