

OPEN SOURCE COMPUTER VISION

Reconocimiento Facial

AGENDA DE HOY

1

Introducción

2

Cascadas

3

Presentación del código

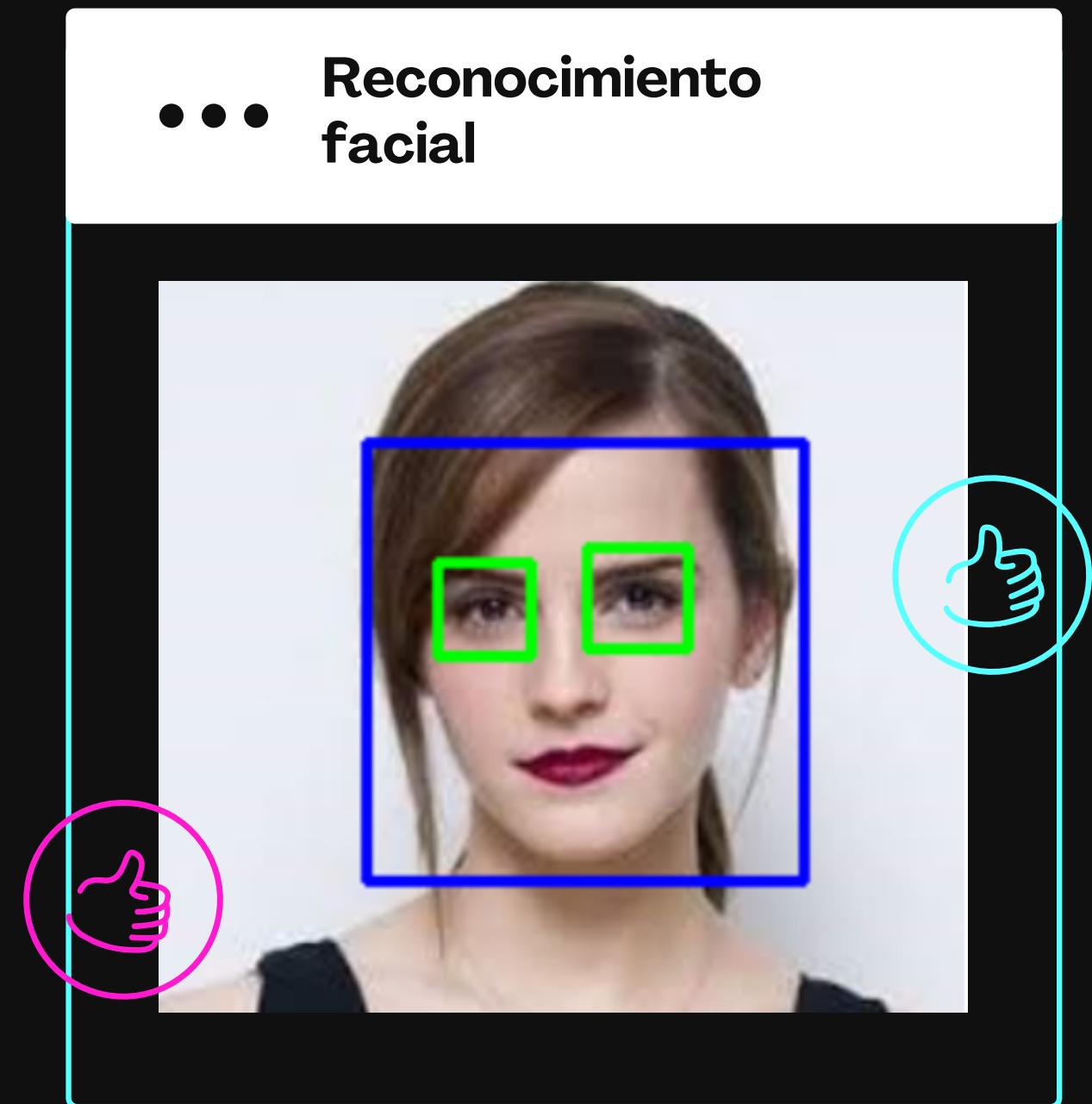
4

Conclusiones

INTRODUCCIÓN

¿Qué es OpenCV?

OpenCV es la biblioteca más popular para computer vision. Escrito originalmente en C / C ++, ahora proporciona enlaces para Python. OpenCV utiliza algoritmos de aprendizaje automático para buscar rostros dentro de una imagen.



CASCADAS

1

Definiendo los componentes del rostro.

