DESARROLLO DE UNA APLICACIÓN PARA RECONOCIMIENTO FACIAL USANDO EL ALGORITMO FISHERFACES

Cristian Ortiz Celi
Escuela de Electrónica y Telecomunicaciones
Universidad Técnica Particular de Loja
Loja, Ecuador
ceortiz2@utpl.edu.ec

Carlos Saca Japa
Sistemas Informáticos y Computación
Universidad Técnica Particular de Loja
Loja, Ecuador
cfsaca@utpl.edu.ec

***Abstract*— This electronic document contains the necessary information to generate a code that, when executed, returns the information of who owns the face that is in front of the web camera; This is achieved first by generating a database with faces ta recognized the same that are compared at the time of running the program and tells us by a text tag who is the person in front of the camera. The code is based on already developed algorithms called eigenfaces and fisherfaces, these mathematical algorithms are already available within the libraries of Open CV, the same ones that form a fundamental part of this code of face recognition. The document consists of a theoretical framework that contains information about the Open CV libraries, as well as the algorithms developed for facial recognition from the mathematical point of view and explained in detail. Subsequently, the developed code is presented and the fundamental parts that make recognition possible are explained. Following with the document, we will find evidence of the tests carried out with their successes and errors to determine how efficiently this project is implemented. At the end we present the results of the tests and the conclusions that were generated.**

Keywords*— *eigenfaces; fisherfaces; reconocimiento facial; opencv.

I. INTRODUCCIÓN

La visión por computadora tiene como objetivo principal simular los procesos de percepción visual de los seres humanos, es por esto que se genera la necesidad de crear sistemas capaces de comportarse como un cerebro humano para desarrollar tareas que nos pueden facilitar la vida, brindarnos seguridad, generar desarrollo social, en fin evolucionar.

Una necesidad que ha surgido en los últimos años ha sido el reconocimiento facial y por lo tanto se han presentado varias soluciones a este problema. Basándose en los precedentes que existen se va a desarrollar una nueva solución, que ayudado de las librerías de acceso libre Open CV se generará un programa capaz de reconocer el rostro de las personas que estén registradas en una base de datos que se crea en primera instancia y al momento de capturar la imagen de un rostro mediante cámara web, obtendremos luego de realizar una relación entre los datos guardados y lo que está capturando una respuesta de reconocimiento que se presenta al usuario etiquetando su rostro con su nombre.

Para el desarrollo de este sistema, en primera instancia debemos considerar algunos factores que caracterizan un rostro como sus ojos, nariz, boca o en su lugar otras características como estilo de cabello, color del mismo, etc. Todo esto para lograr que el programa cumpla su objetivo. Los algoritmos basados en las características geométricas son probablemente la aproximación más intuitiva al reconocimiento facial, aunque luego de realizar varios experimentos resulta que no es el más efectivo porque puede que estas características geométricas no tengan la suficiente información para generar una respuesta confiable.

Es por esta razón que se analizan dos algoritmos más, los mismos que se denominan Eigenfaces y Fisherfaces, posterior en la sección de marco teórico se detalla sus fundamentos matemáticos y su funcionamiento. [1]

II. MARCO TEÓRICO

A. OpenCV

A continuación se realiza una breve descripción acerca de este recurso. Es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Open CV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estérea y visión robótica. El proyecto pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multinúcleo. [1]

B. Métodos para reconocimiento facial

2.1 Eigenfaces [1] [2] [3] [4]

Ahora vamos a referirnos a uno de los algoritmos desarrollados para el reconocimiento facial. El problema con la representación de imagen es su alta dimensión. Las imágenes en escala de grises bidimensionales pxq ocupan un espacio vectorial $m = pq$ -dimensional, por lo que una imagen con 100 x 100 píxeles se encuentra en un espacio de imagen de 10.000 dimensiones. La pregunta es: ¿Todas las dimensiones son igualmente útiles? Sólo podemos tomar una decisión si hay alguna variación en los datos, entonces lo que se busca son los componentes que representan la mayor parte de la información. Karl Pearson (1901) y Harold Hotelling (1933) propusieron independientemente el Análisis de Componentes Principales (PCA) [5] para convertir un conjunto de variables posiblemente correlacionadas en un conjunto más pequeño de variables no correlacionadas. La idea es que un conjunto de datos de alta dimensión es a menudo descrito por variables correlacionadas y por lo tanto sólo unas cuantas dimensiones significativas representan la mayor parte de la información. El método PCA encuentra las direcciones con mayor varianza en los datos, llamados componentes principales.

