RECONOCIMIENTO FACIAL POR EL MÉTODO DE EIGENFACES

Franco Gabriel Caballero Julián

Tecnológico Nacional de México/Instituto Tecnológico de Oaxaca

francogcaballero@gmail.com

Martín Vidal Reyes

Tecnológico Nacional de México/Instituto Tecnológico de Oaxaca

martinvid@gmail.com

Antonia López Sánchez

Tecnológico Nacional de México/Instituto Tecnológico de Oaxaca

shunashi_stine@hotmail.com

Carlos Alberto Jerónimo Ríos

Tecnológico Nacional de México/Instituto Tecnológico de Oaxaca

carlosriostv@gmail.comr

Resumen

En este trabajo se presenta una aplicación de detección facial por Eigenfaces que usa Microsoft Visual Studio con diversas herramientas de programación como C# y librerías de la plataforma Emgu CV asociadas a OpenCV. El desarrollo de la aplicación se codificó en dos partes: una de registro y la otra de reconocimiento. El programa está cargado en una computadora de escritorio con sistema operativo Windows 8 y se usa la webcam integrada. En la etapa de registro el individuo se presenta ante la cámara para tomarle tres fotografías en diferentes instantes. En la etapa de reconocimiento el usuario se presenta ante la cámara y el sistema hace una comparación con todos los registros existentes en la base de datos indicando si el usuario está registrado. Tanto en la etapa de registro como en la etapa de reconocimiento se crean otros formularios que interconectan ambas, utilizan templates del explorador de soluciones por HaarCascade, CANNY_PRUNING y el

objeto EigenObjectRecognizer. El proyecto se sometió a la parte experimental con un universo del tamaño de diez usuarios, en el que ocho usuarios son hombres y dos usuarios mujeres, haciendo diez pruebas por usuario obteniendo una matriz de confusión con resultados del 100% de reconocimiento incluso con usuarios no registrados.

Palabra(s) Clave: Algoritmo, Eigenfaces, HaarCascade, Reconocimiento facial.

Abstract

This paper presents an Eigenfaces facial detection application that uses Microsoft Visual Studio programming software with tools such as C # and Emgu CV libraries associated with OpenCV. The development of the application was codified in two parts: one of records and the other of recognition. The program is loaded on a desktop computer with Windows 8 operating system and the integrated webcam is used. At the registration stage the individual appears before the camera to take 3 photographs at different times. In the recognition stage the user appears before the camera and the system makes a comparison with all the existing registers in the database indicating if the user is registered. Both the registration stage and the recognition stage create other forms that interconnect both, use solution explorer templates by HaarCascade, CANNY_PRUNING, and the EigenObjectRecognizer object. The project was submitted to the experimental part with a universe of the size of ten users, in which eight users are male and two female users, doing ten tests per user obtaining a confusion matrix with results of 100% recognition even with non-users recorded.

Keywords: *Algorithm, Eigenfaces, Facial recognition, HaarCascade.*

1. Introducción

Reconocimiento facial

Las aplicaciones de reconocimiento facial son posibles en la actualidad debido a las capacidades digitales de una cámara de alta resolución asociada a una computadora que de una imagen identifica automáticamente a un individuo. El reconocimiento biométrico del sujeto se hace por medio de un análisis de las

características faciales extraídas de la imagen que se compara con el contenido de una base de datos para su identificación.

En los últimos años, el reconocimiento facial es un área de investigación que se soporta con el desarrollo vertiginoso de las herramientas de programación y los sistemas digitales como los dispositivos procesadores y las memorias; involucra diversas disciplinas, como procesamiento de imágenes, reconocimiento de patrones, visión por computadora y redes neuronales [González y Woods, 2005]. Se le considera en el contexto del campo de reconocimiento de objetos, en el que la cara de una persona es el objeto en tres dimensiones sujeto a condiciones variables de iluminación, distancia, orientación, pose, etc.

