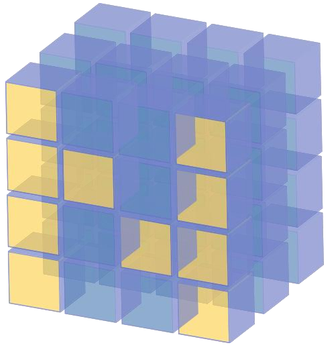
PROCESAMIENTO DIGITAL DE IMAGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Módulos principales



NumPy

matplotlib

OpenCV

pip install opencv-python

PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

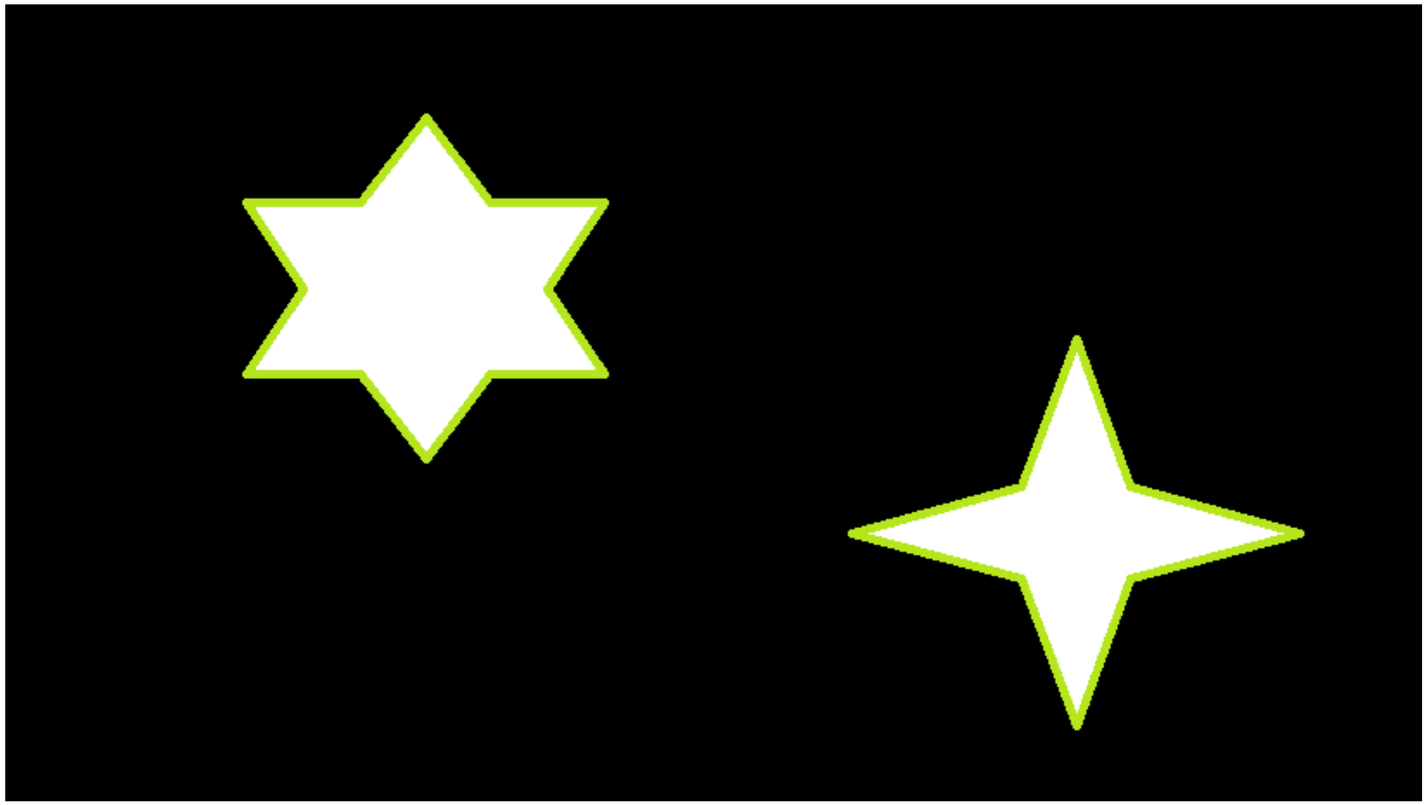
TEMAS QUE VEREMOS HOY

- ¿Qué es un contorno?
- ¿Cómo dibujar un contorno?
- Método de aproximación de contornos
- Características de los contornos
- Momento
- Área del contorno
- Perímetro del contorno
- Aproximación de contorno
- Envoltura convexa
- Revisión de convexidad
- Cuadro delimitador

PROCESAMIENTO DIGITAL DE IMÁGENES

CONTORNO

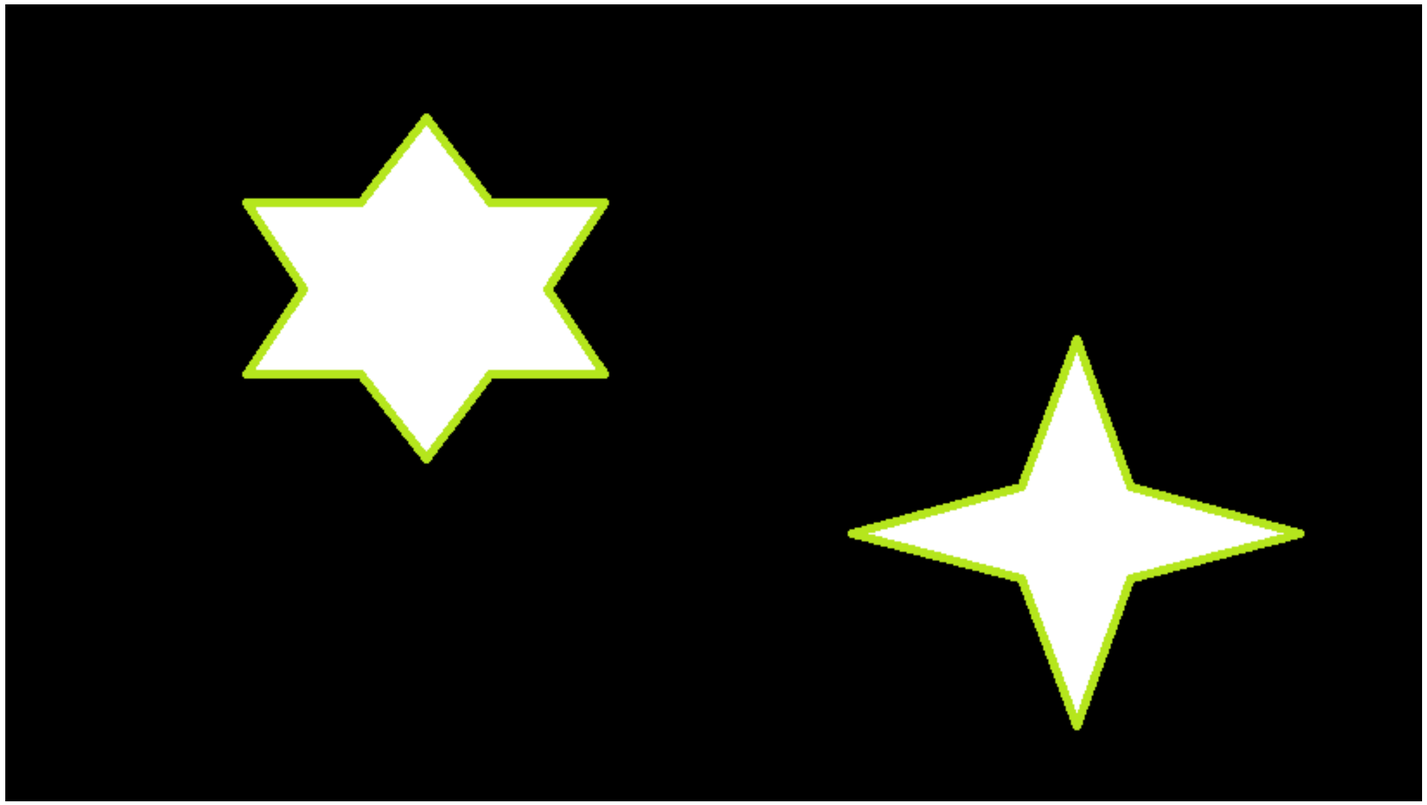
- Es una curva que une los puntos continuos en una imagen
- Los puntos que son unidos se encuentran a lo largo de los bordes
- La curva formada sirve para delimitar la forma de un objeto en la imagen



