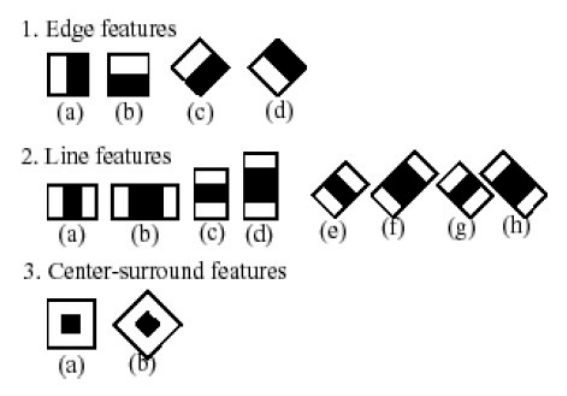
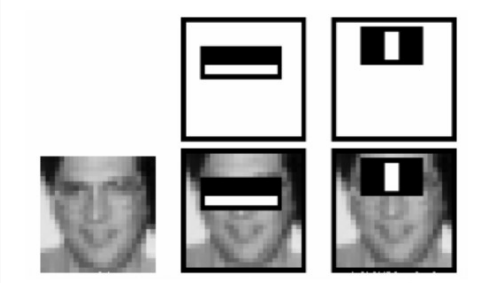
RECONOCIMIENTO FACIAL

Detección vs Reconocimiento

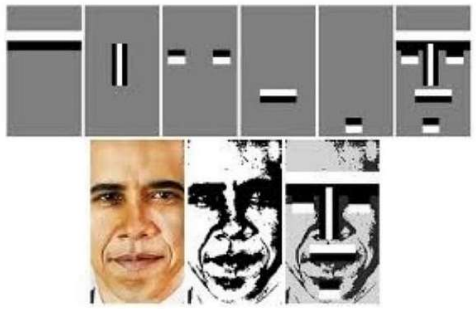
Encuentra el rostro	✓	✓
Identifica rasgos faciales	✓	✓
Funciona para imágenes y videos	✓	✓
Utiliza datos biométricos	✗	✓
Vincula los rasgos faciales con un nombre	✗	✓

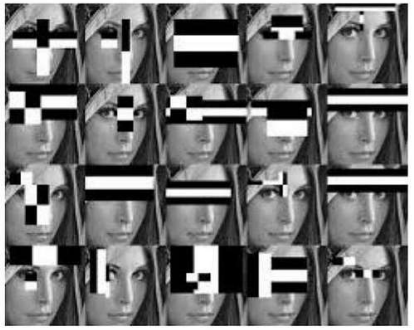
Detección Facial con OpenCV



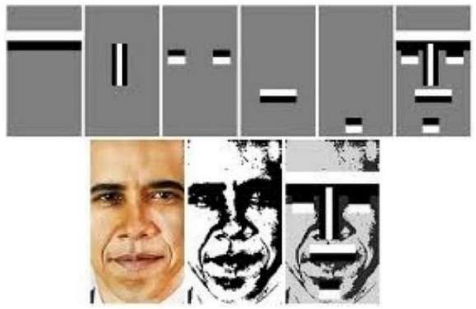


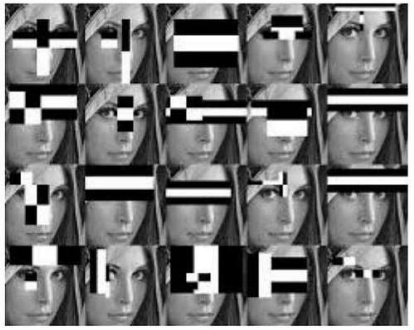
Detección Facial con OpenCV





Detección Facial con OpenCV





Detección Facial

```
import cv2

#Se carga el clasificador Haar (https://github.com/opencv/opencv/tree/master/data/haarcascades)
cascadeClassifier = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')

#Se inicia la webcam
webcam = cv2.VideoCapture(0)
```