Los métodos con mejor desempeño para reconocimiento son el algoritmo fisherface, el método de Eigenfaces, el modelo de cadenas ocultas de Markov, LDA - Linear Discriminant Analysis, el Subespacio de Aprendizaje Multilineal y la Contrapartida de Enlace Dinámico [Viorica y Capitan, 2016].

La tendencia más reciente en el reconocimiento facial es el método que utiliza cámaras 3D para capturar datos sobre la cara de alguien. Con esta tecnología en comparación con el reconocimiento 2D clásico se tienen mejores resultados, ya que no es sensible a las variaciones impredecibles de luz, las diferentes expresiones faciales, el maquillaje y puede identificar puntos de vista.

En este trabajo se utilizan herramientas de software que ya están disponibles desde la plataforma Net de Emgu CV, que contiene las librerías de procesamiento de imágenes compatible con OpenCV compilados desde Visual Studio con el lenguaje de programación C#. No es necesario desarrollar todo el código de programa para crear la aplicación, en el proyecto se declaran las librerías y referencias Emgu, cargar los objetos de captura y de cascadas de Haar para la detección de los rostros que con el uso de la función DO_CANNY_PRUNING para que en la detección de bordes se rechacen las regiones de la imagen que contienen pocos o muchos bordes donde no está el objeto deseado. Los valores de umbral particulares se calibran para la detección de rostros y en este caso se mejora la velocidad de respuesta. CANNY presenta un algoritmo óptimo con buena detección y buena localización con el empleo de máscaras de convolución

adaptable al ambiente a diferencia de otros. Finalmente, el objeto EigenObjectRecognizer realiza la tarea de reconocimiento en tiempo real al comparar la imagen actual con el banco de imágenes de entrenamiento.

Los autores originales Turk y Pentland [1991] en su artículo Eigenfaces for recognition presentan un sumario detallado para el reconocimiento de rostros que aquí se describe en forma abreviada:

- Colectar un conjunto de imágenes de rostros de individuos conocidos con algunas variaciones en la expresión y de iluminación de tamaño M.
- Calcular la matriz L, calcular sus eigenvectores y sus eigenvalores, elegir un nuevo espacio M´ de eigenvectores con los eigenvalores asociados.
- Hacer la combinación lineal de imágenes para formar un subespacio µL en una sumatoria para L= desde 1 hasta M.
- Para cada individuo calcular el vector Uk que define la distancia de cada representación en el libro de código.
- Para cada nueva imagen a identificar, encontrar la distancia mínima.
- Al encontrar coincidencia con un individuo conocido, la imagen se puede agregar a ese conjunto y recalcular una nueva Eigenface.

Eigenfaces

En visión por computadora, se le conoce como eigenfaces al conjunto de vectores propios para el reconocimiento de la cara humana. El método de eigenfaces fue planteado por Sirovich y Kirby [1987] para el reconocimiento y desarrollado unos años después en la clasificación de la cara por Matthew Turk y Alex Pentland [Turk y Pentland, 1991]. El término "eigen" se refiere a un conjunto de vectores propios. La principal virtud de este método es que se puede representar un conjunto de imágenes utilizando una base formada de imágenes "eigen" cuya dimensión es mucho más pequeña que el conjunto original. Los vectores propios resultan de la matriz de covarianza de la distribución de probabilidad sobre el espacio vectorial de alta dimensionalidad de las imágenes de la cara. Los propios eigenfaces forman un conjunto de base de todas las imágenes utilizadas para construir la matriz de covarianza.

El conjunto de eigenfaces se genera por medio de un proceso matemático llamado análisis de componentes principales (PCA) en un conjunto de imágenes que representan diferentes caras humanas. El enfoque de eigenfaces comenzó con la necesidad de encontrar una representación de pocas dimensiones de imágenes de rostros. Kirby y Sirovich demostraron que el Análisis de Componentes Principales se puede utilizar en un grupo de imágenes de la cara para formar un conjunto de características básicas [Turk y Pentland, 1991]. Este conjunto es conocido como "imágenes eigen" y se puede utilizar para reconstruir la colección de imágenes original. Cada rostro original sería reconstruido como una combinación lineal del conjunto base. Se extrae sólo la información crítica de la imagen de prueba y se codifica con la mayor eficacia posible. Fotografías de caras se proyectan en un espacio de características que ilustra mejor la variación entre las imágenes de rostros conocidos. Esta característica del espacio está definida por los vectores propios o "eigenfaces". El vector de pesos expresa la contribución de cada cara eigen a la imagen de nuestra entrada.