PROCESAMIENTO DIGITAL DE IMÁGENES

CONTORNO

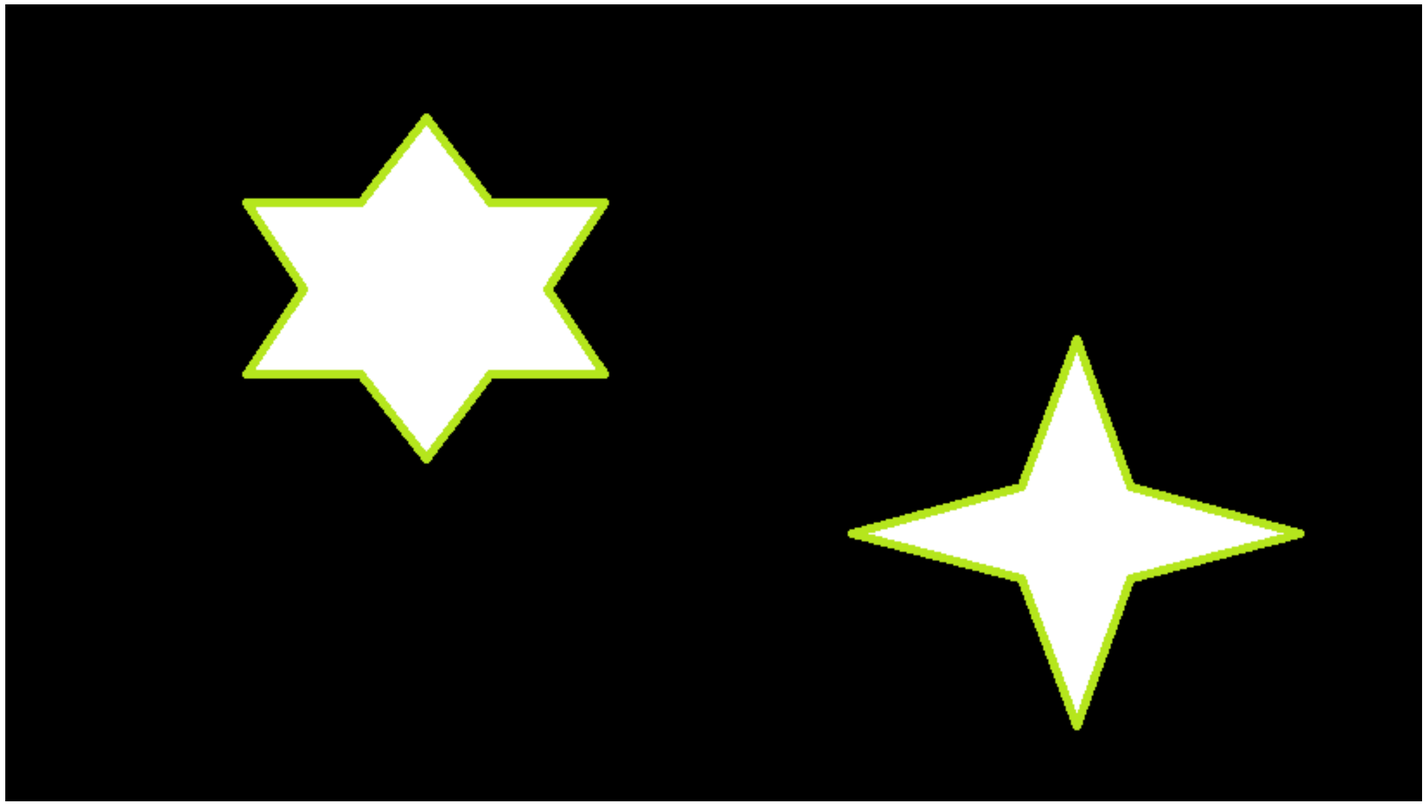
- Es útil para el análisis de formas
- Sirve para la detección de objetos
- Ayuda en el reconocimiento de objetos



PROCESAMIENTO DIGITAL DE IMÁGENES

CONTORNO

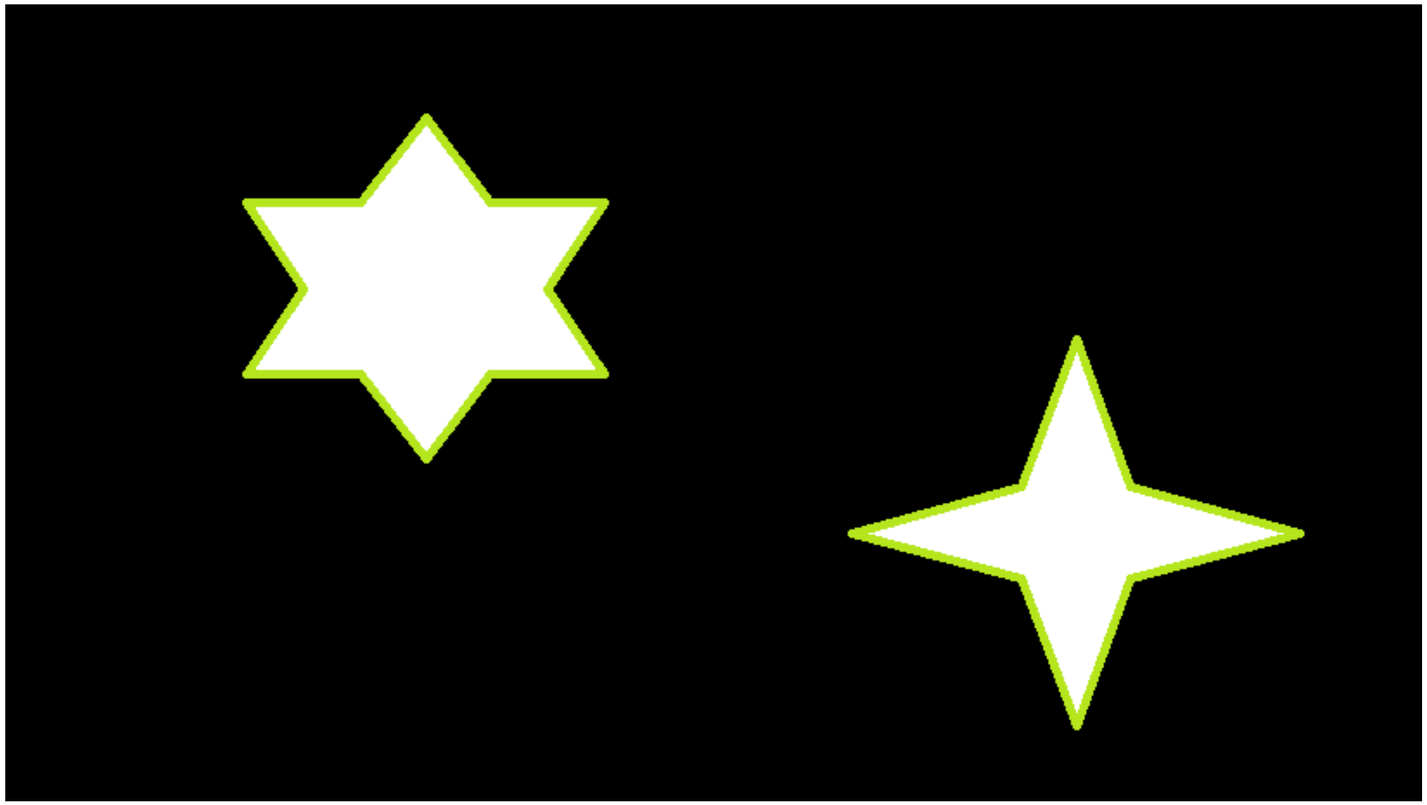
- Con el objetivo de tener mejores resultados, es preferible usar imágenes binarias
- Para ello, se debe aplicar un umbral o el algoritmo de Canny
- Entonces, los contornos pueden ser encontrados