Detección Facial

```
while(1):
    #Se captura un frame de la webcam para analizar
    valid, img = webcam.read()

    if valid:
        #Se convierte la imagen a escala de grises
        img_gris  = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Detección Facial

```
#Usando el clasificador se detectan las caras
coord_caras = cascadeClassifier.detectMultiScale(
    img_gris,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(30, 30),
    #flags=cv2.cv.CV_HAAR_SCALE_IMAGE
    )
```

Detección Facial

```
#Cada cara encontrada se va a marcar con un recuadro rojo
for(x,y,w,h) in coord_caras:
    cv2.rectangle(img,  (x, y), (x+w, y+h),(0,0,255),3)

#En una ventana nueva se muestra el resultado
cv2.imshow('Webcam', img)
```

Reconocimiento Facial

```
#Se cargan la imagen o imagenes
imagen1 = face_recognition.load_image_file('./caras/Pablo.jpg')
imagen2 = face_recognition.load_image_file('./caras/chalamet.jpg')

#Extrae los encodings de los rostros
encoding1 = face_recognition.face_encodings(imagen1)
encoding2 = face_recognition.face_encodings(imagen2)

#Se define un array con los encodings y otro con los nombres
encodings_conocidos = np.array([
    encoding1,
    encoding2
])

nombres_conocidos = np.array([
    "Pablo Garcia",
    "Timothee Chalamet"
])
```

Reconocimiento Facial

```
while(1):
    #Definimos arrays y variables a utilizar
    coord_caras = np.array([]) #Coordenadas de las caras detectadas
    encodings_caras = np.array([]) #Encodings de las caras detectadas
    nombres_caras = [] #Nombre de las personas
    total_distances = np.array([100.0,100.0,100.0,100.0,100.0,100.0]) #Array de "distancias" entre rostros
    procesar = True
    nombre = "Desconocido" #Variable para el nombre

    valid, img = webcam.read()
```

Reconocimiento Facial

```
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) #Convertir de BGR a RGB
img_rgb = cv2.resize(img_rgb, (0,0), fx = 1.0/reduccion, fy= 1.0 /reduccion) #Se aplica la reducción
#Se localizan las caras y se obtienen sus encodings
coord_caras = face_recognition.face_locations(img_rgb)
encodings_caras = face_recognition.face_encodings(img_rgb, coord_caras)
```

Reconocimiento Facial

```
for cara in encodings_caras:
    #Se buscan coincidencias con los encoding conocidos
    coincidencias = face_recognition.compare_faces(encodings_conocidos, cara, 0.2)
    matches = np.array(coincidencias)  #Se convierte en un array de numpy para poder
    face_distances = face_recognition.face_distance(encodings_conocidos, cara) #"Dis
    i=0
```

Reconocimiento Facial

```
if matches.all():
    #Para cada cara sumamos las "distancias"
    for face in face_distances:
        total_distances[i] = np.sum(face)
        i+=1
    best_match_index = np.argmin(total_distances) #Se la mejor coincidencia
    nombre = nombres_conocidos[best_match_index] #Se asigna el nombre de la mejor coincidencia
#Se añade al array de nombres
nombres_caras.append(nombre)
```

Reconocimiento Facial

```
#Recorremos tanto las coordenadas como los nombres
for(top, right, bottom, left), nombre in zip(coord_caras, nombres_caras):
    top *= reduccion
    right *= reduccion
    bottom *= reduccion
    left *= reduccion

    #Cambia el color si es conocido o no
    if nombre == 'Desconocido':
        color = (0,0,255) #Color rojo
    else:
        color = (0,255,0) #Color verde

    #Se dibuja el rectángulo que indica la posición de la cara
    cv2.rectangle(img, (left,  top), (right, bottom), color, 2)
    cv2.rectangle(img, (left, bottom - 20), (right, bottom), color, -1)

    cv2.putText(img,nombre, (left, bottom -6), font, 0.6, (0,0,0), 1)

#En una ventana nueva se muestra el resultado
cv2.imshow('Webcam', img)
```