DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TEMA: IMPLEMENTACION DE UN CONTROL DE ACCESO BIOMETRICO MEDIANTE RECONOCIMIENTO FACIAL

AUTOR: OBANDO CISNEROS, DARÍO ISRAEL

DIRECTOR: ING. PROAÑO ROSERO, VÍCTOR GONZALO MSc.

SANGOLQUÍ

2019

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación, "**IMPLEMENTACION DE UN CONTROL DE ACCESO BIOMETRICO MEDIANTE RECONOCIMIENTO FACIAL**" fue realizado por el señor Obando Cisneros, Darío Israel el mismo que ha sido revisado en su totalidad, analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí 22 abril de 2019

Ing. Víctor Proaño MSc.

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORÍA DE RESPONSABILIDAD

Yo, Obando Cisneros, Darío Israel, declaro que el contenido, ideas y criterios del trabajo de titulación: "**IMPLEMENTACION DE UN CONTROL DE ACCESO BIOMETRICO MEDIANTE RECONOCIMIENTO FACIAL**" es de mi autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Consecuentemente el contenido de la investigación mencionada es veraz.

Sangolquí 22 abril de 2019

Darío Israel Obando Cisneros

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN

Yo, Obando Cisneros, Darío Israel, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: "**IMPLEMENTACION DE UN CONTROL DE ACCESO BIOMETRICO MEDIANTE RECONOCIMIENTO FACIAL**", en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí 22 abril de 2019

Darío Israel Obando Cisneros

DEDICATORIA

A toda mi familia y amigos.

AGRADECIMIENTO

A toda mi familia y amigos que me supieron enseñar mucho durante todo este proceso universitario.

ÍNDICE DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN

El presente proyecto desarrolla un sistema robusto para el control de acceso puesto que tiene dos métodos de verificación de personas, uno de ellos bastante utilizado a nivel comercial como lo son las tarjetas RFID y el otro es un sistema biométrico no intrusivo para reconocer rostros en tiempo real. Para la captura de información de las tarjetas RFID se usa un microcontrolador Arduino donde se conecta la lectora de tarjetas y además se conecta una chapa magnética. Para realizar el reconocimiento facial se usa una computadora y una cámara IP por parte del hardware mientras tanto que los recursos de software son un sistema operativo Linux, en donde se encuentra instalado Python lenguaje de programación que se usa para realizar el sistema en su totalidad, además de algunas librerías de vital importancia para el tratamiento de imágenes como lo son OpenCV, también se usan redes neuronales convolucionales pre-entrenadas para el reconocimiento de caras y la extracción de sus características. Cada usuario debe pasar la tarjeta por el lector RFID en donde se toma el número de tarjeta que tiene que emparejarse con la cara que se reconoce si se cumplen estas dos condiciones se permite el acceso, existe una base de datos que contiene las fotografías de los rostros de las personas y su información relevante como lo son Nombre, Apellido, CI, Edad, Cargo, Número de tarjeta.

Palabras clave:

- **TARJETA RFID.**
- **RECONOCIMIENTO FACIAL.**
- **TRATAMIENTO DE IMÁGENES.**
- **BASE DE DATOS.**
- **REDES CONVOLUCIONALES PRE-ENTRENADAS.**

ABSTRACT

This project develops a robust system for access control since it has two methods of verification of people, one of them quite used commercially as are RFID cards and the other is a non-intrusive biometric system to recognize faces, all this in real time. For the capture of information from the RFID cards an Arduino microcontroller is used where the card reader is connected and a magnetic plate is connected. To perform facial recognition a computer and an IP camera are used by the hardware while the software resources are a Linux operating system, where Python is installed programming language that is used to make the system in its entirety, in addition to some bookstores of vital importance for the treatment of images as they are OpenCV, it is important to emphasize that pre-trained convolutional neural networks are used for the recognition of faces and the extraction of their characteristics. Each user must pass the card through the RFID reader where the card number is taken that has to be paired with the face that is recognized if these two conditions are met access is allowed, there is a database containing the photographs of the faces of people and their relevant information such as Name, Surname, IC, Age, Position, Card number.