PROCESAMIENTO DIGITAL DE IMÁGENES

CONTORNO

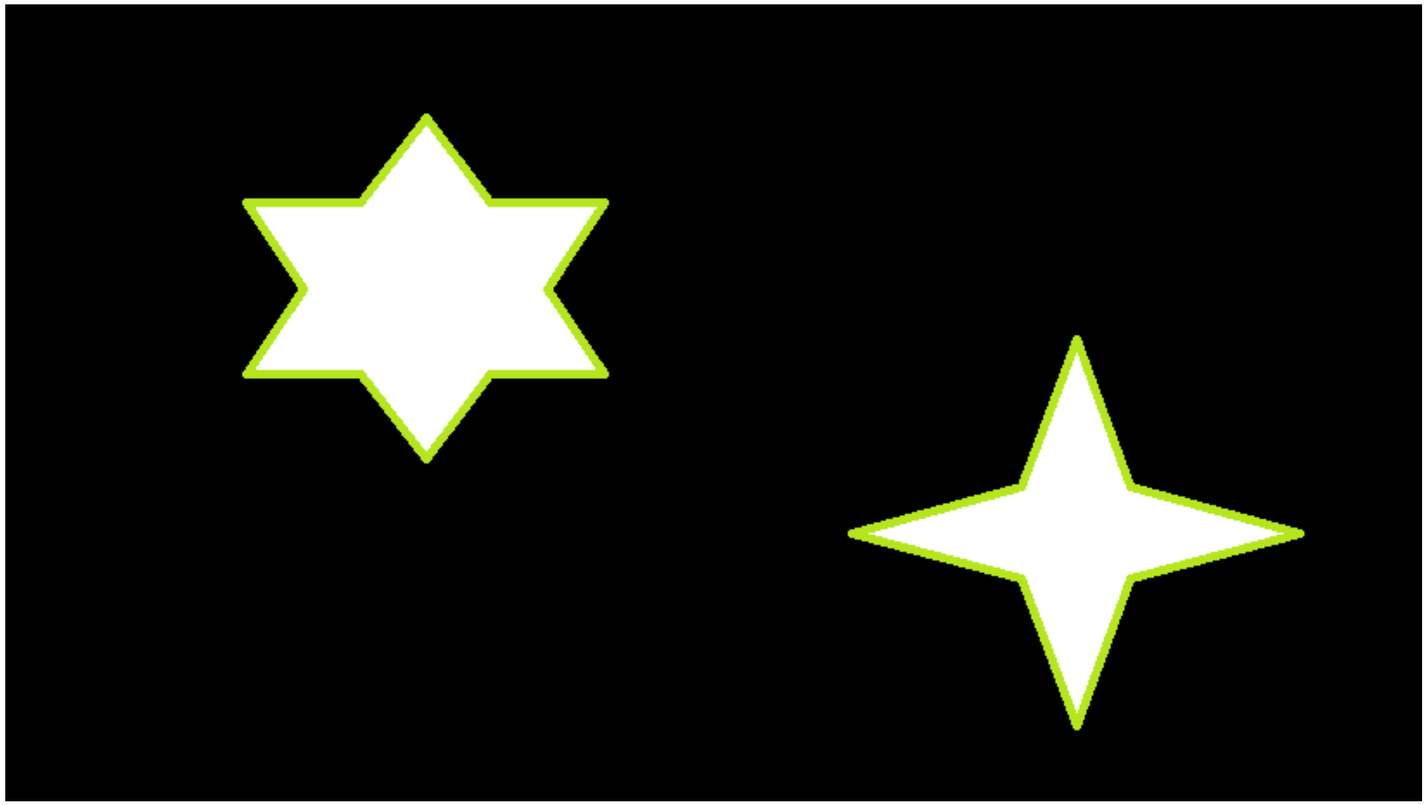
- Mientras que los bordes son computados como puntos, los contornos son curvas cerradas
- Cuando los contornos son obtenidos a partir de los bordes, se requiere conectar dichos bordes, con el objetivo de obtener un contorno cerrado



PROCESAMIENTO DIGITAL DE IMÁGENES

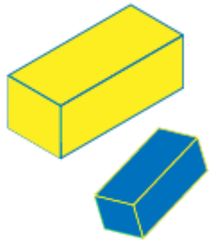
CONTORNO

- Para OpenCV, encontrar contornos es similar a encontrar objetos de color blanco en un fondo de color negro
- Por ello, los objetos que usaremos deberán ser de color blanco y estar en un fondo de color negro



PROCESAMIENTO DIGITAL DE IMÁGENES

Aplicar findContours a una imagen



findContours : Encuentra los contornos en una imagen binaria

Sintaxis: cv2.findContours(imagen, modo, método)

Parametros:
imagen : Imagen de entrada
modo : Modo para obtener los contornos
método : Método para realizar la aproximación de contornos

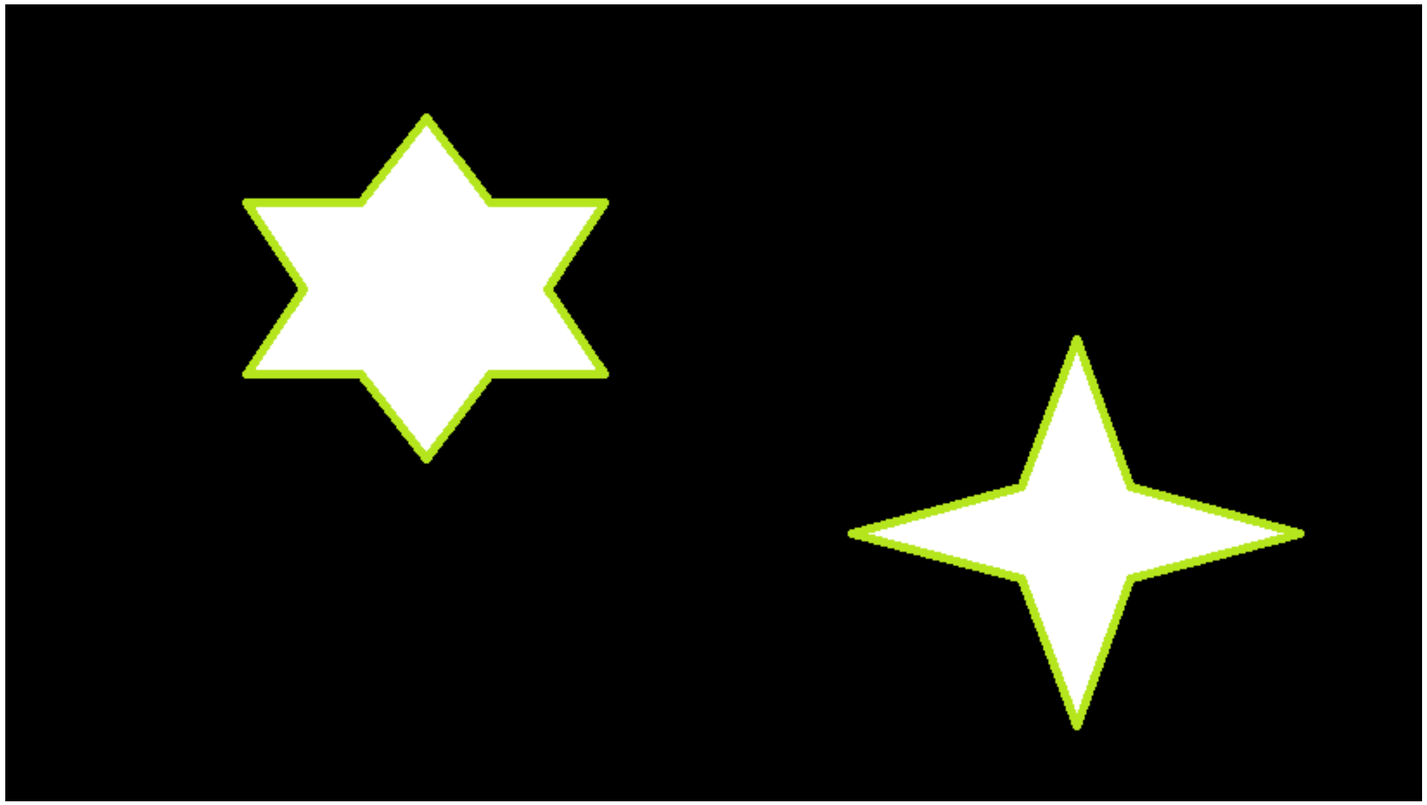
Nota Para el parámetro «modo» se usa «cv2.RETR_EXTERNAL» para indicar que se devuelve solo los contornos exteriores. En el parámetro «método» se usa «cv2.CHAIN_APPROX_NONE» para indicar que se guarden todos los puntos que conforman el contorno. El método «findContours» devuelve los contornos y la jerarquía; siendo el primero una lista que contiene todos los contornos en la imagen, donde cada contorno es un arreglo de coordenadas (x, y)

