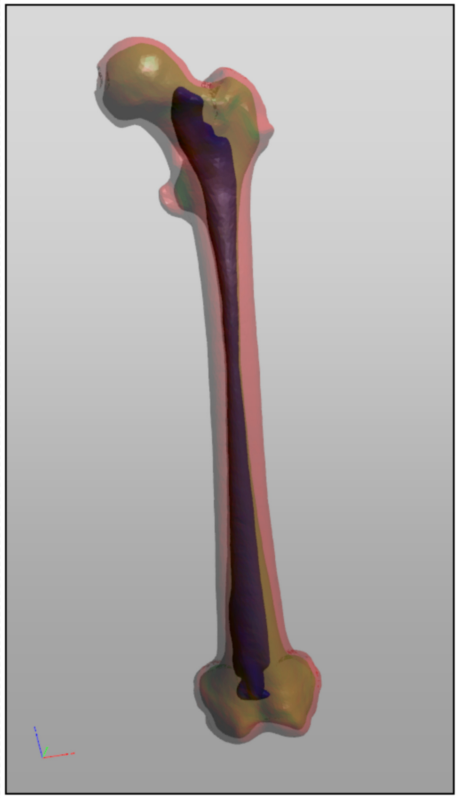
Segmentación de imágenes

Definición

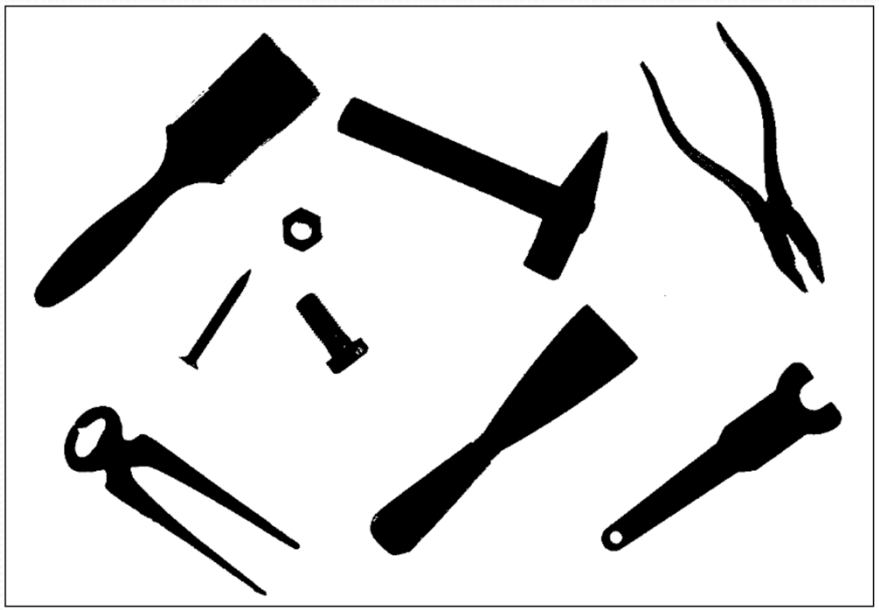
- **Segmentación de imágenes**. Proceso de convertir una imagen en una colección de regiones.
- Se puede procesar solo los segmentos importantes de la imagen en lugar de procesar la imagen completa.

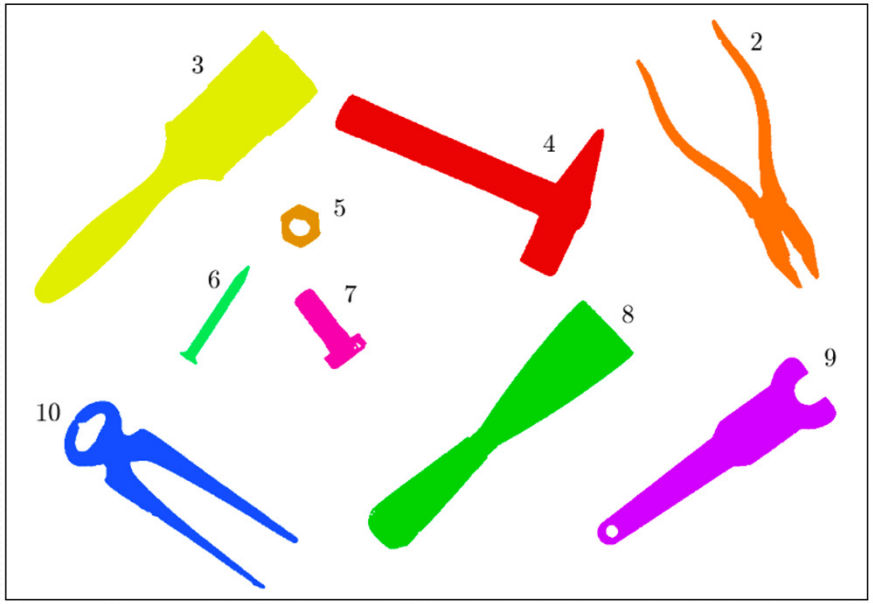
Ejemplo



By Newe A, Ganslandt T - Newe A, Ganslandt T (2013) Simplified Generation of Biomedical 3D Surface Model Data for Embedding into 3D Portable Document Format (PDF) Files for Publication and Education. PLoS ONE 8(11): e79004. doi:10.1371/journal.pone.0079004, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=30052549

Ejemplo





Fuente: DIP-Java, pp. 195, 205

Temario

- Dos métodos:

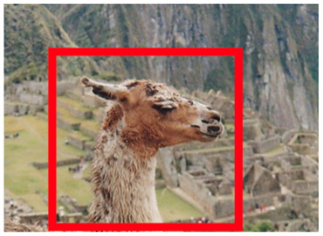
1. GrabCut
2. Watershed

GrabCut

- Es un método para extraer objetos de una imagen.
- Clasifica los pixeles en frente (foreground) y fondo (background).








Fuente: https://www.microsoft.com/en-us/research/wp-content/uploads/2004/08/siggraph04-grabcut.pdf

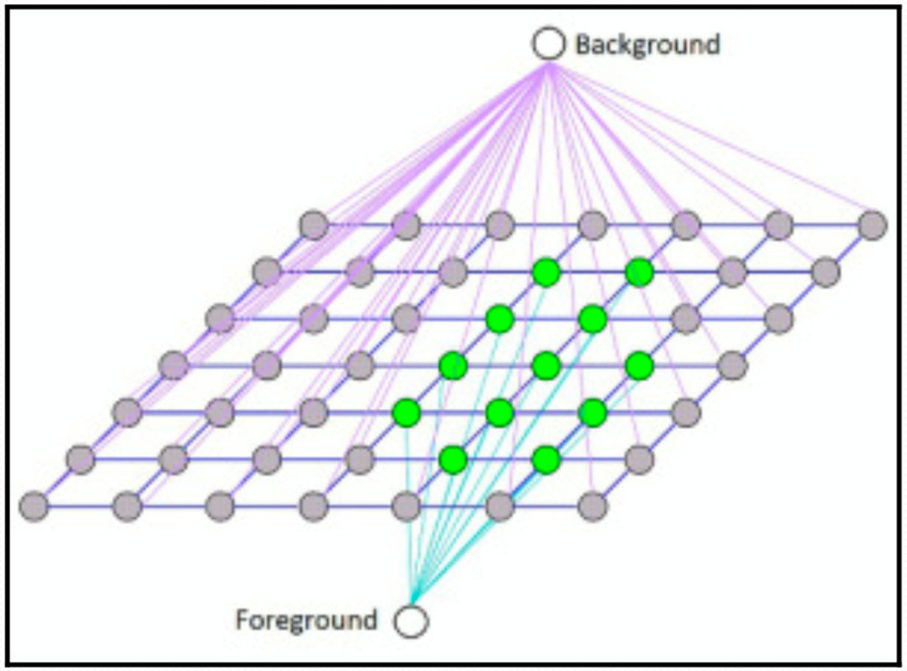
Algoritmo GrabCut

1. Definir un rectángulo que incluya al sujeto de la imagen.
2. El área afuera del rectángulo se clasifica automáticamente como fondo.
3. Los datos contenidos en el fondo se usan como referencia para distinguir áreas del fondo de las áreas del frente en el rectángulo definido en el paso 1.
4. Se utiliza un modelo gaussiano para clasificar los pixeles como fondo probable y frente probable.