2

Las divisiones o classifiers.

3

Dividiendo en etapas la solución,

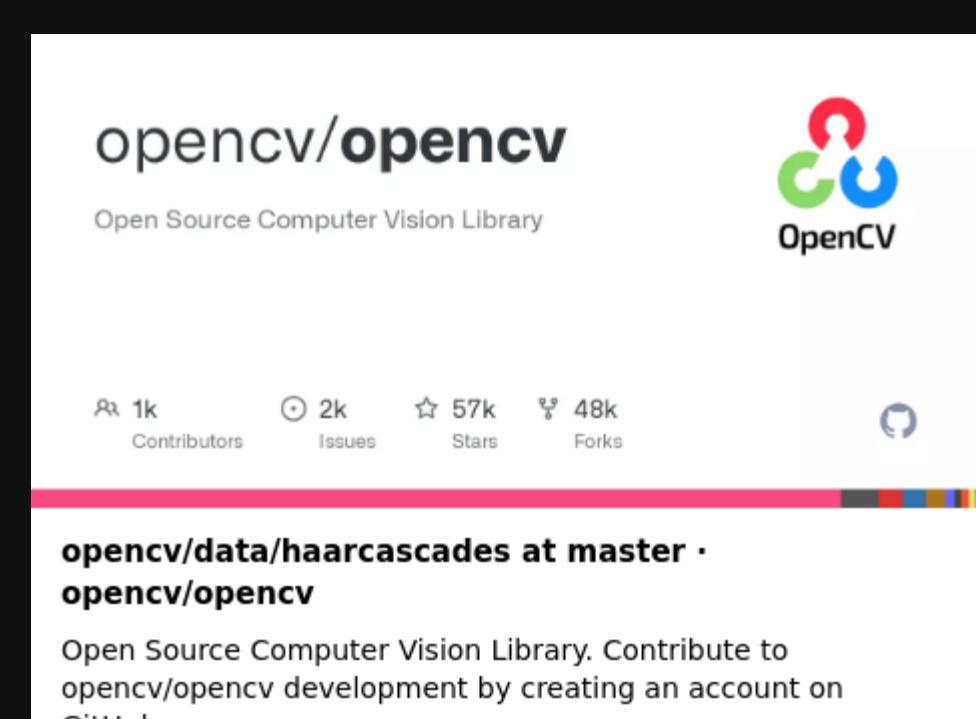
• • •

¿Qué son las cascadas?

Debido a que las caras son tan complicadas, no existe una prueba simple que le diga si encontró una cara o no. En cambio, hay miles de pequeños patrones y características que deben coincidir. Los algoritmos dividen la tarea de identificar la cara en miles de tareas más pequeñas y del tamaño de un bocado, cada una de las cuales es fácil de resolver. Estas tareas también se denominan **classifiers**.

Para la detección de rostros, el algoritmo comienza en la parte superior izquierda de una imagen y se mueve hacia abajo a través de pequeños bloques de datos, mirando cada bloque, preguntando constantemente: “¿Es este un rostro?”. Dado que hay más de 6.000 pruebas por bloque, se tienen que hacer millones de cálculos. Para evitar esto, **OpenCV usa cascadas**. Básicamente, OpenCV divide el problema de detectar rostros en múltiples etapas. Para cada bloque, hace una prueba muy aproximada y rápida. Si la pasa, se hace una prueba más detallada, y así sucesivamente. Puede haber 30 - 50 cascadas, y solo detectará una cara si pasan todas las etapas. La ventaja es que la mayoría de la imagen arrojará un resultado negativo durante las primeras etapas, lo que significa que el algoritmo no perderá tiempo probando las 6.000 funciones que contiene.

Las cascadas en sí mismas son solo un grupo de archivos XML que contienen datos OpenCV que se utilizan para detectar objetos. Inicializa el código con la cascada deseada y listo.



Acceso a las cascadas

OPEN CAMERA



Conociendo el código



¡Sí!



¡Esta es
genial!