Keywords:

- **RFID CARD.**
- **FACIAL RECOGNITION.**
- **IMAGE PROCESSING.**
- **DATABASE.**
- **PRE-TRAINED CONVOLUTIONAL NEURAL NETWORK.**

CAPITULO I

PLANTEAMIENTO DEL PROBLEMA DE INVESTIGACIÓN

1. Antecedentes

La visión artificial es una ciencia que está ganando espacio en distintas aplicaciones tecnológicas como lo son la robótica (Sobrado Malpartida & Tafur Sotelo, 2011), mejora de procesos (Constante Prócel, y otros, 2016) (Pérez Grassi & Puente León, 2006), detección de vehículos en carreteras (Bertozzi, y otros, 2002), reconocimiento de expresiones faciales (Rodríguez Hernández & Duque Méndez, 2015) por nombrar solo algunas. Para (Deshpande & Ravishankar, 2010) la visión artificial es un potenciador de las interfaces hombre máquina mejor conocidas como HCI.

El reconocimiento facial, es un reto que se lo viene atacando desde principios de los años 60 en el campo de la ingeniería (Marques, 2010). En estos años Woody Bledsoe un pionero de la inteligencia artificial se encargaría de realizar el primer reconocimiento facial que se lo denominó hombre-máquina, en este trabajo con la ayuda de un operador se podía extraer las coordenadas de las características físicas de un rostro como el centro de las pupilas, la distancia exterior e interior de los ojos, etc. de un set de fotografías, para después calcular 20 distancias como el ancho de la boca, el ancho de los ojos, distancia entre pupila y pupila, etc. Adicionalmente en este aplicativo se generaba una base de datos en donde el nombre de la persona se asociaba con las distancias calculadas. Por último, en la fase de reconocimiento el conjunto de distancias se comparaba con la distancia correspondiente para cada fotografía, produciendo una distancia entre la fotografía y el registro de la base de datos (Ballantyne,

Boyer, & Hines, 1996). Siguiendo con la línea del tiempo en los años 70 con técnicas heurísticas y antropométricas simples (Hjelmas & Kee Low, 2001), se buscó un reconocimiento facial confiable, pero en esta época no se obtuvo resultados con gran eficiencia alcanzando desde un 45% a 75% de acierto (Marques, 2010). A partir del año 2001 cuando Viola y Jones publican su artículo titulado "Rapid Object Detection using a Boosted Cascade of Simple Features" en el cual realizan tres contribuciones claves, la primera el uso de una imagen integral, la segunda un algoritmo de aprendizaje basado en AdaBoost y la tercera uso de clasificadores en cascada. Los resultados de Viola y Jones son la detección de objetos más rápida y eficaz vista hasta ese entonces llegando a obtener entre un 80% y 95% de confiabilidad (Viola & Jones, 2001).

En un amplio campo de algoritmos de reconocimiento facial se consideran los siguientes puntos:

- **Extractor de características:** es el proceso para obtener características faciales relevantes a partir de los datos. Estas características podrían ser ciertas regiones de la cara, variaciones, ángulos o medidas, que pueden ser relevantes para el ser humano (por ejemplo, espaciado de los ojos) o no. Esta fase tiene otras aplicaciones como el seguimiento de rasgos faciales o el reconocimiento de emociones (Marques, 2010).
- **Reconocimiento facial:** Esta fase implica un método de comparación, un algoritmo de clasificación y una medida de precisión. Esta fase utiliza métodos comunes a muchas otras áreas que también realizan algún proceso de clasificación: ingeniería de sonido, minería de datos, etc. (Marques, 2010).

Es significativo tener en cuenta que un detector facial tiene algunos problemas como lo son el posicionamiento de la persona, la luz existente cuando se toma la imagen, color de la piel y si está en movimiento o no el sujeto de experimento (Marques, 2010). En relación a estos

problemas existen diferentes métodos con los que se obtiene mejores resultados dependiendo de la aplicación, entre ellos destacan.

Extractores de características:

- **HOG:** descriptor de características, la distribución (histogramas) de las direcciones de los gradientes (gradientes orientados) se utilizan como características de representación de una imagen (Mallick, 2016).
- **HAAR:** considera regiones rectangulares vecinas en una ubicación especifica de la ventana de detección, suma la intensidad de pixeles en esta región y calcula la diferencia entre estas sumas. Esta diferencia se utiliza para caracterizar las subsecciones de una imagen (Viola & Jones, 2001).
- **LBP:** representa el patrón que existe de la derivada circular de primer orden de las imágenes, que se genera con la concatenación de las direcciones del gradiente binario (Zhang, Gao, & Zhao, 2010).
- **CNN:** es una clase de red neuronal artificial que utiliza capas convolucionales para filtrar entradas para obtener información útil. La operación de convolución implica combinar datos de entrada (mapa de características) con un núcleo de convolución (filtro) para formar un mapa de características transformado. Los filtros en las capas convolucionales (capas conv) se modifican en función de los parámetros aprendidos para extraer la información más útil para una tarea específica (Corporation N. , 2018).