Ejemplo:

```
import cv2

image = cv2.imread("image_1.png")
image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

valor_umbral = 127
valor_maximo = 255
tipo = cv2.THRESH_BINARY
return_value, image_with_threshold = cv2.threshold(image, valor_umbral, valor_maximo, tipo)

contours, hierarchy = cv2.findContours(image_with_threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
print("Contornos:")
print(contours)
```

CONTORNO

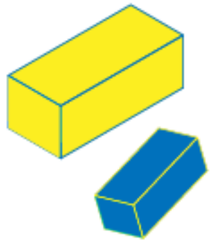
- En OpenCV, para poner los contornos sobre la imagen se hace uso de un método
- Con este método se puede poner cualquier forma sobre la figura siempre y cuando se conozca a priori sus contornos



PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Aplicar drawContours a una imagen



drawContours : Grafica los contornos o rellena sus interiores

Sintaxis: cv2.drawContours(imagen, contornos, índice, color[, grosor])

Parametros:
imagen : Imagen de entrada
contornos : Lista de puntos almacenados como coordenadas (x, y)
índice : Índice del contorno a graficar
color : Color de los contornos, es una tupla en formato BGR
grosor : Grosor de las líneas con las que se dibujan los contornos

Nota Si el parámetro «índice» es igual a un número negativo, entonces se grafican todos los contornos. En caso el parámetro «grosor» fuera un número negativo, se rellena el interior del contorno