- Descripción del algoritmo

Si tenemos $X = \{x_1, x_2, \ldots., x_n\}$ que es un vector aleatorio con observaciones $x_i \in R^d$

1. Obtenemos la media μ

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i \quad (1)$$

Donde:

μ: Media

n: Número total de componentes de vector

x_i: Componente del vector

2. Calculamos la covarianza de la matriz S

$$S = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T \quad (2)$$

3. Se calculan los valores propios (eigenvalues) λ_i y los vectores propios (eigenvectors) v_i de S

$$Sv_i = \lambda_i v_i, i = 1,2, \ldots, n \quad (3)$$

4. Se ordenan los vectores propios de manera descendente respecto a los valores propios correspondiente al valor propio más alto k

Los principales componentes del vector de observación x están dados por:

$$y = W^T(x - \mu) \quad (4)$$

Donde: $W = (v_1, v_2, \ldots, v_k)$

La recostrucción desde las bases PCA están dadas por:

$$x = Wy + \mu \quad (5)$$

Donde $W = (v_1, v_2, \ldots, v_k)$

El método Eigenfaces realiza el siguiente proceso:

- Proyecta todos los elementos de la base datos dentro del subespacio PCA
- Proyecta la imagen de consulta en el subespacio PCA
- Encuentra el vecino más cercano entre las imágenes de entrenamiento proyectadas y la imagen de consulta proyectada.

Todavía queda un problema por resolver. Imagina que se nos dan 400 imágenes de tamaño 100 x 100 píxeles. El Análisis de Componentes Principales resuelve la matriz de covarianza $S = XX^T$, donde $X = 1000x400$ en nuestro ejemplo. Usted terminaría con una matriz 10000 x 10000, aproximadamente 0,8 GB. Para resolver este problema no es factible, así que tendremos que aplicar un truco. Del álgebra lineal se sabe que una matriz M x N con M> N sólo puede tener N - 1 valores propios distintos de cero. Por lo tanto, es posible tomar la descomposición del valor propio $S = X^T X$ de tamaño N x N en su lugar:

$$X^T X v_i = \lambda_i v_i \quad (6)$$

Y así se obtiene los vectores propios de $S = XX^T$ multiplicando a su izquierda con la matriz de datos:

$$XX^T(Xv_i) = \lambda_i(Xv_i) \quad (7)$$

Los vectores propios resultantes son ortogonales, para obtener vectores propios ortonormales necesitan ser normalizados a la longitud unitaria.

2.2 Fisherfaces [1] [2] [3] [4]

Es el momento de hacer una introducción a este algoritmo que por cierto es el elegido para realizar la aplicación objeto de este proyecto. El Análisis de Componentes Principales (PCA), que es el núcleo del método de Eigenfaces, encuentra una combinación lineal de características que maximiza la varianza total en los datos. Si bien esta es claramente una forma poderosa de representar datos, no considera ninguna clase y por lo tanto, mucha información discriminatoria puede perderse al desechar componentes. Imagine una situación donde la variación en sus datos es generada por una fuente externa, que sea la luz. Los componentes identificados por un PCA no contienen necesariamente ninguna información discriminatoria, por lo que las muestras proyectadas se unen y se hace imposible una clasificación.