Los eigenfaces que se crean aparecerán como áreas claras y oscuras que se disponen en un patrón específico. Este patrón es cómo las características diferentes de una cara se destacan para ser evaluados y anotados. Habrá un patrón para evaluar la simetría, si hay algún estilo de vello facial, donde la línea del cabello es, o evaluar el tamaño de la nariz o la boca [Ottado, s.f.]. Otros eigenfaces tienen patrones que son menos fáciles de identificar, y la imagen de la eigenface puede parecer muy poco como una cara. La técnica utilizada en la creación de eigenfaces y su uso para el reconocimiento también se utiliza fuera del reconocimiento facial. Esta técnica también se utiliza para el análisis de escritura a mano, lectura de labios, reconocimiento de voz, interpretación de lenguaje de signos y gestos de manos y análisis de imágenes médicas. Por lo tanto, algunos no usan el término eigenface, pero prefieren usar eigenimage.

Implementación práctica

Los siguientes son los pasos para crear un conjunto de eigenfaces (Viorica y Capitan, 2016):

- Paso1. Consiste en preparar un conjunto de entrenamiento de las imágenes de cara. Tales imágenes constituyen el conjunto de entrenamiento, deben tomarse bajo condiciones equivalentes de iluminación, y se normalizan para alinear los ojos y la boca en todas las imágenes. También en el proceso computacional deben ser remuestreados a una resolución común de píxeles (r×c). Cada imagen se trata como un vector, simplemente concatenando las filas de píxeles en la imagen original, resultando en una sola columna con elementos r×c. Todas las imágenes del conjunto de entrenamiento se almacenan en una sola matriz T, donde cada columna de la matriz es una imagen.
- Paso 2. Restar la media. La imagen media se calcula y luego se obtiene una distancia de por medio de la resta de cada imagen original en T.
- Paso 3. Se calculan los autovectores y los valores propios de la matriz de covarianza S. Cada eigenvector tiene la misma dimensionalidad (número de componentes) que las imágenes originales, y por lo tanto es una representación de esa una imagen. Los autovectores de esta matriz de covarianza son por lo tanto llamados eigenfaces. Son las direcciones en que las imágenes difieren de la imagen media. Normalmente esto será un paso costoso desde el punto de vista computacional (si es posible), pero la aplicabilidad práctica de los eigenfaces se deriva de la posibilidad de calcular los vectores propios de S de manera eficiente, sin computar S explícitamente.
- Paso 4. Se eligen los componentes principales. En el paso 4, se ordenan los valores propios en orden descendente y en consecuencia se organizan los vectores propios. Estos eigenfaces ahora se pueden utilizar para representar caras existentes y nuevas: podemos proyectar una nueva imagen (media-substraída) en los eigenfaces y registrar así cómo esa cara nueva difiere de la cara media.

Los valores propios asociados con cada eigenface representan cuánto las imágenes en el conjunto de entrenamiento varían de la imagen media en esa

dirección. Perdemos información proyectando la imagen en un subconjunto de los vectores propios, pero minimizamos esta pérdida manteniendo esos valores propios con los valores propios más grandes.

Clasificadores cascada de haar

Para aplicaciones en la detección de personas se utilizan los clasificadores cascada de haar, de distribución y uso libre desarrolladas por OpenCV quienes implementan los métodos correspondientes. La librería proporciona una gran variedad de clasificadores para detectar (extensión *.xml), por ejemplo:

- Rostro: “haarcascade_frontalface_alt_tree” y “haarcascade_frontalface_alt”.
- Cuerpo completo: "haarcascade_fullbody".
- Boca: "haarcascade_mcs_mouth".
- Nariz: "haarcascade_mcs_nose".
- Sonrisas: "haarcascade_smile".
- Ojos con o sin lentes: "haarcascade_eye_tree_eyeglasses".
- Otras características faciales específicas.