Clasificadores:

- **Eigenfaces**: conjunto de vectores propios para el reconocimiento de la cara humana (Rodríguez Hernández & Duque Méndez, 2015).
- **Redes Neuronales**: tienen la capacidad de aprender relaciones complejas no lineales

de entrada-salida, usar procedimientos de entrenamiento secuencial y adaptarse a los datos (Kumar Basu, Bhattacharyya, & Kim, 2010).

- **SVM:** clasificadores lineales que maximizan el margen entre el hiperplano de decisión y los ejemplos en el conjunto de entrenamiento. Por lo tanto, un hiperplano óptimo debe minimizar el error de clasificación de los patrones de prueba no vistos (Marques, 2010).
- **Adaboost:** clasificador que propone entrenar una serie de clasificadores débiles de manera interactiva, de modo que cada clasificador o" weak learner" se enfoque en los datos que fueron erróneamente clasificados por su predecesor, de esta manera el algoritmo se adapta y logra tener mejores resultados (Sánchez).

Con toda la connotación apreciada acerca de la visión artificial haciendo énfasis en el reconocimiento facial, cabe recalcar las aplicaciones prácticas como lo son:

- Bioseguridad.
- Conferencias.
- Búsqueda de personas desaparecidas.

Tomando en cuenta toda la información recaudada, en el presente proyecto se propone generar un prototipo que ayude con el reconocimiento facial de un cierto grupo de personas.

2. Justificación

La inteligencia artificial es una nueva tendencia, que crece de manera exponencial por los miles de aplicaciones que se le puede dar, en especial ayuda mucho a la automatización de procesos repetitivos y engorrosos. Trabajar con este tipo de tecnología es un reto interesante para el ámbito de investigación y desarrollo en el país.

En Ecuador la IA es un nicho para desarrollar aplicativos potencialmente comerciales. Reduciendo en parte la brecha tecnológica que se tiene a comparación de países más desarrollados, con los cuales se podría competir en cuanto a precios y calidad. Sumándole a estos acontecimientos, el hecho de que vivimos en una de las zonas más inseguras del mundo y que existe una alta demanda de equipos de seguridad. La biometría contribuye a realizar equipos más seguros.

Realizar el reconocimiento facial que esté ligado a una base de datos con información personal de cada individuo que entre a una oficina se lo puede escalar a un proyecto que vaya de la mano con la seguridad de la ciudadanía, por ejemplo, entrenando al sistema con fotografías e información de delincuentes y buscándolos con cámaras de seguridad de acceso de la policía se podría reducir el esfuerzo que se ocupa en buscar a los antisociales.

El proyecto propuesto engloba algunas áreas del conocimiento adquirido a lo largo de la carrera, con las que se puede lograr poner en práctica los conceptos teóricos, realizando algo útil que sirva para el diario vivir de las personas con el extra de que se puede transformar en un emprendimiento.

