CENTRO DE INGENIERÍA Y DESARROLLO INDUSTRIAL

ESPECIALIDAD DE TECNÓLOGO EN MECATRÓNICA

"RECONOCIMIENTO DE OBJETOS Y ROSTROS CON TÉCNICAS DE VISIÓN POR COMPUTADORA E INTELIGENCIA ARTIFICIAL"

Informe de la Práctica de Entrenamiento Industrial

Nombre de la Empresa o Institución:

CIDESI

Presenta:

Estudiante: Uriel Juárez Álvarez

Tutor Académico: Dr. Leonardo Barriga Rodríguez

Querétaro, Qro. Septiembre del 2018

AGRADECIMIENTOS

A la vida, por permitirme explorar y estudiar sus maravillas a lo largo de todos los años.

A mis padres y hermanos, quienes han sido el motivo más importante para luchar y alcanzar cada una de mis metas, y a los que siempre estaré agradecido por su incondicional amor y apoyo.

A mi novia que me ha acompañado estos últimos cinco años, siempre le estaré agradecido por brindarme su apoyo incondicional.

A mi asesor el Dr. Leonardo Barriga Rodríguez, quien me ha brindado una importante orientación académica para superar el desarrollo de este proyecto, y en lo personal la visión de mi futuro.

A CIDESI y CONACYT, por la oportunidad de estudiar la Especialidad de Tecnólogo en Mecatrónica.

Índice

Índice de figuras

1. Introducción

Hoy día encontramos inteligencia artificial en todas partes. No siempre tiene forma de robot, pero mucha de las veces funciona como uno. Existen algoritmos de reconocimiento facial muy robustos los cuales ya los podemos encontrar en diferentes dispositivos como celulares, computadoras etc.

Los seres humanos realizan reconocimiento de objetos y de rostros automáticamente todos los días sin hacer un esfuerzo. Para realizar esta tarea se necesita de un entrenamiento previo y ya después de un determinado tiempo de entrenamiento esta tarea resulta relativamente sencilla para el ser humano, pero para una computadora es una tarea compleja ya que existen diferentes variables que influyen, por ejemplo: variación de la iluminación, baja resolución, oclusión entre otras.

Al igual que los seres humanos, la computadora también necesita un entrenamiento previo para realizar esta tarea. En la Figura 1 se muestra un ejemplo de reconocimiento de rostros y objetos. En el presente trabajo se muestra el entrenamiento e implementación en una computadora personal utilizando el IDE Spyder para programar en python. Se implementan y comparan dos algoritmos de reconocimiento facial. También se muestra la implementación de la plataforma de Google Cloud en una Raspberry Pi para el reconocimiento de objetos.

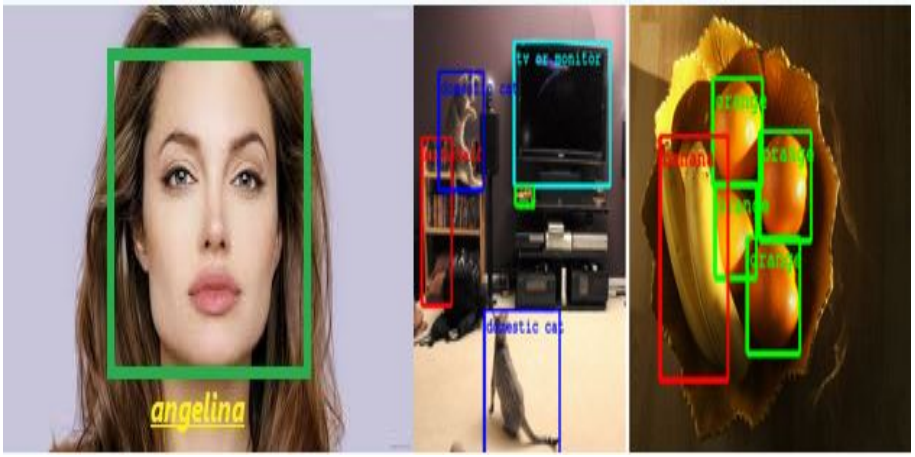

Figura 1: Reconocimiento de objetos y rostros. (C.M.A, 2014) & (Juan C. Sierra, 2014)

2. Planteamiento del problema

En CIDESI se están implementando algoritmos para el reconocimiento facial y de objetos, pero estos algoritmos tienen ventajas y desventajas dependiendo de la aplicación, por lo tanto es necesario realizar pruebas de los algoritmos de LBPH y FisherFaces para conocer su comportamiento

3. Justificación

En los últimos años, CIDESI ha venido incursionando en el desarrollo de algoritmos en la rama de la inteligencia artificial. Estos algoritmos tienen sus ventajas y desventajas dependiendo de su aplicación. Por tal manera se pretende hacer prueba de algunos algoritmos de reconocimiento facial y de reconocimiento de objetos con el fin de ampliar el estado del arte de CIDESI en cuanto a los algoritmos de reconocimiento facial y de objetos.

4. Objetivos

4.1. Objetivo general

El objetivo general es implementar y entrenar LBPH y FisherFaces, algoritmos para reconocimiento facial, además implementar la plataforma de Google Cloud en una Raspberry Pi para realizar la detección de objetos.

4.2. Objetivos específicos

- Buscar el estado del arte de LBPH y FisherFaces, alogoritmos de reconocimiento facial y también el estado del arte de Google Cloud.
- Generar una base de datos de imágenes.
- Extraer características de imágenes para generarar un dataset de entrenamiento.
- Ejecutar los algoritmos de reconocimiento.

5. Marco Teórico

5.1. Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Guido van Rossum, creador de Python, en la convención OSCON 2006 Python fue creado a finales de los ochenta por Guido van Rossum en el Centro para las Matemáticas y la Informática (CWI, Centrum Wiskunde y Informatica), en los Países Bajos, como un sucesor del lenguaje de programación ABC, capaz de manejar excepciones e interactuar con el sistema operativo Amoeba. (¨Python¨, 2018)

5.2. Anaconda (distribución de Python)

Anaconda es un distribución libre y abierta de los lenguajes Python y R, utilizada en ciencia de datos, y aprendizaje automático (machine learning). Esto incluye procesamiento de grandes volúmenes de información, análisis predictivo y cómputos científicos. Está orientado a simplificar el despliegue y administración de los paquetes de software. Las diferentes versiones de los paquetes se administran mediante el sistema de administración del paquete conda, el cual lo hace bastante sencillo de instalar, correr, y actualizar software de ciencia de datos y aprendizaje automático como ser Scikit-team, TensorFlow y SciPy. La distribución Anaconda es utilizada por 6 millones de usuarios e incluye más de 250 paquetes de ciencia de datos válidos para Windows, Linux y MacOS. (¨What is Anaconda? - Anaconda¨, 2018)

5.3. Machine Learning

El aprendizaje automático es un campo de la informática que utiliza técnicas estadísticas para dar a los sistemas informáticos la capacidad de .ªprender"(por ejemplo, mejorar progresivamente el rendimiento en una tarea específica) con datos , sin estar explícitamente programados. (¨Machine learning¨, 2018)

El nombre de aprendizaje automático fue acuñado en 1959 por Arthur Samuel . Desarrollado a partir del estudio de reconocimiento de patrones y teoría de aprendizaje computacional en inteligencia artificial , el aprendizaje automático explora el estudio y la construcción de algoritmos que pueden aprender y hacer predicciones sobre datos - dichos algoritmos se superan siguiendo un programa estrictamente estático instrucciones al hacer predicciones o decisiones basadas en datos, : mediante la construcción de un modelode entradas de muestra. El aprendizaje automático se emplea en una variedad de tareas informáticas en las que el diseño y la programación de algoritmos explícitos con un buen rendimiento es difícil o inviable; las aplicaciones de

ejemplo incluyen filtrado de correo electrónico , detección de intrusos de red y visión por computadora .(¨Machine learning¨, 2018)

5.4. Visión artificial

La visión artificial o visión por computador es una disciplina científica que incluye métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real con el fin de producir información numérica o simbólica para que puedan ser tratados por un computador. Tal y como los humanos usamos nuestros ojos y cerebros para comprender el mundo que nos rodea, la visión por computador trata de producir el mismo efecto para que las computadoras puedan percibir y comprender una imagen o secuencia de imágenes y actuar según convenga en una determinada situación. Esta comprensión se consigue gracias a distintos campos como la geometría, la estadística, la física y otras disciplinas. La adquisición de los datos se consigue por varios medios como secuencias de imágenes, vistas desde varias cámaras de video o datos multidimensionales desde un escáner médico. Esquema de las relaciones entre la visión por computadora y otras áreas afines. Hay muchas tecnologías que utilizan la visión por computador, entre las cuales se encuentran el reconocimiento de objetos, la detección de eventos, la reconstrucción de una escena (mapping) y la restauración de imágenes. (Shapiro & Stockman, 2001)

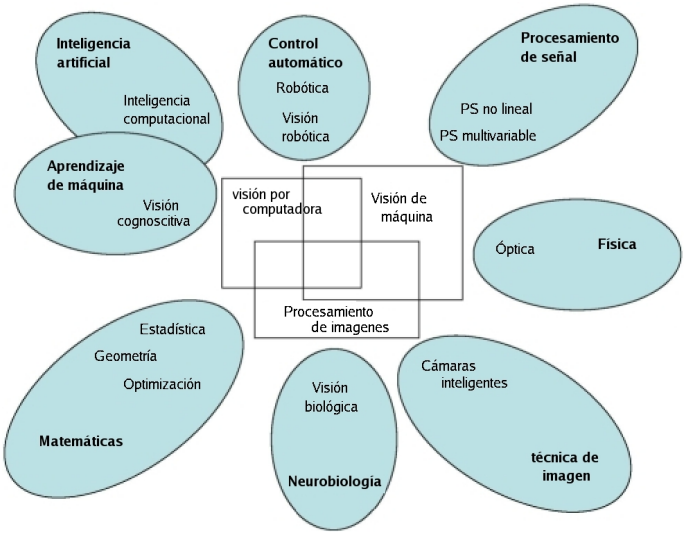

Figura 2: Esquema de las relaciones entre la visión por computadora y otras áreas afines. (Shapiro & Stockman, 2001)

En la Figura 1 se muestra la relacion que existe entre la visión por computadora con otras áreas afines.

5.5. OpenCV

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas. (¨OpenCV library¨, 2018)

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. (¨OpenCV library¨, 2018)

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. OpenCV puede además utilizar el sistema de primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel específicas para procesadores Intel. (¨OpenCV library¨, 2018)

5.6. Raspberry Pi 3

Raspberry Pi es un ordenador de placa reducida, ordenador de placa única u ordenador de placa simple (SBC) de bajo coste desarrollado en el Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de informática en las escuelas. Aunque no se indica expresamente si es hardware libre o con derechos de marca, en su web oficial explican que disponen de contratos de distribución y venta con dos empresas, pero al mismo tiempo cualquiera puede convertirse en revendedor o redistribuidor de las tarjetas RaspBerry Pi, por lo que da a entender que es un producto con propiedad registrada, manteniendo el control de la plataforma, pero permitiendo su uso libre tanto a nivel educativo como particular. En cambio, el software sí es de código abierto, siendo su sistema operativo oficial una versión adaptada de Debian, denominada Raspbian, aunque permite usar otros sistemas operativos, incluido una versión de Windows 10. En todas sus versiones incluye un procesador Broadcom, una memoria RAM, una GPU, puertos USB, HDMI, Ethernet (El primer modelo no lo tenía), 40 pines GPIO y un conector para cámara. Ninguna de sus ediciones incluye memoria, siendo esta en su primera versión una tarjeta SD y en ediciones posteriores una tarjeta MicroSD. (Raspberry Pi 3 Modelo B”, 2018)

5.6.1. Caracteristicas principales de Raspberry Pi 3

- Procesador:
  - Chipset Broadcom BCM2387.
  - 1,2 GHz de cuatro núcleos ARM Cortex-A53.

- GPU
  - Dual Core VideoCore IV ® Multimedia Co-procesador. Proporciona Open GL ES 2.0, OpenVG acelerado por hardware, y 1080p30 H.264 de alto perfil de decodificación.
  - Capaz de 1 Gpixel / s, 1.5Gtexel / s o 24 GFLOPs con el filtrado de texturas y la infraestructura DMA.
- RAM: 1GB LPDDR2.
- Conectividad.
  - Ethernet socket Ethernet 10/100 BaseT
  - 802.11 b / g / n LAN inalámbrica y Bluetooth 4.1 (Classic Bluetooth y LE)
  - Salida de vídeo
    - HDMI rev 1.3 y 1.4.
    - RCA compuesto (PAL y NTSC).
  - Salida de audio
    - jack de 3,5 mm de salida de audio, HDMI
    - USB 4 x Conector USB 2.0
  - Conector GPIO
    - 40-clavijas de 2,54 mm (100 milésimas de pulgada) de expansión: 2x20 tira
    - Proporcionar 27 pines GPIO, así como 3,3 V, +5 V y GND líneas de suministro
  - Conector de la cámara de 15 pines cámara MIPI interfaz en serie (CSI-2).

5.7. Cámara Raspberry Pi

El módulo cámara de 5 megapíxeles esta diseñado específicamente para Raspberry Pi, con un lente de foco fijo. Es capaz de tomar imágenes estáticas de 2592 x 1944, y también es compatible con el formato de video 1080p30, 720p60 y 640x480p60/90. Se conecta al Raspberry Pi por medio de un pequeño conector en la parte superior de la tarjeta y utiliza la interfaz dedicada CSi, diseñado especialmente para la conexión de cámaras. 5 megapíxeles de resolución nativa del sensor con capacidad de 2592 x 1944 píxeles. Soporta 1080p30, 720p60 y 640x480p60/90 en formatos de vídeo.

El propio sensor tiene una resolución nativa de 5 megapíxeles, y cuenta con un lente de foco fijo. En cuanto a las imágenes fijas, la cámara es capaz de tomar imágenes estáticas de 2592 x 1944 píxeles, y también es compatible con video de 1080p30, 720p60 y 640x480p60/90.

5.8. Detección de rostro

La detección de rostros tiene como objetivo encontrar los rostros (ubiación y tamaño) en una imagen y extraer esas imagnes y utilizarlas en un algoritmo de reconocimiento facial.

5.8.1. Detector de rostro Haar Cascade

Los clasificadores en cascada basados en características de Haar es un método efectivo de detección de objetos propuesto por Paul Viola y Michael Jones en su documento, ”Detección rápida de objetos usando una cascada potenciada de características simples.en 2001. Es un enfoque de aprendizaje automático donde la función de cascada está entrenada a partir de muchas imágenes positivas y negativas. Luego se usa para detectar objetos en otras imágenes.
Inicialmente, el algoritmo necesita muchas imágenes positivas (imágenes de caras) e imágenes negativas (imágenes sin caras) para entrenar al clasificador. Por lo que se necesita extraer caracteristicas de él. Para esto, se utilizan las características de haar que se muestran en la Figura 3. Son como un kernel convolucional. Cada característica es un valor único que se obtiene al restar la suma de píxeles en el rectángulo blanco de la suma de píxeles en el rectángulo negro. (Harvey, 2018)

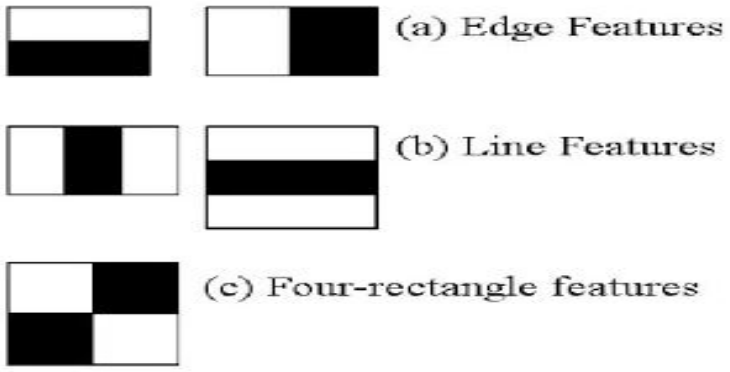

Figura 3: Características de haar. (Harvey, 2018)

Pero entre todas estas características que calcula, la mayoría de ellas son irrelevantes. Por ejemplo, se considera la Figura 4. La fila superior muestra dos buenas características. La primera característica seleccionada parece enfocarse en la propiedad de que la región de los ojos a menudo es más oscura que la región de la nariz y las mejillas. La segunda característica seleccionada se basa en la propiedad de que los ojos son más oscuros que el puente de la nariz. Pero las mismas ventanas que se aplican en las mejillas o en cualquier otro lugar son irrelevantes. Entonces, ¿cómo se seleccionan las mejores características de 160000+ características?. Para esto, se

Figura 4: Características de imagen. (Harvey, 2018)

aplica en todas y cada una de las características en todas las imágenes de entrenamiento. Para cada característica, encuentra el mejor umbral que clasificará las caras en positivas y negativas. Pero, obviamente, habrá errores o clasificaciones erróneas. Seleccionamos las características con una tasa de error mínima, lo que significa que son las características que mejor clasifican la cara y las imágenes que no son caras. (El proceso no es tan simple como esto. Cada imagen recibe el mismo peso al principio. Después de cada clasificación, se aumentan los pesos de las imágenes mal clasificadas. Luego, se repite el mismo proceso. Se calculan nuevas tasas de error. También se generan nuevos pesos. el proceso continúa hasta que se alcanza la precisión requerida o la tasa de error o se encuentra el número requerido de características).

El clasificador final es una suma ponderada de estos clasificadores débiles. Se llama débil porque solo no puede clasificar la imagen, pero junto con otros forma un clasificador fuerte. El documento dice que incluso 200 características proporcionan detección con 95 % de precisión. Su configuración final tenía alrededor de 6000 características. (Imagine una reducción de 160000+ características a 6000 características. Eso es una gran ganancia). (Harvey, 2018)

5.9. Reconocimiento facial

Ya teniendo imágenes faciales ya extraídas, recortadas, redimensionadas y generalmente convertidas a escala de grises, el algoritmo de reconocimiento facial es responsable de encontrar las características que mejor describan la imagen.

5.9.1. Local Binary Pattern

Local Binary Pattern (LBP) es un operador de textura simple pero muy eficiente que etiqueta los píxeles de una imagen mediante el umbral de la vecindad de cada píxel y considera el resultado como un número binario. Debido a su poder discriminativo y simplicidad computacional, el operador de textura LBP se ha convertido en un enfoque popular en diversas aplicaciones. Se puede ver como un enfoque unificador a los modelos estadísticos y estructurales tradicionalmente divergentes del análisis de textura. Quizás la propiedad más importante del operador de LBP en aplicaciones del mundo real es su robustez frente a los cambios monótonos de la escala de grises causados, por ejemplo, por las variaciones de iluminación. Otra propiedad importante es su simplicidad computacional, que hace posible analizar imágenes en entornos desafiantes en tiempo real. Usando el LBP combinado con histogramas podemos representar las imágenes de la cara con un vector de datos simple. Como LBP es un descriptor visual, también se puede usar para tareas de reconocimiento facial, como se puede ver en la siguiente explicación paso a paso.(Ahonen, Hadid & Pietikainen, 2006)

5.9.2. Como funcion Local Binary Pattern

Pasos del algoritmo:
1. Parámetros : el LBPH usa 4 parámetros:

- Radio : el radio se usa para construir el patrón binario local circular y representa el radio alrededor del píxel central. Por lo general, se establece en 1.
- Vecinos : el número de puntos de muestra para construir el patrón binario local circular. Tenga en cuenta que cuantos más puntos de muestra incluya, mayor será el costo computacional. Por lo general, se establece en 8.
- Cuadrícula X : el número de celdas en la dirección horizontal. Cuantas más células, más fina es la cuadrícula, mayor es la dimensionalidad del vector de características resultante. Por lo general, se establece en 8.
- Cuadrícula Y : la cantidad de celdas en la dirección vertical. Cuantas más células, más fina es la cuadrícula, mayor es la dimensionalidad del vector de características resultante. Por lo general, se establece en 8.

2. Entrenando el algoritmo : Primero, necesitamos entrenar el algoritmo. Para hacerlo, necesitamos usar un conjunto de datos con las imágenes faciales de las personas que queremos reconocer. También necesitamos establecer una ID (puede ser un número o el nombre de la persona) para cada imagen, por lo que el algoritmo utilizará esta información para reconocer una imagen de entrada y darle una salida. Las imágenes de la misma persona deben tener la misma identificación. Con el conjunto de entrenamiento ya construido, veamos los pasos computacionales de LBPH.

3. Aplicación de la operación de LBP : el primer paso computacional del LBPH es crear una imagen intermedia que describa la imagen original de una mejor manera,

resaltando las características faciales. Para hacerlo, el algoritmo usa un concepto de ventana deslizante, basado en los parámetros radio y vecinos .

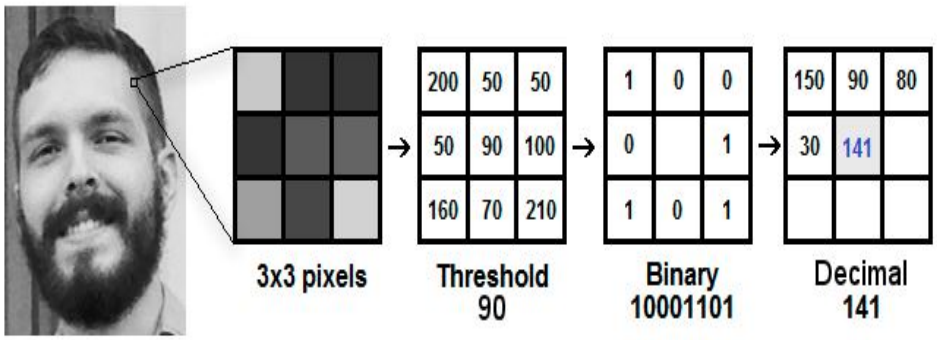

Figura 5: Procedimiento inicial LBP. (Ahonen, Hadid & Pietikainen, 2006)

Basándonos en la imagen de la Figura 5, dividámosla en varios pequeños pasos para que podamos entenderla fácilmente:

- Supongamos que tenemos una imagen facial en escala de grises.
- Podemos obtener parte de esta imagen como una ventana de 3x3 píxeles.
- También se puede representar como una matriz de 3x3 que contiene la intensidad de cada píxel (0 - 255).
- Entonces, necesitamos tomar el valor central de la matriz para usar como umbral.
- Este valor se usará para definir los nuevos valores de los 8 vecinos.
- Para cada vecino del valor central (umbral), establecemos un nuevo valor binario. Establecemos 1 para valores iguales o superiores al umbral y 0 para valores inferiores al umbral.
- Ahora, la matriz contendrá solo valores binarios (ignorando el valor central). Necesitamos concatenar cada valor binario de cada posición de la matriz línea por línea en un nuevo valor binario (por ejemplo, 10001101). Nota: algunos autores utilizan otros enfoques para concatenar los valores binarios (por ejemplo, sentido horario), pero el resultado final será el mismo.
- Luego, convertimos este valor binario en un valor decimal y lo establecemos en el valor central de la matriz, que en realidad es un píxel de la imagen original.
- Al final de este procedimiento (procedimiento LBP), tenemos una nueva imagen que representa mejor las características de la imagen original.
- Nota : El procedimiento de LBP se amplió para usar un número diferente de radio y vecinos, se llama Circular LBP.

Para este algoritmo se puede variar el Parametro ¨P¨ que es el numero de vecinos y también ¨R¨ que es el radio como se observa en la Figura 6. Se puede hacer mediante

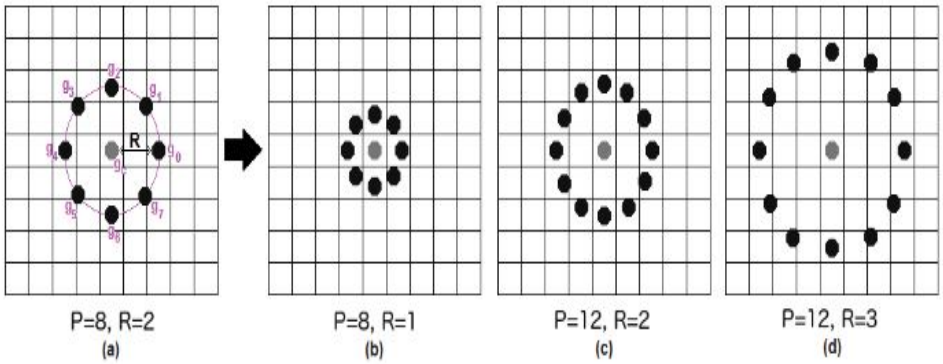

Figura 6: Variación del parametro P. (Ahonen, Hadid & Pietikainen, 2006)

la interpolación bilineal . Si algún punto de datos está entre los píxeles, utiliza los valores de los 4 píxeles más cercanos (2x2) para estimar el valor del nuevo punto de datos.

4. Extracción de los histogramas : ahora, usando la imagen generada en el último paso, podemos usar los parámetros Grid X y Grid Y para dividir la imagen en múltiples cuadrículas, como se puede ver en la Figura 7:

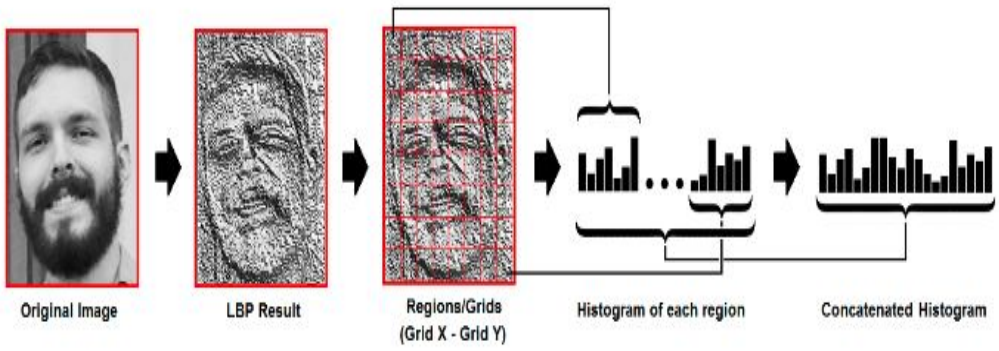

Figura 7: Concatenación de histogramas. (Ahonen, Hadid & Pietikainen, 2006)

Según la Figura 7, podemos extraer el histograma de cada región de la siguiente manera:

- Como tenemos una imagen en escala de grises, cada histograma (de cada cuadrícula) contendrá solo 256 posiciones (0 - 255) que representan las ocurrencias de cada intensidad de píxel.
- Luego, debemos concatenar cada histograma para crear un histograma nuevo y más grande. Supongamos que tenemos rejillas de 8x8, tendremos 8x8x256

= 16.384 posiciones en el histograma final. El histograma final representa las características de la imagen original de la imagen.

5. Realización del reconocimiento facial : en este paso, el algoritmo ya está entrenado. Cada histograma creado se usa para representar cada imagen del conjunto de datos de entrenamiento. Entonces, dada una imagen de entrada, realizamos los pasos nuevamente para esta nueva imagen y creamos un histograma que representa la imagen.

- Entonces, para encontrar la imagen que coincida con la imagen de entrada, solo tenemos que comparar dos histogramas y devolver la imagen con el histograma más cercano.
- Podemos usar varios enfoques para comparar los histogramas (calcular la distancia entre dos histogramas), por ejemplo: distancia euclídea , chi-cuadrado , valor absoluto , etc. En este ejemplo, podemos usar la distancia euclidiana (que es bastante conocida) basada en la siguiente fórmula:

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)} \tag{1}$$

- Entonces la salida del algoritmo es la ID de la imagen con el histograma más cercano. El algoritmo también debe devolver la distancia calculada, que puede usarse como una medida de çonfianza ". Nota : no se deje engañar por el nombre de 'confianza', ya que las confidencias más bajas son mejores porque significa que la distancia entre los dos histogramas es más cercana.
- Entonces podemos usar un umbral y la 'confianza' para estimar automáticamente si el algoritmo ha reconocido correctamente la imagen. Podemos suponer que el algoritmo ha reconocido con éxito si la confianza es menor que el umbral definido.

5.9.3. Fisherfaces

El Fisherface es un método que se encarga del reconocimiento de caras, teniendo en cuenta como se refleja la luz y las expresiones faciales. Éste algoritmo maximiza la relación entre la distribución de las clases y la distribución intra-clases. Fisherface clasifica y reduce la dimensión de las caras utilizando el método Discriminante Lineal de Fisher(FLD) y PCA (conocido como Eigenfaces). Éste método crea una proyección lineal que maximiza las diferentes imágenes de caras proyectadas.(Martinez et al., 2018)
Para calcular los Fisherfaces se define la matriz de distribución entre clases (S_B) y la matriz de distribución intra clases (S_W). u_i es la imagen resultante de X_i (conjunto de N imágenes) y N_i es el número de imágenes de X_i. donde:

$$S_B = \sum_{i=1}^{c} N_i(u_i - u)(u_i - u)^T \tag{2}$$

$$S_W = \sum_{i=1}^{c} \sum_{X_k \in X} N_i (x_k - u)(x_k - u)^T \tag{3}$$

W_{opt} es una matriz ortonormal que maximiza la relación de la matriz de distribución entre clases de imágenes proyectadas y las futuras proyecciones que habrá dentro de éste mismo determinante.

$$W_{opt} = \underset{w}{argmax} \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|} = \left[W_1 W_2 \ddots W_m\right] \tag{4}$$

La siguiente matriz, tiene c-1 valores propios diferentes de cero. Por lo tanto, el límite superior de m es c-1 (c es el número de clases).

$$S_B w_i = S_W w_i Y_i \tag{5}$$

En cuanto al reconocimiento de caras la matriz S_W es N-c (donde N es el número de imágenes de entrenamiento).? Por lo tanto, se puede escoger una matriz W en la que la distribución intra-clases de las imágenes proyectadas pueda ser cero. Utilizando el método PCA se reduce la dimensión del espacio de características a N-c. Seguidamente aplicamos la técnica FLD, que reduce la dimensión a c-1.(Martinez et al., 2018)

$$W_{pca}^T = W_{fld}^T W_{pca}^T \tag{6}$$

$$W_{pca}^T = \underset{w}{argmax} \left|W^T S_T W\right| \tag{7}$$

$$W_{fld} = \underset{w}{argmax} \frac{\left|W^T W_{pca}^T S_B W_{pca} W\right|}{\left|W^T W_{pca}^T S_W W_{pca} W\right|} \tag{8}$$

En la Figura 8 se muestra un ejemplo de un conjunto de imagenes donde se ejecuto el procedimiento anterior.

Figura 8: Conjunto de imagenes FisherFaces. (Martinez et al., 2018)

5.10. Google Cloud

La API Prediction de Google ofrece capacidades de coincidencia de patrones y aprendizaje automático. Después del aprendizaje con los datos de preparación, API Prediction es capaz de predecir un valor numérico o seleccionar una categoría que describa un nuevo fragmento de datos. Estas capacidades te permiten crear aplicaciones para realizar tareas como predecir las películas o los productos que podrían gustarle a un usuario, determinar si un correo electrónico es spam, valorar si los comentarios publicados son positivos o negativos, o bien estimar lo que podría gastar un usuario un día específico.
La mayoría de consultas de predicción tardan menos de 200 ms. Puede obtenerse un rendimiento superior. Los datos se replican en varios centros de datos a través de Google Cloud Storage.

5.10.1. Características de API Prediction

Aprendizaje automático para analizar datos y hacer predicciones

- Aprendizaje con datos de usuario: Reúne un conjunto de datos de preparación para crear un modelo predictivo propio adaptado a las necesidades del usuario.
- Estimación o clasificación: Los servicios de Google estan diseñados para colaborar. Usa los resultados de Google BigQuery o importa conjuntos de datos de hasta 2,5 GB de Cloud Storage para crear un modelo predictivo. Además, se puede utilizar la API Prediction directamente desde App Engine.
- Actualización del modelo conforme a los datos de preparación.
- Acceso a la API: La API RESTful está disponible a través de bibliotecas de muchos lenguajes conocidos, como Python, JavaScript y .NET.
- Complemento Smart Autofill de Hojas de Cálculo: Smart Autofill (función autocompletar inteligente) es un complemento de las Hojas de Cálculo de Google que usa la API Prediction para aprovechar el aprendizaje automático directamente en las hojas de cálculo.

5.11. Marco de referencía

- D. C. Gallardo (2015) Realizo reconocimiento facial orientado a acceso de sistemas de seguridad utilizando principalmente una Raspberry Pi. Para la detección de rostros utilizó el algoritmo de Viola-Jones y para el reconocimiento utilizó el algoritmo de EigenFaces.

- E.X. Sánchez & E. S. Gallardo (2010) Realizó reconocimiento facial con Raspberry Pi utilizando el algoritmo de reconocimiento LBPH despues evalúo el porcentaje de error que este tiene.

- A. E. López, Cyntia Mendazo, L. A. Reyes, E. A. Rivas, J.M. Ramos & J.C. Pedraza (2015) En este trabajo se desarrolló un sistema de autenticación facial mediante un algoritmo PCA modificado utilizando un sistema embebido con arquitectura ARM. El lenguaje utilizado fue C++ y Python.

- M.F. Perilla (2016) Realizó una investigacíon en cuanto a la efectividad, consumo y otros factores relevantes de los distintos algoritmos para el reconocimiento facial enfocado a la seguridad. Según sus resultados en base a una serie de pruebas como lo son: rendimiento, tiempos de ejecución, ID positivos, falsos positivos y falsos negativos. Los algoritmos que tienen menores tiempos de ejecución y una buena efectividad son los algoritmos Kernel PCA y Eigenfaces.

- Rubén Palacios (2018) En este trabajo se realizó reconocimiento de imagenes mediante el hardware Raspberry Pi y tensorflow como plataforma de entrenamiento. Cabe mencionar que se realizó una base de datos bastante pequeña por lo cual los recursos computacionales de la Raspberry Pi fueron suficientes.

6. Metodología

En la Figura 9 se muestra la metodología que se siguio para realizar el reconocimiento facial.

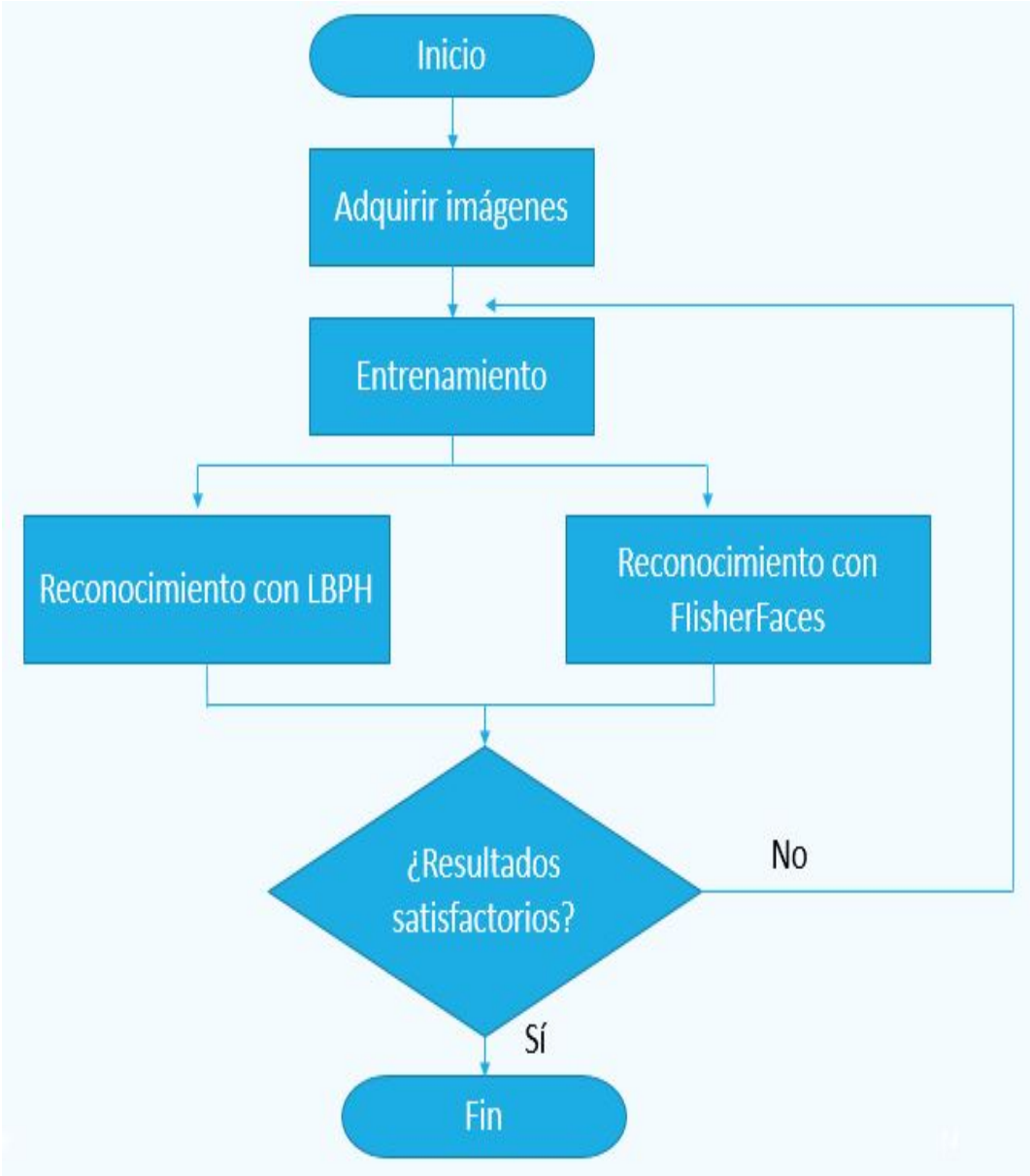

Figura 9: Metodología propuesta.

7. Resultados

7.1. Reconocimiento facial con LBPH y FisherFaces

Para el reconocimiento facial se utilizó el IDE Syder de la plataforma de Anaconda. Las librerias que se utilizarón principalmente fuerón las siguientes:

- Numpy: es una extensión de Python para funciones matemáticas de alto nivel para operar con vectores o matrices.
- cv2: es la extensión de openCV que contiene herramientas para vision artificial.
- PIL (Python Imaging Library): esta extension ofrece soporte en el manejo de imagenes.
- os: es un modulo que permite acceder a funcionalidades dependientes del sistema operativo. Sobre todo permite manipular los directorios.

7.1.1. Identificación de rostro

Para realizar la detección del rostro se utilizó un algoritmo de detección haarcascade. En la Figura 10 se muestra algunas pruebas realizadas.

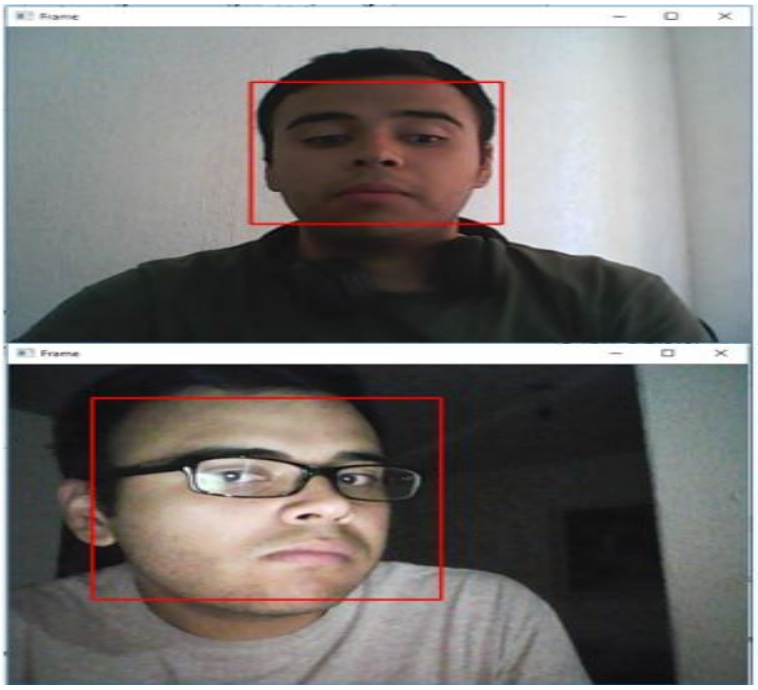

Figura 10: Detección de rostro.

En la Figura 11 se muestra el código realizado el cual tiene como principal función detectar el rostro y guardar un cierto numero de rostros con ID definido por el usuario.

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jun  1 20:18:19 2018
@author: Uriel Juárez Álvarez
"""
import cv2

faceDetect=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cam=cv2.VideoCapture(1)
id=input('Introduce nombre de usuario: ')
muestra=65
while (True):
    ret,img=cam.read();
    if ret is True:
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    else:
        break
    faces=faceDetect.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5, minSize=(100, 100),
flags=cv2.CASCADE_SCALE_IMAGE)
    for (x,y,w,h) in faces:
        muestra=muestra+1
        cv2.rectangle(img,(x,y),(x+w,y+w),(0,0,255),2)
        cv2.imwrite('DataSet/usuario.'+str(id)+'.'+str(muestra)+'.jpg',gray[y:y+h,x:x+w])
        cv2.waitKey(1000)
    cv2.imshow("Frame",img);
    cv2.waitKey(1000)
    if muestra > 85:
        break
cam.release()
cv2.destroyAllWindows()
```

Figura 11: Código para generar base datos (imagenes de rostros).

Con el código anterior se generó el conjunto de imagenes mostrado en la Figura 12, este conjunto fue exclusivamente utilizado para entrenar el algoritmo de LBPH.

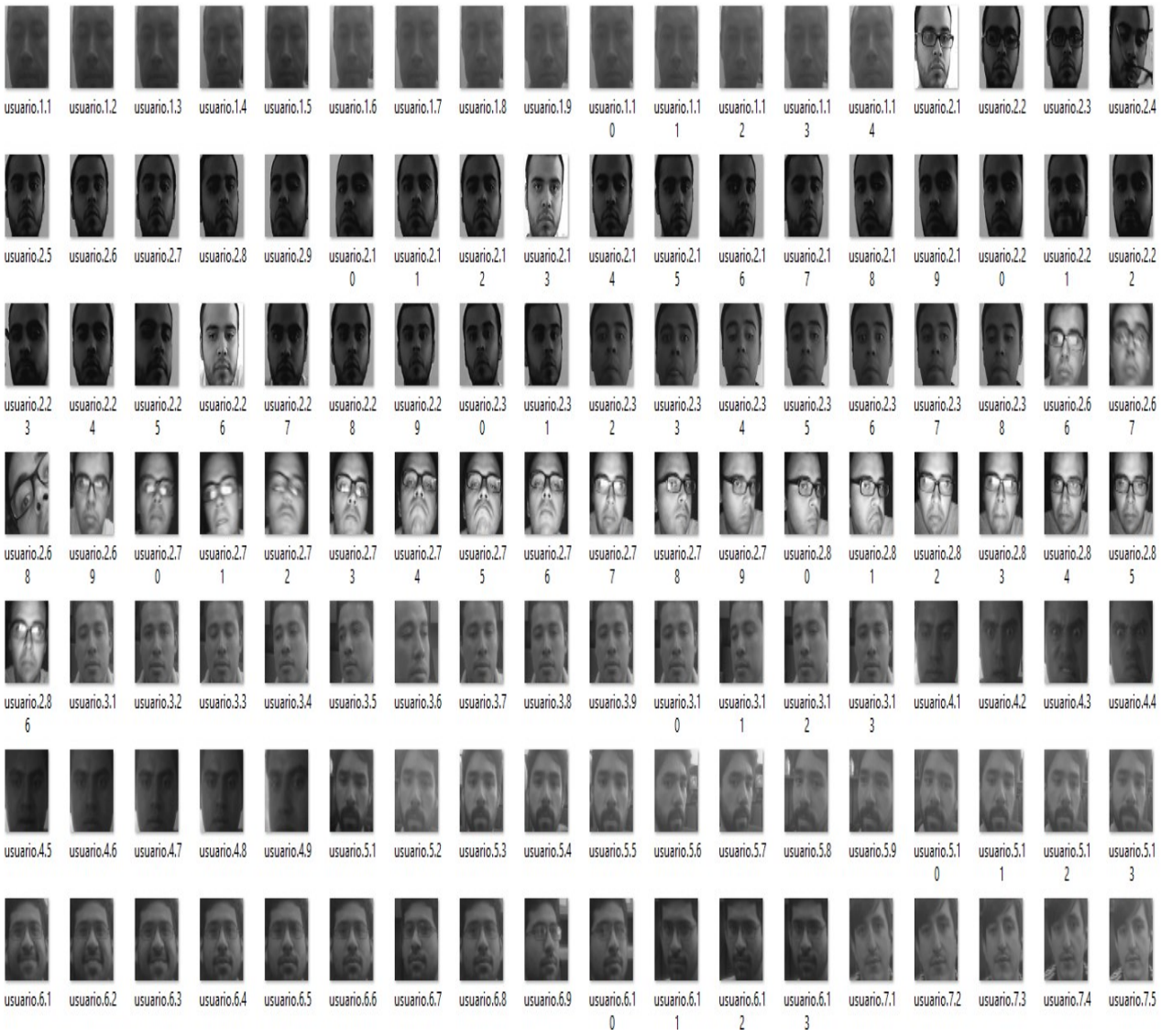

Figura 12: Rostros de compañeros de especialidad.

En la Figura 12 se muestra el conjunto de imagenes que se utilizo para entrenar el algoritmo de FisherFaces, se puede observar que tiene menor numero de muestras que el otro conjunto.

Figura 13: Seguno conjunto de rostros de compañeros de especialidad.

7.1.2. Entrenamiento

Para el entrenamiento se utilizó el conjunto de imagenes mostrado anteriormente (Figura 12 y 13). Se extraen las caracteristicas principales de cada imagen y tambien se extrae el ID correspondiente de cada imagen. En la Figura 14 se muestra el código realizado para el entrenamiento de reconocimiento facial. Este código es practicamente el mismo para realizar el entrenamiento a LBPH y FisherFaces. Con esto se obtiene un archivo (trainingData.yml), al tener este archivo ya es posible realizar el reconocimiento facial.

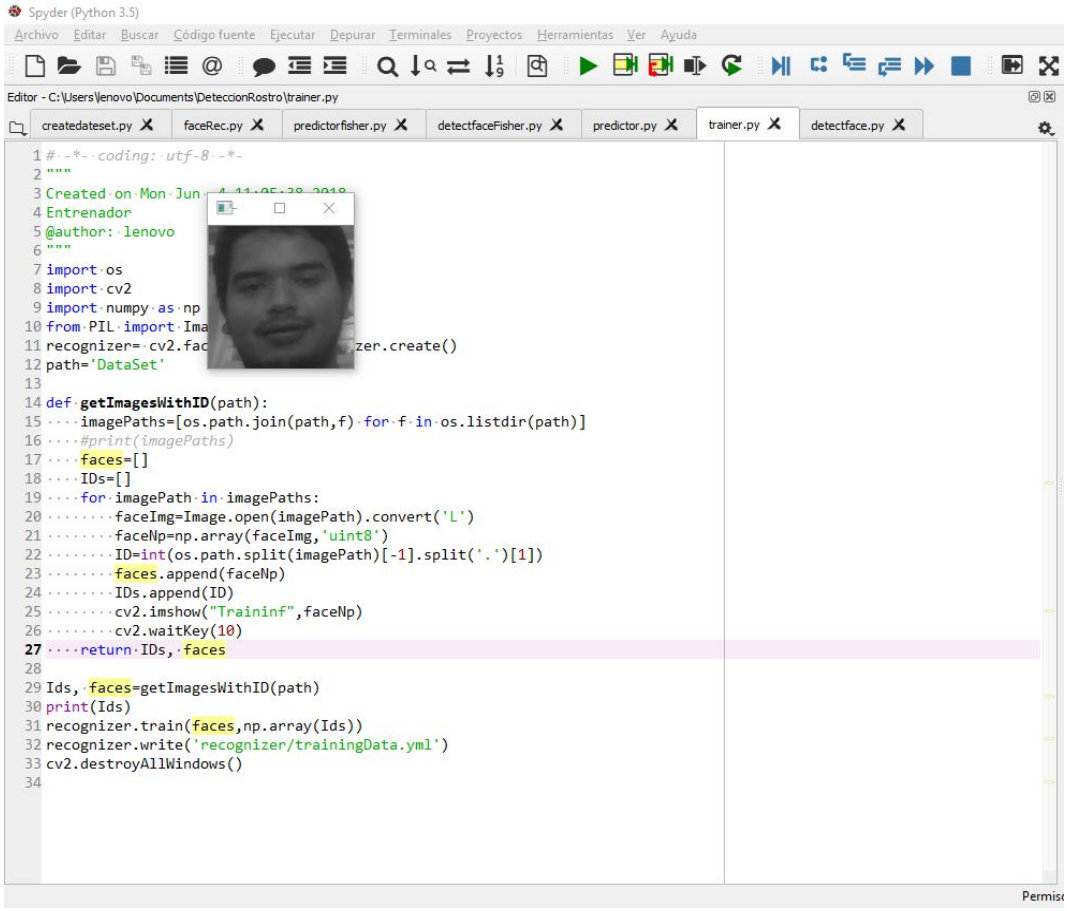

Figura 14: Código para detectar rostro utilizando un algoritmo haarcascade.

7.1.3. Reconocimiento

En la Figura 15 se muestra el algoritmo de FisherFaces en ejecución, se experimento con diferente intensidad luz. En los cuatro experimentos arrojo el ID correcto y esto fue con un numero menor de muestras de entrenamiento.

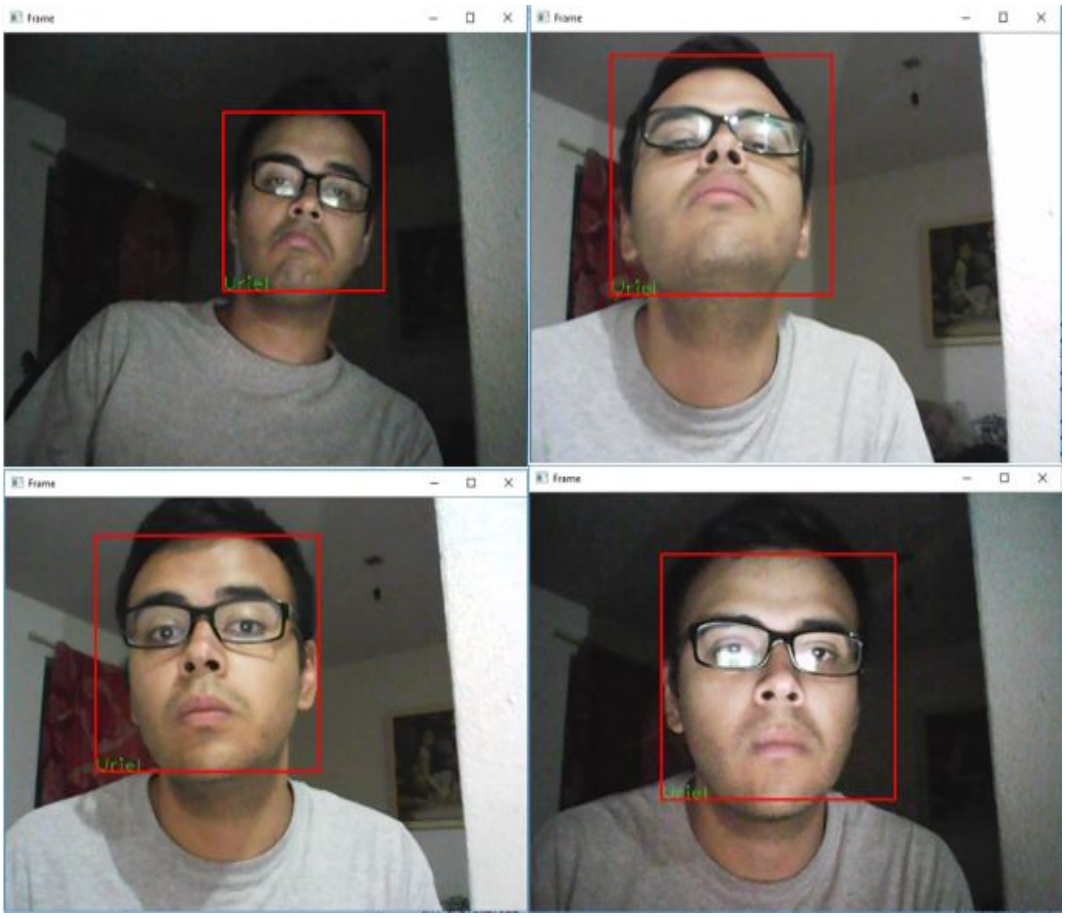

Figura 15: Reconocimiento facial con el algoritmo FisherFaces.

En la Figura 16 se muestra el algoritmo LBPH corriendo, al igual que el algoritmo de FisherFaces se hicieron cuatro experimentos con diferentes intensidades de luz. Se puede puede observar que aqui arrojo un ID incorrecto. Cabe mencionar que para este algoritmo se utiliza un numero mayor de muestras en comparación al de FisherFaces.

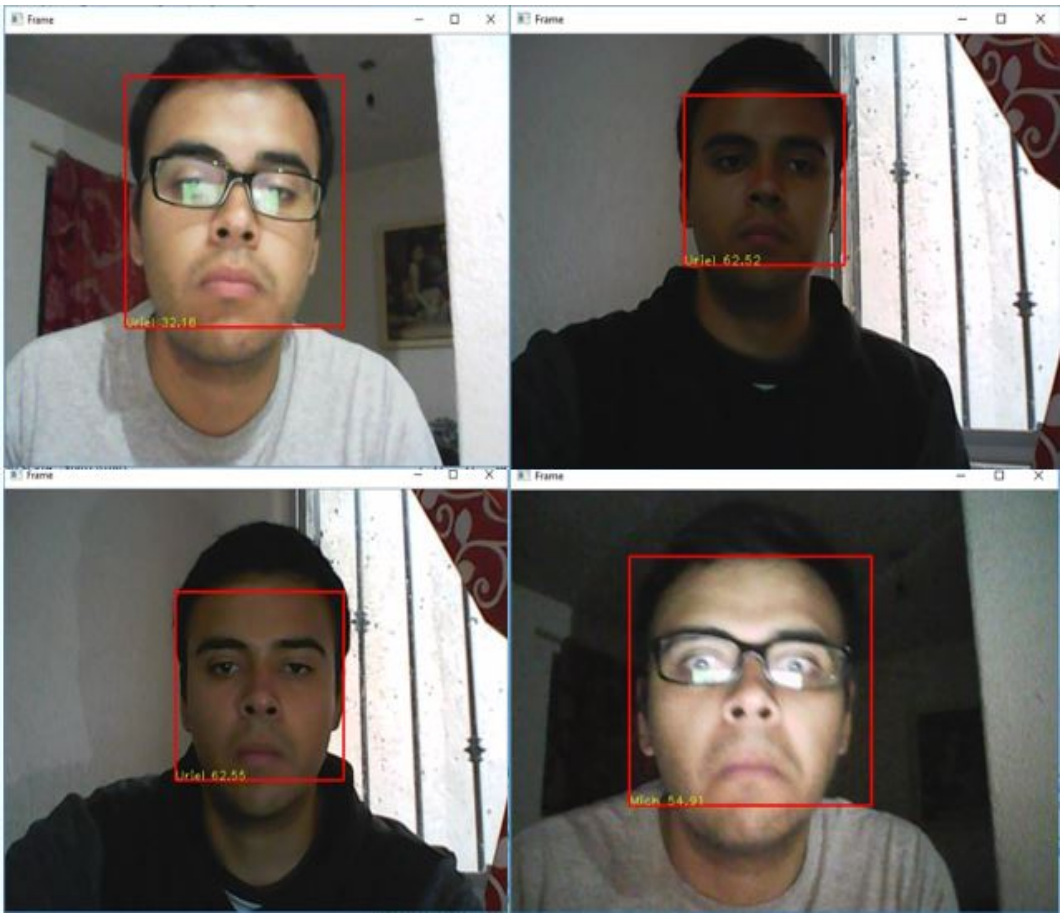

Figura 16: Reconocimiento facial con el algoritmo LBPH.

7.2. Reconocimiento de objetos con GoogleCloud y Raspberry Pi 3

La API Prediction de Google ofrece capacidades de coincidencia de patrones y aprendizaje automático. Después de aprendizaje con los datos de preparación, API Prediction es capaz de predecir un valor numérico o seleccionar una categoría que describa un nuevo fragmento de datos.

Así es como funciona:

Figura 17: Funcionamiento de API prediction.

El codigó de la Figura 18 lo ofrece la plataforma de GoogleCloud, este solo se modifico para para las etiquetas. Aquí el usuario realiza una captura de imagen, despues hace una consulta a la plataforma y esta regresa un archivo 'JSON' el cual tiene una lista de los objetos detectados con un porcentaje de confiabilidad.

Se observa en la Figura 19 una consulta realizada.

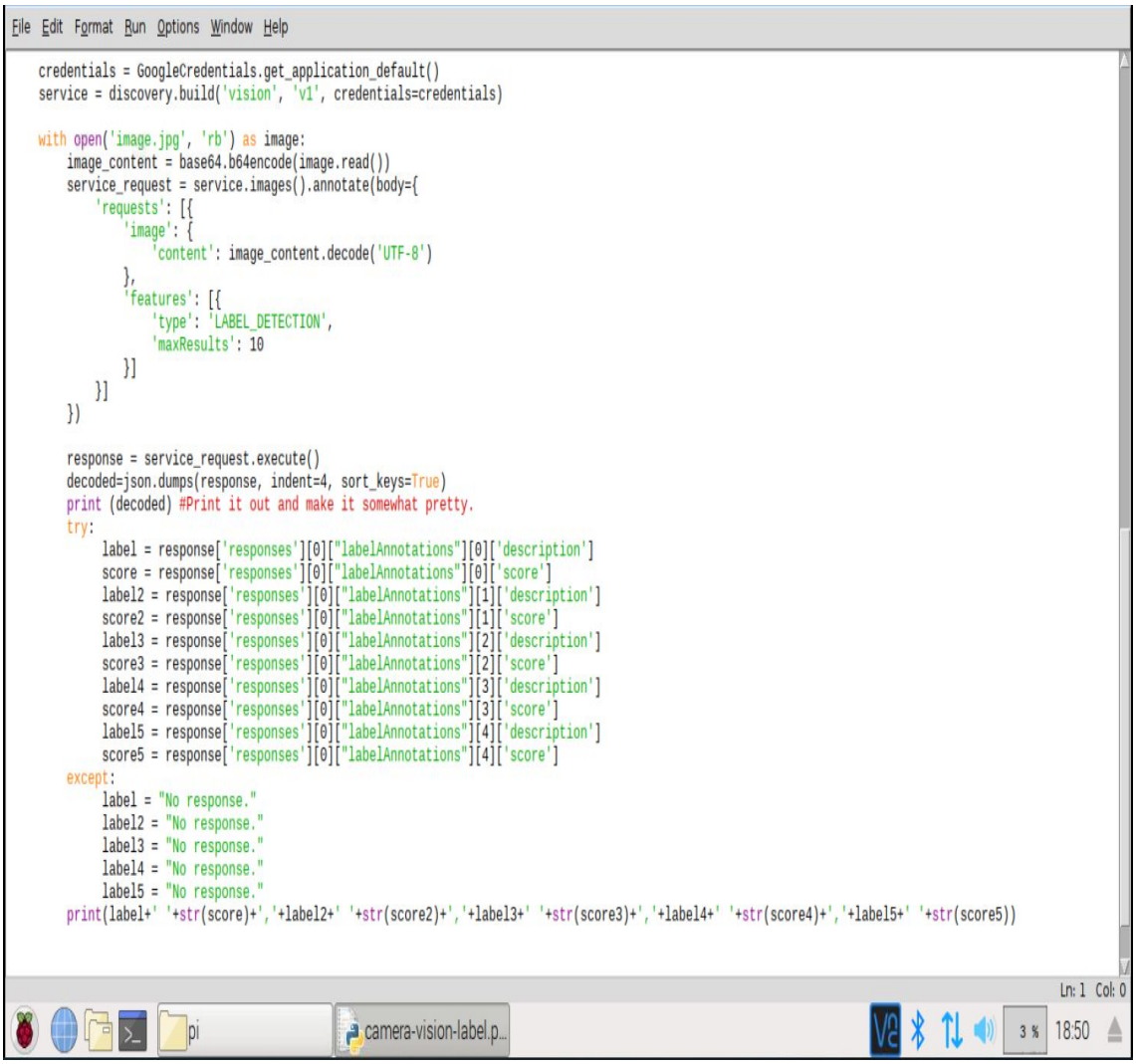

Figura 18: Codigó 'Detect Label'.

Nos dice que es un producto con una certeza del 76.66 % y en segundo lugar dice que es un dispositivo electronico con 75.18 %.

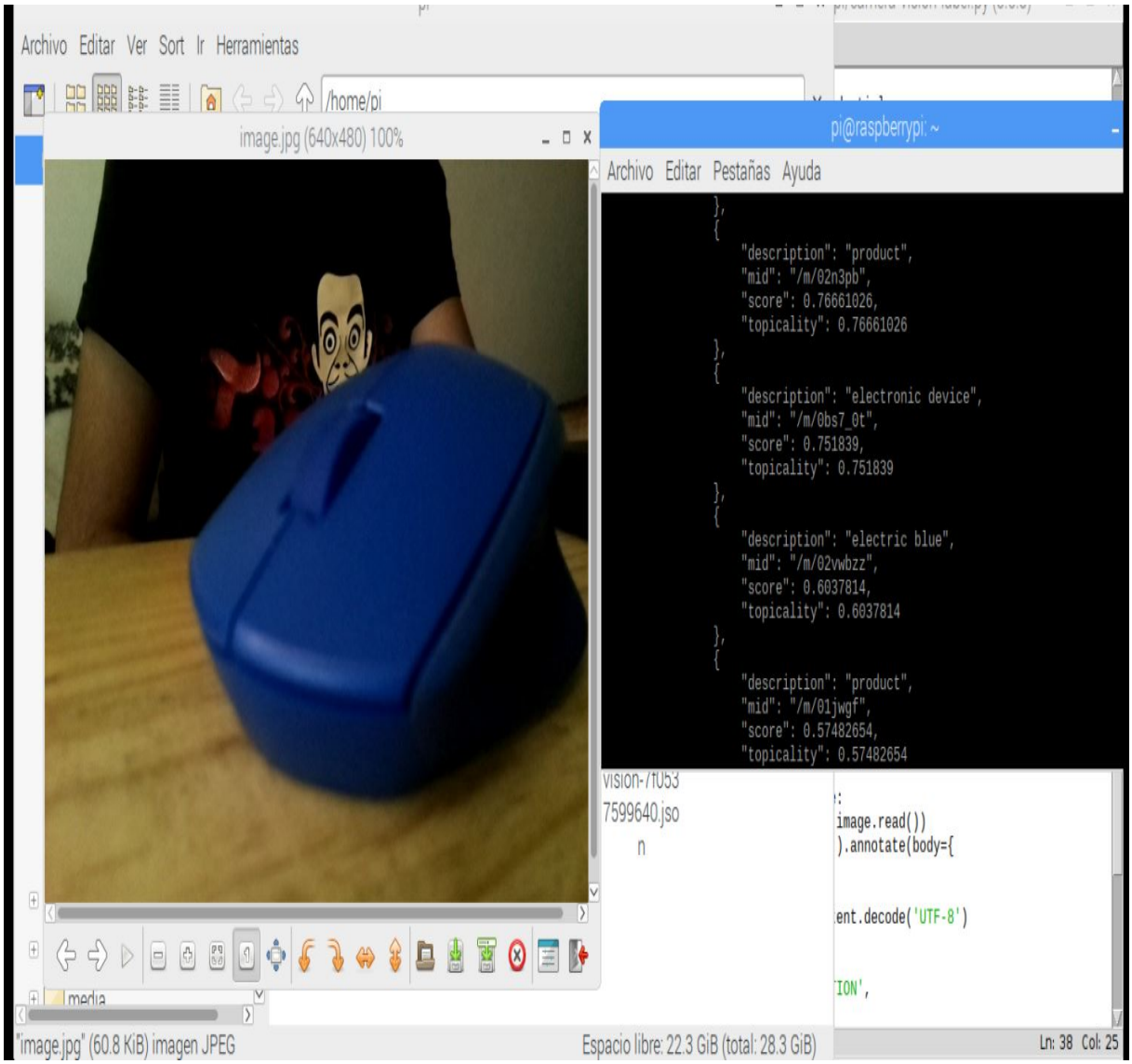

Figura 19: Primera prueba con el código ’Detect Label’.

En esta prueba realizada de la Figura 20 arrojo un resultado más erroneo. Pero se puede observar que detecto la boca del personaje con una certeza de 69.40 % pero también detecto un personaje ficticio con 60.88 % y ya que no había iluminación, detecto que era de noche con un 58.20 %.

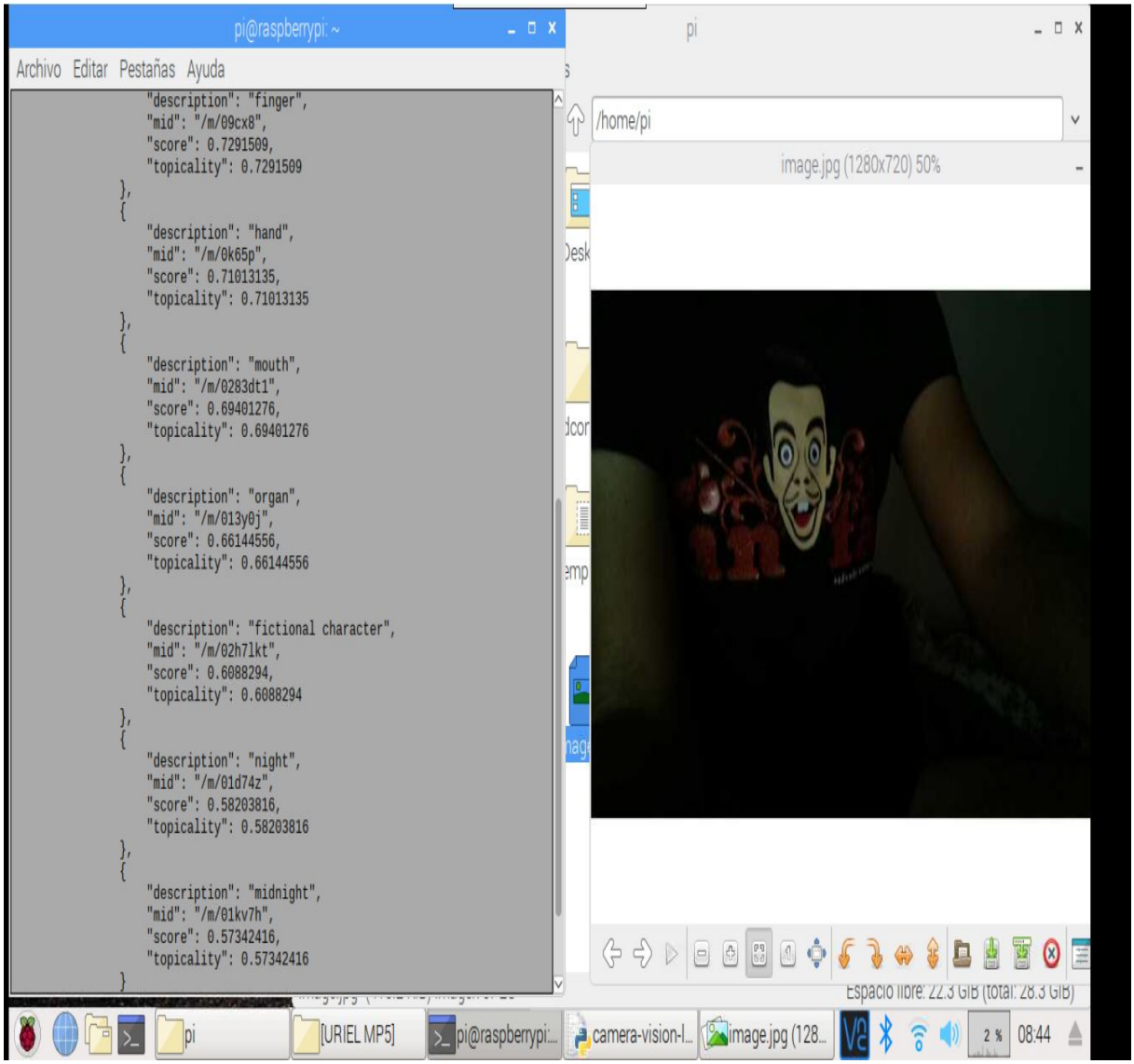

Figura 20: Segunda prueba con el código 'Detect Label'.

8. Actividades adicionales

8.1. Programa para semi-automatizar un proceso de calibración de acelerometros con LabVIEW y una NI-DAQ

Además de lo realizado anteriormente se colaboró con el departamento de metrología en el laboratorio de vibraciones. Donde se realiza la calibración de acelerómetros triaxiales (3 ejes).

Los vibradores electródinámicos (shakers, shock and vibration exciters), son equipos que generan vibraciones para el análisis o ensayos de vibraciones en componentes, piezas y sistemas discretos. Dependiendo del nivel de vibración de cada equipo, existen modelos de rangos altos, que permiten generar fuerzas dinámicas importantes

para el análisis modal en grandes estructuras y ver así la respuesta de dicha estructura a diferentes frecuencias. Estos equipos se utilizan principalmente para para el análisis sísmico, dinámica estructural, mecánica estructural, busqueda de frecuencias de resonancia, etc.
Este shaker por se excita con una señal senoidal de alto voltaje. Con una tarjeta de adquisición de datos (NI-DAQ) se genera una señal senoida pequeña despues esta señal se aumenta con un amplificador para asi poder excitar el Shaker. Con la misma tarjeta de adquisicón se adquieren las señales del acelerometro, en el caso de que el acelerometro fuerá triaxial se adquieren las señales de los tres ejes.

8.2. Funcionamiento de programa

Este programa tiene un modo manual y otro automático.

En el **modo manual** como se observa en la Figura 21 solo se excita el shaker y se adquieren las señales de los ejes del acelerómetro, y se obtiene la transformada de Fourier para para obtener el espectro de frecuencia de la señal del dominio-tiempo original. Se pone un numero de pasos y se llena automáticamente al presionar ¨Ok Button¨, y en la parte de frecuencia, tiempo y amplitud se llena manualmente. Ya que se llenó la tabla con los valores deseados se presiona el botón de ¨Play¨ y automáticamente empezara a excitarse el shaker con los primeros parámetros puestos por el usuario hasta que el tiempo termine dentro de ese paso, ya al terminar el primer paso seguirá con los siguientes hasta llegar al paso final y se prendera un indicador (Time has Elapsed). A todo este paralelamente se registrarán valores de la frecuencia obtenida al aplicar la transformada de Fourier y también del tiempo en el que se presentó.

En el **modo automático** (Figura 21) se ingresa una frecuencia mínima, frecuencia máxima, incremento y tiempo. Al ingresar estos parámetros se presiona ¨Ok Button¨ y automáticamente se llenará la tabla. Ya al tener la tabla con los parámetros necesarios, se presiona ¨Play¨ y empieza a ejecutar el barrido y también el registro los datos.

En las Figuras 22, 23 y 24 se muestra el diagrama a bloques, se tuvo que hacer uso de dos ciclos while, ya que los ciclos requieren distintos tiempos de ejecución, ya que uno es para

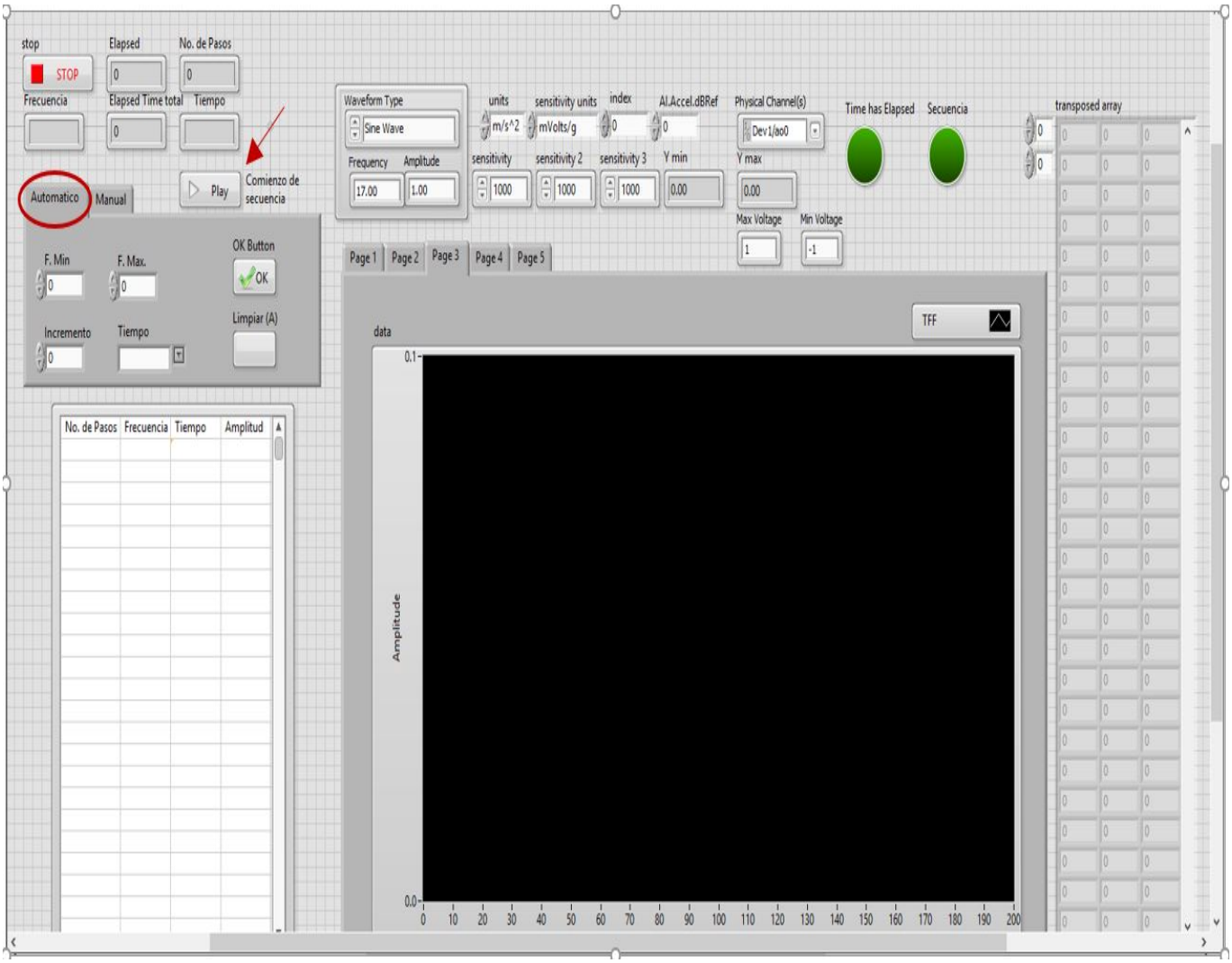

Figura 21: Panel frontal (modo automatico y manual).

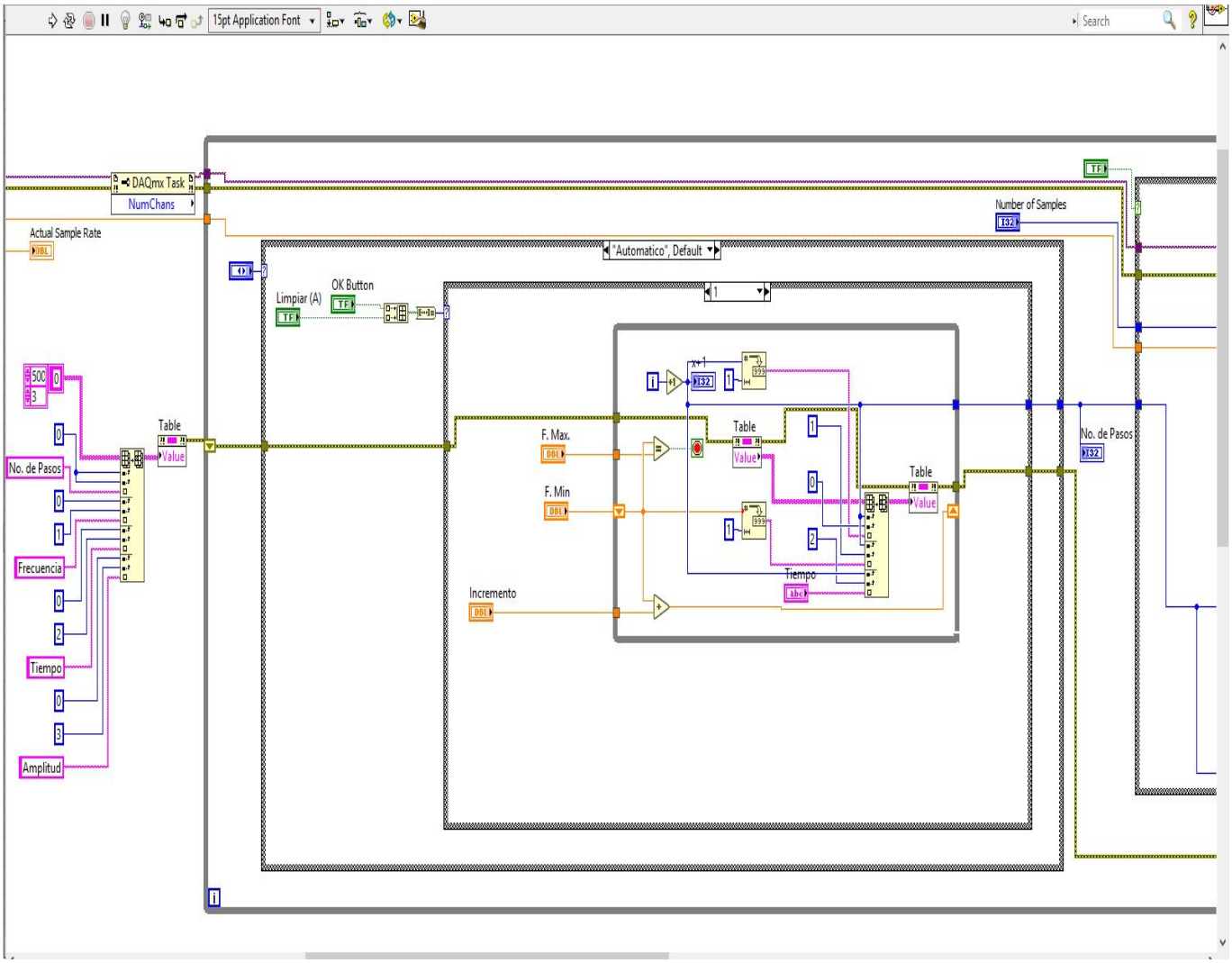

Figura 22: Captura uno de panel frontal.

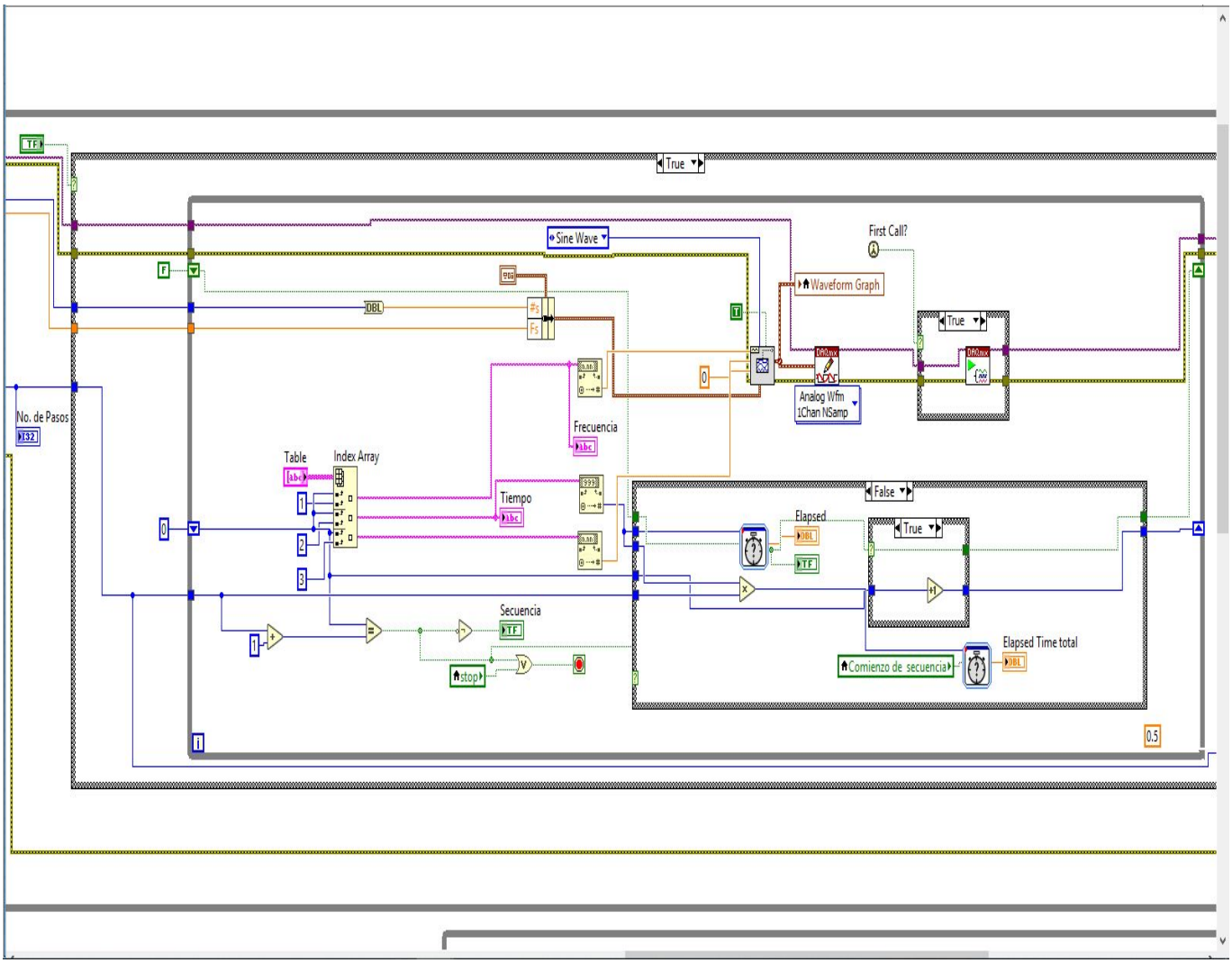

Figura 23: Captura dos de panel frontal.

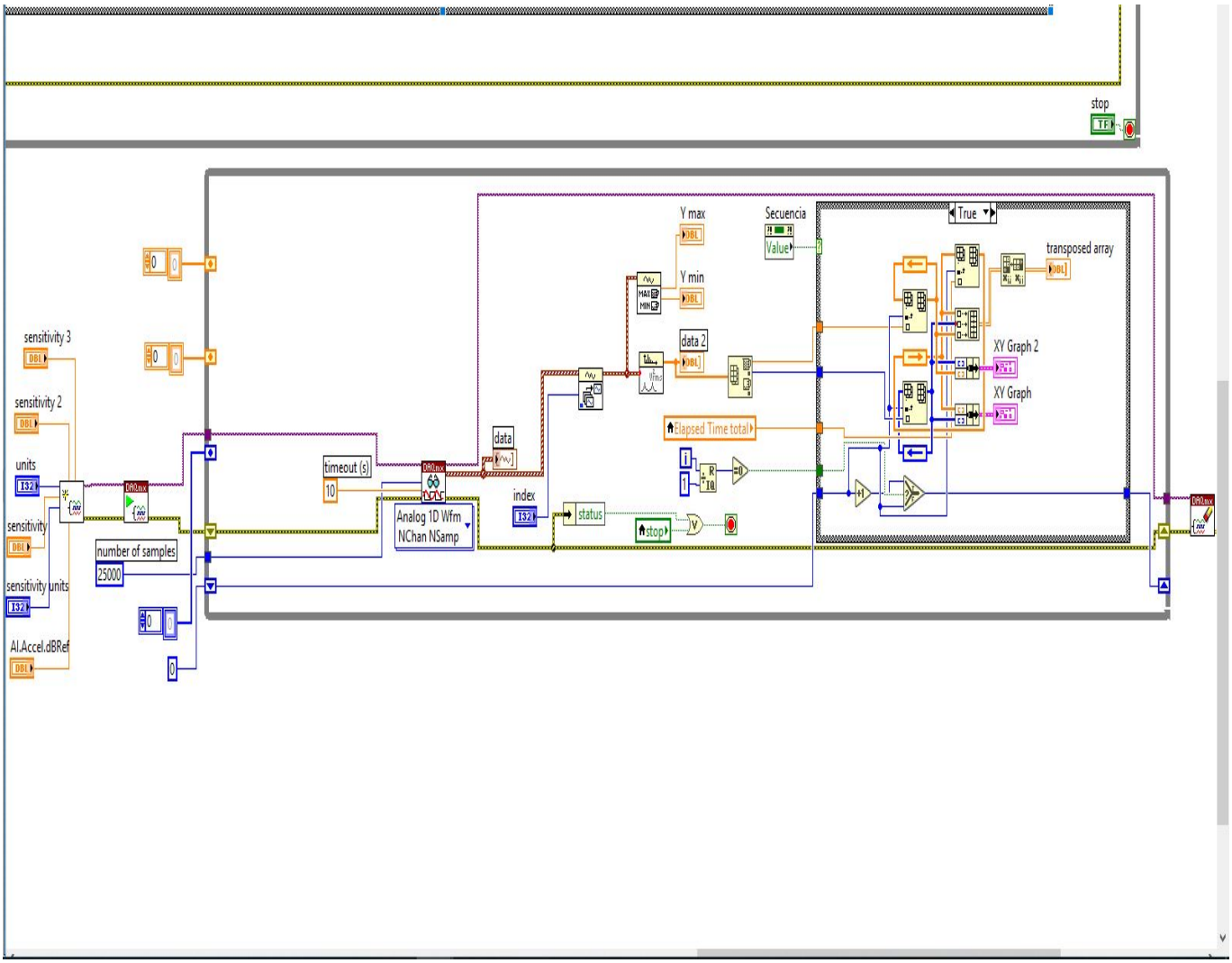

Figura 24: Captura tres de panel frontal.

9. Conclusiones

En cuanto al reconocimiento facial se probaron dos algoritmos de predicción de rostros como lo son: LBP y FisherFaces. Se puede decir que el LBP resulto ser un algoritmo más efectivo en un ambiente controlado. FisherFaces es un algoritmo bastante eficiente en cuanto a los recursos computacionales según el artículo de M.F. Perilla, y en una experiencia propia este necesita menos entrenamiento para lograr un buen desempeño. A pesar de que LBP es un algoritmo diseñado para detectar rugosidades resulta bastante eficiente para el reconocimiento facial, pero tiene la desventaja de demandar más recursos computacionales en comparación al de FisherFaces.

GoogleCloud es una plataforma que tiene Datasets bastantes grandes por lo cual es muy eficiente para la detección objetos, al implementar esta plataforma en la Raspberry Pi resulta muy fácil ya que no requiere mayores recursos computacionales pero lo que requiere es una buena conexión a internet.

En el caso del programa de LabVIEW, se logró hacer un avance para la automatización del proceso de calibración, pero este programa todavía requiere trabajo para automatizar por completo el proceso. Ya que una calibración requiere la estimación de la sensibilidad de los ejes de un acelerómetro en base a un acelerómetro patrón y esto se realiza excitando el shaker en un rango de frecuencias predefinido por el usuario con el fin de encontrar frecuencias naturales de este.

El trabajo realizado a largo de estos cuatro meses en la especialidad ha sido de mucho provecho ya que ha dejado una gran enseñanza lo que servirá para desarrollar trabajos futuros y más que nada en la parte del desarrollo de software y la automatización.

10. Bibliografía

Ahonen, T., Hadid, A., & Pietikainen, M. (2006). Face Description with Local Binary Patterns: Application to Face Recognition. IEEE Transactions On Pattern Analysis And Machine Intelligence, 28(12), 2037-2041. doi: 10.1109/tpami.2006.244

C.M.A, (2014). Reconocimiento Facial con OpenCV. Recuperado de https://opencv2.wordpress.com/2014/03/24/reconocimiento-facial-con-opencv/

Harvey, A. (2018). OpenCV: Face Detection using Haar Cascades. Recuperado de https://docs.opencv.org/3.3.0/d7/d8b/tutorial_ py _ face_ detection.html

Kolda, M. (2018). Analyzing your BigQuery usage with Ocado Technology?s GCP Census. Recuperado de https://cloudplatform.googleblog.com/2018/01/analyzing-your-BigQuery-usage-with-Ocado-Technologys-GCP-Census.html

López Sandoval, A., Mendoza Martínez, C., Reyes Cruz, L., Rivas Araiza, E., Ramos Arreguín, J., & Pedraza Ortega, J. (2015). Sistema de autenticación Facial mediante la implementación del algoritmo PCA modificado en sistemas embebidos con arquitectura ARM. In Asociación Mexicana de Mecatrónica A.C:. Querétaro.

Martinez, A., Pumarola, A., Agudo, A., Martinez, A., Sanfeliu, A., & Moreno-Noguer, F. et al. (2018). dblp: Aleix M. Martinez. Recuperado de https://dblp.org/pers/m/Martinez:Aleix_ M

Machine learning. (2018). Recuperado de https://en.wikipedia.org/wiki/Machine_ learning

Sánchez Quevedo, e. (2018). Diseño y construcción de un sistema de autentificación con reconocimiento facial mediante procesamiento de imágenes con la utilización de software libre y tecnología Raspberry pi. Presentación, universidad de las fuerzas armadas de ecuador.

OpenCV library. (2018). Recuperado de http://opencv.org/

Perilla Benítez, M. (2016). Investigación / Desarrollo de uso de la Raspberry Pi para aplicaciones biometricas de identificación y seguridad. IEEE.

Platero Plazas, D. (2015). Reconocimiento de imágenes faciales orientado a controles de acceso y sistemas de seguridad (Licenciatura). UNIVERSIDAD DISTRITAL ¨FRANCISCO JOSÉ DE CALDAS¨.

Python. (2018). Recuperado de https://es.wikipedia.org/wiki/Python

Raspberry Pi 3 Modelo B (2018). Recuperado de https://www.pccomponentes.com/raspberry-

pi-3

Shapiro, L., & Stockman, G. (2001). Computer vision. Upper Saddle River, NJ: Prentice Hall.

What is Anaconda? - Anaconda. (2018). Recuperado de https://www.anaconda.com/what-is-anaconda/