En este proyecto se utiliza: "haarcascade_frontalface_default.xml" para hacer la detección del rostro.

Detección de personas

El proceso para hacer la detección de personas por medio de su rostro consiste de cuatro etapas fundamentales:

- La primera es cargar el clasificador para detección de rostros "haarcascade_frontalface_default.xml" que tiene los datos necesarios para el procesamiento de los rostros. Se utiliza un clasificador haar brindado por las librerías de OpenCV, el cual es un método de detección de objetos efectivo propuesto por Pablo Viola y Michael Jones en su artículo [Viola y Jones, 2001], el clasificador trabaja en un enfoque basado en el aprendizaje de las máquinas donde la función de cascada es entrenada de una gran

cantidad de imágenes positivas y negativas y se utiliza para detectar objetos en las imágenes para nuestro caso detección de rostros.

- La segunda etapa consiste en cargar nuestra imagen por medio de tomas que se adquieren de la cámara.
- La tercera parte y una de las más usadas para el procesamiento de imágenes es pasar nuestra imagen a escala de grises. Esto permite trabajar con menos información de los pixeles y sobretodo detectar más detalles.
- Por último, buscar en la imagen en escala de grises el rostro y enmarcarlo con un recuadro reflejado sobre la imagen original para indicar el rostro.

Captura y reconocimiento de rostros

Para el sistema de reconocimiento de rostros se realiza un análisis de características faciales extraídas de una imagen o fotograma que posteriormente son comparadas con las de una base de datos y de esta manera se logra identificar el rostro. Este procedimiento implica utilizar un algoritmo para la detección de rostros, de tal manera este proyecto implementa el algoritmo EigenFaces sin embargo, existen otros algoritmos como por ejemplo Logic Binary Pattens Histograms (LBPH) [Ahonen, Hadid y Pietikäinen, 2004], y el algoritmo FisheFaces [Belhumeur, Hespanha y Kriegman, 1997]. Dichos algoritmos son facilitados en clases por las librerías OpenCV y así se modera su implementación al ser declarados en el programa.

El programa está dividido en dos partes, la primera se encarga de detectar el rostro, almacenarlo en una base de datos y crear un archivo para que posteriormente se puedan leer las imágenes guardadas en la ruta indicada. La segunda parte del programa se encarga de leer los rostros almacenados en la base de datos y usar el algoritmo para hacer el reconocimiento. La librería OpenCV cuenta con clases (Facerecognizer) que facilitan el trabajo a la hora de hacer reconocimiento facial, la clase facerecongnizer implementa tres algoritmos: EigenFaces, FisherFases y LBHP, podemos implementar el que mejor se ajuste a nuestras necesidades. De tal forma que con ayuda de los métodos y algoritmos que proporciona OpenCV ya no es necesario desarrollarlos, sólo basta con

declararlos en el programa y así identificar a las personas que previamente almacenaron sus rostros.

2. Metodología

Crear un formulario de registro

Iniciamos insertando cuatro "Label", dos "TextBox", cinco "Button", un "groupBox", un imblmagenGuardado y un imageBoxFrameGrabber con los que dibujaremos la plantilla que se presenta en la figura 1. Estas herramientas las encontramos en "Toolbox".

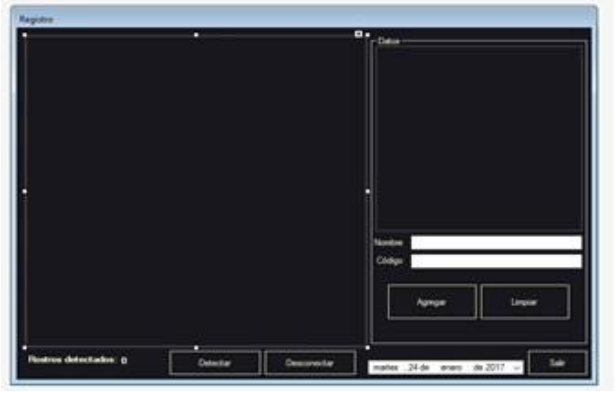


Figura 1 Plantilla del formulario para el registro.

