Reconocimiento Biométrico Aplicando SVD

Diego Fabian Ledesma Motta
Dilan Stiven Polanco Valencia
Nataly Andrea Portillo Velasco

I. INTRODUCCIÓN

A partir del siguiente proyecto se busca desarrollar un sistema de identificación y verificación de la existencia de huellas dactilares en un conjunto de imágenes conocidas, haciendo uso de la descomposición de valores singulares (SVD) y una vez finalizado el proceso de verificación planteado determinar si dicha huella de entrada tiene acceso o no.

II. ESTRATEGIA DE SOLUCIÓN

De manera general el reconocimiento de huellas se encuentra en el escenario de identificación o reconocimiento. La identificación es una búsqueda de cierta huella en particular dentro de un conjunto grande de imágenes, dónde la biometría de una huella desconocida es comparada con todas las biometrías dentro de la matriz de entrenamiento, de cada una de estas comparaciones se toma aquella biometría más cercana y así se asigna dicha identidad. Si las comparaciones no alcanzan un límite establecido, el sistema puede decir que la huella desconocida no se reconoce dentro de dicho directorio.

Para el desarrollo del algoritmo de este sistema de reconocimiento biométrico a través de huellas dactilares se hizo uso del lenguaje de programación Python y se han realizado respectivas pruebas usando un directorio local de imágenes que van a representar el conjunto de entrenamiento; este directorio se encuentra dividido en sub carpetas que albergan la cantidad de 5 images de huellas dactilares pertenecientes a una misma persona, además de esta carpeta se extrajeron un conjunto de imágenes que representaría el grupo con el cual serán realizadas las pruebas para determinar si existe o no, en la base de datos. Dentro de ese conjunto de pruebas se tienen imágenes que están dentro de la matriz de entrenamiento y otras que no fueron incluidas en la misma, para ser sometidas a un debido proceso de verificación.

El conjunto de huellas dactilares utilizadas para el proyecto consta de 250 imágenes en escala de grises, con un tamaño de 200x200 píxeles y en un formato jpg, este total de huellas corresponden a 50 personas distintas.

La estrategia de solución dada para este problema de reconocimiento biométrico, inicia con la lectura de las rutas de alojamiento de las imágenes dentro del directorio local '***huellas'***, mediante las siguientes líneas de código:

```
#Obtener dirección de las carpetas de las imágenes de entrenamiento
path = []
for i in os.listdir("huellas"):
        path.append(os.getcwd()+"/huellas/"+i)
```

Posteriormente se procede a determinar la cantidad total de imágenes que se encuentran almacenadas en la base de datos, obteniendo como resultado un total de 245 imágenes, ya que se decidió extraer una de las sub carpetas que tenía la base de datos, este cálculo se realizó de la siguiente manera:

```
#Cantidad total de imágenes de entrenamiento
number = 0
for i in path:
    number += len(glob.glob(i+"/*.jpg"))
```

Una vez obtenido el total de las imagenes, lo siguiente fue la creación de la matriz de entrenamiento cuyas dimensiones estarán establecidas por el número total de imágenes y la resolución de las mismas, para ello fue necesario calcular la resolución de las imágenes, teniendo en cuenta que cada imagen es de 200x200 píxeles la resolución sería de 40.000. Por ende la matriz de

entrenamiento tiene unas dimensiones de 40000x245. Esta sección fue implementada de la siguiente manera:

```
#Resolución de las imágenes
m=200
n=200
resolution= m*n

#Definición matriz de entrenamiento
#La función shape=(filas,columnas)
MTraining = npy.empty(shape=(resolution,
number), dtype='float64')
```

