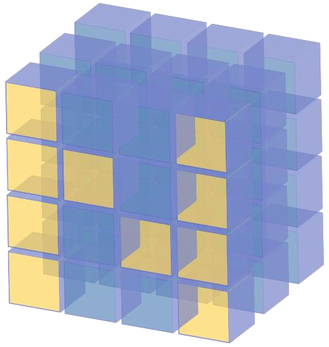
PROCESAMIENTO DIGITAL DE IMAGENES

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

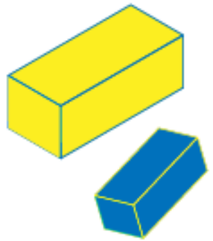
TEMAS QUE VEREMOS HOY

- Cuadro delimitador
- Círculo mínimo de inclusión
- Ajustar una elipse
- Ajustar una línea
- Relación de aspecto
- Extensión
- Solidez
- Diámetro equivalente
- Orientación
- Máscara y las coordenadas de sus píxeles

PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

CARACTERÍSTICAS DE LOS CONTORNOS (CUADRO DELIMITADOR)



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

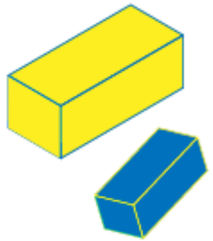
Cuadro recto:

- 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



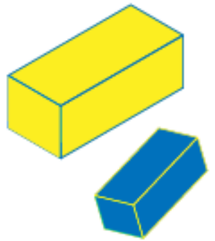
boundingRect : Calcula el cuadro delimitador

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

CARACTERÍSTICAS DE LOS CONTORNOS (CUADRO DELIMITADOR)

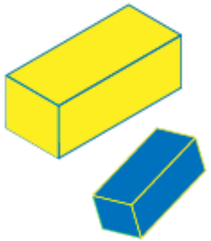


Cuadro rotado:

- 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

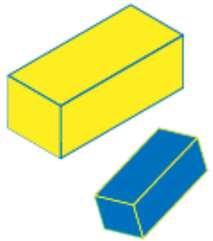
CARACTERÍSTICAS DE LOS CONTORNOS (CUADRO DELIMITADOR)



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

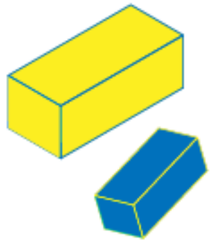
Sintaxis: cv2.minAreaRect(curva)

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

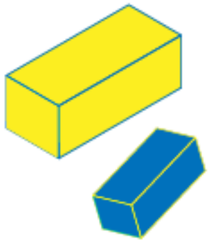
Nota Este método normalmente se usa luego del método «minAreaRect» con el objetivo de obtener los 4 vértices del cuadro y de esta forma poder graficarlo

PROCESAMIENTO DIGITAL DE IMÁGENES

Ejemplo:

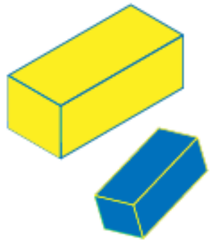
```python
import cv2
import numpy as np

image_as_BGR = cv2.imread("image_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)

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()
```

CARACTERÍSTICAS DE LOS CONTORNOS (CÍRCULO MÍNIMO DE INCLUSIÓN)



- Consiste en determinar una circunferencia para una objeto
- Esta circunferencia debe encerrar al objeto
- En el caso que la circunferencia se rellene, el objeto quedaría oculto
- Además, el área de la circunferencia debe ser la mínima posible
- En OpenCV, se obtiene mediante el método «minEnclosingCircle»

Obtener círculo delimitador



minEnclosingCircle : Calcula el círculo delimitador

Sintaxis: cv2.minEnclosingCircle(curva)

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

Nota El círculo delimitador se calcula sobre un contorno, tal que su área que delimita el objeto sea mínima. Este método devuelve el centro y radio del círculo delimitador

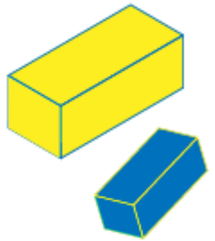
PROCESAMIENTO DIGITAL DE IMÁGENES

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)

contour = contours[0]

(x, y), radius = cv2.minEnclosingCircle(contour)

center = (int(x), int(y))
radius = int(radius)
color = (0, 255, 0) #BGR
cv2.circle(image_as_BGR, center, radius, color, 2)