3. Alcance

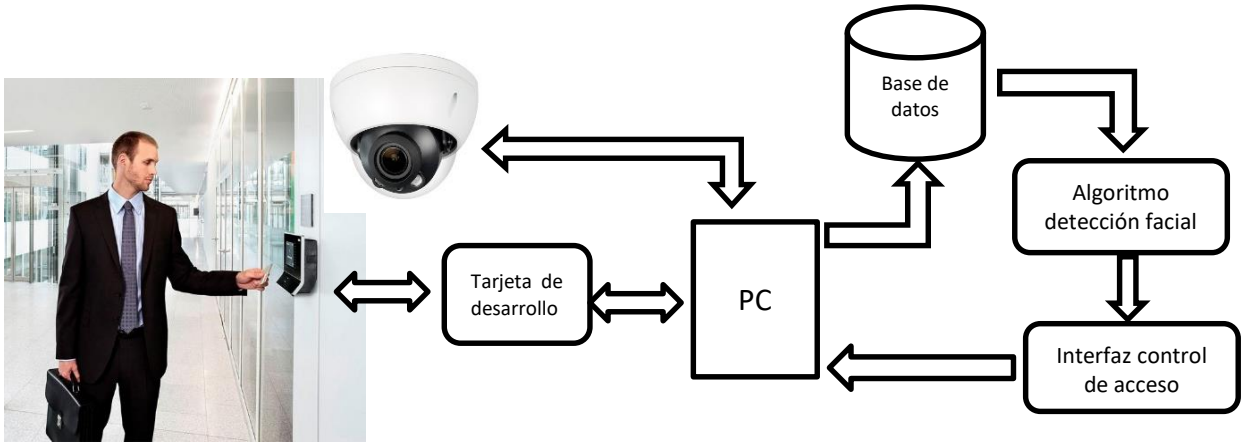


Figura 1. Descripción del proyecto

El presente proyecto de investigación tiene como objetivo el realizar un prototipo funcional para el reconocimiento de usuarios que ingresan a una oficina mediante un sistema de visión artificial, obteniendo sus datos como nombre, cédula de identidad, edad, etc. en primera instancia mediante el uso de un programa de registro, con esta información más varias imágenes del rostro de los usuarios se busca entrenar a un sistema de visión por computador para reconocer caras. Posterior al entrenamiento el sistema de forma autónoma debe diferenciar entre cada una de las personas que estén registradas en la base de datos con el fin de controlar el acceso. El sensor que se utilizará para el aprendizaje de máquina será una cámara, también un lector de tarjetas RFID será parte del proyecto, además hay que tomar en cuenta una tarjeta de desarrollo donde conectar el lector RFID. El trabajo se divide en cuatro etapas que son las siguientes:

Primera etapa desarrollo de un aplicativo para obtener el número de la tarjeta RFID, siendo esta la señal para tomar una fotografía del rostro del usuario, lo cual se implementará mediante

una tarjeta de desarrollo que tendrá conectado un lector RFID y una computadora para la conexión de la cámara.

Segunda etapa se genera una base de datos que contenga las dos contemplaciones nombradas en la primera etapa, realizando una concatenación de las imágenes con su respectivo número de tarjeta con el fin de asignarle la correspondiente información personal del usuario.

Tercera etapa se realiza un procedimiento de detección de rostros entrenado con la base de fotografías e información personal de usuario, obtenidas en la segunda etapa del proyecto, adicionalmente en esta etapa se realizará una interfaz de control de acceso. La cual está integrada por la tarjeta de desarrollo y la PC.

Cuarta etapa pruebas del sistema con usuarios tanto registrados como no en la base de datos, para analizar en cuanto porcentaje es confiable el sistema de control de acceso biométrico.

4. Objetivos

4.1 Objetivo General

Desarrollar un prototipo funcional para el reconocimiento facial de usuarios con el fin de obtener con solo una imagen en tiempo real su información personal y en base a esta controlar su acceso a un establecimiento.

4.2 Objetivo Específico

- Desarrollar un sistema para conectar una cámara, un lector RFID, una tarjeta de desarrollo y una computadora con el fin de capturar datos.
- Realizar una base de datos de fotografías confiables considerando puestas de luz, colores, etc. para entrenar al sistema de la mejor forma posible.
- Diseñar una interfaz de usuario intuitiva y de fácil manejo enfocada en controlar el acceso a la oficina vía reconocimiento facial.

CAPITULO II

FUNDAMENTOS TEÓRICOS

2.1 Introducción a la Biometría

La biometría es una ciencia que estudia los diferentes rasgos físicos de las personas y los estandariza de tal forma que convierte las características de cada individuo en información útil que pueda ser procesada, en la actualidad generalmente por un computador (Jain, Bolle, & Pankanti, Biometrics Personal Identification in Network Society, 2006). Los sistemas biométricos se usan para identificación de personas, las características biológicas que deben procesar estos denominados sistemas se rigen según las 4 condiciones (Vilda, 2006) denominadas en la tabla 1.

Tabla 1
Características biológicas de un sistemas biométrico

Condición	Descripción
Universal	Toda persona debe poseer la característica biométrica.
Distintiva	Las diferencias entre las características biométricas de dos personas diferentes deben ser lo suficientemente significativas para permitir distinguirlas.
Permanente	Debe permanecer relativamente invariante a lo largo del tiempo
Coleccionable	Debe poder ser medida y cuantificable

Fuente: (Vilda, 2006)

Usar como medios biométricos las huellas dactilares, el iris de los ojos, las líneas marcadas en las palmas de la mano o el reconocimiento facial para este caso es viable puesto que cumple con las condiciones antes nombradas en la tabla 1.

2.2 Breve evolución de los sistemas biométricos

Los gobiernos necesitan sistemas de identificación de individuos para mantener el control de la sociedad, porque estos pueden ayudar a la clasificación de personas entre un sin número de categorías como sexo, etnia, color de piel, etc., así como también en la seguridad y otras aplicaciones para la organización de la sociedad.