El Análisis Discriminante Lineal (LDA) [6] [7] realiza una reducción de dimensionalidad específica de clase y fue inventado por el gran estadístico Sir R. A. Fisher. Él lo utilizó con éxito para clasificar las flores en su paper de 1936. El uso de medidas múltiples en problemas taxonómicos. Con el fin de encontrar la combinación de características que separan mejor entre las clases el análisis discriminante lineal maximiza la

relación de entre clases y dentro de las clases de dispersión, en lugar de maximizar la dispersión global. La idea es simple: las mismas clases se agruparán juntas, mientras que las diferentes clases estarán lo más lejos posible una de la otra en la representación de dimensiones inferiores. Esto también fue reconocido (LDA) [6] [7] por Belhumeur, Hespanha y Kriegman, por lo que aplicaron un Análisis Discriminante para hacer frente al reconocimiento.

- Descripción del algoritmo

Teniendo un vector aleatorio x con muestras de $\mathbf{C}$ clases:

$$X = \{X_1, X_2, \dots, X_c\}$$
$$X_i = \{x_1, x_2, \dots, x_n\}$$

Las matrices de dispersión S_B y S_W se calculan:

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T \quad (8)$$

Donde μ es la media total (1), y μ_i es la media de la clase $i \in \{1, \dots, c\}$, la ecuación se detalla a continuación:

$$\mu_i = \frac{1}{|X_i|} \sum_{x_j \in X_i} x_j \quad (9)$$

El algoritmo clásico de Fisher ahora busca una proyección W, que maximice el criterio de separabilidad de clase:

$$W_{opt} = \arg max_W \frac{|W^T S_B W|}{|W^T S_W W|} \quad (10)$$

Una solución a este problema de optimización viene dado mediante la resolución del problema General de valor propio:

$$S_B v_i = \lambda_i S_w v_i \quad (11)$$
$$S_W^{-1} S_B v_i = \lambda_i v_i \quad (12)$$

Hay un problema por resolver: El rango de S_W es como mucho $(N - c)$, con N muestras y c clases. En los problemas de reconocimiento de patrones el número de muestras N es casi siempre más pequeño que la dimensión de los datos de entrada (el número de píxeles), por lo que la matriz de dispersión S_W se vuelve singular. Esto se resolvió mediante la realización de un análisis de componentes principales en los datos y la proyección de las muestras en el espacio dimensional $(N - c)$. A continuación, se realizó un Análisis Discriminante Lineal sobre los datos reducidos, ya que S_W ya no es singular.

La solución se puede reescribir como:

$$W_{pca} = \arg max_W |W^T S_T W| \quad (13)$$

$$W_{fld} = \arg max_W \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_W W_{pca} W|} \quad (14)$$

La transformación de la matriz W, que proyecta una muestra en el espacio dimensional $(c - 1)$, viene dada por:

$$W = W_{fld}^T W_{pca}^T \quad (15)$$

III. DESARROLLO DE LA APLICACIÓN

Esta aplicación ha sido desarrollada en lenguaje de programación Python 2.7+, usando librerías de Open CV 3.0 y aplicando el algoritmo Fisherfaces [1] que ya se mencionó antes en este documento. Además cabe aclarar que se desarrolló bajo el sistema operativo Ubuntu 16.04.

Cuenta con dos etapas, la primera es la de entrenamiento y la segunda el reconocimiento. Empezaremos refiriéndonos a la parte de adquisición o captura de rostros mediante el código que lo hemos denominado "capture.py"; al momento de ejecutar este programa se inicia la web cam para hacer una captura del rostro que se encuentre en frente de la misma, estos datos se almacenan en un directorio que lo hemos denominado "rostros", y posterior estos datos serán comparados al ejecutar el programa denominado "reconocimiento.py" para entregar una respuesta en pantalla en tiempo real.