Algoritmo GrabCut

5. Cada pixel se conecta virtualmente con los pixeles que lo rodean a través de aristas virtuales.
6. A cada arista se le asigna una probabilidad de ser frente o de ser fondo.
7. Cada pixel o se conecta a una terminal frente o a una terminal fondo.

Algoritmo GrabCut

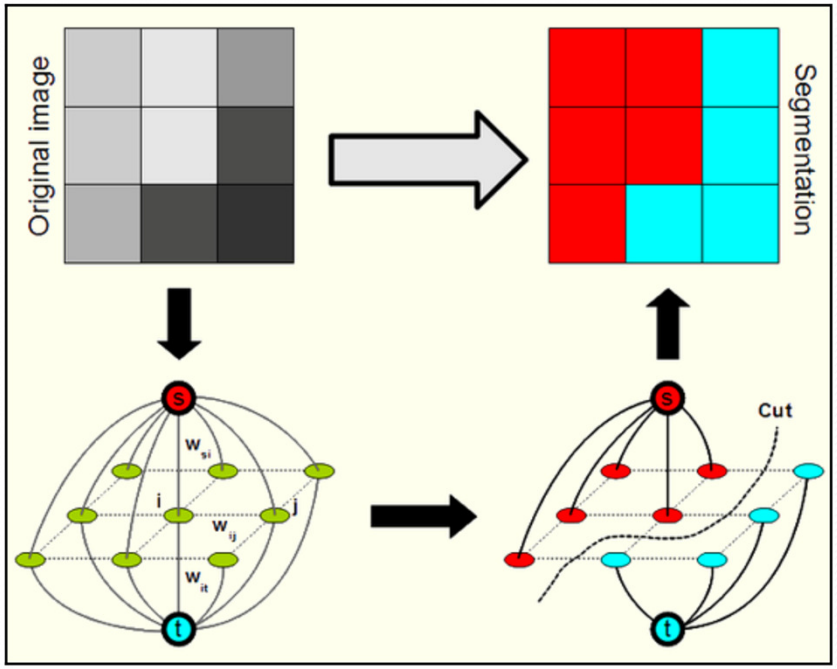


Fuente: LO4CV-P3, p. 99

Algoritmo GrabCut

8. Después de que cada pixel ha sido conectado a alguna terminal (frente o fondo), las aristas entre pixeles pertenecientes a diferentes terminales se cortan y la imagen queda segmentada en 2 partes.

Algoritmo GrabCut



Fuente: LO4CV-P3, p. 99

GrabCut en OpenCV

cv.grabCut(img, mask, rect, bgdModel, fgdModel, iterCount[, mode]) -> mask, bgdModel, fgdModel

- img – imagen de entrada de 3 canales de 8 bits
- mask – máscara de un canal de 8 bits
- rect – ROI que contiene un objeto segmentado
- bgdModel – matriz temporal para el fondo
- fgdModel – matriz temporal para el primer plano
- iterCount – número de iteraciones que debe realizar el algoritmo antes de devolver el resultado
- mode – modo de operación (p. e. cv.GC_INIT_WITH_RECT)

GrabCut en OpenCV

- Método cv2.grabCut().
- La matriz resultado contiene valores entre 0 y 3:

Enumerator	
GC_BGD Python: cv.GC_BGD	an obvious background pixels
GC_FGD Python: cv.GC_FGD	an obvious foreground (object) pixel
GC_PR_BGD Python: cv.GC_PR_BGD	a possible background pixel
GC_PR_FGD Python: cv.GC_PR_FGD	a possible foreground pixel

Fuente: https://docs.opencv.org/4.x/d7/d1b/group__imgproc__misc.html#gad43d3e4208d3cf025d8304156b02ba38

Ejemplo

- Extraer la estatua del ángel:

Ejemplo

- Se carga la imagen y se crea una máscara llena con ceros con la misma forma que la imagen.

```
# File: grabcut.py
source = cv.imread('../images/statue_small.jpg')
result = source.copy()
mask = np.zeros(result.shape[:2], np.uint8)
```

Ejemplo

- Se crean los modelos para el frente y el fondo llenos de ceros.

```
background = np.zeros((1, 65), np.float64)
foreground = np.zeros((1, 65), np.float64)
```

- GrabCut se va a inicializar con un rectángulo que contenga al sujeto.
- Los modelos del frente y el fondo se determinan con la áreas afuera del rectángulo inicial.
- Se define el rectángulo:

```
rect = (100, 1, 421, 378)
```

Ejemplo

- Ahora se invoca a GrabCut y se le pasan los modelos vacíos, la mascara y el rectángulo inicial.

```
cv.grabCut(result, mask, rect, background, foreground, 5,
                                        cv.GC_INIT_WITH_RECT)
```

- El 5 indica el número de iteraciones que el algoritmo va a correr sobre la imagen.

Ejemplo

- Al terminar GrabCut, mask tiene valores entre 0 y 3.
- 0 (cv.GC_BGD) es un pixel obvio de fondo.
- 1 (cv.GC_FGD) es un pixel obvio de frente.
- 2 (cv.GC_PR_BGD) es un pixel probable de fondo.
- 3 (cv.GC_PR_FGD) es un pixel probable de frente.
- Para visualizar el resultado, se desea pintar de negro el fondo de la imagen original y dejar el frente intacto.

Ejemplo

- Se crea una nueva máscara que tenga 0 en dónde la máscara original tiene 0 o 2 (fondo obvio y probable) y 1 en dónde la máscara original tenga 1 y 3 (frente obvio y probable).
- Se multiplica la nueva máscara por la imagen original para poner el fondo en negro y dejar intactos los pixeles del frente.

```
mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8’)
result = result * mask2[:, :, np.newaxis]
```

Ejemplo

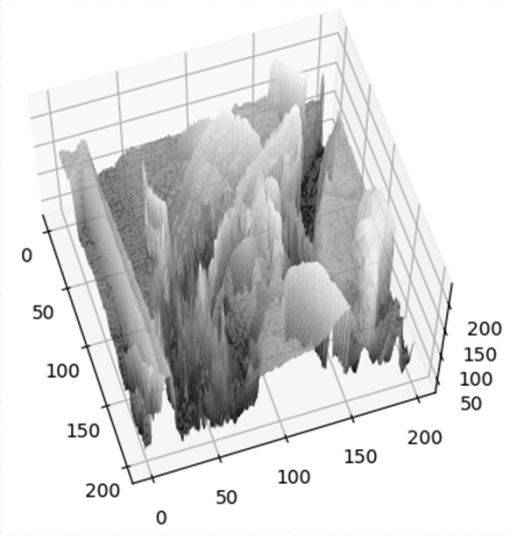
- Por último, se guarda una imagen con la imagen original y el resultado lado a lado.

```
stack_image = np.hstack((source, result))
cv.imwrite('statue_small_grabcut.jpg', stack_image)
```