Definida la matriz de entrenamiento, se procede a almacenar en ella las 245 imagenes que conformaran el conjunto de entrenamiento, para hacer esto, inicialmente se leen las imagenes en formato blanco y negro, obteniendo como resultado una matriz que contiene la información de cada imagen, una vez obtenida esta matriz lo siguiente es convertirlo en un arreglo de tipo flotante y adicionalmente linealizarlo. Realizado estos pasos se procede a almacenar cada imagen en la matriz de entrenamiento en forma de columna. La implementación de este procedimiento se realizó de la siguiente manera:

```
#Bucle de lectura de imágenes de entrenamiento
column = 0
for dir in path:
   for filename in sorted(os.listdir(dir)):
      pathname = os.path.join(dir,filename)
      #Leer imagen en BN
      img = cv2.imread(pathname,0)
#Convertir en array de tipo flotante y linealizarlo
            img_array =
npy.array(img,dtype='float64').flatten()
#Escribir los arreglos en columnas de la matriz de
entrenamiento
            MTraining[:,column] = img_array[:]
      column += 1
```

A la matriz de entrenamiento se le decide calcular el promedio de cada 5 huellas las cuales representan la identidad de una persona y almacenarlo en columnas en una matriz de promedios, para posteriormente lograr una huella promedio por persona que contendrá los datos o información más significativos de ella. Se hace esto con el fin de determinar si realmente el tratamiento de datos realizado es correcto y al graficar el promedio se logre mostrar una imagen coherente, como la siguiente:

Imagen 1. gráfica del promedio de huellas de una persona.

Cabe aclarar que esta información no será utilizada para determinar la existencia de la huella dentro de la base de datos ya que su precisión no es lo suficientemente acertada, pues según el libro guía '***Numerical linear algebra in data mining***' este método arroja un precisión de un 75%.

Por esta razón y para obtener una tasa de éxito más alta se implementa el procedimiento del cálculo en descomposición en valores singulares (SVD).

SVD:

La descomposición en valores singulares, SVD(por sus siglas en inglés), permite obtener, entre otras cosas, una base para el espacio columna de la matriz de imágenes de huellas de cada persona, lo cual permitirá más adelante un método de mejor precisión que la comparación con la huella promedio, el cual consiste en buscar la combinación lineal de las bases más próxima a la imagen que se esté probando. Con este objetivo, se usó el siguiente código para almacenar, en dos matrices y un arreglo, los resultados del SVD de cada persona por separado.

```
for i in range(49):
```

```
U[:,i*5:(i+1)*5], S[:,i],
V[:,i*5:(i+1)*5] =
npy.linalg.svd(MTraining[:, i*5: (i+1)*5],
full_matrices = False)
```

En donde U almacena la matriz ortogonal de las bases para el espacio columna de la matriz de huellas de cada persona.

Al ser las dimensiones de cada matriz de huellas de cada persona, de (40000,5), con sus cinco columnas independientes, la matriz U tendrá para cada persona el mismo tamaño, por lo que la matriz U que almacena las bases del espacio columna de las 49 personas tiene un tamaño de (40000,245).

En este punto, es interesante el hecho de que cada vector de una base tiene el mismo tamaño que la imagen de una huella en la matriz de entrenamiento, y podemos, por tanto, graficarla:

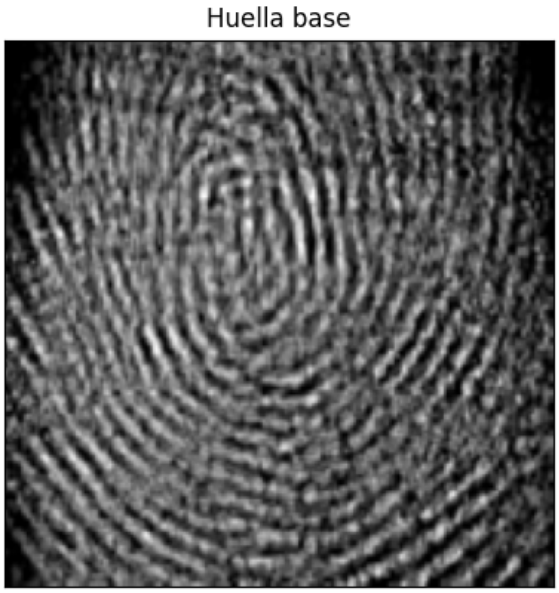


Imagen 2. Gráfica de la primera huella singular del SVD de una persona.