Paso 1: Abrir la cámara

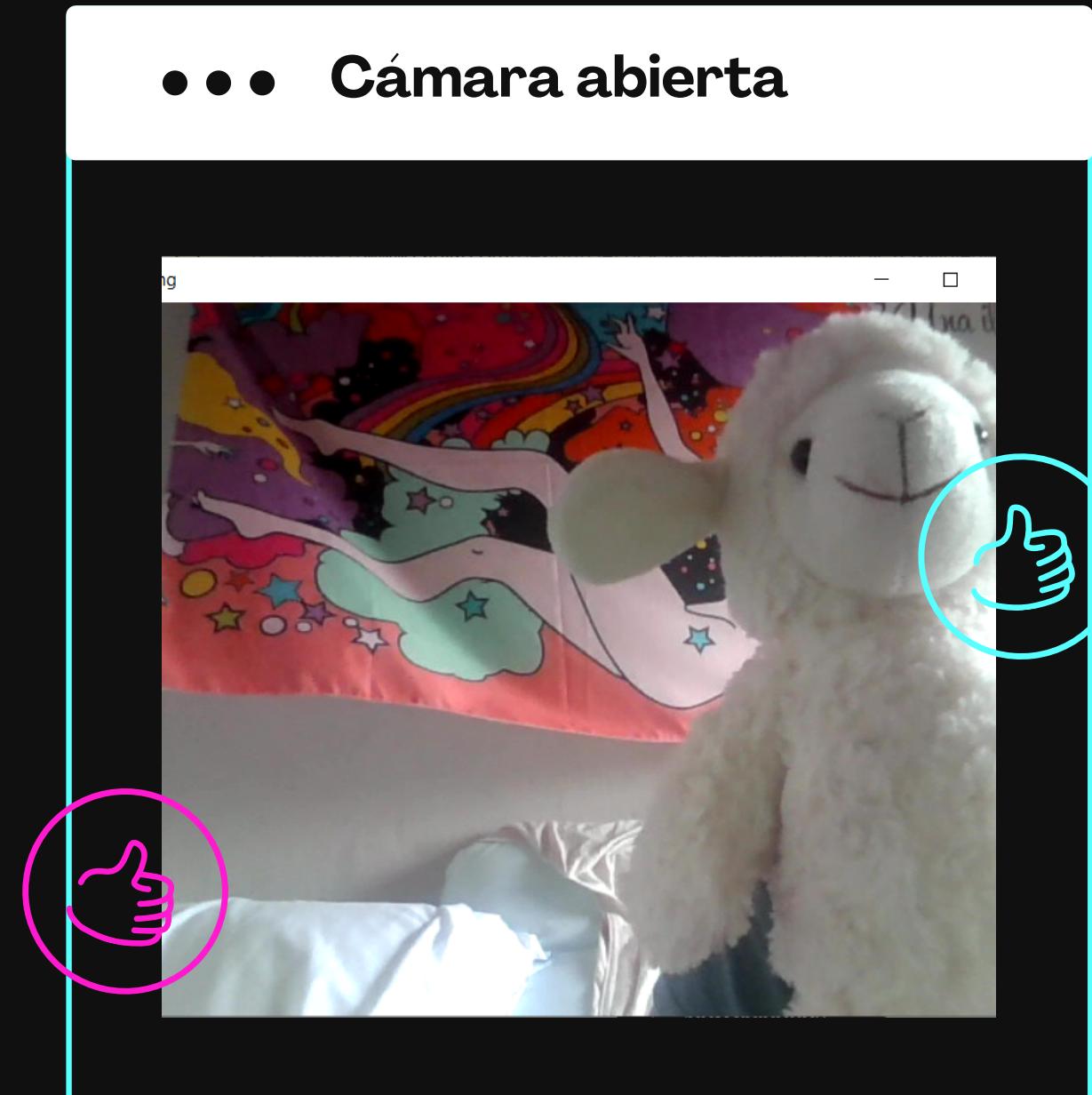
```
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

while(True):
    ret, frame = cap.read()
    cv2.imshow('Img', frame)

    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
    cap.release()
    cv2.destroyAllWindows()
```

• • • Cámara abierta



COLOR RECOGNITION

• • •

ColorRecognition.py

```
import numpy as np
import cv2

#Se crea la variable cap con cv2 para capturar frame por frame
cap=cv2.VideoCapture(0)

#Se crea un while infinito para poder mostrar el frame constantemente
while (True):
    ret,frame = cap.read()
    hav = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    #HSV = hue sat value
    lower_red = np.array([50,150,150])
    upper_red = np.array([180,255,255])

    #Se crea una mascara para filtrar el color
    mask = cv2.inRange(hav, lower_red,upper_red)
    #Se crea el resultado del filtro
    res = cv2.bitwise_and(frame, frame, mask = mask)

    #Se muestra el resultado del visor original, la mascara y el resultado final
    cv2.imshow('Visor',frame)
    cv2.imshow('Mask',mask)
    cv2.imshow('Result',res)

    #Se crea un condicional para finalizar el bucle infinito al presionar "q"
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

    cap.release()
    cv2.destroyAllWindows()
```