No sólo este formulario es el que se utiliza, sino que serán dos formularios más, uno donde se visualiza lo que observa la cámara y el otro de registro. Para el formulario de registro, en el mismo archivo de programa en el que se está trabajando insertamos otro formulario en "Solution Explorer", ahí en una lista en la carpeta en C# demos clic derecho y agregamos "New Solution Explorer View". En ese formulario agregamos los recursos y el código que permita cargar todas las imágenes de entrenamiento, figura 2.

Crear un formulario de reconocimiento

De igual forma que en el primer formulario es para esta segunda parte ya que estarán conectados para que funcione la red de identificación, esta ventana es la que permite guardar los registros y a la vez es la que permite que la persona

detectada sea reconocida mediante un texto que presenta su identificador en la pantalla. En este formulario se insertan 4 "Label" y 3 "Button", como se muestra en la figura 3, (estas herramientas se encuentran en Toolbox) y un imageBoxFrameGrabber. Proseguimos editando el formulario y las etiquetas. Para esto seleccionamos el "Form 1" y en la ventana "Properties" editaremos los campos haciendo que todo coincida.

```
            Labelsinfo = File.ReadAllText(Application.StartupPath +
"/TrainedFaces/TrainedLabels.txt");
            string[] Labels = Labelsinfo.Split('%');
            NumLabels = Convert.ToInt16(Labels[0]);
            ContTrain = NumLabels;
            string LoadFaces;

               for (int tf = 1; tf < NumLabels + 1; tf++)
               {
                  LoadFaces = "face" + tf + ".bmp";
                  trainingImages.Add(new Image<Gray,
byte>(Application.StartupPath + "/TrainedFaces/" + LoadFaces));
                  labels.Add(Labels[tf]);
               }
         }
         catch (Exception e)
         {
            MessageBox.Show(e + "No hay ningun rosto registrado).", "Cargar
rostros", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
         }
      }
```

```
//Detector de Rostros

         MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(face, 1.2, 10,
HAAR_DETECTION_TYPE.FIND_BIGGEST_OBJECT, new Size(20, 20));

//Acción para cada elemento detectado

         foreach (MCvAvgComp f in facesDetected[0])
         {
            t = t + 1;
            result = currentFrame.Copy(f.rect).Convert<Gray, byte>().Resize(173,
168, INTER.CV_INTER_CUBIC);

//Dibujar el cuadro para el rostro

            currentFrame.Draw(f.rect, new Bgr(Color.LightGreen), 1);

            if (trainingImages.ToArray().Length != 0)
            {
```

Figura 2 Un bloque de código para leer las imágenes y su detección.

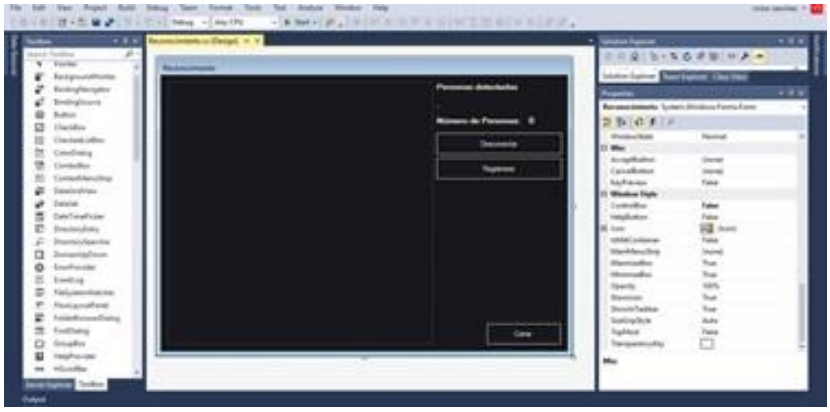


Figura 3 Formulario de reconocimiento.

