PEC 3

Reconocimiento de las personas por la imagen de la cara y el iris

Pablo Riutort Grande

11 de mayo de 2020

1.

Sea f el conjunto de matrices formado por (f1,..., f10) y n el conjunto de matrices formado por (n1,..., n10)

a)

- **c1**: Los valores de la primera fila son muy similares a los de la matrices f2, f4, f5 y f6. Los valores de la primera columna también tienen muchas similitudes con todas las de las caras excepto con f1. Aunque también existen las mismas similitudes con la matriz n1, dadas las múltiples similitudes con el conjunto f y las pocas con el conjunto n podría tratarse de una cara.
- **c2**: Los valores de esta matriz no son muy similares a ninguna del conjunto f, sin embargo, sí que hay muchas similitudes con las del conjunto n por lo que podría no tratarse de una cara.
- **c3**: Aplicando los mismos criterios de comparación de c1 y dado que estas dos matrices se parecen mucho entre sí, se podría pensar que también se trata de una cara.
- **c4**: Esta matriz tiene muy poca similitud con el conjunto f, sin embargo, tiene muchas similitudes con las matrices de n, en especial con n2, n3, n6, n7 y n10. Por tanto, se puede pensar que no se trata de una cara.
- **c5**: Las 3 primeras filas de esta matriz son idénticas a las de la matriz f4, sin embargo las 3 últimas filas son idénticas a las de la matriz n4. "A ojo" no es posible deducir si es una cara o no.
- **c6**: Las 2 primeras columnas son idénticas a las de la matriz n5 y los demás valores poseen más similitud con el conjunto n que con el f así que podría ser que no fuera una cara.

b) En general, he ido comparando la primera fila y la primera columna de la matriz a clasificar (c) con matrices del conjunto más pequeño (f) para encontrar similitudes. He considerado que había suficiente similitud si cada uno de los valores comparados no distaba demasiado de su homólogo en la matriz comparada, es decir, si la diferencia entre ambos valores c_{ij} y f_{ij} era baja.
Si no veía mucha similitud entre la matriz c y ninguna del conjunto f, entonces, he pasado a buscar similitudes de la misma forma en matrices del conjunto más grande (n).

c) Los siguientes valores se han calculado mediante el programa adjunto en el anexo Filtros de Haar[A]

	C^1	C^2	C^3	C^4	C^5	C^6
P_1^c	0.958	3.719x10e^{-12}	0.957	2.881x10e^{-13}	1.759x10^{-6}	0.916
P_2^c	0.867	0.120	0.845	0.126	0.827	0.610
P_3^c	0.840	0.137	0.830	0.147	0.545	0.053
P_4^c	0.958	8.266x10e^{-6}	0.953	4.807x10e^{-5}	0.960	0.371
$\prod_{i=1}^{4} P_i^c$	0.669	5.076x10e^{-19}	0.641	2.582x10e^{-19}	7.638x10e^{-7}	0.011
P_1^{nc}	0.041	0.999	0.042	0.999	0.999	0.083
P_2^{nc}	0.132	0.879	0.154	0.873	0.172	0.389
P_3^{nc}	0.159	0.862	0.169	0.852	0.454	0.946
P_4^{nc}	0.041	0.999	0.046	0.999	0.039	0.628
$\prod_{i=1}^{4} P_i^{nc}$	3.609x10e^{-5}	0.758	5.085x10e^{-5}	0.744	0.003	0.019
Clase	Cara	No cara	Cara	No cara	No cara	No cara

d) Resultados de aplicar únicamente Haar$_1$[B] en el conjunto de imágenes:

	C^1	C^2	C^3	C^4	C^5	C^6
P_1^c	0.958	3.719x10e^{-12}	0.957	2.881x10e^{-13}	1.759x10^{-6}	0.916
$\prod_{i=1}^{4} P_i^c$	0.958	3.719x10e^{-12}	0.957	2.881x10e^{-13}	1.759x10^{-6}	0.916
P_1^{nc}	0.041	0.999	0.042	0.999	0.999	0.083
$\prod_{i=1}^{4} P_i^{nc}$	0.041	0.999	0.042	0.999	0.999	0.083
Clase	Cara	No cara	Cara	No cara	No cara	Cara

Al no haber más probabilidades que la correspondiente a Haar_1 no existe un conjunto de probabilidades que multiplicar y por eso $P_1^c = \prod_{i=1}^{4} P_i^c$ y $P_1^{nc} = \prod_{i=1}^{4} P_i^{nc}$.
Cabe destacar la discrepancia obtenida en la imagen C^6 donde en el ejercicio anterior fue catalogada como "No cara" y esta vez el resultado ha variado a "Cara" y, por tanto, no se obtiene la misma clasificación de las seis imágenes.