PROCESAMIENTO DIGITAL DE IMÁGENES

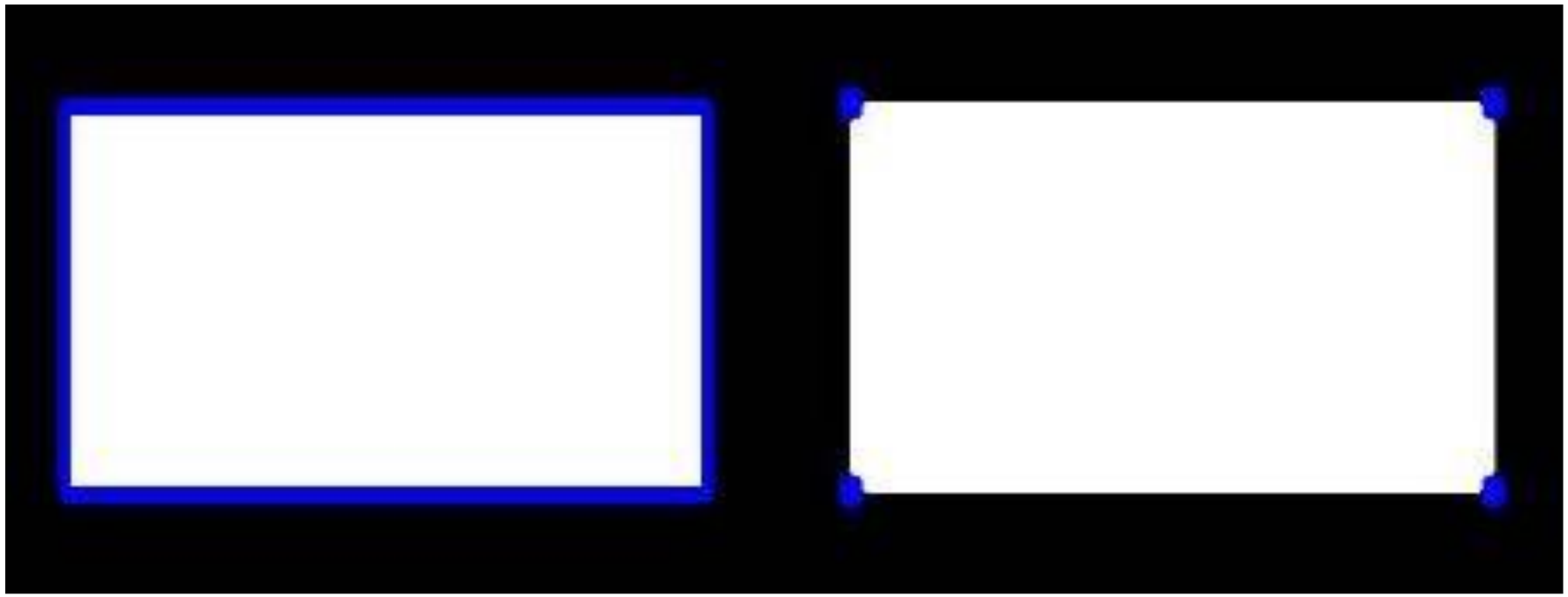
UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Ejemplo:

```python
import cv2

image_as_BGR = cv2.imread("image_1.png")
image_as_GRAY = cv2.cvtColor(image_as_BGR, cv2.COLOR_BGR2GRAY)

valor_umbral = 127
valor_maximo = 255
tipo = cv2.THRESH_BINARY
return_value, image_with_threshold = cv2.threshold(image_as_GRAY, valor_umbral, valor_maximo, tipo)

contours, hierarchy = cv2.findContours(image_with_threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

color = (0, 255, 0) #BGR
cv2.drawContours(image_as_BGR, contours, -1, color, 2)

cv2.imshow("Imagen con los contornos", image_as_BGR)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

MÉTODO DE APROXIMACIÓN DE CONTORNOS

- Hace referencia al tercer argumento del método «findContours»
- Si usamos «cv2.CHAIN_APPROX_NONE», entonces se almacenan todos los puntos que conforman el contorno
- Sin embargo, no siempre es necesario almacenar todos los puntos que representan el contorno

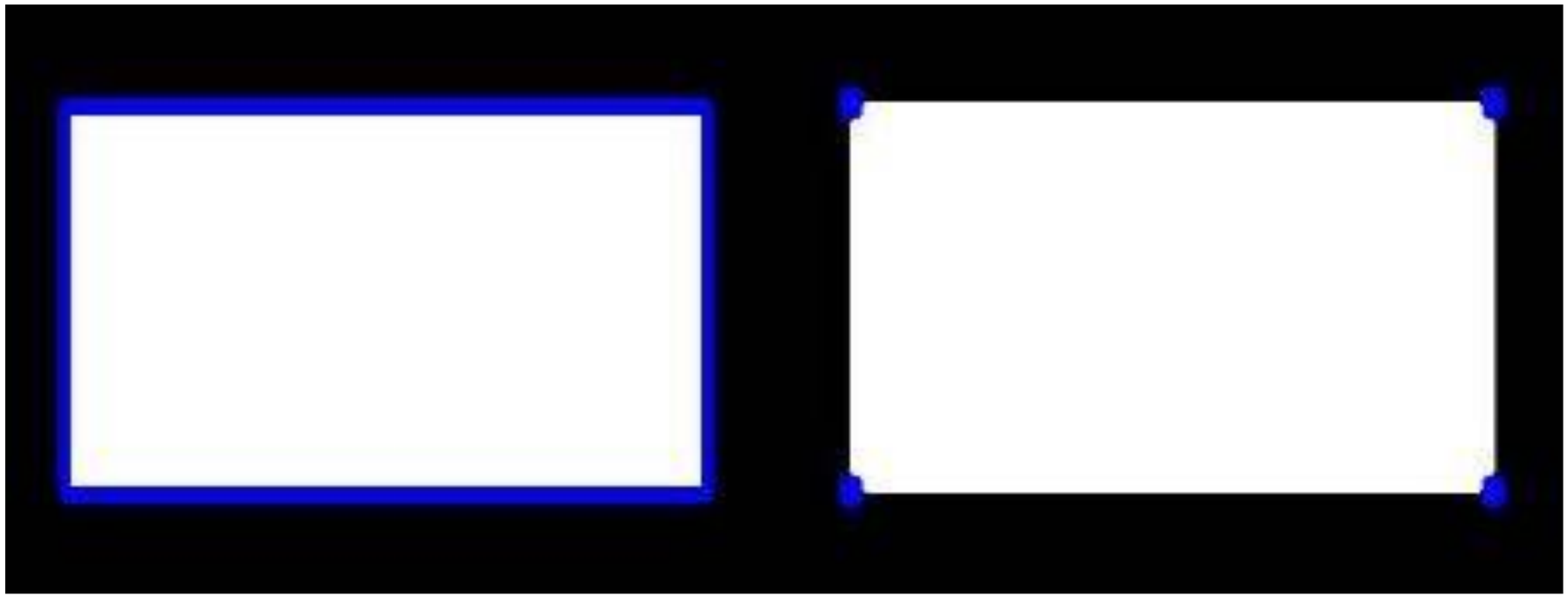


PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

MÉTODO DE APROXIMACIÓN DE CONTORNOS

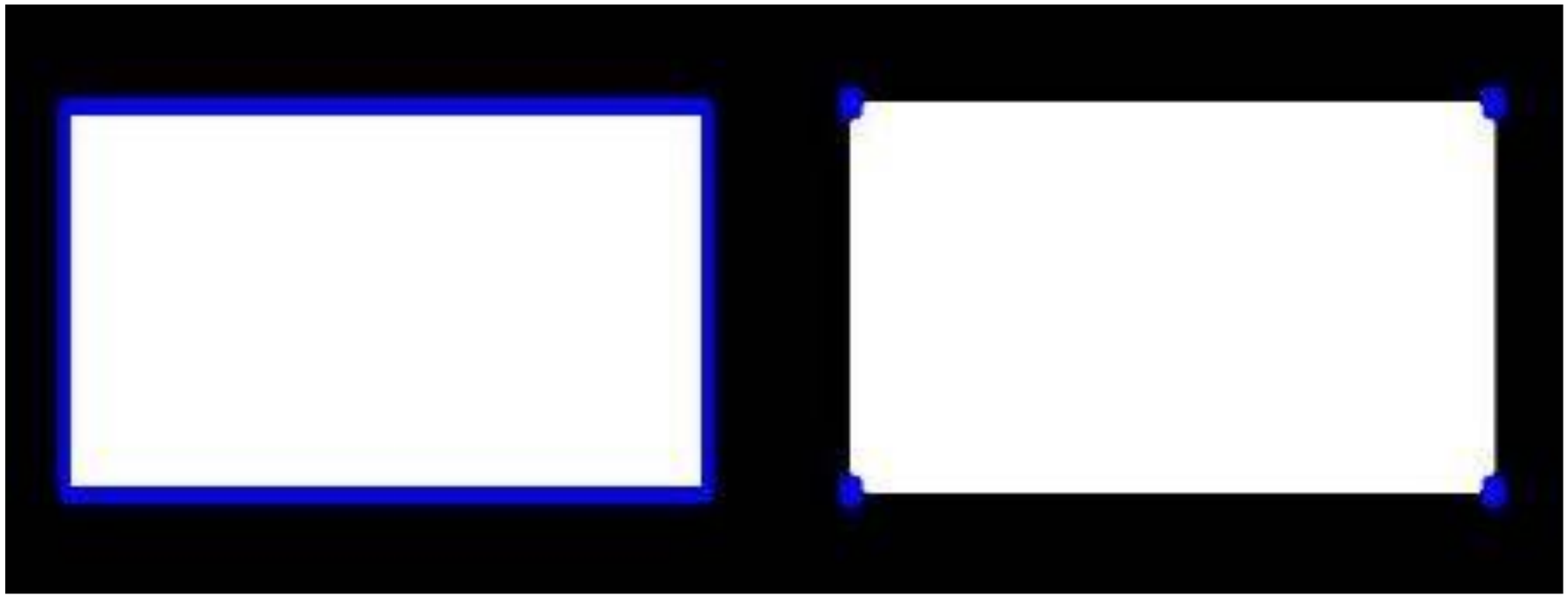
- En el caso de encontrar los contornos de una línea recta no es necesario almacenar todos los puntos que conforman el contorno
- Sólo se requiere los 2 puntos extremos a una línea recta para poder representar esa línea
- Esto se consigue cuando usamos «cv2.CHAIN_APPROX_SIMPLE», el cual se encarga de eliminar todos los puntos redundantes
- Además, comprime el contorno, de tal forma que es requerido menos memoria



PROCESAMIENTO DIGITAL DE IMÁGENES

MÉTODO DE APROXIMACIÓN DE CONTORNOS

- La imagen adjunta muestra la diferencia entre «cv2.CHAIN_APPROX_NONE» y «cv2.CHAIN_APPROX_SIMPLE»
- Se dibuja un círculo o un rectángulo en todas las coordenadas de la lista del contorno, con el objetivo de representar estos puntos
- La imagen del lado izquierdo requiere más de 700 puntos, mientras que la imagen en la derecha sólo requiere 4 puntos

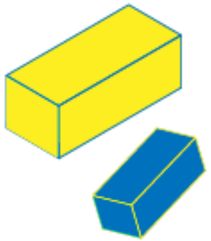


PROCESAMIENTO DIGITAL DE IMÁGENES

Ejemplo:

```python
import cv2

image_as_BGR = cv2.imread("image_2.png")
image_as_GRAY = cv2.cvtColor(image_as_BGR, cv2.COLOR_BGR2GRAY)

valor_umbral = 127
valor_maximo = 255
tipo = cv2.THRESH_BINARY
return_value, image_with_threshold = cv2.threshold(image_as_GRAY, valor_umbral, valor_maximo, tipo)

contours, hierarchy = cv2.findContours(image_with_threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
print("Contornos:")
print(contours)

color = (0, 255, 0) #BGR
cv2.drawContours(image_as_BGR, contours, -1, color, 2)

cv2.imshow("Imagen con los contornos", image_as_BGR)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

CARACTERÍSTICAS DE LOS CONTORNOS



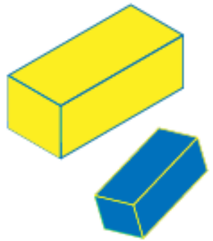
- Las características de los contornos son:

Momento
Área del contorno
Perímetro del contorno
Aproximación de contorno
Envoltura convexa
Revisión de convexidad
Cuadro delimitador
Círculo mínimo de inclusión
Ajustar una elipse
Ajustar una línea

PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

CARACTERÍSTICAS DE LOS CONTORNOS (MOMENTO)



- El momento de una imagen es un determinado promedio ponderado (momento) de las intensidades de los píxeles en la imagen

- También se puede tener una función que represente los momentos de una imagen, los cuales son generalmente elegidos para tener alguna propiedad o interpretación en la imagen

- Permiten describir los objetos luego de su segmentación

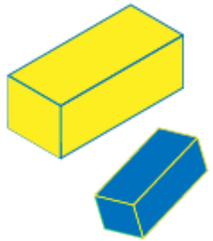
- Los momentos de una imagen permiten obtener propiedades en la imagen, como el área o centro de masa de un objeto

- Mediante OpenCV se puede obtener un diccionario que contenga todos los valores de momento calculados

- A partir de los momentos obtenidos se puede extraer datos, como el centroide

PROCESAMIENTO DIGITAL DE IMÁGENES

Calcular los momentos



moments : Calcula los momentos de un polígono o forma rasterizada

Sintaxis: cv2.moments(arreglo)

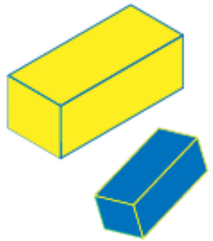
Parametros:
arreglo : Arreglo de puntos (x, y)

Nota El parámetro «arreglo» puede ser una imagen de 1 canal, representada mediante una matriz de 2 dimensiones que contiene valores en 8 bits o punto flotante. También puede ser un arreglo de coordenadas, tal que cada coordenada (x, y) debe ser del tipo «np.int32» o «np.float32»

Ejemplo:

```python
import cv2

image = cv2.imread("image_1.png")
image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

valor_umbral = 127
valor_maximo = 255
tipo = cv2.THRESH_BINARY
return_value, image_with_threshold = cv2.threshold(image, valor_umbral, valor_maximo, tipo)

contours, hierarchy = cv2.findContours(image_with_threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

contour = contours[0]

moments = cv2.moments(contour)

print("Momentos:")
print(moments)
```

CARACTERÍSTICAS DE LOS CONTORNOS (ÁREA DEL CONTORNO)



- El área del contorno se puede determinar mediante el método «contourArea», o a partir del momento «moments[‘m00’]»

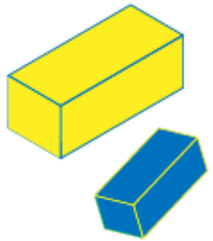
CARACTERÍSTICAS DE LOS CONTORNOS (PERÍMETRO DEL CONTORNO)

- Es conocido como longitud de arco
- Se puede determinar mediante el método «arcLength»

PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Obtener el área del contorno



contourArea : Calcula el área de contorno

Sintaxis: cv2.contourArea(contorno[, orientación])

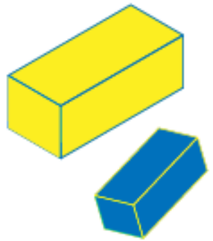
Parametros:
contorno : Lista de puntos almacenados como coordenadas (x, y)
orientación : Flag que determina si se debe o no retornar el área con un signo

Nota Si el parámetro «orientación» es igual a «True», se devolverá un área con signo, el cual dependerá de la orientación del contorno (en sentido horario o antihorario). De este modo, se puede obtener la orientación de un contorno mediante el signo del área. Por defecto, el parámetro «orientación» es igual a «False», lo que indica que retorna el valor absoluto del área

Ejemplo:

```python
import cv2

image = cv2.imread("image_1.png")
image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

valor_umbral = 127
valor_maximo = 255
tipo = cv2.THRESH_BINARY
return_value, image_with_threshold = cv2.threshold(image, valor_umbral, valor_maximo, tipo)

contours, hierarchy = cv2.findContours(image_with_threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

contour = contours[0]

area = cv2.contourArea(contour)

print("Área:")
print(area)
```

Obtener el perímetro del contorno



arcLength : Calcula una longitud de curva o un perímetro de contorno

Sintaxis: cv2.arcLength(curva, cerradura)

Parametros:
curva : Es una lista de puntos almacenados como coordenadas (x, y)
cerradura : Flag que indica si la curva está o no cerrada

Nota Si el parámetro «cerradura» es igual a «True», indica que la forma producida por la curva es un contorno cerrado. Si el parámetro «cerradura» es igual a «False», indica que el primer parámetro es sólo una curva

PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

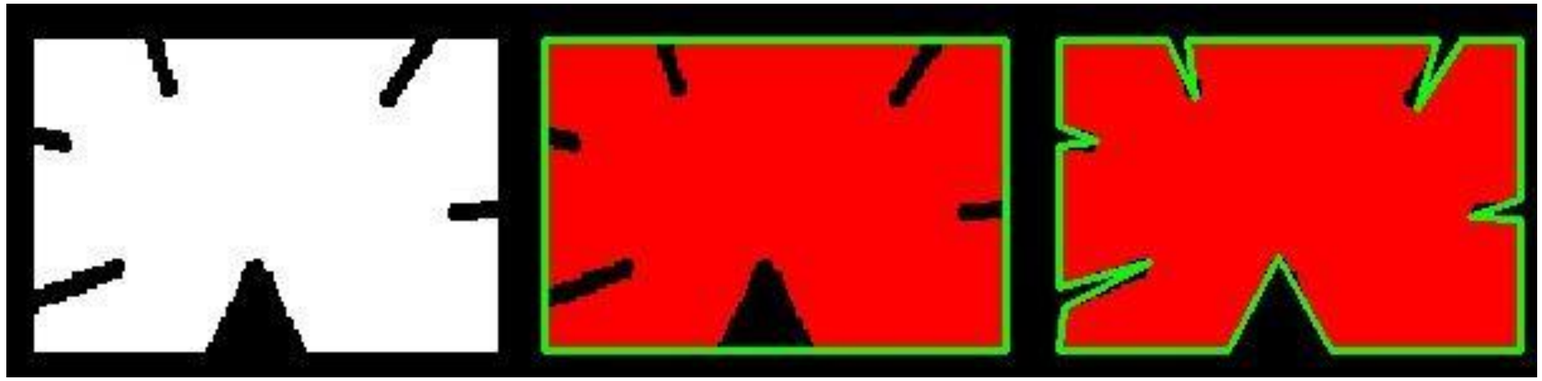
Ejemplo:

```python
import cv2

image = cv2.imread("image_1.png")
image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

valor_umbral = 127
valor_maximo = 255
tipo = cv2.THRESH_BINARY
return_value, image_with_threshold = cv2.threshold(image, valor_umbral, valor_maximo, tipo)

contours, hierarchy = cv2.findContours(image_with_threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

contour = contours[0]

perimeter = cv2.arcLength(contour, True)

print("Perímetro o Longitud de curva:")
print(perimeter)
```

CARACTERÍSTICAS DE LOS CONTORNOS (APROXIMACIÓN DE CONTORNO)

- Permite aproximar la forma de un contorno hacia otra forma con una menor cantidad de vértices, dependiendo de la precisión especificada

