Seguridad HPC a través del Reconocimiento Facial en Pastillero Inteligente

Cairo Matias, Lamberto Jofré Victor Enrique, Retamar Alejandro Ruben, Segovia Elías, Veltri Gonzalo.

[1]Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina
maticairo@gmail.com, enquibe@gmail.com, alejandro_r_retamar@hotmail.com,
segoviaelias92@gmail.com, veltrigonzalo@gmail.com

Resumen. Nuestra investigación consiste en agregar una funcionalidad más al sistema embebido proporcionando un mecanismo de seguridad a través del reconocimiento facial que brindan los sensores biométricos del dispositivo Android, usando la tecnología HPC mediante el paralelismo GPU del dispositivo para lograr un óptimo, rápido y eficiente reconocimiento.

Palabras claves: HPC, paralelismo, GPU, biométrico, openCV, Sistema Embebido, Android.

1 Introducción

El propósito de la investigación consiste en proporcionar mayor seguridad al pastillero inteligente.

Nuestro pastillero permite realizar una planificación de las pastillas que el usuario debe tomar durante la semana a través del dispositivo Android que se conecta vía Bluetooth al pastillero. Al llegar el día y la hora indicadas, el pastillero libera la pastilla correspondiente, enciende una luz y activa una alarma. Si el pastillero estuviera al alcance de niños u otra persona que no sea el usuario, la pastilla podría ser extraída, lo cual sería sumamente peligroso. Debido a esto, el objetivo de esta investigación es que mediante un reconocimiento facial del dispositivo Android el pastillero sólo despida la pastilla una vez que fue reconocido el rostro del usuario que se registró en la aplicación.

Actualmente existen varios métodos de reconocimiento facial, estos son por ejemplo, el algoritmo fisherface [1], el método de Eigenfaces [2], 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] [3].

Cuando se emplea cualquiera de estos métodos de reconocimiento en el sistema se puede considerar que la persona pueda estar en movimiento y aun así se la reconozca,

este sistema necesita de una fuente de luz, lo cual puede implicar un inconveniente en ambientes con falta de iluminación; aún así también existen problemas a personas con cambios faciales.

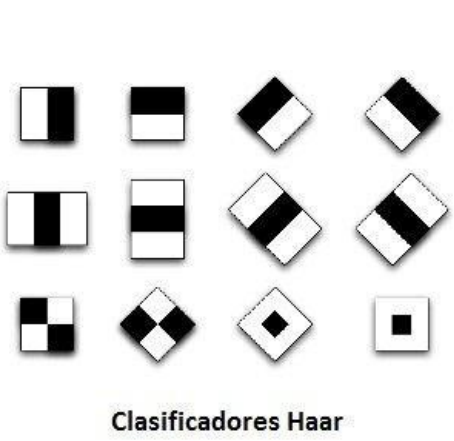
La funcionalidad que se agregará a nuestra aplicación consiste en que el usuario realice capturas de su rostro de diferentes ángulos, las cuales se guardarán en la base de datos del sistema del pastillero alojada en la nube, de esta manera, cuando se produzca el reconocimiento facial, la aplicación validará el rostro con las capturas almacenadas por el usuario. Para lograr esto utilizaremos el módulo de GPU del dispositivo para procesar una cantidad considerable de muestras capturadas de forma paralela para lograr la detección en un tiempo de respuesta lo más corto posible.

En la actualidad, son muchas las aplicaciones móviles similares que utilizan mecanismos de detección de rostros. Algunos ejemplos de esto son las aplicaciones desarrolladas por los bancos para entrar a la cuenta bancaria, Facebook, Instagram y Snapchat para etiquetar a personas o aplicar filtros, etc.

2 Desarrollo

Para el reconocimiento facial usaremos la biblioteca openCV, que implementa el algoritmo o clasificador Haar, y fue el primer framework de detección de objetos propuesto por Paul Viola y Michael Jones en 2001 [4], el cual permitió el análisis de imágenes en tiempo real, haciendo uso de una función matemática (Wavelet Haar) propuesta por Alfred Haar en 1909.

Los clasificadores haar, definen regiones rectangulares sobre una imagen en escala de grises (imagen integral) que, al estar formada por un número finito de rectángulos, posibilita la obtención de un valor escalar que consiste en sumar los pixeles de cada rectángulo, en base a una serie de clasificadores en cascada. Cada clasificador determina si la subregión se trata del objeto buscado o no. A diferencia de otros algoritmos, este solo invierte capacidad de procesamiento a las subregiones que posiblemente representen un rostro.



Clasificadores Haar

Cálculo de haar sobre imágen

Este sistema tiene un porcentaje de aciertos bastante alto, aunque su éxito también dependerá del tipo de cámara utilizada, la iluminación de la sala, etc.

3 Explicación del algoritmo.

Primeramente se guardan en el servidor los clasificadores Haar, estos archivos son "modelos" de los rostros humanos, se generarán modelos particulares para cada usuario, donde se logrará un entrenamiento con un promedio de los distintos face code obtenido de cada imagen.

Luego cuando el sistema tiene que verificar la captura de la imagen del usuario capturada a través del dispositivo, el algoritmo busca en dicha imagen todas las posibles combinaciones de estos patrones.

Pseudocódigo:

Se debe incluir la biblioteca openCV (se recomienda en su última versión).

```
let CV = OpenCV();
if (Camera.isOpened()) {  // cuando se abre la camara
int[][] MatrizRgb = Camera.read(); // se lee la imagen capturada
int[][] MatrizRgbGry = cvtColor(MatrizRgb, CV_RGBA2GRAY) // Pasa la imagen a escala de grises

int cantHaar_Like = getServidorCantHaar_Like(); // se busca la cantidad de figuras patrones
```

detectarPatron<<<1,cantHaar_Like>>>(MatrizRgbGry, resultado); // ejecuta todas las comparaciones en paralelo ejecutando en bloque tantos hilos en la GPU como patrones se tenga

```
---
__global__ void detectarPatron(MatrizRgbGry, resultado){
if(MatrizRgbGry==Patron)
        resultado++;
}
---

if(resultado >= promedio)
        return true;
else
        return false;
```

Si la cantidad de comparaciones es mayor al promedio se llega a la conclusión de que la imagen captura coincide con la del usuario.

4 Pruebas que se deben realizar

Se deben realizar las siguientes pruebas:

- Probar cómo funciona el algoritmo con una baja cantidad de tomas capturadas, mientras más tomas realizadas mayor es la probabilidad de reconocimiento.
- Probar cómo es la detección con niveles bajos de iluminación en el ambiente.
- Probar cómo se comporta con distintas calidades de cámara en diferentes dispositivos, a mayor calidad (resolución) mejor es la detección.

5 Conclusiones

- El complemento de usar HPC en sistemas embebidos es de vital importancia debido a la cantidad de tiempo que se puede ahorrar sabiendo como funciona dicha tecnología; además, en nuestro caso, nos permite brindar una funcionalidad que proporciona seguridad al usuario.
- Aunque hoy en día son cada vez más los dispositivos que trabajan con sensores biométricos e incorporan mecanismos de reconocimiento facial, es difícil trabajar con esta tecnología debido a que requisitos como calidad de imagen e iluminación deben ser óptimos, sumado a las fallas debido a los cambios faciales de la persona.
- En conjunto con la funcionalidad de reconocimiento facial investigada y desarrollada en este artículo, se abren puertas para la investigación sobre algoritmos que permitan, por ejemplo, saber de forma rápida qué proveedor de

pastillas (farmacia) se encuentra disponible en el momento en que el pastillero informa que se están acabando las pastillas.

6 Referencias

1. Ottado, G. (s.f). Reconocimiento de caras: Eigenfaces y Fisherfaces; 2017.
2. Face Recognition Machine Vision System Using Eigenfaces Fares Jalled, Moscow Institute of Physics & Technology, Department of Radio Engineering & Cybernetics [8 mayo 2017].
3. LDA - Linear Discriminant Analysis, el Subespacio de Aprendizaje Multilineal y la Contrapartida de Enlace Dinámico [Viorica y Capitan, 2016].
4. ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION 2001 Rapid Object Detection using a Boosted Cascade of Simple Features
5. Face Detection and Tracking using OpenCV The SIJ Transactions on Computer Networks & Communication Engineering (CNCE), Vol. 1, No. 3, July-August 2013.