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

cv2.waitKey(0)

cv2.destroyAllWindows()
```

CARACTERÍSTICAS DE LOS CONTORNOS (AJUSTAR UNA ELIPSE)



- Consiste en determinar una elipse para una objeto
- Esta elipse se ajusta al objeto en cuestión
- Un cuadro delimitador rotado ayuda a obtener esta elipse
- Se usa el cuadro rotado para representa la elipse inscrita
- Esta elipse no necesariamente encierra al objeto totalmente
- La causa de esto es que al obtener la elipse inscrita en el cuadro rotado se puede dejar fuera regiones del objeto, siendo que el cuadro rotado no es un cuadro de área mínima tal que encierre todo el objeto
- En OpenCV, se obtiene mediante el método «fitEllipse»

Obtener elipse

fitEllipse : Se ajusta a una elipse alrededor de un conjunto de puntos 2D

Sintaxis: cv2.fitEllipse(curva)

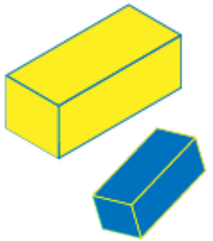
Parametros:
curva : Es una lista de puntos almacenados como coordenadas (x, y)

Nota Este método devuelve el cuadro delimitador rotado en el que la elipse se encuentra inscrita. Se calcula una elipse que se ajuste a la lista de puntos que forman el contorno lo mejor posible, este ajuste es en términos de mínimos cuadrados

Ejemplo:

```
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)

contour = contours[0]

ellipse = cv2.fitEllipse(contour)

color = (0, 255, 0) #BGR
cv2.ellipse(image_as_BGR, ellipse, color, 2)

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

cv2.waitKey(0)

cv2.destroyAllWindows()
```

CARACTERÍSTICAS DE LOS CONTORNOS (AJUSTAR UNA LÍNEA)

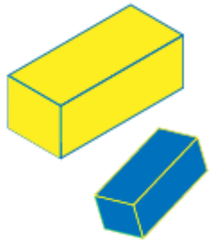


- Consiste en ajustar una línea a un conjunto de puntos
- El conjunto de puntos representa el contorno del objeto
- El ajuste se realiza mediante la minimización de una distancia
- La distancia representa que tan alejado se encuentra a línea a ajustar respecto del conjunto de puntos
- En OpenCV, se obtiene mediante el método «fitLine»

PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Obtener línea



fitLine : Se ajusta a una elipse alrededor de un conjunto de puntos 2D

Sintaxis: cv2.fitLine(curva, distancia, parámetro, radio, ángulo)

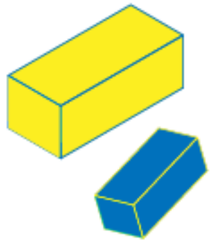
Parametros:
curva : Es una lista de puntos almacenados como coordenadas (x, y)
distancia : Tipo de distancia usada para ajustar la línea
parámetro : Es el parámetro numérico «C» usado en algunos tipos de distancia
radio : Precisión para el radio
ángulo : Precisión para el ángulo

Nota El radio indicado en el parámetro «radio» es la distancia entre el origen de coordenadas y la línea a ajustar. Este método usa una técnica iterativa, mediante la precisión del radio y el ángulo puede evitar realizar iteraciones innecesariamente. Normalmente, los parámetros «radio» y «ángulo» suelen ser «0.01». Por otro lado, «parámetro» suele ser 0, debido que así un valor óptimo es elegido

Ejemplo:

```python
import cv2

image_as_BGR = cv2.imread("image_1.png")
image_as_GRAY = cv2.cvtColor(image_as_BGR, cv2.COLOR_BGR2GRAY)
rows, columns, channels = image_as_BGR.shape

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]

(vx, vy, x, y) = cv2.fitLine(contour, cv2.DIST_L2, 0, 0.01, 0.01)

right_y = int(((columns - x) * vy / vx) + y)
left_y = int((-x * vy / vx) + y)

color = (0, 255, 0) #BGR
cv2.line(image_as_BGR, (columns, right_y), (0, left_y), color, 2)

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

cv2.waitKey(0)

cv2.destroyAllWindows()
```

PROPIEDADES DE LOS CONTORNOS

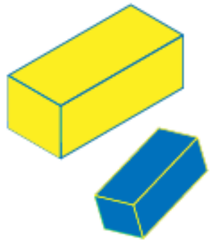


- Las propiedades de los contornos son:

Relación de aspecto
Extensión
Solidez
Diámetro equivalente
Orientación
Máscara y las coordenadas de sus píxeles
Puntos extremos

PROCESAMIENTO DIGITAL DE IMÁGENES

CARACTERÍSTICAS DE LOS CONTORNOS (RELACIÓN DE ASPECTO)



- Se usa el ancho y la altura del contorno del objeto
- Es la razón entre el ancho y la altura

$$R = \frac{ancho}{altura}$$

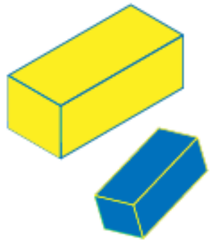
PROCESAMIENTO DIGITAL DE IMÁGENES

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)

contour = contours[0]

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

aspect_ratio = w / h

print("La relación de aspecto es {}".format(aspect_ratio))
```