Todas las imágenes capturadas son a color y con una resolución de 112x92 pixeles; se recomienda tener una base de datos de al menos 500 fotos por cada candidato a reconocer; entre mayor entrenemos a la aplicación mejor respuesta nos dará. Se debe considerar las variaciones de luz en el ambiente y se recomienda entrenar la aplicación en varios escenarios para lograr adaptabilidad. A continuación, en la fig. 1 podemos apreciar un ejemplo de los rostros capturados y que forman pare de la base de datos.

a)

b)

Fig.1 Rostros capturados para la base de datos, a) Cristian, b) Carlos

En la Fig.2 se observa la ventana que se genera para hacer la captura de los candidatos a ser reconocidos posteriormente por la aplicación.

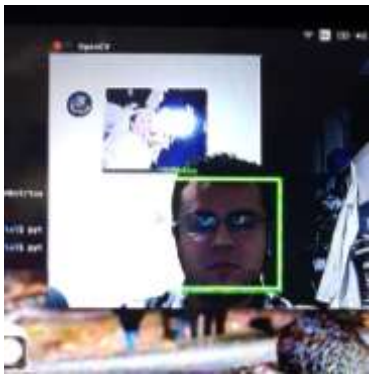

Fig.2 Ventana de captura de rostros

A continuación la Fig.3 y Fig.4 muestran las capturas de los autores de esta aplicación realizando la captura y reconocimiento de sus rostros.

Fig.3 Captura de rostro vista en tercera persona

Fig.4 Captura de rostro en primera persona

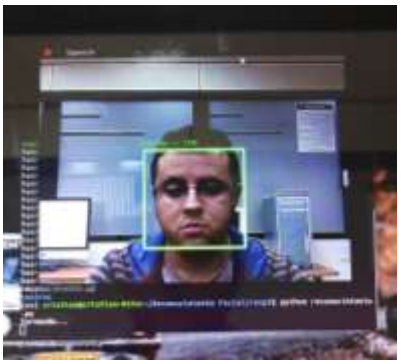

Fig.5 Reconocimiento rostro de uno de los autores

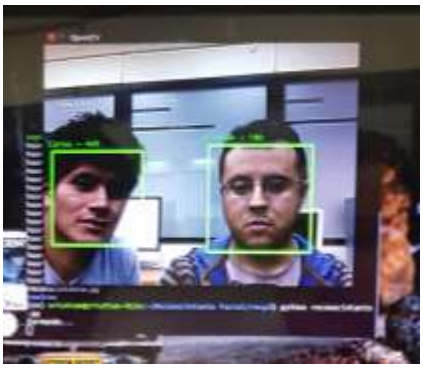

Fig.6 Reconocimiento del rostro de los autores

Fig.7 Reconocimiento y presencia de cara desconocida

En la Fig.7 podemos observar un reconocimiento exitoso de uno de los rostros que se encuentra en la base de datos, además podemos apreciar que se captura un rostro desde una fotografía, como este rostro no consta en la base de datos nos presenta el texto “Desconocido”.

Se destaca que las capturas realizadas para almacenar en la base de datos se hicieron en un salón con buena iluminación, esto es un factor importante pues determinará si la respuesta al realizar el reconocimiento es positiva o negativa.

IV. RESULTADOS

Presentamos en la siguiente tabla (Tabla.1) un resumen de las pruebas realizadas con la aplicación y posterior podremos determinar qué tan preciso es. En la tabla constará la hora del día en que se realizó cada prueba y si el reconocimiento dio un resultado positivo o negativo.

# Prueba	Hora del día	Resultado positivo	Resultado negativo
1	9:00	X	
2	9:30	X	
3	10:00	X	
4	10:30	X	
5	11:00	X	
6	11:30	X	
7	12:00	X	
8	12:30	X	
9	15:00	X	
10	15:30	X	
11	16:00	X	
12	19:00		X
13	19:30		X
14	20:00	X	
15	20:30		X
16	21:00		X
17	21:30		
18	22:00	X	
19	22:30		X
20	23:00	X	

Tabla 1. Registro de pruebas de la aplicación

Si analizamos la tabla anterior rápidamente podemos determinar el porcentaje de efectividad y error de nuestra aplicación, además podemos decir que funciona de mejor manera durante el día comparando con las horas de la noche, el factor de iluminación juega un papel fundamental en esta aplicación pues la mayoría de las capturas para la base de datos se hicieron durante el día, horas de la mañana específicamente y vemos que es justo en estas horas donde el reconocimiento nos da resultados positivos. Volviendo a los porcentajes podemos decir que si las 20 pruebas son el 100 % de efectividad 14 de estas según la Tabla.1 equivale al 70%, es decir de presentar reconocimientos correctos y un 30% de falla o error, que corresponde a las 6 pruebas que dieron negativo al reconocimiento.

