Reconocimiento Facial con OpenCV y OpenMP

Alberto Gisele Romina, Benitez Gerardo Abel, Coradini Gonzalo Fabián, Vázquez Lucia Maité

Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina
gisele.alberto08@gmail.com, gerardobenitez@gmail.com, gonzalocoradini@gmail.com, luciamaitev@gmail.com

Resumen. El siguiente documento es un trabajo de investigación donde analizaremos el funcionamiento del reconocimiento facial, su estado del arte y, a su vez, explicaremos la forma de implementación al proyecto Smartiquin. Para ello, aprovecharemos el poder de paralelización que ofrece OpenMP, para tener resultados en el menor tiempo posible, además de asegurar la correspondencia o no de la imagen capturada a través de la cámara IP, y contrastarlas con aquellas registradas en la base de datos.

Palabras claves: Reconocimiento Facial, Cámara IP, OpenCV, OpenMP.

1. Introducción

La investigación que se desarrolla en este documento está aplicada al sistema Smartiquin, donde buscamos añadir un método de seguridad alternativo para evitar accesos indeseados al mismo. El botiquín cuenta con una cerradura magnética que funciona mediante un electroimán como primera opción, y actualmente la apertura de la misma solo puede realizarse a través de una aplicación instalada en un smartphone. La segunda opción de apertura que vamos a agregar será mediante el reconocimiento facial, vamos a usar una cámara IP para capturar imágenes de los rostros frente al botiquín y mediante el algoritmo seleccionado, vamos a procesar las mismas utilizando threads para paralelizar la etapa de reconocimiento.

Uno de los desafíos es elegir el algoritmo adecuado para resolver la problemática de reconocimiento facial, debido a la dificultad de analizar los rasgos faciales en las imágenes captadas, existe variabilidad en la posición, iluminación, gestos, sombra, etc. Por último tomar ventaja de la paralelización de procesamiento ofrecida por la tecnología OpenMP con el fin de ofrecer resultados en un tiempo menor.

El reconocimiento facial es considerado como la medición biométrica más natural entre las que existen (huellas dactilares, lectura de iris, reconocimiento de voz, etc.), ya que es el método más utilizado por los seres humanos para reconocerse unos a otros.

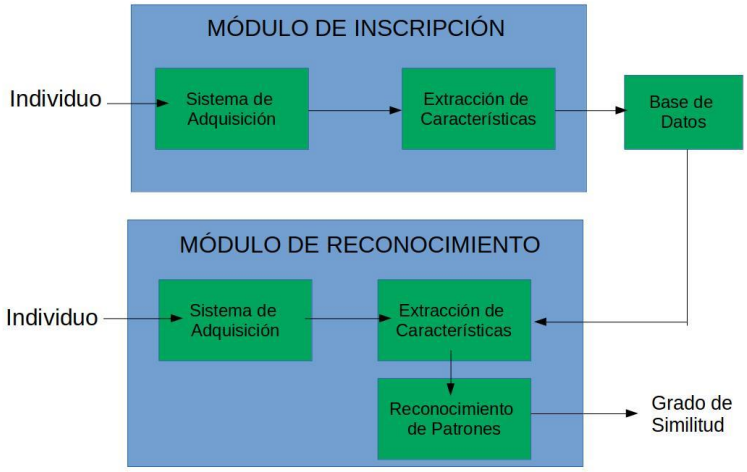
Aplicaciones como Google Fotos y Facebook, han incorporado el reconocimiento facial para distinguir a personas en tus fotos, y así poder etiquetarlas y clasificarlas. El iPhone X incorporó Face ID para autenticar usuarios [7]. En un uso más amplio, podemos ver la implementación del mismo en tiempo real en aeropuertos internacionales, o aquí en Buenos Aires, para identificar entre las personas que transitan por distintos lugares de Capital Federal, si están prófugos de la justicia, si es que se encuentran en la base de datos de personas buscadas en Argentina.

2. Desarrollo

Existen ocho algoritmos que pueden ser usados en reconocimiento facial [4] [5], que son: PCA, LDA, ICA, LBP, FDA, EBGM, KPCA, HMM, entre otros. Nosotros nos centraremos en el algoritmo PCA (Principal Components Analysis), ya que es un método matemático que se utiliza para reducir el número de variables, de forma que pasemos a tener el mínimo número de nuevas variables, llamadas componentes principales, y estos representan a todas las antiguas de la forma más representativa posible. En el caso puntual de reconocimiento facial, este algoritmo es llamado Eigenfaces[1], y funciona proyectando las imágenes faciales sobre un espacio de facciones que engloba las variaciones significativas entre las imágenes faciales conocidas.

[1]

Podemos dividir el proceso en dos módulos identificables: El módulo de inscripción, donde se lleva a cabo el registro del individuo a la base de datos (el entrenamiento del modelo), y el segundo es donde se realiza el reconocimiento a partir de los datos recolectados en el módulo anterior [2]. Dentro de nuestra investigación, utilizaremos librerías open source como OpenCV y OpenMP. La primera fue diseñada para procesamiento de imágenes, y entre las funcionalidades destacadas, se incluyen la detección y reconocimiento de objetos. OpenMP fue creada para soportar multiprogramación, por lo que es fundamental para trabajar con hilos concurrentes en la ejecución del algoritmo.



[2]

Etapa de entrenamiento

Cómo primer paso, se debe detectar un rostro frente a la cámara ubicada en el botiquín, una vez ubicado, se dá lugar la detección de los ojos del sujeto dentro de esa región. Para obtener mejores resultados, es necesario que se tomen varias fotos por individuo, con distintas expresiones, y tomarse bajo condiciones equivalentes de iluminación. Estas se normalizan para alinear los ojos y la boca en todas las imágenes, y redimensionarlas a un tamaño común para todas de wxh pixeles. Esto se llevará a cabo cada vez que se registre un nuevo usuario en Smartiquin.

Una vez se tienen todas las imágenes de entrenamiento preprocesadas, se procede a la extracción de características, cuyo primer paso consiste en aplicar la técnica PCA al conjunto de imágenes preprocesadas para calcular los autovectores que formarán la base ortonormal del nuevo subespacio (Eigenfaces). Por último se proyectan las imágenes de entrenamiento preprocesadas sobre las eigenfaces, de tal forma que se obtienen los coeficientes de las imágenes de entrenamiento en el nuevo subespacio.

Etapa de reconocimiento

Al igual que en la etapa de entrenamiento, algunos de los primeros pasos se repiten: Detectar el rostro y los ojos, luego realizar el preprocesado de la imagen captada. A partir de las eigenfaces ya calculadas, se proyectan cada uno de los rostros en el nuevo subespacio para obtener los coeficientes de las imágenes de estos rostros. Con estos nuevos coeficientes, más los previamente calculados en la etapa de entrenamiento, puede llevarse a cabo el reconocimiento. Para ello, con el teorema de Pitágoras, se calcula la distancia entre la proyección del rostro en cuestión y cada una de las proyecciones de las imágenes de entrenamiento. La proyección cuya distancia sea menor, indicará a qué individuo pertenece, y así proporcionar la autenticación deseada en nuestro proyecto.

3. Explicación del algoritmo

V_MIN depende del grado de coincidencia que deseemos tenga el algoritmo, 90% por ejemplo, para ser considerado válido.

```
registroUsuario() {
  capturarImagen();
  procesarImagen();
  generarMatriz();
  insertarEnBD();
}
```

```
reconocimientoUsuario() {
   capturarImagen();
   procesarImagen();
   generarProyección();
   if (compararCoeficientes() > = V_MIN)
{
      mostrarTexto(“Usuario Validado”);
      abrirBotiquin();
} else {
```

```
mostrarTexto("Usuario No Registrado");
}
```

capturarImagen: Esta función va a contener los dos métodos que van a estar encargados de delimitar el recuadro de captura, y que servirán para el preprocesado de imágenes posterior.

- *deteccionRostro*: Esta función primeramente detectará la región del rostro, utilizando el clasificador Haas proporcionados en OpenCV: "haarcascade_frontalface",
- *deteccionOjos*: Una vez detectada la región, pasará a realizar la detección de ambos ojos por separados con los clasificadores: "haarcascade_lefteye" y "haarcascade_righteye". Conociendo la posición de los ojos, se puede llegar a determinar el ángulo de rotación de la cara para cuando se realice la alineación.

procesarImagen: Aquí se aplican una serie de transformaciones a cada imagen detectada para la extracción de características, eliminando el pelo, orejas, y solo dejando visible el rostro. Posteriormente se aplica la rotación, el escalado, el recorte y la transformación a escala de grises. Se debe emplear el mismo tamaño para cada rostro ecualizado.

generarMatriz: Por simplicidad, englobamos en esta función la aplicación del algoritmo de PCA para obtener las Eigenfaces, generando los vectores correspondientes para las imágenes, y la matriz de covarianza. *generarProyeccion*: Aquí la imagen de prueba a ser procesada, se proyecta en un nuevo subespacio a partir de las Eigenfaces calculadas en generarMatriz, y se obtiene el coeficiente de ésta nueva imagen.

compararCoeficientes: Es en esta función donde realizamos la paralelización con memoria compartida, lanzando hilos para que cada uno haga el reconocimiento simultáneamente, reduciendo el tiempo de ejecución. Se comparan los coeficientes de las proyecciones de las imágenes guardadas en la base de datos con las de la imagen captada.

4. Pruebas que pueden realizarse

Una de las pruebas que pueden realizarse, es la implementación del algoritmo con y sin el empleo de OpenMP, empleando una base de datos grandes, para así demostrar de forma certera, la diferencia en tiempos de ejecución al procesar las imágenes. Para este caso, utilizaremos alguna base de datos que esté en la nube, o creada por nosotros con un número considerable de individuos con sus variantes de imágenes.

Otra prueba, que podríamos realizar al emplear una cámara 3D, es verificar que si proporcionamos una fotografía de un usuario registrado frente a la lente, el sistema no valide el ingreso y avise de este evento, ya que una imagen plana no proporciona la misma profundidad que es capaz de detectar una cámara 3D.

5. Conclusiones

A modo de resumen, en este informe se propuso principalmente enunciar las consecuencias de incorporar el reconocimiento facial al sistema Smartiquin como método de seguridad adicional para la habilitación de la apertura del mismo, centrándonos en el algoritmo PCA - explicado en la sección 2. Gracias al mínimo número de nuevas variables que se consiguen con éste método matemático y al paralelismo en el procesamiento de las mismas, se logra una implementación eficiente del reconocimiento facial.

Si bien el algoritmo elegido es uno de los más usados, por robustez y facilidad de implementación, tiene sus desventajas en consideración a la luz ambiente y posición del rostro al querer realizar la identificación, por lo que en alguna futura investigación, podemos desarrollar el algoritmo de Fisherfaces, que si considera cambios de luz, posición y expresiones faciales.

A su vez, para futuras investigaciones, podríamos incorporar procesamiento GPU y ver cómo éste puede mejorar el desempeño de la paralelización.

6. Referencias

1. Lopez, E.: Estado del Arte del Reconocimiento Facial (2014)
2. Garduño Santana, M.A., Díaz-Sánchez, L. E. , Tabarez Paz, I., Romero Huertas, M.: Estado del Arte en Reconocimiento Facial (2017)
3. Arcos Acuña, Y.F., Moreno Meneses, D.A., Niño Pacheco, W.D.: Prototipo de Seguridad Mediante Reconocimiento Facial por Medio del Algoritmo de Eigenfaces (2015)
4. Dominguez Pavón, S.: Reconocimiento Facial Mediante el Análisis de Componentes Principales (PCA) (2017)
5. Suryaprasad, J., Sandesh, D. S., Priyanka, I., Pravalika, G.N., Aman Kumar: Parallel Implementation and Performance Evaluation of Facial Recognition Algorithms Using Open Source Technologies (2017)
6. Aprende Machine Learning https://www.aprendemachinelearning.com
7. Gemalto https://www.gemalto.com/latam/sector-publico/biometria/reconocimiento-facial