Conociendo el código

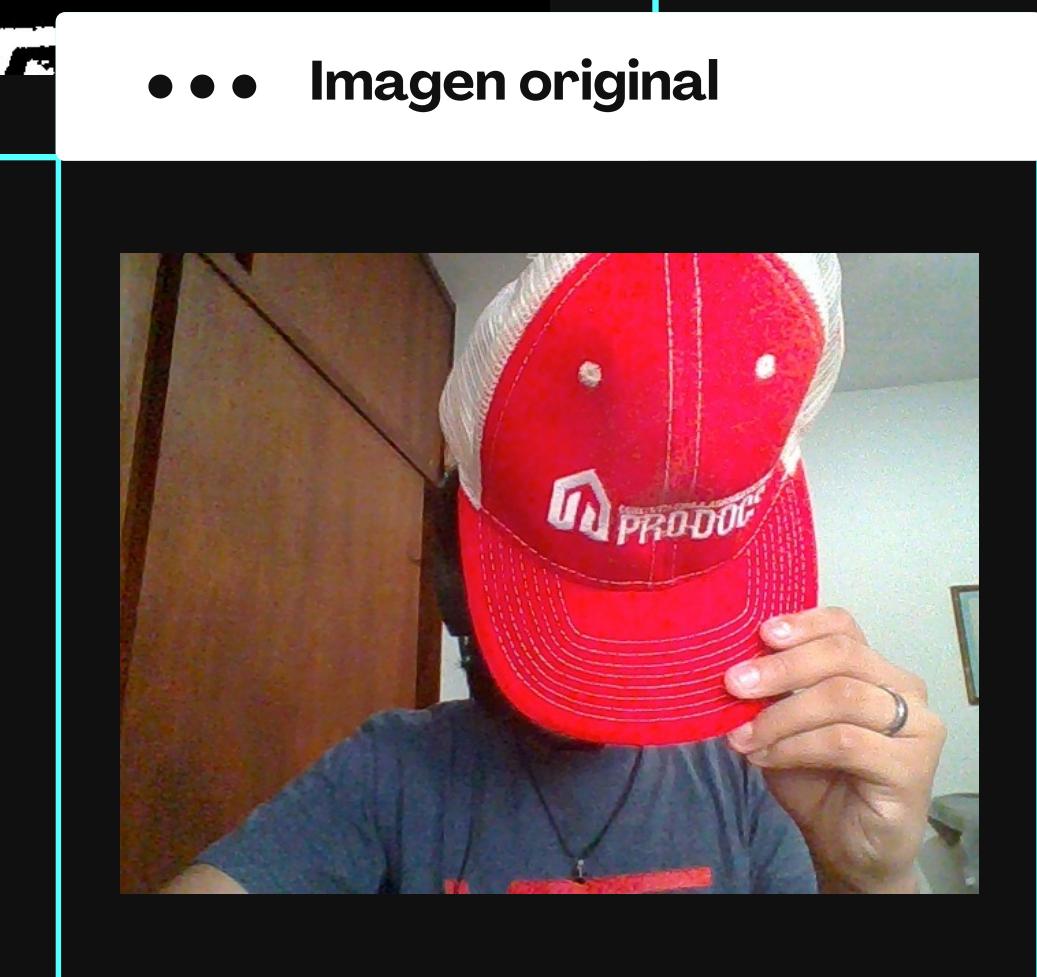


¡Sí!



¡Esta es
genial!

COLOR RECOGNITION



FACIAL RECOGNITION



Conociendo el código

•

```
import cv2

# The two xml docs to read facial expressions and eyes.
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

# Captures frame by frame from a camera.
capture = cv2.VideoCapture(0)

# Loop runs if capturing has been initialized.
while(True):
    # Reads frames from a camera
    ret, img = capture.read()

    # Converts to grayscale each frame
    grayscale = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Detects faces of different sizes in the input image
    faces = face_cascade.detectMultiScale(grayscale, 1.3, 5)
```