De la misma manera que en el registro insertamos otro formulario, agregamos los recursos y el código que permita realizar las operaciones en la etapa de

reconocimiento por HaarCascade, CANNY_PRUNING y el objeto EigenObjectRecognizer, figura 4.

```
//Detector de rostros
            MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(face, 1.2, 10,
HAAR_DETECTION_TYPE.FIND_BIGGEST_OBJECT, new Size(90, 90));

//Acción para cada elemento detectado
            foreach (MCvAvgComp f in facesDetected[0])
            {
                t = t + 1;
                result = currentFrame.Copy(f.rect).Convert<Gray, byte>().Resize(173,
168, INTER.CV_INTER_CUBIC);

//Dibujar el cuadro para el rostro detectado
                currentFrame.Draw(f.rect, new Bgr(Color.YellowGreen), 1);
                NamePersons[t - 1] = name;
                NamePersons.Add("");
```

```
//Obtenga un marco de gris del dispositivo de captura
            gray = grabber.QueryGrayFrame().Resize(640, 480,
INTER.CV_INTER_CUBIC);

//Detector de rostros
            MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(face, 8.0, 10,
HAAR_DETECTION_TYPE.DO_CANNY_PRUNING, new Size(90, 90));

//Acción para cada elemento detectado
            foreach (MCvAvgComp f in facesDetected[0])
            {
                TrainedFace = currentFrame.Copy(f.rect).Convert<Gray, byte>();
                break;
            }
```

Figura 4 Un bloque de código en la etapa de reconocimiento.

La etapa de entrenamiento

Cuando la persona no está dada de alta en el programa es necesario pulsar el botón de detectar en la aplicación para que la cámara entre en funcionamiento y reconozca el rostro de la persona que esté frente a la cámara, entonces será necesario rellenar un pequeño formulario. Al agregar la persona será necesario que se ingrese tres veces al sistema la misma persona, por la cual se crea tres matrices en escala de grises en los registros de las fotografías de los usuarios y se guarda por vectores, figura 5.

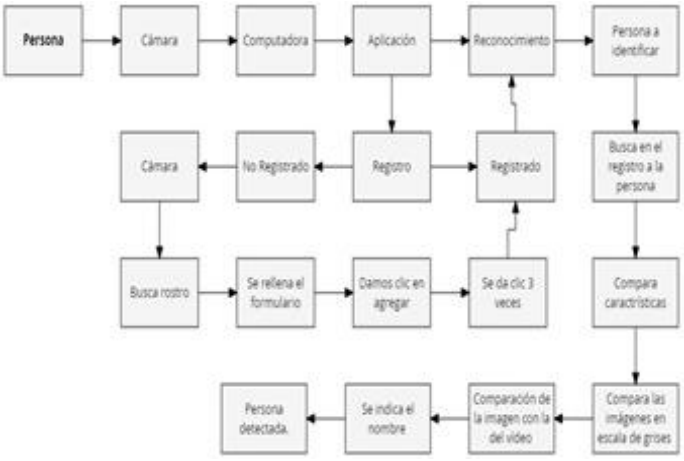
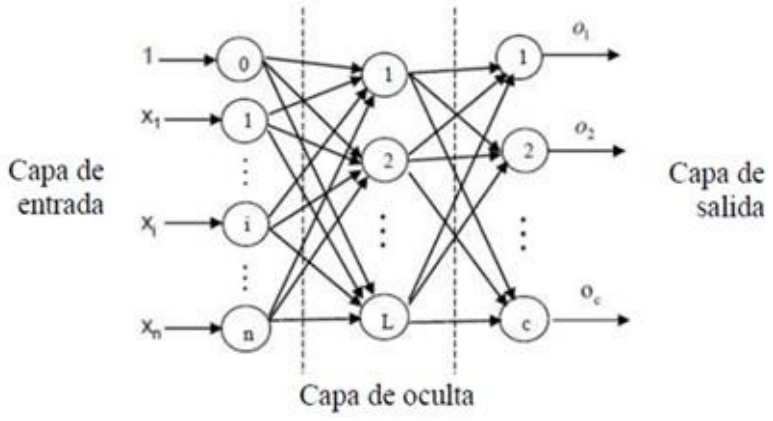


Figura 5 La etapa de entrenamiento.