Tener estas bases del espacio columna permite poder usarlas para determinar si una huella pertenece o no al conjunto de huellas permitidas. Esto se logra con el algoritmo de clasificación de bases del SVD, explicado en el libro guía con el caso de los dígitos, que traídos al contexto de las huellas digitales, parte de los siguientes supuestos:

1. Cada huella está bien caracterizada por las primeras pocas huellas singulares de su propio tipo(en este caso, de la misma persona).
2. Una expansión en términos de las primeras pocas huellas singulares discrimina bien entre las diferentes personas.
3. Si una huella desconocida puede ser mejor aproximada en una base particular de imágenes singulares, que en las otras, entonces es probable que la huella pertenezca a la persona a quien corresponde dicha base.

Pero ¿Cómo se aproxima una huella desconocida en una base particular?

Problema de mínimos cuadrados:
Al hablar de aproximar una huella en una base particular, se hace referencia a tomar un determinado número de componentes de la base(columnas), y buscar la combinación lineal de los mismos que más se parece a la huella desconocida, es decir, se busca la combinación lineal de las columnas de la base que genera el vector más cercano al vector que representa la huella desconocida.

El primer paso es establecer un criterio para medir qué tan cerca está un vector de otro, para esto, se toma la distancia euclidiana entre los mismos, que es el equivalente a calcular la norma euclidiana de la resta de ambos vectores;

También hay que definir cuántas columnas se usarán de cada base para esta búsqueda. Por medio de pruebas se encontró que usar las cinco huellas base es significativamente mejor que usar sólamente tres, y no genera un tiempo de cómputo desproporcionado, por lo que se decidió usar todas las columnas(5) de cada base.

A continuación hay que definir una función que calcule la distancia de una determinada combinación lineal a la huella desconocida, a partir de un arreglo que contiene los coeficientes que multiplican a las columnas de la base:

```
def optimalRepresentation(startingX):
    x0 = startingX[0]
    x1 = startingX[1]
    x2 = startingX[2]
```

```
    x3 = startingX[3]
    x4 = startingX[4]
 dist =
npy.linalg.norm(testImageVec-(x0*basis[:,0]
+ x1*basis[:,1] + x2*basis[:,2] + x3*basis[:,3]
                    + x4*basis[:,4]))
    return dist
```

Finalmente, hay que minimizar la función optimalRepresentation, a partir de un vector aleatorio inicial, para lo cual, se usa el método de mínimos cuadrados:

```
result = minimize(optimalRepresentation,
startingX, method = 'SLSQP')
```

En el código anterior, se usa la herramienta minimize, perteneciente a scipy (scipy.optimize.minimize), especificando como método de solución el parámetro 'SLSQP'(Sequential Least Squares Programming), el cual resuelve el problema de qué tan bien puede ser una huella desconocida, representada en una determinada base.

Al hacer esto con la base de cada persona, podemos acudir al tercer supuesto del método para determinar si la huella desconocida corresponde a alguno de los conjuntos de huellas presentes en la matriz de entrenamiento.

De modo que si tras encontrar la representación óptima de la huella desconocida en todas las bases que se poseen, existe una base para la cual la representación es mucho más acertada que las demás, se puede asumir que la huella pertenece a la persona a la cual corresponde esa base, o determinar que la huella no pertenece a ninguna de las personas de las cuales se poseen huellas en la matriz de entrenamiento.

Finalmente hay que especificar un criterio para poder denominar una representación como "mucho más acertada" que las demás. A partir de varias pruebas, se observó que cuando una huella no pertenece a la persona a la cual corresponde una determinada base, la mejor representación alcanza una cercanía de varias decenas de miles, en muchos casos, entre 18 mil y 50 mil, mientras que si la huella pertenece a la persona a la cual corresponde la base, suelen obtenerse distancias del orden de $1x10^{-5}$. Confirmando el primer supuesto del método empleado. Por lo tanto, establecer como criterio que la distancia entre la mejor representación de una huella y la huella misma sea menor a 1, es suficiente para garantizar una buena discriminación:

```
acceso = False
for i in range(49):
    basis = U[:,(i*5):(i+1)*5]
    result = minimize(optimalRepresentation,
            startingX, method = 'SLSQP')
    if (result.fun < 1):
        acceso = True
```

Imponer otro criterio del estilo de especificar que esa distancia sea menor que 1 y difiera en más de 5 mil de la siguiente representación más cercana, resulta innecesario, ya que la segunda condición se cumple en todas las pruebas realizadas.

