## ****Código explicado por secciones****

### 1. Importación de librerías

python

Copiar código

import cv2 # OpenCV: procesamiento de imágenes

Solo se importa OpenCV (no necesitas numpy en esta versión).

### 2. Inicialización de la cámara

python

Copiar código

camara = cv2.VideoCapture(0)

if not camara.isOpened():

print("No se puede abrir la cámara")

exit()

* cv2.VideoCapture(0): abre la cámara por defecto.
* Se verifica si está disponible. Si no, se imprime un mensaje y se termina el programa.

### 3. Mensaje al usuario

python

Copiar código

print("Presiona 'q' para salir")

Indica al usuario cómo salir del programa.

### 4. Bucle principal

python

Copiar código

while True:

ret, frame = camara.read()

* Se entra en un bucle infinito que capturará cuadros (frame) continuamente.
* ret es True si la captura fue exitosa.

### 5. Verificar captura

python

Copiar código

if not ret:

print("Error al capturar imagen")

break

* Si ret es False, se rompe el bucle: algo salió mal con la cámara.

### 6. Conversión a escala de grises

python

Copiar código

gris = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

* Convierte la imagen de color (formato BGR de OpenCV) a escala de grises.
* Esto facilita operaciones como umbralización y contornos, reduciendo ruido visual.

### 7. Umbralización (binarización)

python

Copiar código

\_, binaria = cv2.threshold(gris, 100, 255, cv2.THRESH\_BINARY\_INV)

* Convierte la imagen en blanco y negro (binaria):
  + Los píxeles con valor menor a 100 se vuelven **blancos** (255).
  + Los mayores o iguales se vuelven **negros** (0).
* El sufijo INV invierte el resultado: los objetos oscuros quedan blancos, y el fondo queda negro.

### 8. Detección de contornos

python

Copiar código

contornos, \_ = cv2.findContours(binaria, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

* Detecta contornos blancos sobre el fondo negro.
* RETR\_EXTERNAL: solo busca contornos externos (ignora contornos internos).
* CHAIN\_APPROX\_SIMPLE: simplifica la cantidad de puntos del contorno (más rápido y eficiente).

### 9. Procesamiento de contornos

python

Copiar código

for contorno in contornos:

area = cv2.contourArea(contorno)

if area > 500:

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

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

cv2.putText(frame, f"Area: {int(area)}", (x, y - 10),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 1)

#### Para cada contorno:

* contourArea: mide el área del objeto detectado.
* Si el área es mayor a 500 píxeles, se considera un **objeto válido**.
* boundingRect: obtiene el rectángulo más pequeño que encierra ese contorno.
* cv2.rectangle: dibuja el rectángulo verde.
* cv2.putText: muestra el valor del área sobre el rectángulo.

### 10. Mostrar resultados

python

Copiar código

cv2.imshow("Camara", frame)

cv2.imshow("Binaria", binaria)

* Camara: muestra el video con los rectángulos dibujados.
* Binaria: muestra la imagen binarizada que se usó para detectar objetos.

### 11. Salida al presionar 'q'

python

Copiar código

if cv2.waitKey(1) & 0xFF == ord('q'):

break

* Espera 1 ms por una tecla.
* Si el usuario presiona 'q', se rompe el bucle.

### 12. Limpieza

python

Copiar código

camara.release()

cv2.destroyAllWindows()

* Libera la cámara.
* Cierra todas las ventanas abiertas por OpenCV.