main.py

```
# For cycle:
for(x, y, w, h) in faces:
    # Draws a rectangle in a face
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 255, 0), 2)
    roi_gray = grayscale[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]

    # Detects eyes of different sizes in the input image
    eyes = eye_cascade.detectMultiScale(roi_gray)

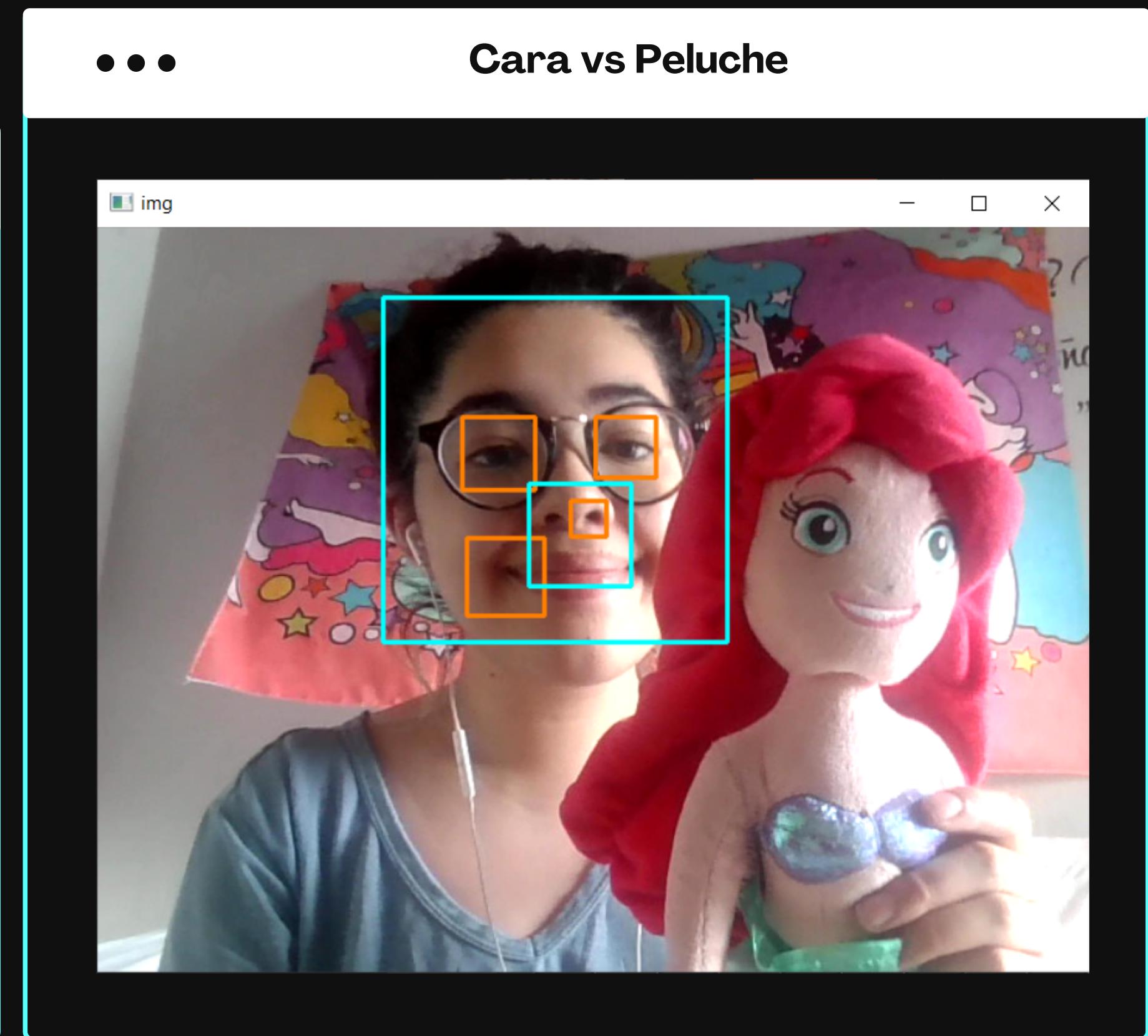
    # Draws rectangles in eyes
    for(ex, ey, ew, eh) in eyes:
        cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh),
                     (0, 127, 255), 2)

    # Displays an image in a window
    cv2.imshow('img',img)

    # Waits for Esc key to stop
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

capture.release()
cv2.destroyAllWindows()
```

FACIAL RECOGNITION



CONCLUSIÓN

El proyecto representa de forma muy sencilla una manera de enmardetectar distintos elementos en una imagen en vivo utilizando CV2 en Python. Se utilizaron no solo las herramientas aprendidas, sino también se aprovecharon otros recursos nuevos en su implementación. El caso más importante son las cascadas y las librerías de matrices.