Detección de Rostros en Tiempo Real

Mariano Rivera-Ramos
mariano.rivera@cimat.mx

Basado en el paper: Rapid Object Detection using a Boosted Cascade of Simple Features, P.Viola (2001)

Detección de rostros

- ¿Qué significa detección en tiempo real?

Detección de rostros

- ¿Qué significa detección en tiempo real?

- ¿Qué significa detección en un video en tiempo real?

Detección de rostros

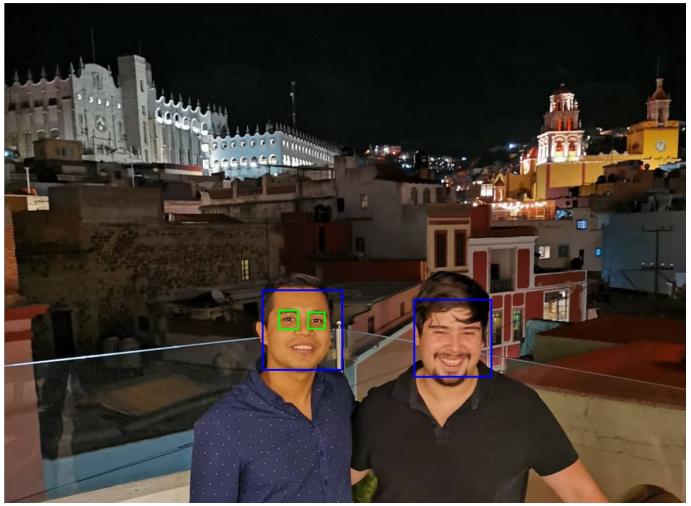
- ¿Qué significa detección en tiempo real?

- ¿Qué significa detección en un video en tiempo real?

- ¿Qué significa detección en imágenes?

Detección de rostros

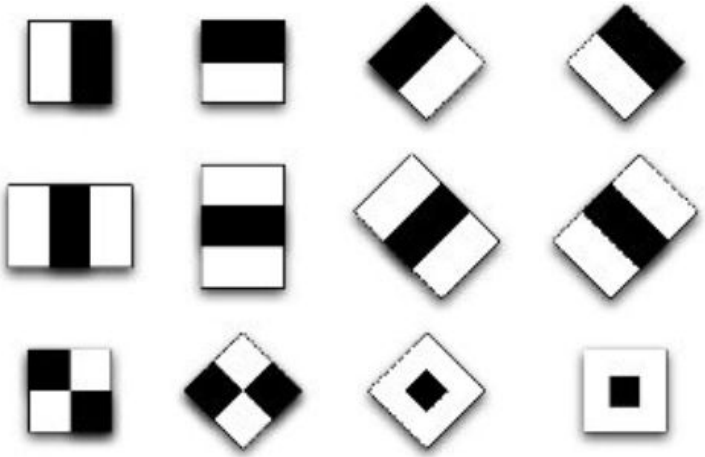
- ¿Qué significa detección en tiempo real?
- ¿Qué significa detección en un video?
- ¿Qué significa detección en imágenes?