III. PROCESO DE EVALUACIÓN Y RESULTADOS

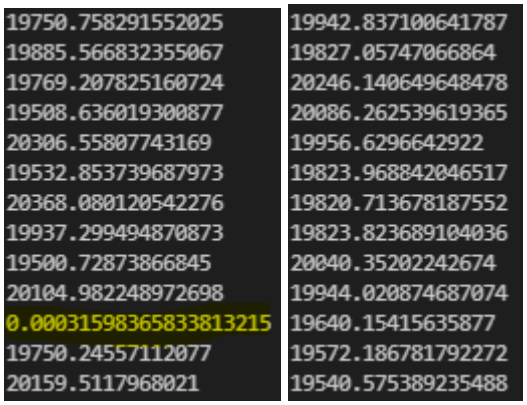
Para realizar el proceso de evaluación y resultados lo que se hizo fue realizar una gran variedad de pruebas con imágenes existentes en la matriz de entrenamiento, y algunas otras que no fueron usadas para crear la misma.

Una de las pruebas realizadas se hizo con la imagen **11.jpg** la cual no fue incluida dentro de la matriz de entrenamiento y una vez terminado todo el proceso planteado en el código se obtienen los siguientes resultados luego de calculada la distancia y su minimización.

17914.91477374386	18255.435226469686
18512.644314526315	18383.829779829754
18148.90762602231	17915.56460242581
17854.08984616668	19138.224518052826
18559.581375414833	18230.905170274316
17756.30645531055	18085.554008849474
18830.77733404883	17994.256356008373
18043.571834315946	17387.414241145147
18145.171434270655	17870.40291561511
18735.63301351945	18035.96345446994
18215.290605041544	17648.67293091301
18102.235925064004	17585.473056876155
18255.435226469686	17863.932860105364

Como se puede observar los resultados dan unos valores bastante grandes, teniendo en cuenta que para la realización del cálculo se tomaron 5 bases de la imagen.

Por otra parte se realizó la siguiente prueba con la imagen **22.jpg** la cual si se encontraba incluida dentro de la matriz de entrenamiento y los resultados que se obtuvieron fueron los siguientes.



En este caso podemos observar que en el momento que la imagen es encontrada, el valor de la distancia y minimización que retorna el algoritmo es un valor inferior a 1, demostrando que su precisión es bastante alta.

Teniendo en cuenta los resultados obtenidos, se decide que la condición de parada, y el criterio que nos define si la huella se encuentra o no en la base de datos será que al calcular la distancia y minimización, si esta nos retorna un valor menor que 1, esto significa que la imagen si ha sido encontrada.

IV. CONCLUSIONES

Finalmente, luego de realizado todo el procedimiento para determinar la solución del problema planteado inicialmente, podemos concluir que, por una parte fue un reto bastante difícil de solucionar que nos ayudó a interiorizar un poco más el tema de descomposición de valores singulares y adicionalmente nos sirvió para buena investigación e implementación de código que funcionara de la manera más eficiente posible desde nuestra percepción.

Por otro lado se pudo observar que el método implementado a partir del SVD tiene una tasa de éxito bastante alta, debido a que las pruebas realizadas retornaban resultados satisfactorios. Pensamos que estos resultados fueron tan eficientes ya que se decidió tomar las 5 bases de la imagen para realizar el cálculo de la distancia, lo cual genera una tasa de éxito bastante alta.

Generalizando podemos decir que el programa cumple con lo planteado inicialmente, determinando a partir de una huella de entrada si esta tiene acceso o no a un área o servicio.

V. BIBLIOGRAFÍA

- *Numerical linear algebra in data mining*
- *https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html*