2.

a) Para este ejercicio se ha utilizado el programa de edición de imágenes Gimp y sus distintas herramientas de selección. Gimp considera el punto origen (0, 0) en el extremo superior izquierdo de la imagen.

- Estimación del centro pupilar: **(123, 126)**
  Para esta estimación se ha creado un cuadrado que contenga a la pupila y se han dibujado sus diagonales. La intersección de estas nos dará el punto que corresponde al centro del cuadrado que coincidirá con el de la pupila.

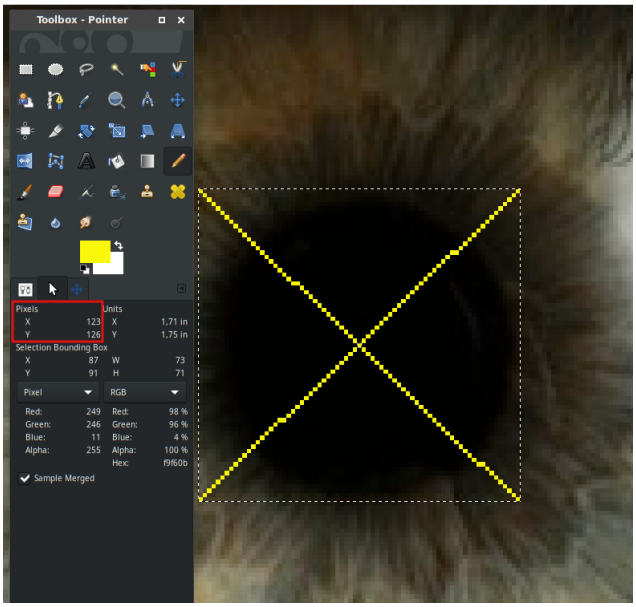

Figura 1: Intersección de las diagonales del cuadrado que contiene la pupila nos da el centro pupilar

- Estimación del radio de la pupila: **36px**.
  Esta estimación se ha hecho con la herramienta de Medida de distancias de Gimp y además la coordenada y del otro extremo del segmento es 162: $162 - 126 = 36px$.

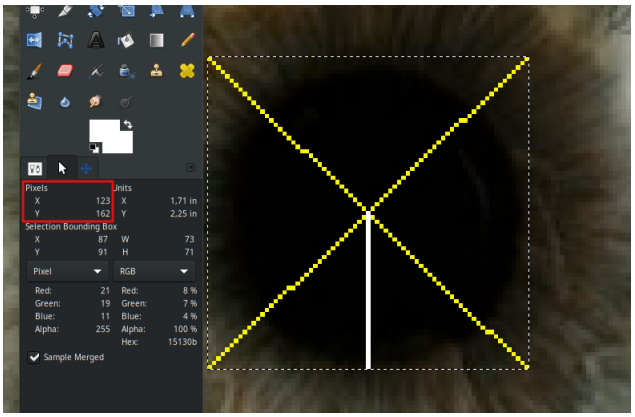

Figura 2: Segmento (blanco) recto desde el centro de la pupila hasta un lado del cuadrado

- Estimación del radio límbico: **134px**
  De igual forma que hemos calculado el radio de la pupila podemos calcular el radio límbico, solo hay que extender la selección rectangular de un lado hasta el límite del iris con la esclerótida.

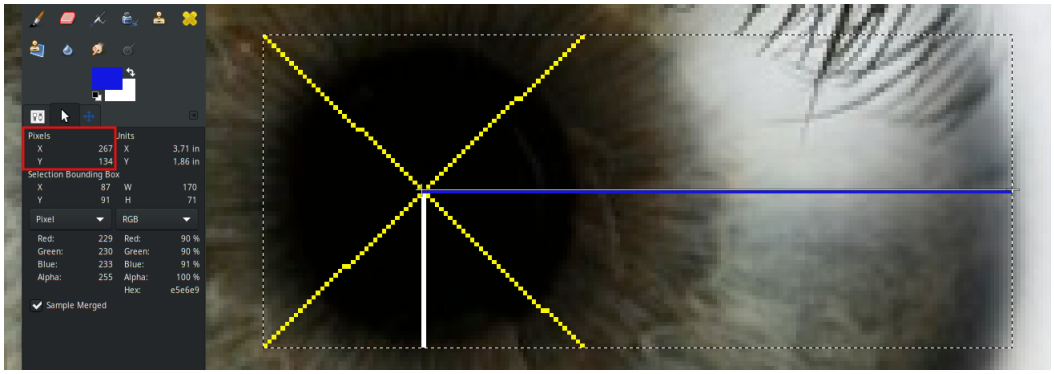

Figura 3: Segmento (azul) recto desde el centro de la pupila hasta el límite del iris

b) Considerando las siguientes variables:

$$x = 150$$
$$y = 200$$
$$x_0 = 123$$
$$y_0 = 126$$
$$r_p = 36px$$
$$r_l = 134px$$

El ángulo se calcula como:

$$\alpha = \arctan 2(x - x_0, y - y_0)$$
$$\alpha = \arctan(\tfrac{y-y_0}{x-x_0})$$
$$\alpha = \arctan(\tfrac{74}{27})$$
$$\alpha = 1{,}22rad$$

Para calcular la posición de la coordenada dada en la imagen desenrollada según el formato de Daugman, cada píxel en lugar de definirlo con el sistema de coordenadas cartesiano (x, y) se define en un sistema de coordenadas polar (r, α). Donde r representa el radio dentro del iris y α es el ángulo.

El radio r se define como:

$$r = \sqrt{(x - x_0)^2 + (y - y_0)^2}$$
$$r = \sqrt{(27)^2 + (74)^2}$$
$$r = 78{,}77px$$

Daugman usa los siguientes valores para la imagen normalizada: $M = 256$ y $N = 8$.
La coordenada x' en el nuevo sistema de coordenadas polares se calcula como:

$$x' = \frac{(M-1)(\alpha+\pi)}{2\pi}$$
$$x' = \frac{255(1{,}22+\pi)}{2\pi}$$
$$x' = 156$$

La coordenada y' se calcula como:

$$y' = \frac{(N-1)(r-r_p)}{r_l-r_p}$$
$$y' = \frac{7(78{,}77-36)}{134-36}$$
$$y' = 3$$

Posición donde irá a parar este punto en la imagen desenrollada según el formato de Daugman:

$$V(x', y') = (156, 3)$$

A. Características de Haar

Este programa hace una extracción de las características de Haar de los conjuntos de imágenes proporcionados (f y n) y hace el análisis estadístico de las mismas: media y desviación típica.
Posteriormente, analiza el conjunto de imágenes a determinar si son cara o no (c), aplica las características de Haar de las mismas y determina la probabilidad de que sea una cara o no a través de las funciones probabilísticas.

El algoritmo es el siguiente:

1. Para los dos conjuntos de imágenes proporcionados:
   I. Extraer la matriz de la imagen a tratar y pasarla una lista de listas de enteros.
   II. Extraer vector a_j^i: Aplicar los cuatro filtros de Harr sobre la lista y construir una matriz donde las filas correspondes a la característica de Haar (j) y las columnas el número de la imágen al que se le ha aplicado en el conjunto (i).
   III. Calcular la media y la desviación típica de cada vector y guardarlo en una lista indexable por número de característica de Haar.
2. Para cada imagen a clasificar del conjunto c:
   I. Extraer la matriz de la imagen a tratar y pasarla una lista de listas de enteros.
   II. Construir vector cnn_j^i: Aplicar los cuatro filtros de Harr sobre la lista y construir una matriz donde las filas correspondes a la característica de Haar (j) y las columnas el número de la imagen al que se le ha aplicado en el conjunto (i).
   III. Para cada valor j del vector cnn_j^i:
      - Para cada valor i del vector cnn_j^i:
        I. Obtener la media y desviación típica correspondiente a la característica $Haar_j$.
        II. Obtener la probabilidad a priori de que sea cara $P(x_i|\text{“}cara''\text{)}$ y de que sea no cara $P(x_i|\text{“}nocara''\text{)}$ a partir de la media y la desviación típica proporcionada.
        III. Obtener la probabilidad de que sea cara P_c y de que sea no cara P_{nc}.
        IV. Si estamos iterando sobre la última carácterística de Haar, obtenemos el producto de las probabilidades de cara y no cara y determinamos la clase a la que pertence el la matriz analizada. Si $\prod_{i=1}^{n} P_i^c(x_i) > \prod_{i=1}^{n} P_i^{nc}(x_i)$ entonces es de la clase “Cara”, en caso contrario se trata de la clase “No cara”.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*
from numpy import mean, std, sqrt, prod, pi, e
from pprint import pprint


def get_matrix(filename):
    matrix = []
    with open(filename, "r") as f:
        for line in f.readlines():
            row = line.strip().split("\t")
            matrix.append([int(r) for r in row])
    return matrix


class Haar:
    window = []

    def __init__(self, matrix, x, y, h, w):
        self.window = [row[x:w] for row in matrix[y:h]]

    def filter_1(self):
        split = int(len(self.window) / 2)
        white, black = 0, 0

        for row in self.window[:split]:
            white += sum(row)
```

```
        for row in self.window[split:]:
            black += sum(row)

        return white - black

    def filter_2(self):
        split = int(len(self.window) / 2)
        white, black = 0, 0

        for row in self.window:
            white += sum(row[:split])
            black += sum(row[split:])

        return white - black

    def filter_3(self):
        split = int(len(self.window) / 3)
        white, black = 0, 0

        for row in self.window:
            white += sum(row[:split])
            remain = row[split:]
            black += sum(remain[:split])
            white += sum(remain[split:])

        return white - black

    def filter_4(self):
        split = int(len(self.window) / 2)
        white, black = 0, 0

        for row in self.window[:split]:
            black += sum(row[:split])
            white += sum(row[split:])

        for row in self.window[split:]:
            white += sum(row[:split])
            black += sum(row[split:])

        return white - black


def get_haar_values(obj_list, window):
    haar_values = [[], [], [], []]
    for f in obj_list:
        matrix = get_matrix(path + f + ".txt")
        haar = Haar(matrix, **window)
        haar_values[0].append(haar.filter_1())
        haar_values[1].append(haar.filter_2())
        haar_values[2].append(haar.filter_3())
        haar_values[3].append(haar.filter_4())

    return haar_values


def a_priori(mean, std, x):
    exp = (-((x - mean) ** 2)) / (2 * std ** 2)
    return (1 / (std * sqrt(2 * pi))) * e ** exp


def p_face(priori_face, priori_no_face):
    return priori_face / (priori_face + priori_no_face)


def p_no_face(priori_face, priori_no_face):
    return priori_no_face / (priori_face + priori_no_face)


if __name__ == "__main__":

    path = "PEC3_ficheros/"

    window = {"x": 0, "y": 0, "w": 6, "h": 6}
```

```
    faces = ["f" + str(i) for i in range(1, 11)]
    no_faces = ["n" + str(i) for i in range(1, 11)]

    vector_faces = get_haar_values(faces, window)
    faces_mean_stds = [(mean(h), std(h)) for h in vector_faces]
    vector_no_faces = get_haar_values(no_faces, window)
    no_faces_mean_stds = [(mean(h), std(h)) for h in vector_no_faces]

    cnn = ["c" + str(i) for i in range(1, 7)]
    vector_cnn = get_haar_values(cnn, window)
    results = [{} for _ in range(len(cnn))]

    for haar_index, value in enumerate(vector_cnn):
        for image_index, x_i in enumerate(value):
            faces_mean, faces_std = faces_mean_stds[haar_index]
            priori_face = a_priori(faces_mean, faces_std, x_i)
            no_faces_mean, no_faces_std = no_faces_mean_stds[haar_index]
            priori_no_face = a_priori(no_faces_mean, no_faces_std, x_i)

            img = results[image_index]
            img[f"Pc{haar_index+1}"] = p_face(priori_face, priori_no_face)
            img[f"Pnc{haar_index+1}"] = p_no_face(priori_face, priori_no_face)

            # if we are about to check next image, calculate probability product
            if haar_index == 3:
                img["Prod(Pc)"] = prod(
                    [v for k, v in img.items() if k.startswith("Pc")]
                )
                img["Prod(Pnc)"] = prod(
                    [v for k, v in img.items() if k.startswith("Pnc")]
                )

                img["class"] = (
                    "Face" if img["Prod(Pc)"] > img["Prod(Pnc)"] else "No face"
                )

    pprint(results)
```

B. Característica de Haar$_1$

Versión modificada del programa Filtros de Haar[A] para devolver únicamente resultados de características de Haar$_1$.

```

def get_haar_values(obj_list, window):
    """Modified function in order to return only Haar1"""
    haar_values = [[]]
    for f in obj_list:
        matrix = get_matrix(path + f + ".txt")
        haar = Haar(matrix, **window)
        haar_values[0].append(haar.filter_1())
```

```
            img = results[image_index]
            img[f"Pc{haar_index+1}"] = p_face(priori_face, priori_no_face)
            img[f"Pnc{haar_index+1}"] = p_no_face(priori_face, priori_no_face)

            img["Prod(Pc)"] = prod([v for k, v in img.items() if k.startswith("Pc")])
            img["Prod(Pnc)"] = prod([v for k, v in img.items() if k.startswith("Pnc")])

            img["class"] = "Face" if img["Prod(Pc)"] > img["Prod(Pnc)"] else "No face"
```