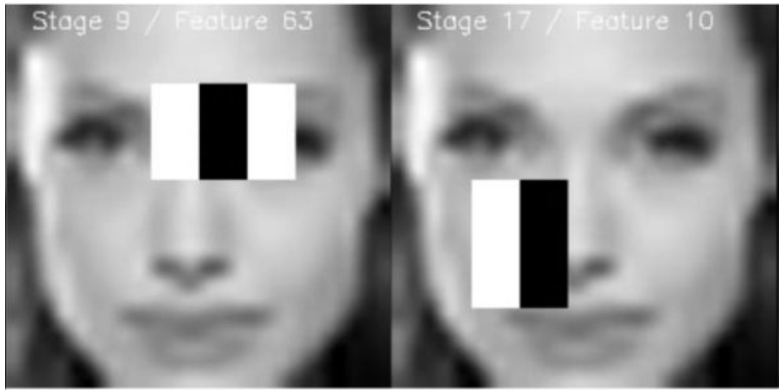


Detección en imágenes

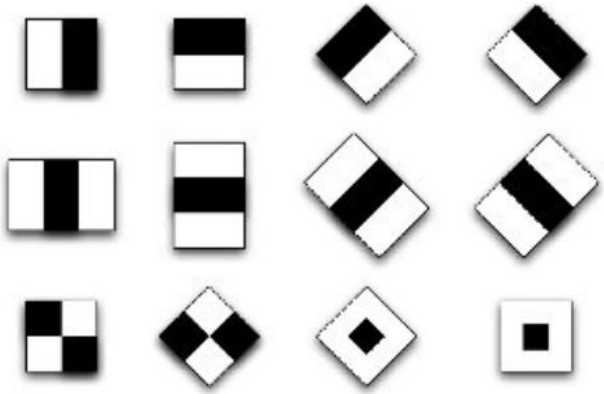
Método a utilizar: Haar-cascade.

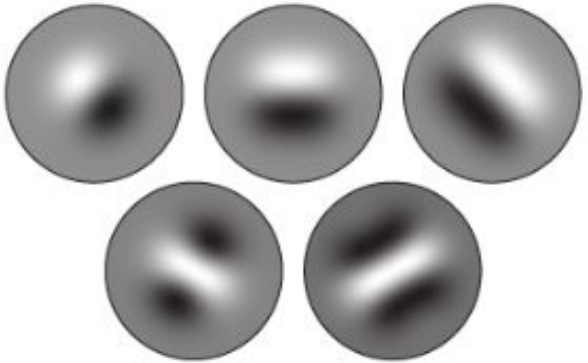
CARACTERÍSTICAS:





Características

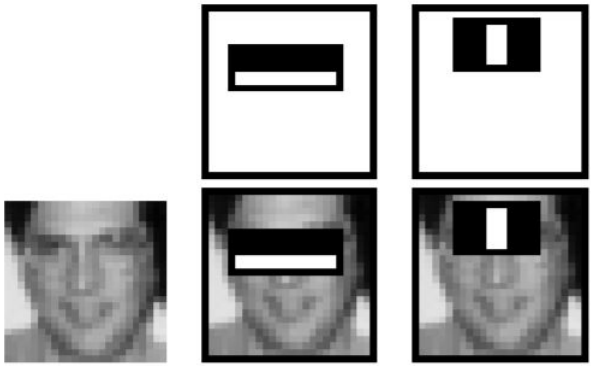




- Fáciles de representar requieren a lo más 4 puntos para representar características rectangulares.
- Fácil computacionalmente o no demandantes computacionalmente.

AdaBoosting

Usando un algoritmo de AdaBoost, seleccionamos las mejores características de clasificación en cada iteración.



Algorithm 1 Proceso Boosting

1: Dados los ejemplos $(x_1, y_1), ...(x_n, y_n)$ donde $y_i = \{0, 1\}$, para ejemplos negativos y positivos de caras respectivamente.
2: Inicializamos los pesos, $w_{1,i} = \frac{1}{2m}, \frac{1}{2i}$ para $y_i = \{0, 1\}$, respectivamente, donde m y l son los números de negativos y positivos respectivamente
3: **for** $t = 0, 1, 2 \ldots, T$ **do**
4: Normalizamos los pesos,

$$\frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

, talque w_t es una función de probabilidad.
5: Para cada característica j, entrenamos un clasificadpr h_j, el cual es restringido a una sola característica. El error evaluado es:

$$\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|.$$

6: Elejimos los clasificadores h_t, con los menores errores ϵ_j
7: Actualizamos los pesos:

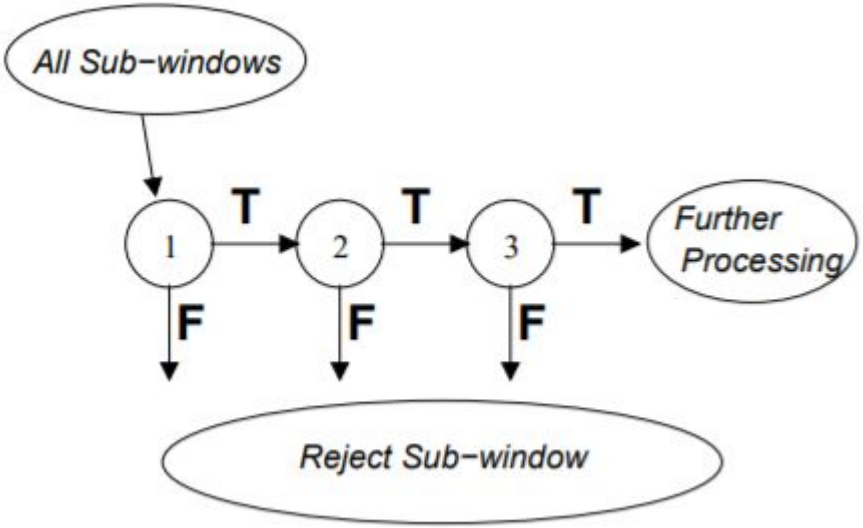
$$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$