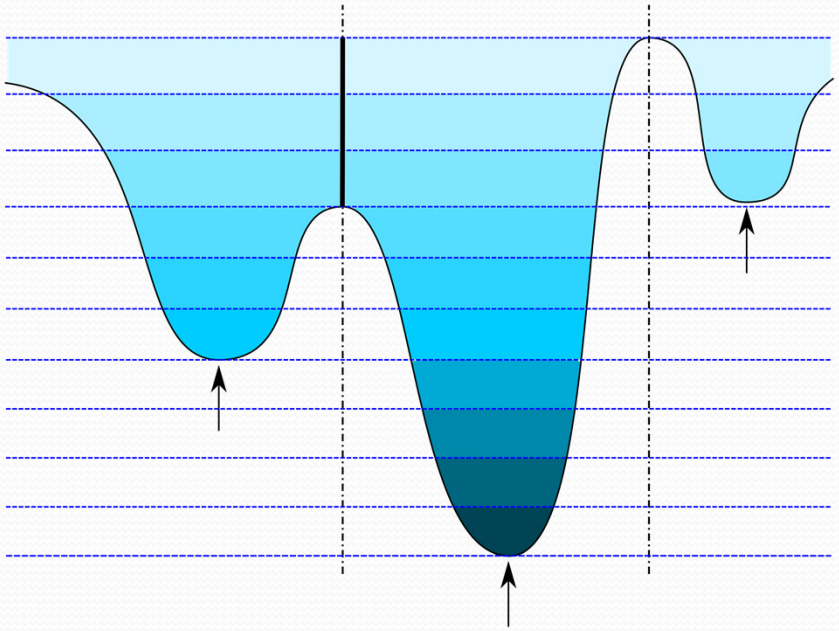
Resultado

Watershed

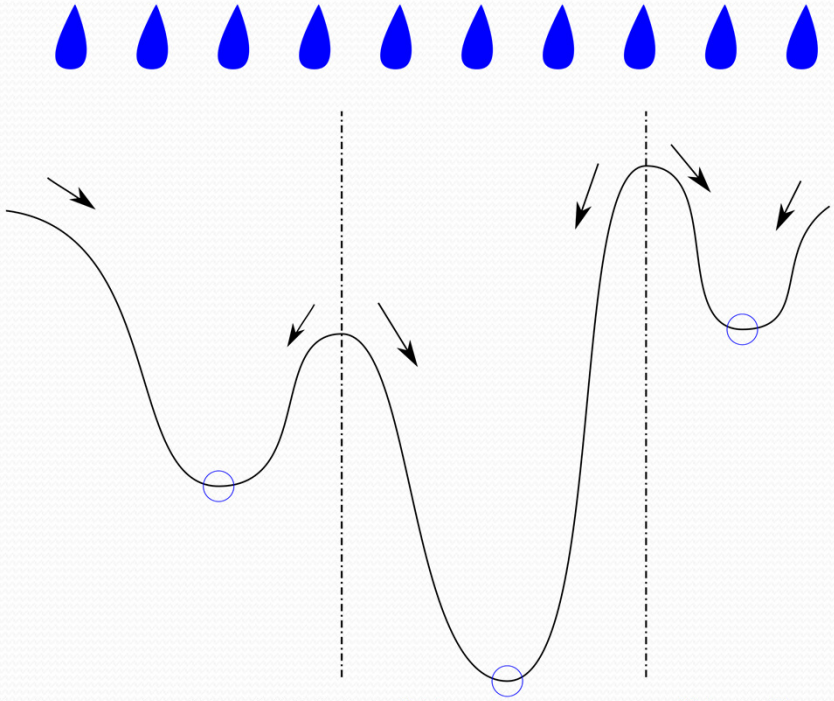
- Es un algoritmo para segmentar una imagen.
- Interpreta una imagen en escala de grises como un mapa topográfico.
- La intensidad de cada pixel es la altura del terreno.



Watershed – inundación



Watershed – lluvia



De Martynas Patasius - Trabajo propio, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10495705

Requisitos

1. Operaciones morfológicas.
2. Componentes conectados.
3. Transformada de distancia.

Operaciones morfológicas

Imagen original

Erosión

Dilation

Opening

Closing

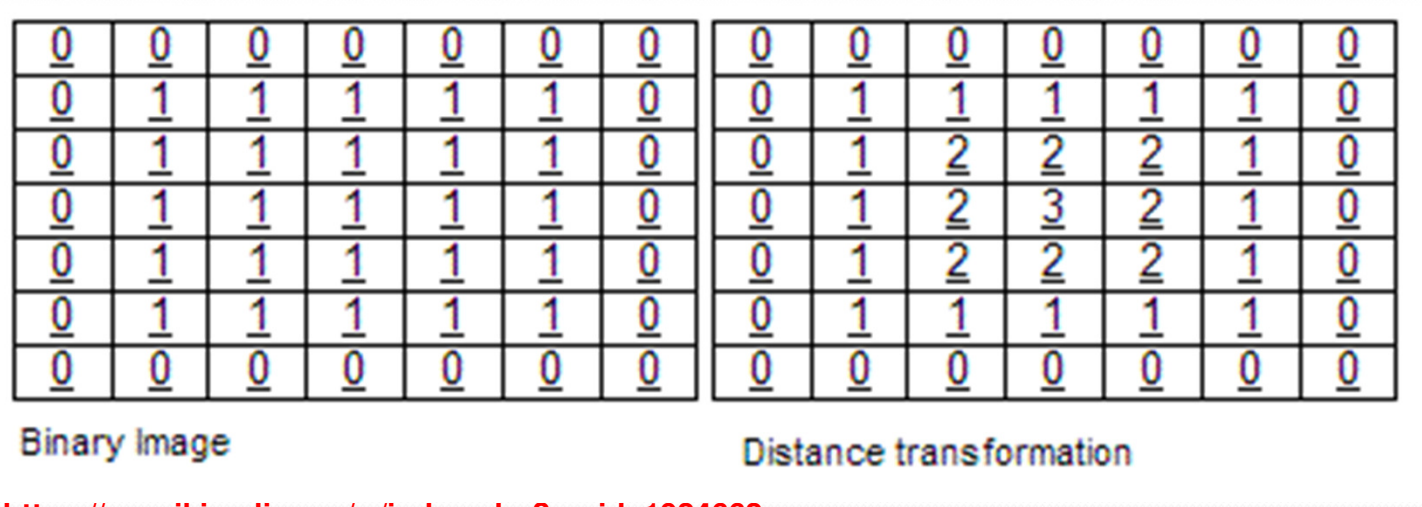
Morphological Gradient

Componentes conectados

- Determina la conectividad de regiones similares en una imagen binaria.
- Método cv2.connectedComponentsWithStats().

Transformación de distancia

- Calcula la distancia al pixel cero más cercano para cada pixel de la imagen binaria de origen.



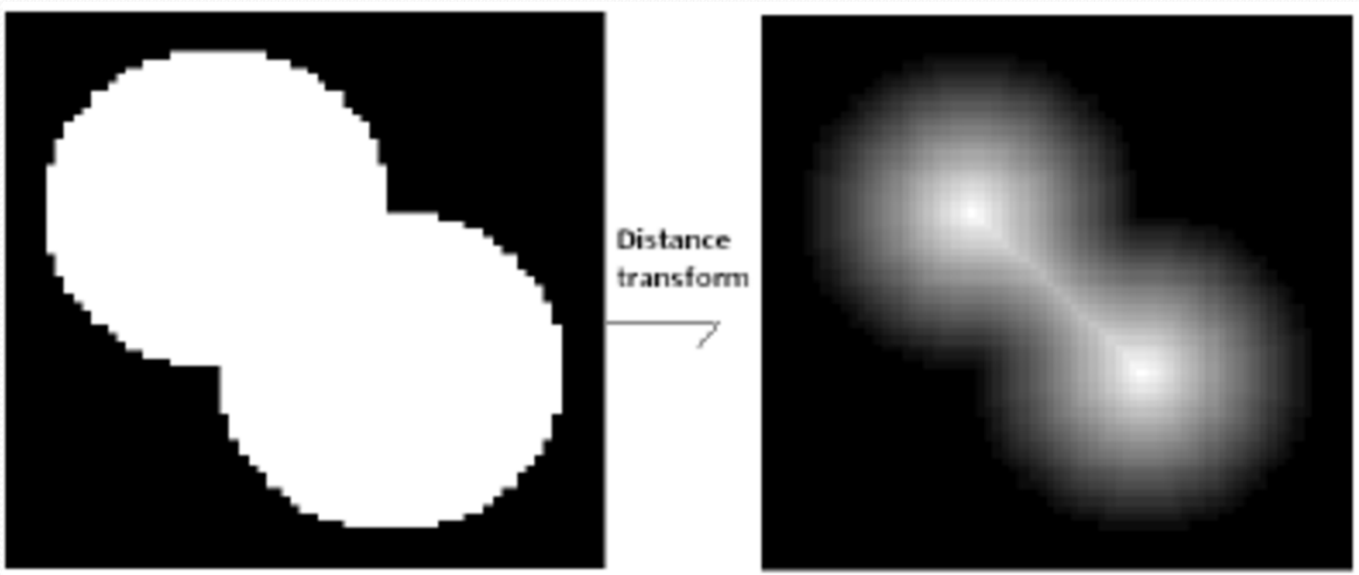
0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