La etapa de reconocimiento

Una vez que se registra la persona en la base datos damos en salir en el formulario de registro, llamar al formulario de reconocimiento en el cual la persona a identificar se encuentra frente a la cámara. El programa busca en su registro a la persona, compara todas las imágenes guardadas por medio de una red neuronal que en la aplicación es transparente. El método _faceRecognizer.Predict” ”) es un método predictivo en redes neuronales recurrentes de tiempo discreto en la que a partir de la historia pasada de una o más variables la red neuronal debe proporcionar una predicción lo más correcta posible de su valor futuro, su implementación es con los argumentos propios del objeto EigenObjectRecognizer, haciendo distintas comparaciones entre ellas para determinar quién es la persona, figura 6.



Fuente [García, 2014]

Figura 6 Etapa de comparación por redes neuronales.

Turk y Pentland [1991] plantean el uso de una red neuronal de tres capas donde en la capa de entrada se tienen las imágenes de los rostros, la capa intermedia corresponde a los valores de las eigenfaces tal que esa capa corresponde al producto punto de la imagen de entrada y las eigenfaces. La capa de salida produce la proyección espacial de la imagen de entrada con su peso correspondiente a cada eigenfaces.

En esta etapa lo que hace el programa es que cuando el usuario es detectado por la cámara el programa hará una comparación con todos los registros existente en

el programa la cual es llamada como capa de entrada, en la capa oculta aquí el programa comienza a relacionar características que tienen todos los usuarios eliminando a aquellos que no tienen similitud con la persona a reconocer, en la capa de salida es donde la red artificial ya tiene una respuesta a nuestro usuario y al sistema identificando quien es la persona.

3. Resultados

Presentación del formulario de registro

Ahora se muestra cómo funciona el formulario de registro (cuando esté en función), para realizar esta prueba es necesaria verificar que la webcam esté funcionando correctamente. Ejecutamos la aplicación, y damos clic al botón "Registrarse", figura 7. Al dar clic en registrarse aparece una nueva ventana y hay que darle clic en "Detectar" paro poder registrar los usuarios en nuestra base de datos. Una vez que la cámara entre en funcionamiento, detectara al usuario marcando el rostro con un recuadro color amarillo, ahora lo primero que haremos es rellenar el formulario donde nos pide nombre y código (se muestra las opciones en la figura 8).

Figura 7 Proceso de registro.

Figura 8 Captura de datos.

Una vez que tengamos esta parte (es importantes primero rellenar el nombre y código) daremos clic en el botón de "Agregar", figura 9.

Es importante mencionar que para que el programa tenga un mínimo de errores se realicen tres veces agregar (no es necesario salir del programa se puede hacer allí

mismo), por último, damos en limpiar para borrar datos y agregar a nuevos usuarios. Una vez que esté limpio el formulario damos clic en "Desconectar" y "Salir", ver figura 10.

Figura 9 Toma de tres fotografías.

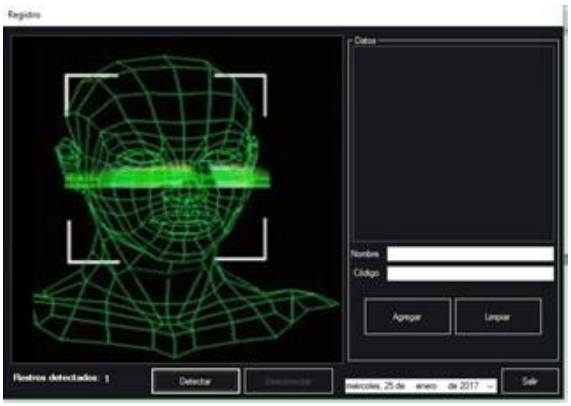


Figura 10 Limpiar registros.

Detección y reconocimiento

Ahora reiniciamos la aplicación para que se refresque con los datos nuevos ingresados y podemos ver como que ahora detecta a la persona que fue registrada marcando su nombre, figura 11.