Donde $e = 0$, si se clasifico correctamente, $e = 1$ en otro caso y $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
8: **end for**
9: El clasificador final es:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otro caso} \end{cases}$$

Donde $\alpha_t = log\frac{1}{\beta_t}$

Cascada de decisión

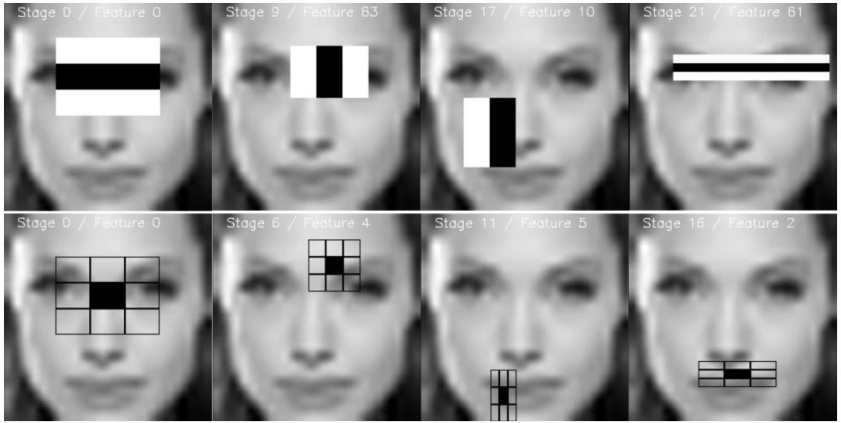


Cada elemento de la cascada es un modelo de clasificación creado mediante AdaBoost.

El modelo final a utilizar es un modelo de 32 etapas con un total de 4297 características.

Ejemplo

Caras de base de datos, consiste en 4196 caras.



Ejemplo de características en diferentes etapas.

Aplicación

Librerías

```
import cv2
import sys
import numpy
```

Bases

```
#Bases a utilizar
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
smile_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +'haarcascade_smile.xml')
```

Imágenes

```
img = cv2.imread('C:/Users/Panda/Desktop/det ros/aaaa3.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

caras= face_cascade.detectMultiScale(
        gray,
        scaleFactor=1.2,
        minNeighbors=5,
        minSize=(20, 20))


for (x,y,w,h) in caras:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
    ojos = eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in ojos:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)


cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Aplicación

Tiempo Real

```
video = cv2.VideoCapture(0)
video.set(3,1280) # Ancho de ventana
video.set(4,920) # Alto de ventana

while True:
    ret, img= video.read()
    gray= cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    #Reconocimento de caras
    caras= face_cascade.detectMultiScale(
            gray,
            scaleFactor=1.2,
            minNeighbors=5,
            minSize=(20, 20))

    for (x,y,w,h) in caras:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]

    #Reconocimiento de ojos
    ojos = eye_cascade.detectMultiScale(
            roi_gray,
            scaleFactor= 1.5,
            minNeighbors=10,
            minSize=(5, 5))

    for (ex, ey, ew, eh) in ojos:
            cv2.rectangle(roi_color, (ex, ey), (ex + ew, ey + eh), (0, 255, 0), 2)

    #Reconocimiento de sonrisas
    sonrisas = smile_cascade.detectMultiScale(
            roi_gray,
            scaleFactor= 1.5,
            minNeighbors=15,
            minSize=(25, 25))

    for (xx, yy, ww, hh) in sonrisas:
            cv2.rectangle(roi_color, (xx, yy), (xx + ww, yy + hh), (0, 255, 0), 2)


    cv2.imshow('video',img)

    #Esc para salir
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

video.release()
cv2.destroyAllWindows()
```

Notas

Para instalar la libreria Open CV , o cv2, correr:

pip install opencv-python