CARACTERÍSTICAS DE LOS CONTORNOS (EXTENSIÓN)



- Hace uso del contorno y rectángulo delimitador
- Es la razón entre el área del contorno y el área del rectángulo delimitador

$$E = \frac{Área\_del\_contorno}{Área\_rectángulo\_delimitador}$$

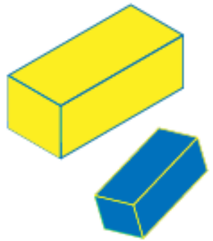
PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Ejemplo:

```
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)

contour = contours[0]

contour_area = cv2.contourArea(contour)

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

bounding_rectangle_area = w * h

extent = contour_area / bounding_rectangle_area

print("La extensión es {}".format(extent))
```

CARACTERÍSTICAS DE LOS CONTORNOS (SOLIDEZ)



- Hace uso del contorno y la envoltura convexa
- Es la razón entre el área del contorno y el área de la envoltura convexa

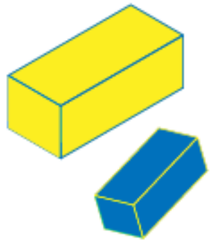
$$S = \frac{Área\_del\_contorno}{Área\_envoltura\_convexa}$$

PROCESAMIENTO DIGITAL DE IMÁGENES

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)

contour = contours[0]

contour_area = cv2.contourArea(contour)

convex_hull = cv2.convexHull(contour)

convex_hull_area = cv2.contourArea(convex_hull)

solidity = contour_area / convex_hull_area

print("La solidez es {}".format(solidity))
```

CARACTERÍSTICAS DE LOS CONTORNOS (DIÁMETRO EQUIVALENTE)



- Hace referencia al diámetro de un círculo
- El área del círculo debe ser igual al área del contorno

$$D = \sqrt{\frac{4 * Área\_contorno}{\pi}}$$

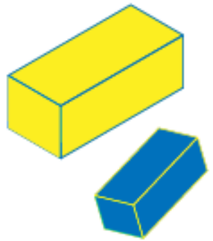
PROCESAMIENTO DIGITAL DE IMÁGENES

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO

Ejemplo:

```
import cv2
import numpy as np

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)

contour = contours[0]

area = cv2.contourArea(contour)

equivalent_diameter = np.sqrt(4 * area / np.pi)

print("El diámetro equivalente es {}".format(equivalent_diameter))
```

CARACTERÍSTICAS DE LOS CONTORNOS (ORIENTACIÓN)



- Para obtener la orientación de un objeto se hace uso de su elipse circunscrita
- Esta elipse contiene 2 ejes
- Se hace uso del menor eje de la elipse circunscrita
- El menor eje se junta con la dirección horizontal
- El ángulo entre el menor eje y la dirección horizontal es la orientación
- La orientación representa el ángulo al que está dirigido el objeto

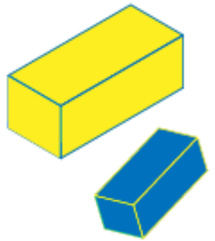
PROCESAMIENTO DIGITAL DE IMÁGENES

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)

contour = contours[0]

ellipse = cv2.fitEllipse(contour)

(center_x, center_y), (minor_axis_length, major_axis_length), angle = ellipse

print("La orientación es {}".format(angle))

color = (0, 255, 0) #BGR
cv2.ellipse(image_as_BGR, ellipse, color, 2)

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

cv2.waitKey(0)

cv2.destroyAllWindows()
```

CARACTERÍSTICAS DE LOS CONTORNOS (MÁSCARA Y LAS COORDENADAS DE SUS PÍXELES)



- Sirve para obtener la máscara de un objeto
- También permite obtener los píxeles que conforman el objeto
- Para esto, primero se obtiene se obtiene los contornos de un objeto
- Segundo, se crea una imagen de color negro, del mismo tamaño que la imagen original
- Tercero, se dibuja los contornos sobre la imagen creada
- Finalmente, se obtiene las coordenadas de los píxeles que conforman el contorno del objeto

PROCESAMIENTO DIGITAL DE IMÁGENES

Ejemplo:

```python
import cv2
import numpy as np

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

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)

mask = np.zeros((rows, columns), np.uint8)

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

pixel_points = cv2.findNonZero(mask)

print("Los puntos de píxeles son: {}".format(pixel_points))

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

cv2.imshow("Máscara de la imagen", mask)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

UMAKER | CENTRO DE CAPACITACIÓN DE DESARROLLO TECNOLÓGICO