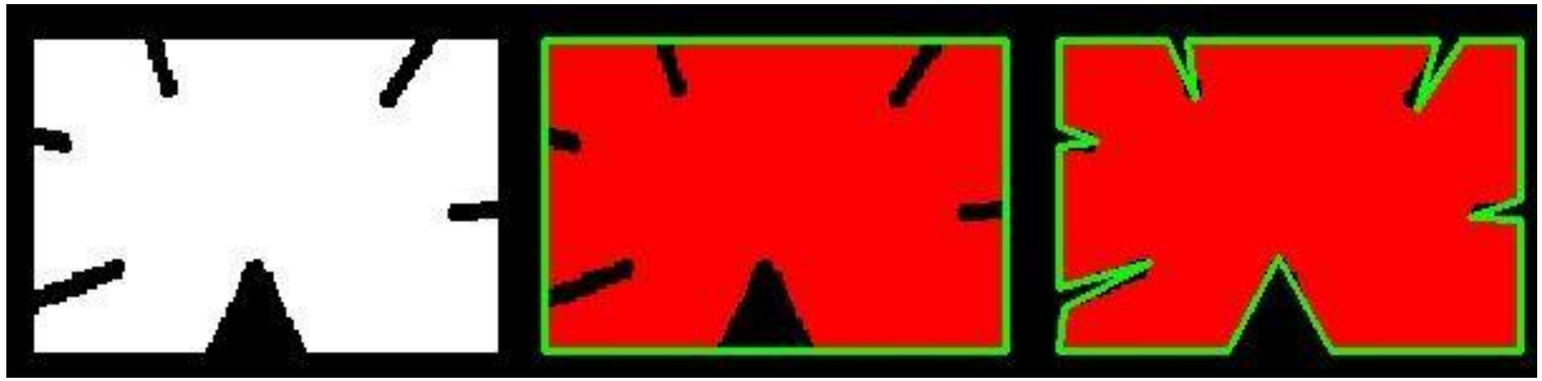
- Por ejemplo, se trata de encontrar un cuadrado en una imagen, pero se obtiene una forma incorrecta del cuadrado. Entonces, se aproxima esta forma a la de un cuadrado



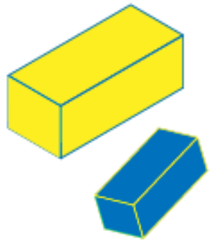
CARACTERÍSTICAS DE LOS CONTORNOS (APROXIMACIÓN DE CONTORNO)

- Mediante el método «approxPolyDP» podemos conseguir aproximar la forma de la figura en el lado izquierdo hacia la figura que se encuentra en la mitad

- Para realizar esta aproximación, se requiere indicar la distancia máxima entre el contorno y el contorno aproximado. Acorde a su valor se obtendrá una u otra aproximación; por ejemplo, un valor bajo devolverá como aproximación la figura que se encuentra a la derecha



Aproximar un contorno



approxPolyDP : Aproxima una curva poligonal

Sintaxis: cv2.approxPolyDP(curva, distancia, cerradura)

Parametros:
curva : Es una lista de puntos almacenados como coordenadas (x, y)
distancia : Es la distancia máxima entre la curva original y su aproximación
cerradura : Flag que indica si la curva está o no cerrada

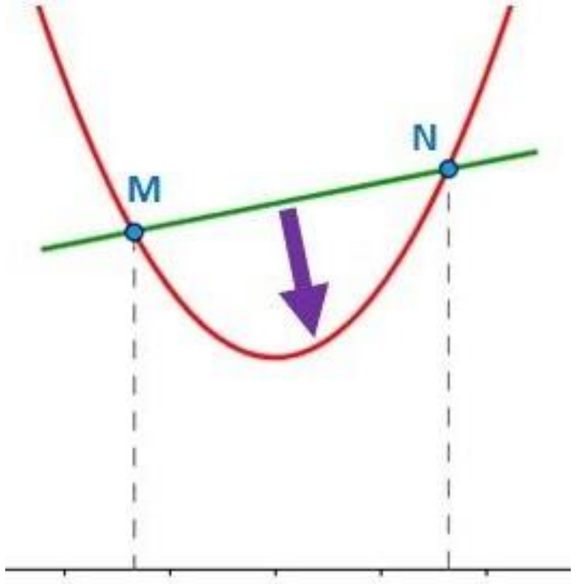
Nota Si el parámetro «cerradura» es igual a «True», indica que la curva aproximada está cerrada. Esto implica que si primer y último vértice estén conectados. Si el parámetro «cerradura» es igual a «False», indica que la curva aproximada no está cerrada

Ejemplo:

```python
import cv2

image_as_BGR = cv2.imread("image_3.png")
image_as_GRAY = cv2.cvtColor(image_as_BGR, cv2.COLOR_BGR2GRAY)

valor_umbral = 127
valor_maximo = 255
tipo = cv2.THRESH_BINARY
return_value, image_with_threshold = cv2.threshold(image_as_GRAY, valor_umbral, valor_maximo, tipo)

contours, hierarchy = cv2.findContours(image_with_threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

contour = contours[0]

perimeter = cv2.arcLength(contour, True)

distance = 0.005 * perimeter
approximate_contour = cv2.approxPolyDP(contour, distance, True)

color = (0, 255, 0) #BGR
cv2.drawContours(image_as_BGR, [approximate_contour], -1, color, 2)

cv2.imshow("Imagen con los contornos", image_as_BGR)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

CARACTERÍSTICAS DE LOS CONTORNOS (ENVOLTURA CONVEXA)

- La envoltura convexa tiene un resultado similar a la aproximación de contorno
- En algunos casos puede producir el mismo resultado que la aproximación de contorno
- La envoltura convexa comprueba una curva en busca de defectos de convexidad, para posteriormente corregirla
- Una curva es convexa si al escoger 2 puntos M y N, el segmento que los une queda por encima de la curva



PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

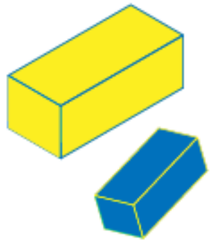
CARACTERÍSTICAS DE LOS CONTORNOS (ENVOLTURA CONVEXA)

- En términos generales, las curvas convexas son aquellas que siempre están abultadas, o al menos planas
- Si la curva se encuentra abombada en el interior, entonces se dice que tiene defectos de convexidad
- En la imagen, la línea roja muestra la envoltura convexa de la mano
- Las flechas muestran los defectos de convexidad, que son las desviaciones máximas locales de la envoltura de los contornos

PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Encontrar la envoltura convexa



convexHull : Encuentra la envoltura convexa de un conjunto de puntos

Sintaxis: cv2.convexHull(curva[, envoltura[, sentido[, puntos]]])

Parametros:
curva : Es una lista de puntos almacenados como coordenadas (x, y)
envoltura : Es el resultado del método
sentido : Flag que determina como se debe orientar la envoltura convexa
puntos : Determina los puntos a devolver

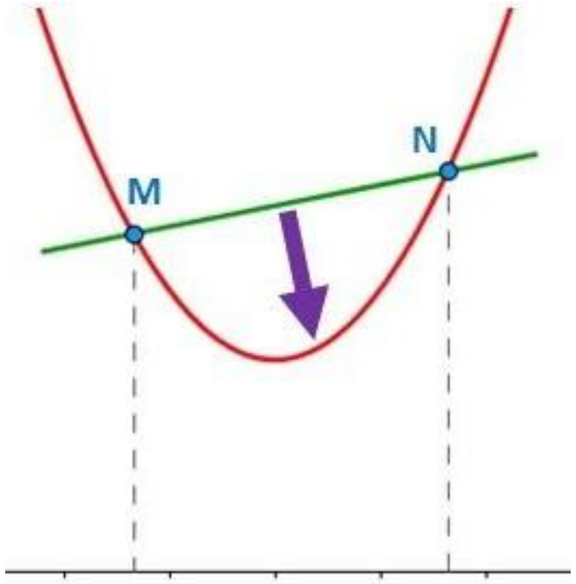
Nota Si el parámetro «sentido» es igual a «True», la envoltura convexa es orientada en sentido horario; caso contario, se orienta en sentido antihorario. El parámetro «puntos» por defecto es igual a «True», por lo que devuelve las coordenadas de los puntos de la envoltura; si es igual a «False», devuelve los índices de la lista de los puntos del contorno, correspondientes a los puntos de la envoltura

Ejemplo:

```python
import cv2

image_as_BGR = cv2.imread("image_3.png")
image_as_GRAY = cv2.cvtColor(image_as_BGR, cv2.COLOR_BGR2GRAY)

valor_umbral = 127
valor_maximo = 255
tipo = cv2.THRESH_BINARY
return_value, image_with_threshold = cv2.threshold(image_as_GRAY, valor_umbral, valor_maximo, tipo)

contours, hierarchy = cv2.findContours(image_with_threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

contour = contours[0]

convex_hull = cv2.convexHull(contour, returnPoints=True)
print("Envoltura convexa:")
print(convex_hull)

color = (0, 255, 0) #BGR
cv2.drawContours(image_as_BGR, [convex_hull], -1, color, 2)

cv2.imshow("Imagen con los contornos", image_as_BGR)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

CARACTERÍSTICAS DE LOS CONTORNOS (REVISIÓN DE CONVEXIDAD)

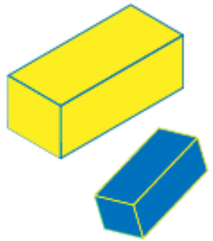
- Sirve para determinar si una curva es o no convexa
- En nuestro caso, nos permite identificar si el contorno es o no convexo
- En OpenCV, se puede obtener mediante el método «isContourConvex»; sin embargo, el contorno debe ser simple, esto implica que no se intersecte consigo mismo



PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Determinar convexidad



isContourConvex : Verifica si un contorno es o no convexo

Sintaxis: cv2.isContourConvex(curva)

Parametros:
curva : Es una lista de puntos almacenados como coordenadas (x, y)

Nota En el caso que el contorno se intersecte consigo mismo, el resultado del método «isContourConvex» será indefinido

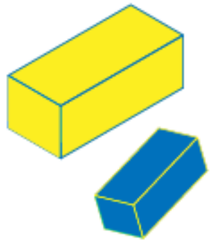
PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Ejemplo:

```
import cv2

image_as_BGR = cv2.imread("image_2.png")
image_as_GRAY = cv2.cvtColor(image_as_BGR, cv2.COLOR_BGR2GRAY)

valor_umbral = 127
valor_maximo = 255
tipo = cv2.THRESH_BINARY
return_value, image_with_threshold = cv2.threshold(image_as_GRAY, valor_umbral, valor_maximo, tipo)

contours, hierarchy = cv2.findContours(image_with_threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

contour = contours[0]

print("Contorno:")
print(contour)

is_convex_contour = cv2.isContourConvex(contour)

print("\n\nEs un contorno convexo:")
print(is_convex_contour)
```

CARACTERÍSTICAS DE LOS CONTORNOS (CUADRO DELIMITADOR)



- Existe 2 tipos de cuadros delimitadores: con y sin rotación

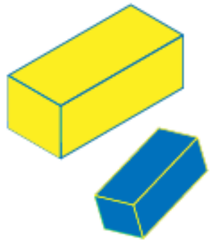
Cuadro recto:

- Es el tipo sin rotación
- No considera la rotación del objeto
- Se obtiene mediante el método «boundingRect»
- La coordenada superior izquierda del cuadro es denotada como (x, y)
- El ancho y la altura del cuadro son denotados como (w, h)

PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Obtener cuadro delimitador



boundingRect : Calcula el cuadro delimitador

Sintaxis: cv2.boundingRect(curva)

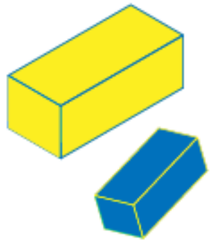
Parametros:
curva : Es una lista de puntos almacenados como coordenadas (x, y)

Nota El cuadro delimitador que contenga al objeto representado mediante la curva (contorno) debe ser lo más pequeño posible

Ejemplo:

```python
import cv2

image_as_BGR = cv2.imread("image_4.png")
image_as_GRAY = cv2.cvtColor(image_as_BGR, cv2.COLOR_BGR2GRAY)

valor_umbral = 127
valor_maximo = 255
tipo = cv2.THRESH_BINARY
return_value, image_with_threshold = cv2.threshold(image_as_GRAY, valor_umbral, valor_maximo, tipo)

contours, hierarchy = cv2.findContours(image_with_threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

contour = contours[0]

x, y, w, h = cv2.boundingRect(contour)

start_point = (x, y)
end_point = (x + w, y + h)
color = (0, 255, 0) #BGR
cv2.rectangle(image_as_BGR, start_point, end_point, color, 2)

cv2.imshow("Imagen con los contornos", image_as_BGR)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

CARACTERÍSTICAS DE LOS CONTORNOS (CUADRO DELIMITADOR)

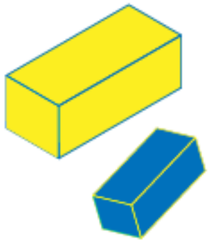


Cuadro rotado:

- Es el tipo con rotación
- El cuadro delimitador será el área mínima necesaria para encerrar un objeto
- Considera la rotación del objeto
- Se obtiene mediante el método «minAreaRect»
- La coordenada superior izquierda del cuadro es denotada como (x, y)
- El ancho y la altura del cuadro son denotados como (w, h)
- Adicionalmente, se obtiene el ángulo de rotación

PROCESAMIENTO DIGITAL DE IMÁGENES

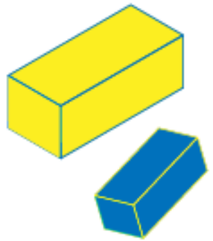
CARACTERÍSTICAS DE LOS CONTORNOS (CUADRO DELIMITADOR)



Cuadro rotado:

- Para graficar este cuadro se requieren sus 4 esquinas
- Las esquinas se obtienen mediante el método «boxPoints»
- Una vez que se obtiene las 4 esquinas del cuadro, se estila convertir sus tipo de dato a «np.int64»
- Entonces, se hace uso del método «drawContours» para graficar el cuadro delimitador rotado sobre la imagen original
- En el método «drawContours» se pasará las 4 esquinas del cuadro rotado dentro de una lista y se usará el valor «0» como índice, para indicar que se grafique estas 4 esquinas

Obtener cuadro delimitador



minAreaRect : Calcula el cuadro delimitador rotado

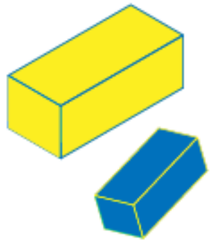
Sintaxis: cv2.minAreaRect(curva)

Parametros:

curva : Es una lista de puntos almacenados como coordenadas (x, y)

Nota El cuadro delimitador rotado posee un área que encierra los puntos, tal que dicha área es la más pequeña posible, al punto de ser un área mínima

Encontrar los 4 vértices del cuadro rotado



boxPoints : Encuentra los cuatro vértices de un cuadro girado

Sintaxis: cv2.boxPoints(cuadro_rotado)

Parametros:
cuadro_rotado : Es un cuadro rotado

Nota Este método normalmente se usa luego del método «minAreaRect» con el objetivo de obtener los 4 vértices del cuadro y de esta forma poder graficarlo

PROCESAMIENTO DIGITAL DE IMÁGENES

Ejemplo:

```python
import cv2
import numpy as np

image_as_BGR = cv2.imread("image_4.png")
image_as_GRAY = cv2.cvtColor(image_as_BGR, cv2.COLOR_BGR2GRAY)

valor_umbral = 127
valor_maximo = 255
tipo = cv2.THRESH_BINARY
return_value, image_with_threshold = cv2.threshold(image_as_GRAY, valor_umbral, valor_maximo, tipo)

contours, hierarchy = cv2.findContours(image_with_threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

contour = contours[0]

(x, y), (w, h), angle = cv2.minAreaRect(contour)

box = cv2.boxPoints(((x, y), (w, h), angle))

box = np.int64(box)

color = (0, 255, 0) #BGR
cv2.drawContours(image_as_BGR, [box], -1, color, 2)

cv2.imshow("Imagen con los contornos", image_as_BGR)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO