Materia: Control Microprocesado
Miguel Braidot

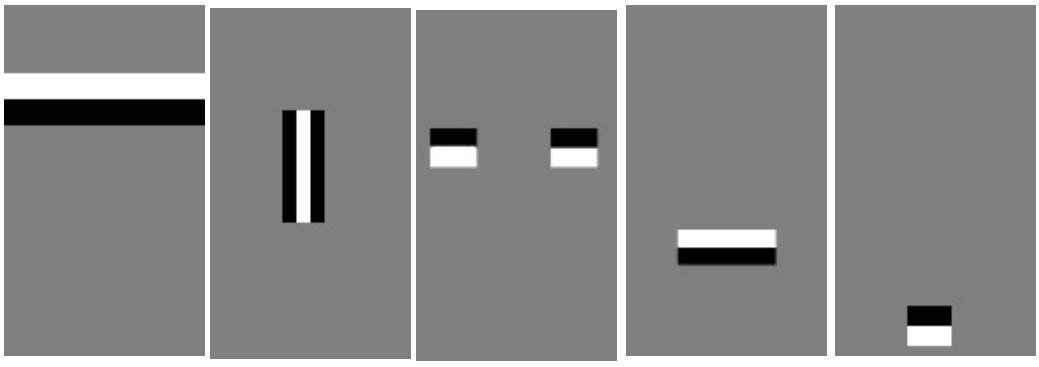
ROSTROS
Reconocimiento e Identificación

Para nosotros, los humanos, determinar que lo que vemos es un rostro y no otro objeto es una tarea casi natural, nuestro cerebro lo resuelve en fracciones de segundo; sin embargo, para una computadora esta tarea no es trivial.

1 EL RECONOCIMIENTO

Detectar una cara en una foto, es más simple que identificar si se trata de una cara conocida.
Para determinar que una foto o el cuadro de un video contiene una cara, o varias, debemos definir la estructura general de un rostro. Por suerte, las caras humanas no difieren mucho entre sí; todos tenemos narices, ojos, boca, frente, mejillas; y todos estos componen la estructura de una cara.

Consideremos la siguiente imagen:



Cada una de estas imágenes representa una característica de una cara humana. Combinándolas todas, obtenemos un acercamiento a lo que podría ser un rostro:

Si podemos determinar que esta estructura se encuentra en una imagen, podemos decir con cierto grado de certeza, si la imagen contiene un rostro o no.
Reuniendo datos estadísticos sobre cuáles de estas características componen una cara y cómo, podemos entrenar un algoritmo para usar las características correctas en la posición correcta, y, por lo tanto, reconocer un rostro.

El proceso consiste en recorrer la imagen secuencialmente por zonas, buscando si está presente alguna de las características de una cara, en principio y para no sobrecargar el algoritmo, se buscan los datos más gruesos y si se determina que puede existir una cara, se realizan pasadas posteriores buscando datos más finos. A este proceso se lo denomina "*proceso en cascada*".

Esta es una explicación simplificada del método desarrollado por Viola-Jones [1].

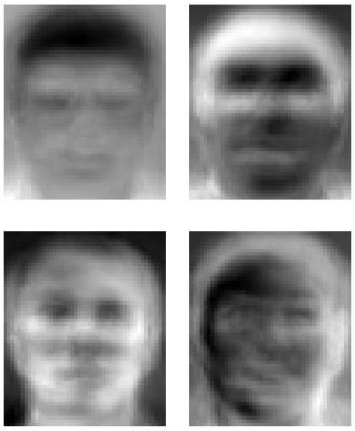
2 IDENTIFICACIÓN DE ROSTROS

Una vez que podemos determinar que en una imagen hay rostros humanos, nos adentramos en una tarea más ambiciosa, identificar a quién pertenece ese rostro.
Para esto se utilizan métodos matemáticos más complejos, como álgebra lineal y estadística.

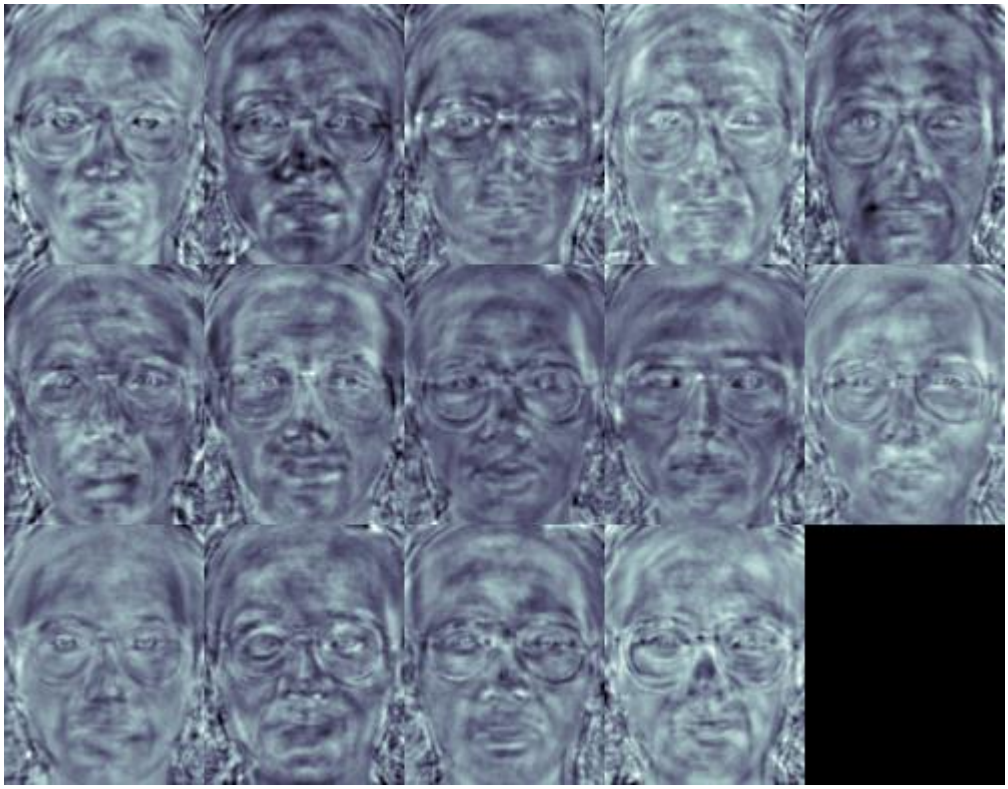
Para facilitar dicha tarea, existe una librería de software desarrollada originalmente en IBM y luego convertida en Open Source donde ha continuado su desarrollo.
Esta librería se llama OpenCV [2], y provee tres métodos de reconocimiento de rostros: Eigenfaces, Fisherfaces y Local Binary Patterns Histrogram (LBPH).

Los tres métodos realizan el reconocimiento comparando la imagen de la cara con algún conjunto pre-entrenado de caras conocidas. En el entrenamiento le mostramos caras al algoritmo, y le decimos a quien pertenecen. Cuando le pedimos que nos identifique un rostro, el algoritmo utiliza los rostros que aprendió previamente para compararlos y estimar la respuesta con un cierto grado de certeza.

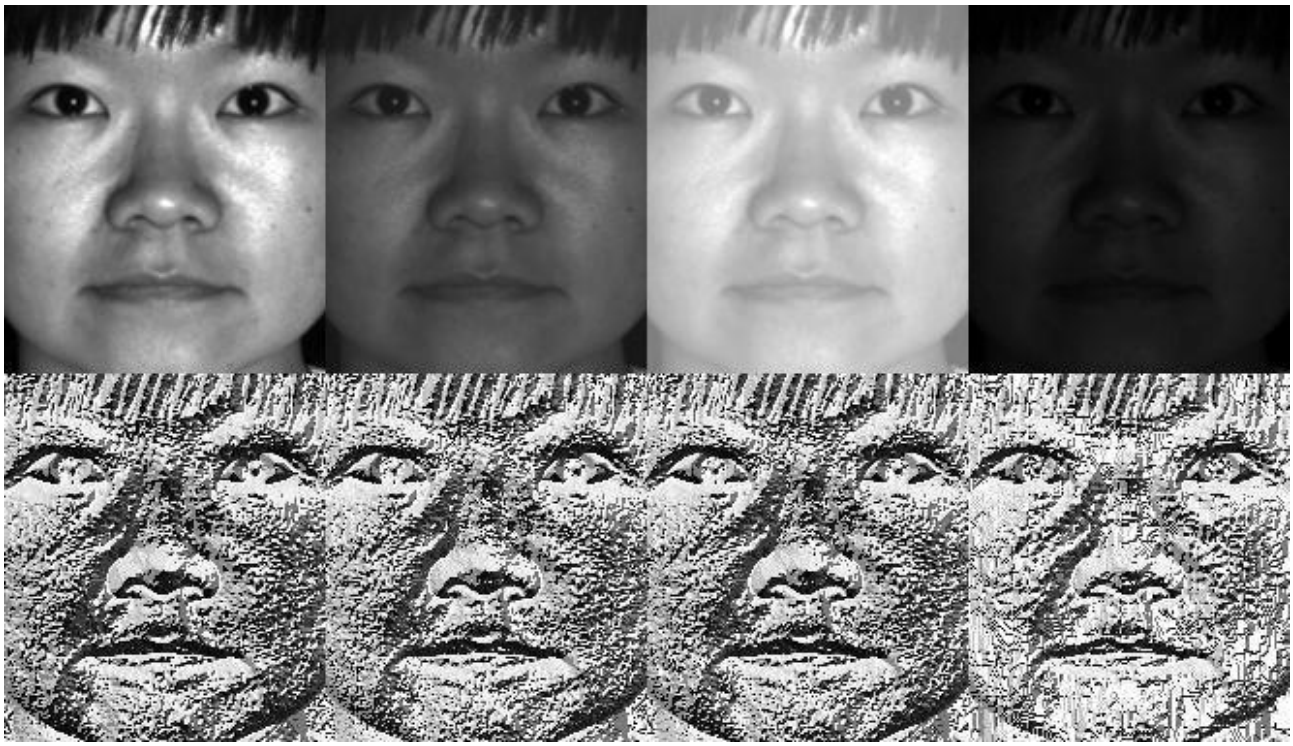
A continuación, se muestran ejemplos del procesamiento que realiza cada uno de estos algoritmos:



Eigenfaces



Fisherfaces



Local Binary Patterns Histrogram

3 APLICACIÓN DE IDENTIFICACIÓN

Para el desarrollo de la aplicación de reconocimiento e identificación de rostros, se ha elegido la plataforma Raspberry Pi 3, por su alta disponibilidad en el mercado, popularidad y potencia de cálculo (la cual es suficiente en su versión 3). Además, por contar con un módulo de cámara para la captura del video.
El lenguaje de programación de preferencia, ha sido Python 3.5 por su sencillez y disponibilidad de librerías para la manipulación de video haciendo uso del módulo de cámara de la Raspberry.
Como algoritmo de reconocimiento, se utilizó OpenCV 3.3.0 en su versión Python.

ESTRUCTURA DE ARCHIVOS Y DIRECTORIOS

- **/Caras/**: Almacena las imágenes tomadas por la cámara, que se utilizarán para entrenar al algoritmo de reconocimiento.
- **/FPS/**: Librería para utilizar el resto de los núcleos del procesador.
- **/haar/**: Archivos pre-entrenados para reconocer una cara dentro de una imagen.
- **/caripela_recognition.py**: Intenta reconocer e identificar una cara dentro del video. Si no es conocida, le toma fotos para usar en el algoritmo de entrenamiento.
- **/caripela_training.py**: Si hay fotos de una persona desconocida, las usa para entrenar al algoritmo de identificación.
- **/conocidos.xml**: Almacena las características de las caras pre-entrenadas que utilizará el algoritmo de identificación.
- **/conocidos.csv**: Almacena los nombres de las caras conocidas.

MODO DE USO – INICIALIZACIÓN

Solo debe ejecutarse una vez, al iniciar Raspbian. Esto inicia un ambiente virtual de python, el mismo ya tiene instaladas las librerías necesarias para el funcionamiento del proyecto.

- cd ~/Documents/Proyectos/caripela_detection/
- source ~/.profile
- workon cv3

MODO DE USO EN RECONOCIMIENTO

- python caripela_recognition.py

Este programa inicia la ventana de video solicitando que el usuario se ubique frente a la cámara para iniciar el reconocimiento. Solo tomará caras que tengan más de 250 px de ancho, por lo que el sujeto debe ubicarse relativamente cerca de la cámara.

Si reconoce una cara, dibuja un recuadro donde está ubicada, e intenta identificarla usando el archivo de caras pre-entrenadas. Si ya estaba entrenada, es decir es una cara conocida, muestra el nombre de la persona en el recuadro, junto con el índice de confianza en la predicción (un número menor indica más confianza).

Si no es una cara conocida, toma 20 fotos separadas por un número configurable de fotogramas, para luego utilizarlas en el programa de entrenamiento.

MODO DE USO EN ENTRENAMIENTO

- python caripela_training.py

Toma las fotos ubicadas en el directorio ***Caras***, y solicita al usuario que indique si se va a realizar una actualización del entrenamiento de una cara conocida, o se va a agregar una nueva cara. En este caso se solicitará el nombre de la persona a reconocer.

4 REFERENCIAS

[1] Viola-Jones. (https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework)
[2] OpenCV. (https://opencv.org/)
[3] Raspberry Pi. (https://www.raspberrypi.org/)
[4] GitHub del proyecto. (https://github.com/mabraidot)