Binary Image

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	2	2	2	1	0
0	1	2	3	2	1	0
0	1	2	2	2	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

Distance transformation

Transformación de distancia



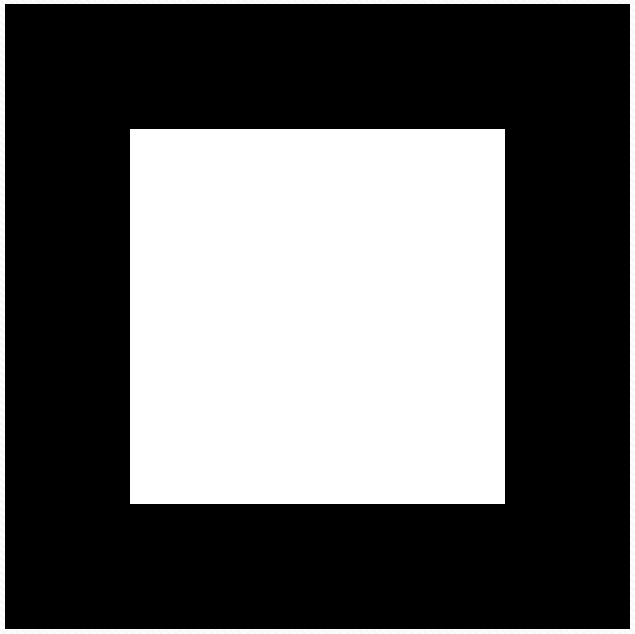
https://www.researchgate.net/figure/Distance-transform-of-the-binary-image-composed-with-a-white-pixel-form-and-a-black-pixel_fig2_29599685

Transformación de distancia en OpenCV

- Método cv2.distanceTransform().
- Se puede escoger el tipo de distancia.
- Las más comunes son:
- L1 – distancia de Manhattan.
- L2 – distancia euclidiana.

Ejemplo

- Disponible en segmentation/distance_transform.py.

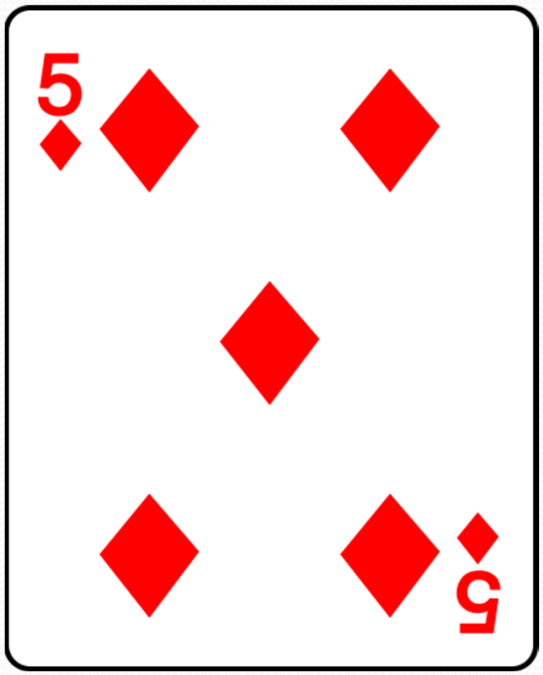


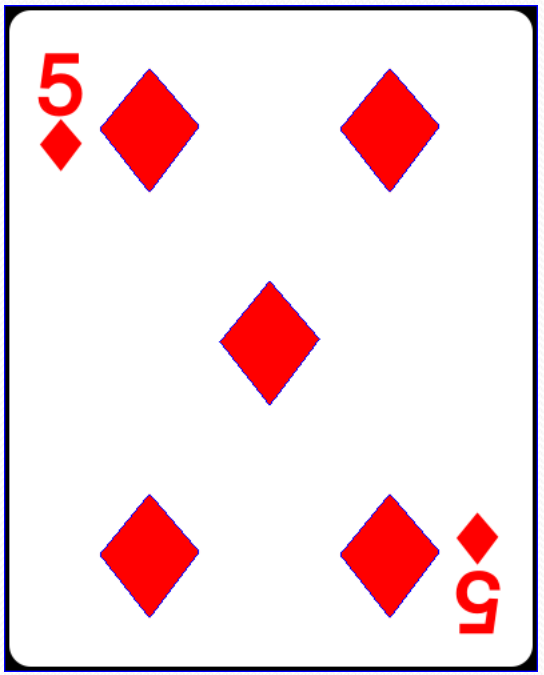
Watershed in OpenCV

- Método cv2.watershed().

Segmentación con watershed

- Ejemplo: marcar los cinco diamantes de la carta.

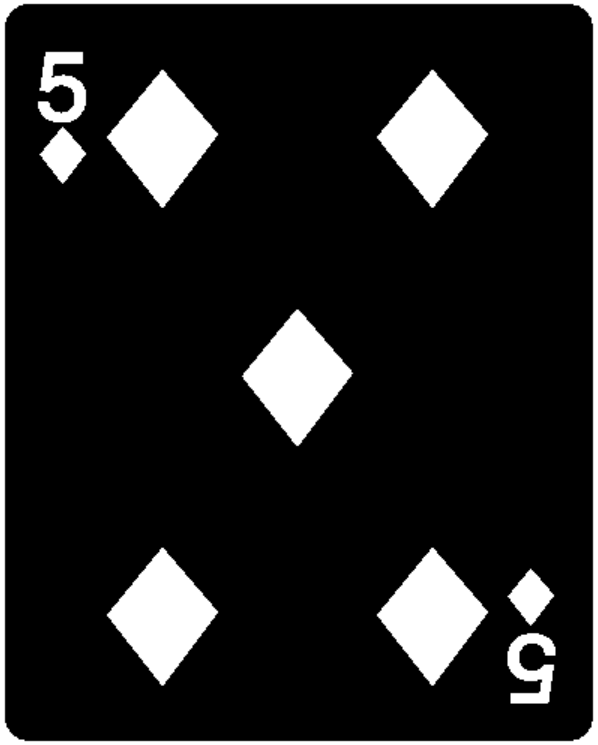




Segmentación con watershed

```
# File: watershed_no_overlap.py
image = cv.imread('../images/5_of_diamonds.png')
# Binarize the image
gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
_, thresh = cv.threshold(gray, 0, 255, cv.THRESH_BINARY_INV |
                                       cv.THRESH_OTSU)
cv.imwrite('01_thresh.png', thresh)
```

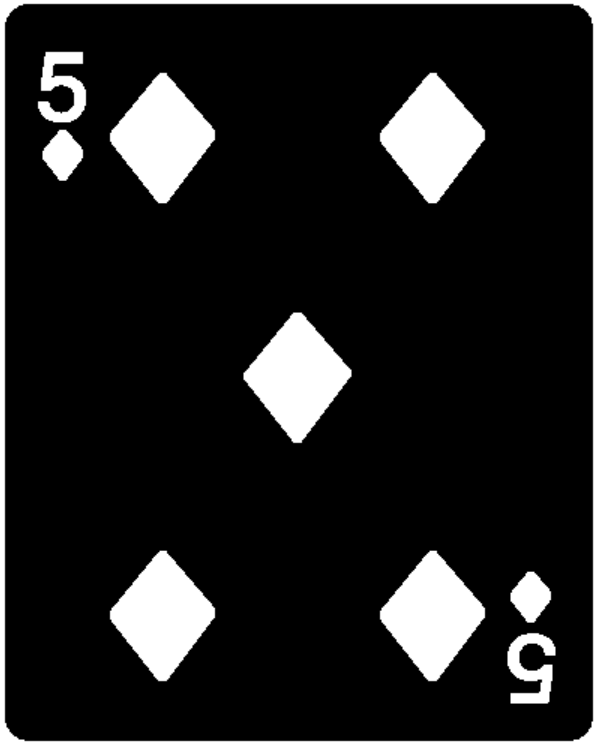
Segmentación con watershed



Segmentación con watershed

```
# Remove noise
kernel = np.ones((3, 3), np.uint8)
opening = cv.morphologyEx(thresh, cv.MORPH_OPEN, kernel, iterations=2)
cv.imwrite('02_opening.png', opening)
```

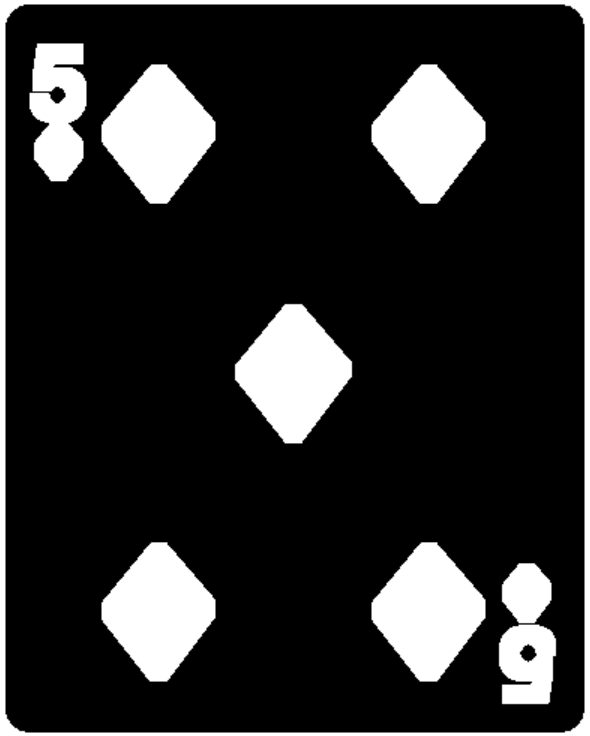
Segmentación con watershed



Segmentación con watershed

```
# Find the sure background region
sure_bg = cv.dilate(opening, kernel, iterations=3)
cv.imwrite('03_sure_bg.png', sure_bg)
```

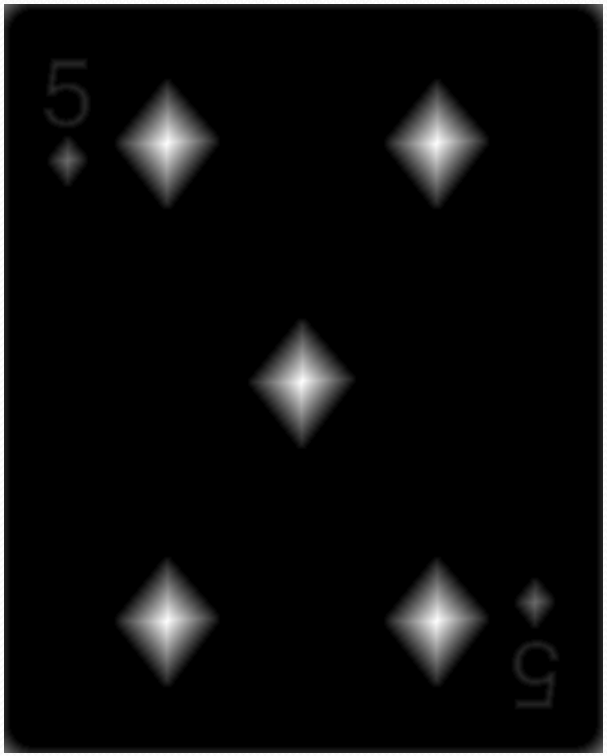
Segmentación con watershed

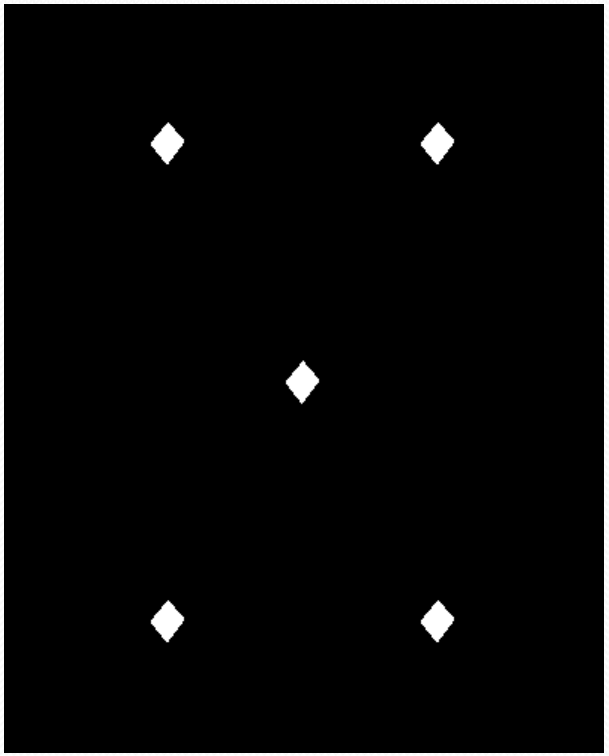


Segmentación con watershed

```
# Find the sure foreground region
dist = cv.distanceTransform(opening, cv.DIST_L2, 5)
norm = cv.normalize(dist, dst=None, alpha=0, beta=255,
                    norm_type=cv.NORM_MINMAX)
cv.imwrite('04_distances.png', norm)
_, sure_fg = cv.threshold(dist, 0.7 * dist.max(), 255, 0)
sure_fg = sure_fg.astype(np.uint8)
cv.imwrite('05_sure_fg.png', sure_fg)
```

Segmentación con watershed

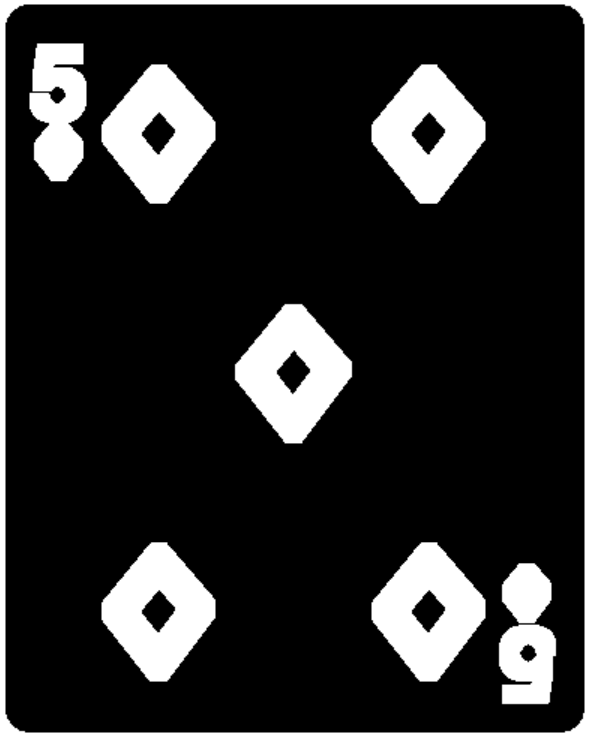




Segmentación con watershed

```
# Find the unknown region
unknown = cv.subtract(sure_bg, sure_fg)
cv.imwrite('06_unknown.png', unknown)
```

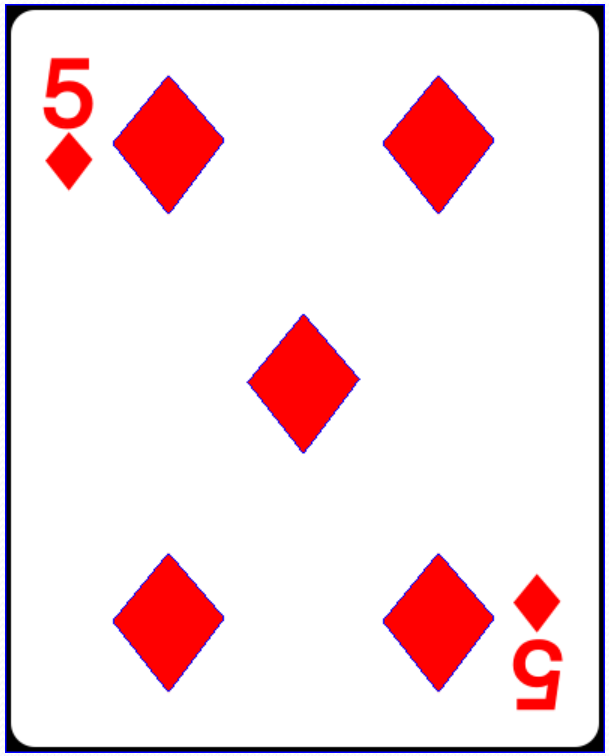
Segmentación con watershed



Segmentación con watershed

```
# Label the foreground objects
_, markers = cv.connectedComponents(sure_fg)
# Add one to all labels so that sure background is not 0, but 1
markers += 1
# Label the unknown region as 0
markers[unknown == 255] = 0
markers = cv.watershed(image, markers)
image[markers == -1] = [255, 0, 0]
cv.imwrite('07_result_watershed.png', image)
```

Segmentación con watershed

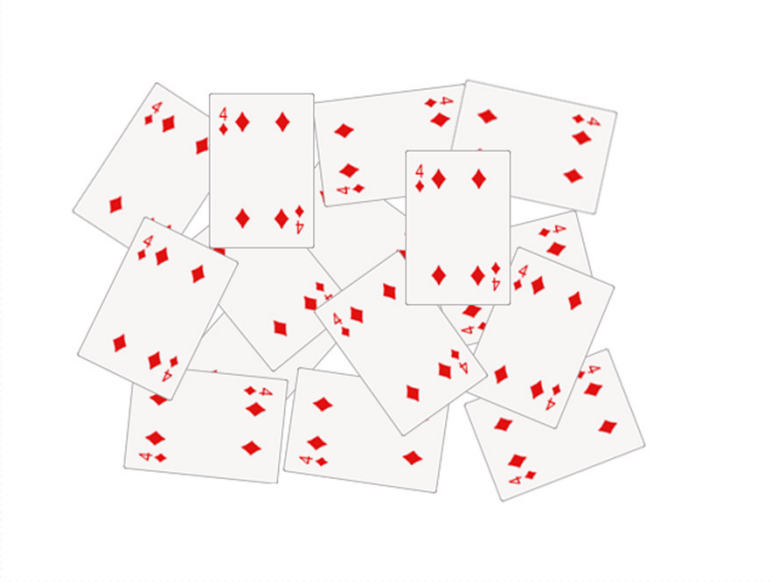


Resumen

1. Leer la imagen
2. Binarizar la imagen (threshold)
3. Eliminar ruido (opening)
4. Encontrar el background seguro (dilation)
5. Encontrar el foreground seguro (transformación de distancia y threshold)
6. Encontrar el área desconocida (background – foreground)
7. Etiquetar los componentes conectados del foreground
8. Aplicar watershed

Ejemplo

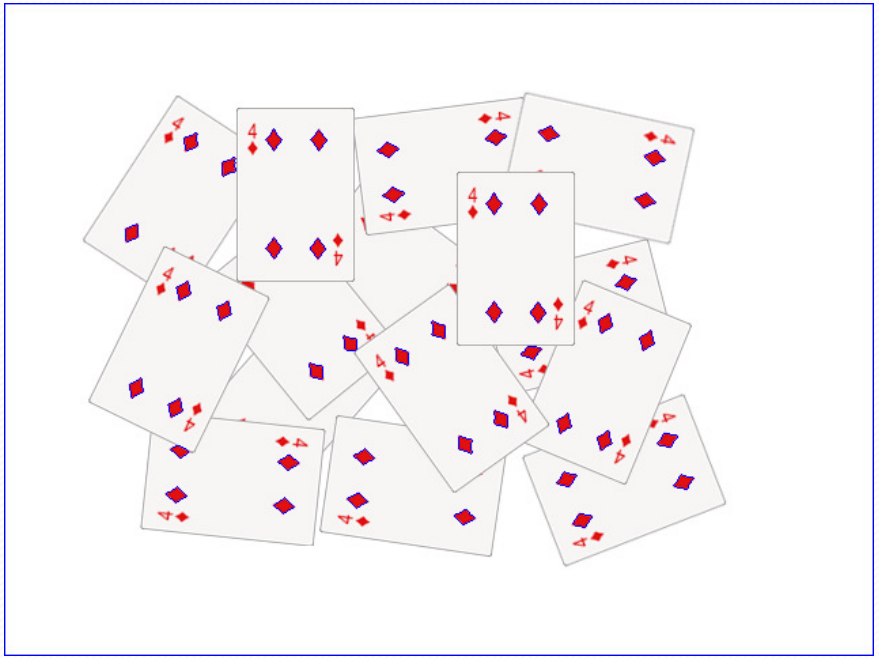
- Colorear las cartas por separado.



- Notar que las cartas están traslapadas.

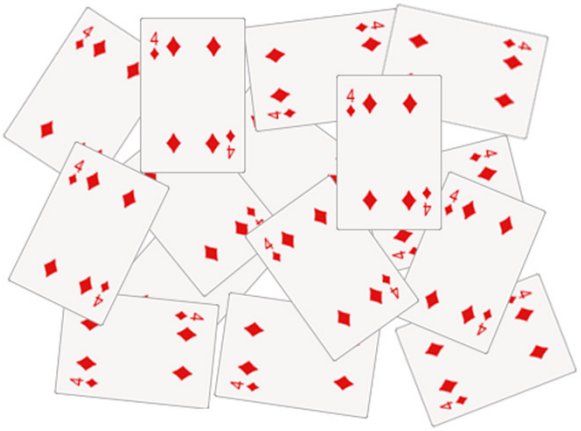
Ejemplo

- Si se usa el programa anterior se obtiene:



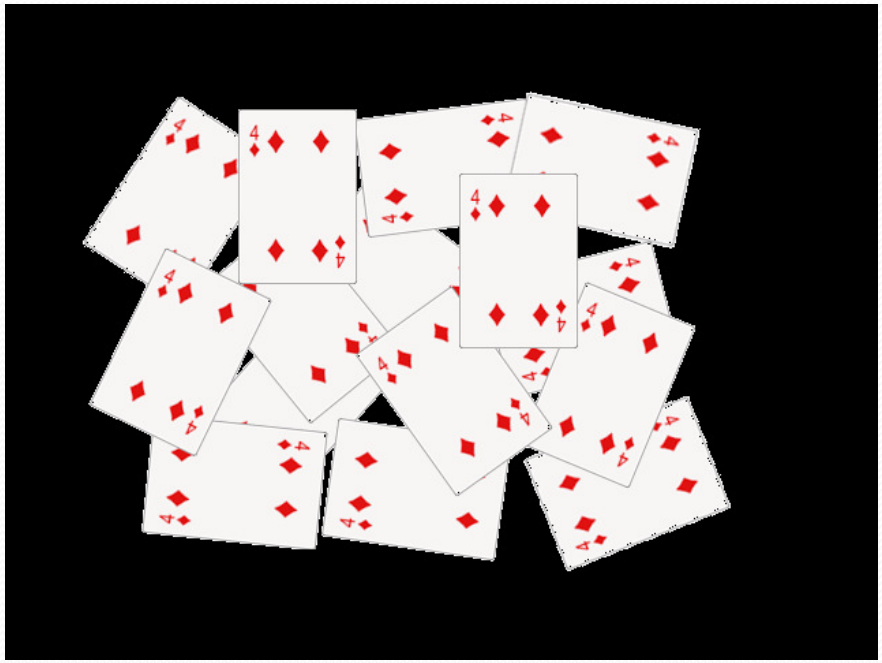
Solución

- Programa segmentation/watershed_cards.py.
- Paso 1: leer la imagen.



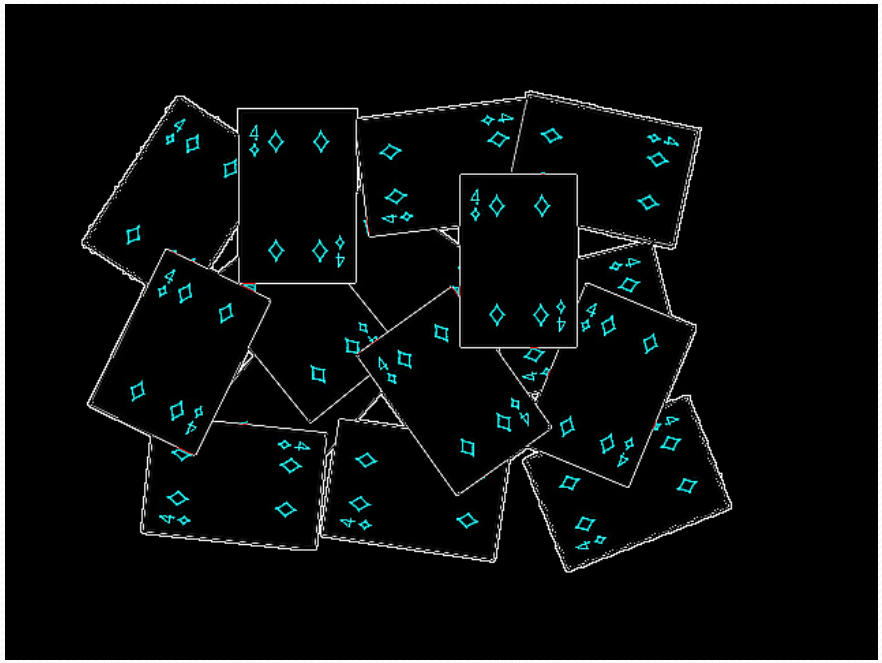
Solución

- Paso 2: cambiar el fondo a negro.



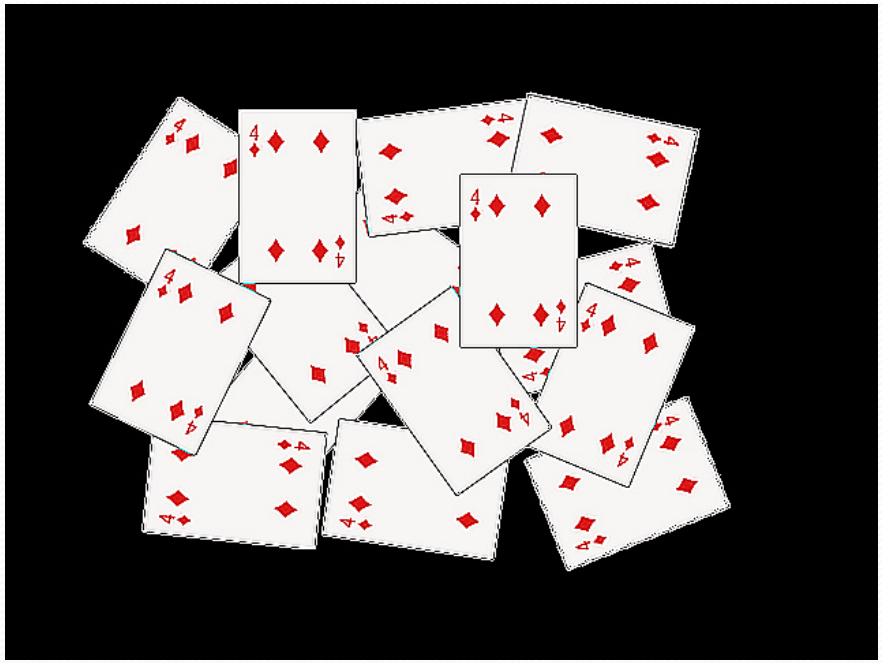
Solución

- Paso 3: aplicar un filtro laplaciano.



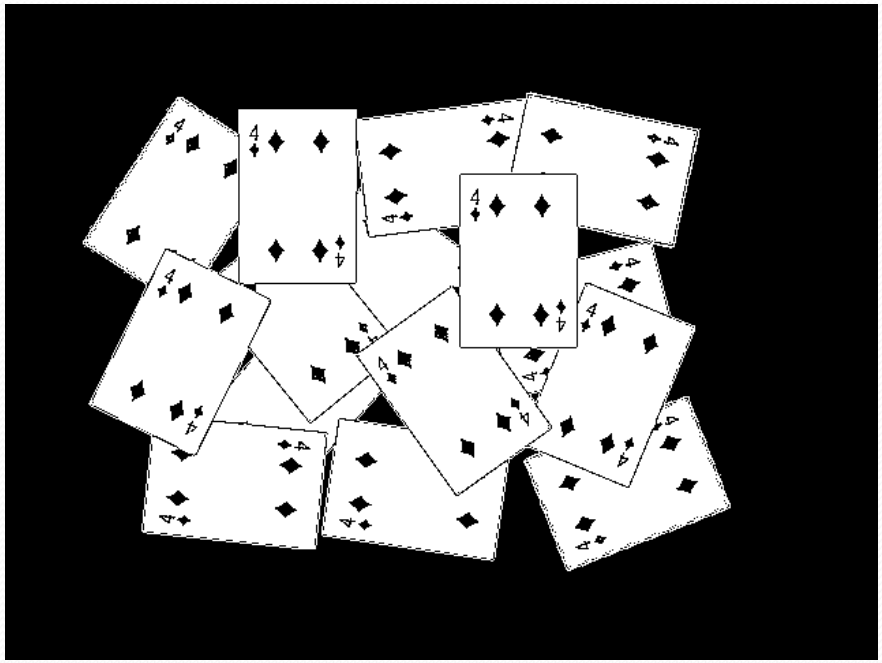
Solución

- Paso 4: con el laplaciano, hacer la imagen sharp.



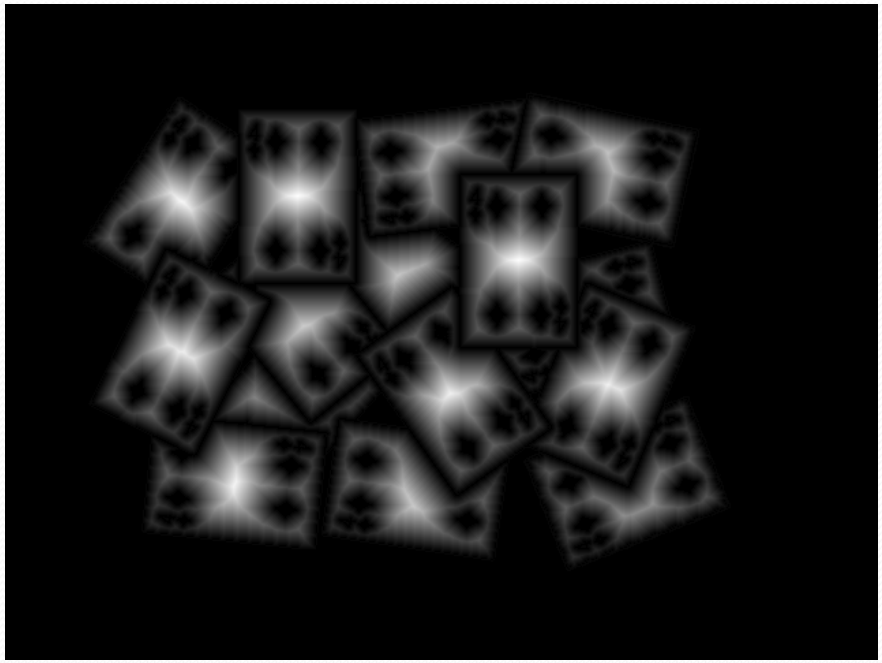
Solución

- Paso 5: binarizar la imagen.



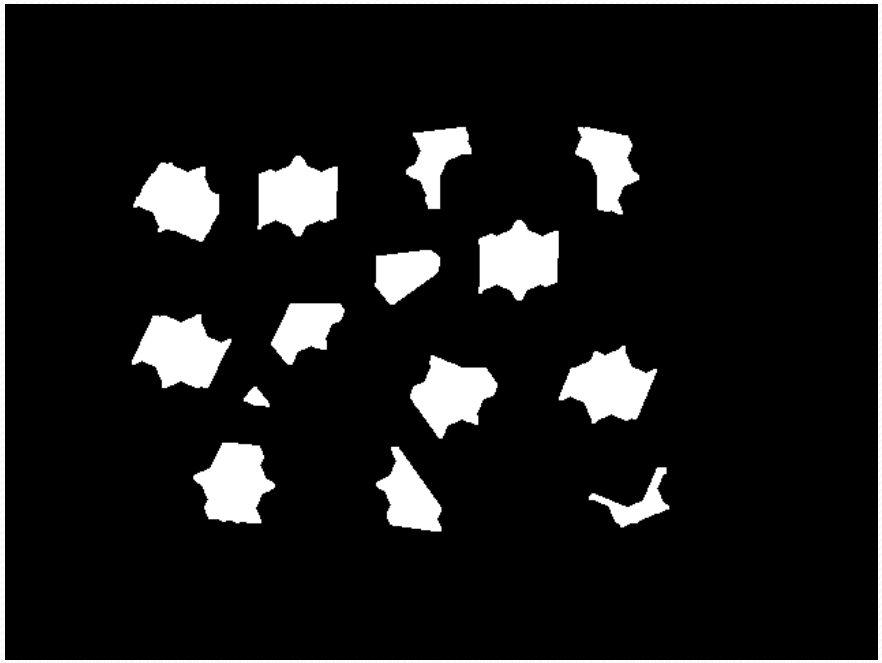
Solución

- Paso 6: ejecutar la transformación de distancia.



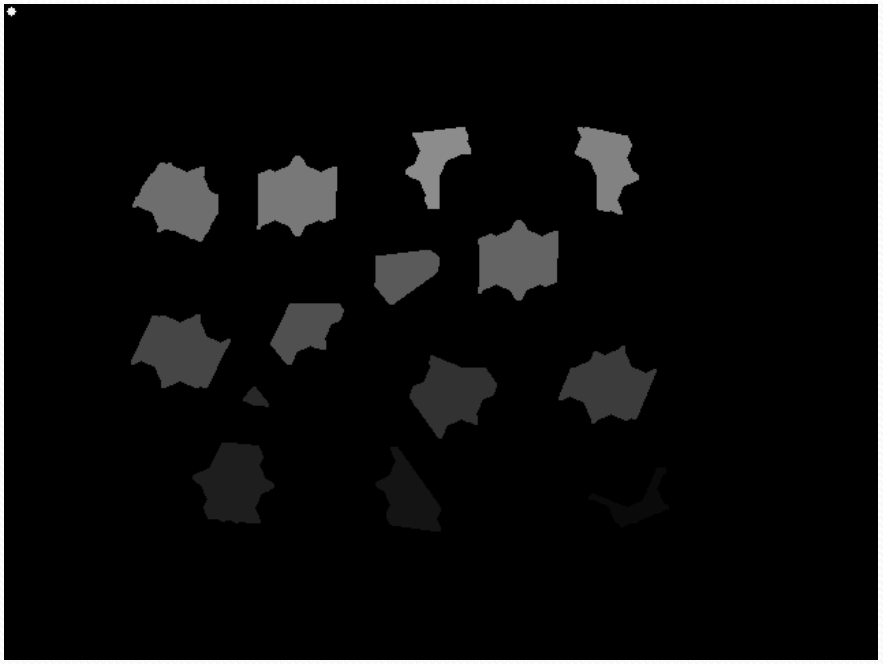
Solución

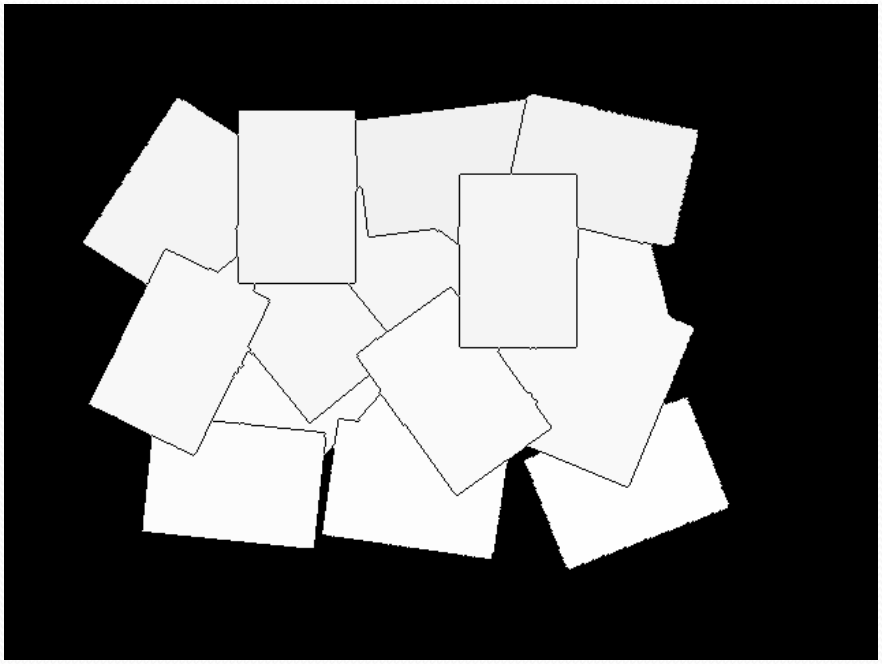
- Paso 7: binarizar la transformación de distancia para obtener los picos.



Solución

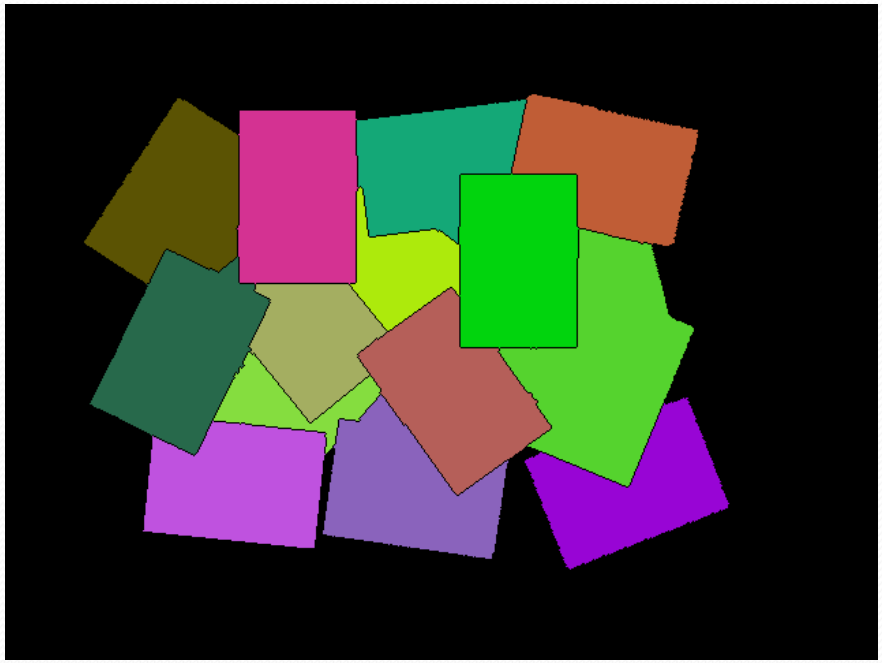
- Paso 8: encontrar los markers.





Solución

- Paso 9: ejecutar watershed y colorear las cartas de forma aleatoria.



Otro ejemplo

- Disponible en segmentation/watershed_coins.py.