COMPUTACIÓN GRÁFICA

PRACTICA CALIFICADA 1

DE LA CRUZ VALDIVIESO, PEDRO LUIS
AZAÑA VEGA, LUIS ANGEL
CALAGUA MALLQUI JAIRO ANDRE
RIVAS GALINDO HUGO
ESPINOZA PARI FRANKLIN

Conceptos Previos

THRESHOLDING

TÉCNICA USADA EN EL PROCESAMIENTO DE IMÁGENES DIGITALES, MEDIANTE EL CUAL SE CONVIERTE UNA IMAGEN (GENERALMENTE EN ESCALA DE GRISES) DE UNA MATRIZ DE PÍXELES CON VALORES DE INTENSIDAD COMPRENDIDOS ENTRE 0 Y 256

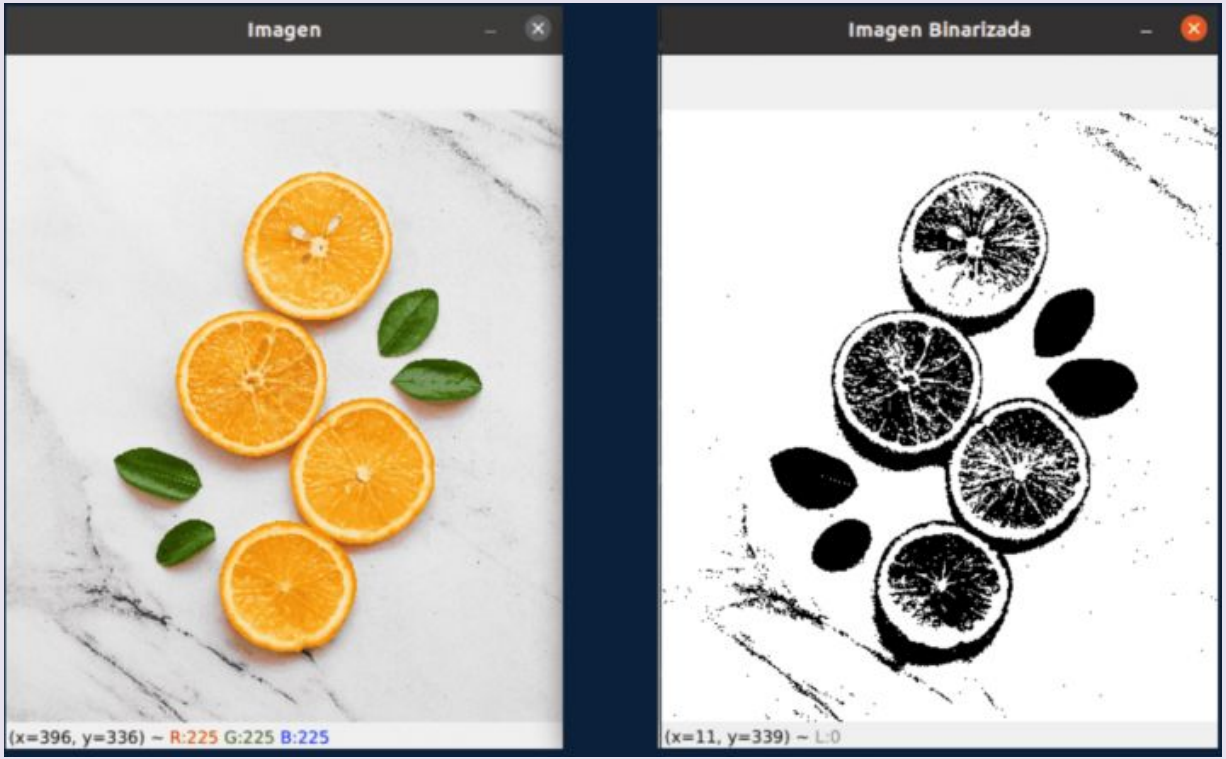


Imagen
(x=396, y=336) ~ R:225 G:225 B:225
Imagen Binarizada
(x=11, y=339) ~ L:0

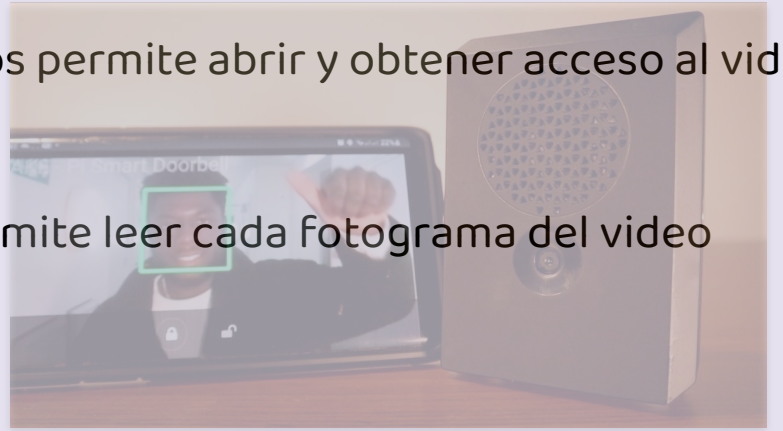
OpenCV

OpenCV es una biblioteca líder en visión por computadora, nos proporciona un conjunto de herramientas robustas y eficientes para la implementación, en esta presentación, de un sistema de reconocimiento de personas en movimiento. Cuenta con una amplia gama de funciones y algoritmos optimizados que OpenCV permite la detección, seguimiento y análisis de personas en tiempo real en entornos diversos.