Figura 11 Reconocimiento de la persona.

4. Discusión

En la tabla 1 se exhiben los resultados del experimento que se realizó para los propósitos específicos de este artículo, con un universo de 10 usuarios de los que

se hicieron tres registros de entrenamiento con 10 pruebas de identificación a cada uno, dando como resultado que el programa entregue el 100% de efectividad lo cual hasta cierto punto puede ser engañoso porque el mismo método no ofrece tal porcentaje. Es de suponerse que resulta más confiable pues entre más fotos se tengan en el registro, el programa tendrá un mínimo de error.

Tabla 1 Matriz de confusión del experimento.

	Us. 1	Us 2.	Us. 3	Us. 4	Us. 5	Us. 6	Us. 7	Us. 8	Us. 9	Us.10
Us. 1	10									
Us. 2		10								
Us. 3			10							
Us. 4				10						
Us. 5					10					
Us. 6						10				
Us. 7							10			
Us. 8								10		
Us. 9									10	
Us.10										10

El reconocimiento es inmediato, puede considerarse un tiempo de respuesta en tiempo real, pero en un universo de usuarios más amplio, el tiempo computacional de comparación tendrá un mayor retardo lo que es un pendiente por resolver a futuro, incluso para saber el tiempo de demora en el reconocimiento habrá que realizar algunas pruebas para tener un dato fidedigno. Para las primeras pruebas que se realizaron solo se tomaba una fotografía lo cual hacia que el programa tuviera confusión con los usuarios de rasgos similares. Todavía es necesario experimentar con el prototipo con más usuarios y en diferentes condiciones de iluminación, así como con diferentes cámaras con mejor resolución. Otros estudiantes siguen trabajando en el proyecto ahora en el desarrollo de la implementación de una aplicación de seguridad en el acceso a sus espacios académicos que permita emitir una alarma que alerte al vigilante ante la presencia de personas no autorizadas para su registro y revisión.

5. Bibliografía y Referencias

[1] Ahonen, T., Hadid, A., y Pietikäinen, M. (2004). Face Recognition With Local Binary Patterns. Machine Vision Group, 469/481.

[2] Anil K. Jain, Ajay Kumar, "Biometrics of Next generation: An overview", to appear in Second Generation Biometrics Springer, 2010.

[3] Belhumeur P. N., Hespanha J., y Kriegman D. (1997). Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 711/720.

[4] Gonzalez, R., y Woods, R. (2002). Digital Image Processing (Vol. Second Edition). United States of America: Prentice Hall.

[5] Gonzalez, R. E. Woods, S. L. Eddins. (2005). "Digital Image Processing using Matlab", Prince Hall.

[6] García, J. (2014). Apuntes de la cátedra de Análisis y Extracción de Información. Máster en Teledetección (2014-2015), Universidad de Valencia. Valencia, España.

[7] Jiménez, J. G. (1999). Visión por Computador. Paraninfo.

[8] Lienhart, R., y Maydt, J. (2002, septiembre). An Extended Set of Haar-like Features for Rapid Object Detection. IEEE ICIP, 900-903.

[9] Ottado, G. (s.f). Reconocimiento de caras: Eigenfaces y Fisherfaces.

[10] Rost Bradski, G., y Kaehler , A. (2008). Learning OpenCV Computer Vision With The OpenCV Library. United States of America: O`REILLY.

[11] Turk, M. y Pentland, A. (1991). Eigenfaces for recognition. Vision and modeling group. Massachusetts Institute of technology.

[12] Viola, P., y Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE CVPR.

[13] Viorica, P., Capitan, F., (2016). "Aplicación para detección y reconocimiento facial en interiores". Trabajo final de grado Ingeniería Electrónica, Robótica y Mecatrónica. Escuela Técnica Superior de Ingeniería. Universidad de Sevilla.

[14] Zisserman, A. (2003). Multiple View Geometry in Computer Vision (Vol. Second Edition). Canberra, Australia: CAMBRIDGE.