V. CONCLUSIONES

- Es evidente que los algoritmos desarrollados para reconocimiento facial han avanzado mucho, pero también podemos notar que se puede mejorar, y no en los fundamentos matemáticos, sino en hacerlos más robustos mediante la creación de aplicaciones como las que se presenta en este proyecto. Se debe considerar las versiones de las librerías pues carecen de adaptabilidad con versiones superiores o inferiores, es por esta razón que al inicio de la sección discusión se especifica las versiones con las que se ha desarrollado esta aplicación.
- Los factores externos siguen siendo un condicionante al momento de procesar imágenes en tiempo real y es que las variaciones de iluminación que se pueden presentar en pequeños lapsos de tiempo son varias, esto genera inconvenientes pues disminuye la garantía de que, en este caso el reconocimiento se ejecute correctamente en todos los escenarios, esto ya se evidenció en la tabla comparativa de pruebas realizadas.
- Para realizar el entrenamiento ejecutamos el código "capture.py", y se aconseja establecer dentro del mismo que se capturen al menos 500 fotos por cada candidato que se almacenará en la base de datos de la aplicación. Y dentro del código "reconocimiento.py", establecer una predicción menor o igual a 200 y mayor que 100.

VI. BIBLIOGRAFÍA

[1] OpenCV, "Face Recognition with OpenCV-Open CV 2.4.13,2 documentation" «www.opencv.org,» 2014. [En línea]. Available: http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html. [Último acceso: 10 Enero 2017].

[2] VII. S. Emami, "Introduction to Face Detection and Face Recognition" «www.shervinemami.info,» 4 Febrero 2012. [En línea]. Available: http://www.shervinemami.info/faceRecognition.html. [Último acceso: 18 Diciembre 2016].

[3] P. Wagner, "Face Recognition with GNU Octave/Matlab" «www.iti.pk.edu.pl,» 18 Julio 2012. [En línea]. Available: http://iti.pk.edu.pl/~chmaj/APSC/facerec_octave.pdf. [Último acceso: 16 Enero 2017].

[4] OpenCV Documentation, "Face Recognition with OpenCV-Open CV 2.4.13,2 documentation" «Opencv.org,» 2011. [En línea]. Available: http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html.

[5] R. Azad, B. Azad y I. tavakoli Kazerooni, "Optimized Method for Real-Time Face Recognition System Based on PCA and Multiclass Support Vector Machine" «www.acsij.org,» Noviembre 2013. [En línea]. Available: http://www.acsij.org/documents/v2i5/ACSIJ-2013-2-5-264.pdf. [Último acceso: 16 Enero 2017].

[6] M. A. Iborra, "Analisis comparativo de métodos basados en subespacios alicados al reconocimiento de caras" «www.researchgate.net,» Septiembre 2006. [En línea]. Available: https://www.researchgate.net/profile/Marcelo_Armengot/publication/237270192_Analisis_comparativo_de_metodos_basados_en_subespacios_aplicados_al_reconocimiento_de_caras/links/53fd83680cf2dca80003484d.pdf. [Último acceso: 16 Enero 2017].

[7] Mika Sebastian, G. Ratsch, J. Weston, B. Scholkopf, K. Muller y R. Muller, "Fisher Discriminant Analysis With Kernels" «www.courses.cs.tamu.edu,» 1999. [En línea]. Available: http://courses.cs.tamu.edu/rgutier/csce666_f13/mika1999kernelLDA.pdf. [Último acceso: 16 Enero 2017].