Funciones a usar:

`cv2.VideoCapture("video.mp4")`: Nos permite abrir y obtener acceso al video deseado.

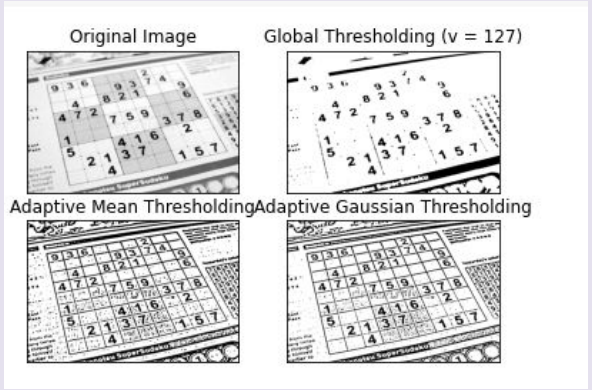
`cv2.VideoCapture().read()`: Nos permite leer cada fotograma del video secuencialmente.



`cv2.cvtColor()`: Nos permitirá convertir los fotogramas a escalas de grises.

`cv2.absdiff()`: Nos permite calcular la diferencia absoluta entre dos fotogramas .

`cv2.threshold()`: Nos permite aplicar un umbral, para así obtener una imagen binaria que resalta las áreas de cambio significativo.

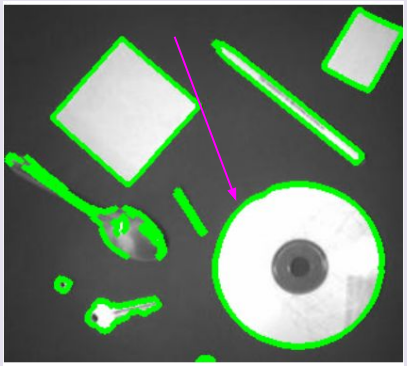


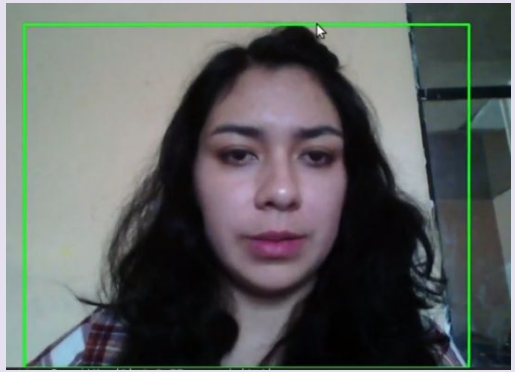
`cv2.findContours()`: Nos permite identificar los contornos de los objetos en movimiento.

`cv2.contourArea()`: Nos permite calcular el área en píxeles del contorno.

`cv2.boundingRect()`: Devuelve las coordenadas x e y, además del ancho y el alto de los contornos.

`cv2.rectangle()`: Nos permite dibujar un rectángulo.





02

Código

Explicación del código

Creación de ventana

```python
video = cv2.VideoCapture("../recursos/soy_un_video.mp4")  # Inicializa una captura de video
width = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))  # Ancho original del video
height = int(video.get(cv2.CAP_PROP_FRAME_HEIGHT))  # Alto original del video

cv2.namedWindow('Frame', cv2.WINDOW_NORMAL)  # Crea una ventana redimensionable
cv2.resizeWindow('Frame', width, height)  # Establece las dimensiones originales
```

```
hugo@asus:~/Graficas/computacionGrafica/sprint_2$ python3 test.py
El ancho del video es 1920
El alto del video es 1080
```

Haciendo uso del módulo cv2, se crea una ventana en la que se mostrará el video. Luego se redimensionará la ventana según las dimensiones del video.

Captura de un frame

```python
i = 0  # Contador

while True:
    ret, frame = video.read()  # Intentamos leer cada frame del video
    if not ret:
        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  # Convertir a escala de grises

    if i == 20:
        bgGray = gray.copy()  # Hacer una copia de la imagen en escala de grises como fondo

    if i > 20:
        dif = cv2.absdiff(gray, bgGray)
        _, th = cv2.threshold(dif, 40, 255, cv2.THRESH_BINARY)
```

Se usa un contador para realizar una separación entre la captura inicial del frame y de los frames posteriores.

Captura de un frame

Se muestra la imagen original y la imagen convertida a escala de grises

Captura de un frame

Captura de un frame

Contorno de un frame

```python
if i > 20:
    dif = cv2.absdiff(gray, bgGray)
    _, th = cv2.threshold(dif, 40, 255, cv2.THRESH_BINARY)

    # Encontrar contornos en la imagen umbralizada
    conts, _ = cv2.findContours(th, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Dibujar contornos en una copia del frame original
    frame_with_contours = frame.copy()

    for c in conts:
        area = cv2.contourArea(c)
        if area > 9000:
            x, y, w, h = cv2.boundingRect(c)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)  # Dibujar rectángulo
```

cv2.findCountours(imagen binarizada, modo de recuperación de contornos, método de aproximación de contornos.)
cv2.rectangle(ventana, punto origen, punto opuesto al origen, color, grosor)

Terminar programa

```
        # Mostrar el frame escalado en la ventana
        cv2.imshow('Frame', scaled_frame)

    i += 1
    if cv2.waitKey(1) & 0xFF == ord('q'):  # Si se presiona la tecla 'q', termina el programa
        break

video.release()  # Liberar los recursos del video
cv2.destroyAllWindows()  # Cerrar todas las ventanas al finalizar
```

03

Conclusiones