La biometría viene siendo parte activa de la vida diaria de las personas desde el año de 1879 con la aparición del primer sistema biométrico planteado y ejecutado por Louis Bertillon jefe del departamento de policía de París quien utilizó 20 medidas del cuerpo humano, que se debían fotografiar y archivar, con el objetivo de saber si la persona es un criminal recurrente, asesino en serie o violador (Caplan & Torpey, 2018).

El sistema Bertillon fue siendo reemplazado de a poco por la obtención de la huella dactilar (fingerprinting), desarrollado por Carlo Ginzburg, este método tiene dos ventajas en relación al Bertillon. En primer lugar es menos invasivo, la segunda ventaja es, al no tener tantas variables se necesita ser menos perspicaz para obtener los datos de identificación y clasificación por lo que cualquier persona lo puede realizar (Caplan & Torpey, 2018).

Estos dos sistemas son de suma importancia para el crecimiento de las tecnologías que hoy en día conocemos porque aportaron la suficiente cantidad de pautas para conseguir con el paso del tiempo técnicas cada vez más sofisticadas para la clasificación de los individuos. En la actualidad la identificación y clasificación de las personas se ha expandido tanto que es muy normal observar a la entrada de un edificio o en una casa lectores de huellas o cámaras para identificación de iris, entre un sin número más de aparatos biométricos.

2.3 Reconocimiento facial

En la actualidad los sistemas de reconocimiento facial se los puede resumir de la siguiente forma, adquisición de la imagen, algoritmo para reconocer una cara en la imagen, extraer las características de la cara, comparar con una base de datos en donde se encuentra almacenada las características de los rostros que van a utilizar el sistema biométrico y por ultimo arrojar un resultado.

La ventaja de utilizar una cara es, utilizar la medición geométrica espacial de los puntos clave de las caras, esto quiere decir las distancias respectivas entre los ojos y la nariz o los ojos y la boca, entre muchas otras combinaciones que se pueden generar, dotando al sistema de herramientas suficientes para formar un sistema confiable (Woodward, John, Horn, Gatune, & Thomas, 2003).

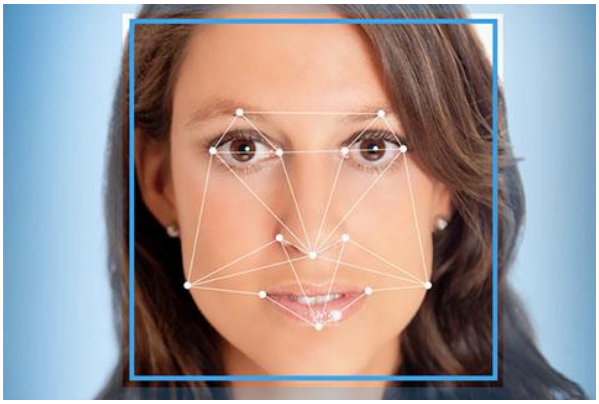


Figura 2. Biometría facial ejemplo

2.3 Procesamiento digital de imágenes.

Es importante empezar por la definición de una imagen digital, según (Ambarjarafi, 2014) una imagen digital es una función en un plano de dos dimensiones (2D) dada por la siguiente expresión:

$$f(x, y) = \delta$$

En donde:

$$x, y = coordenadas\ en\ el\ espacio\ de\ la\ imagen$$

$$\delta = intensidad\ de\ grises\ en\ ese\ punto$$

Entonces si los valores de la función f son todos finitos se denomina una imagen digital. Se toma como referencia el plano de la figura 3.

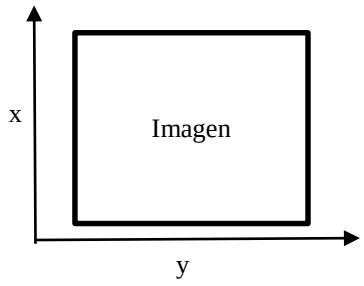


Figura 3. Referencia de plano

En la figura 4 se nota como se expresa una imagen haciendo uso de la función f. El valor de δ varía entre 0 y 255. La imagen consta de 1024 pixeles de ancho y 683 de largo.

Figura 4. Ejemplo función f

La figura 4 sirve como una representación de la función f, tomando en cuenta las tres secciones seleccionadas mediante los cajones tomates que encierran pixeles o una cierta área de la imagen, empezando con la esquina superior izquierda tendríamos una $f(1,1) = 80$, el cajón de la mitad $f(310{:}\,320,450{:}\,500) = \begin{bmatrix} 80 & \dots & 103 \\ \dots & 10 & \dots \\ 205 & \dots & 90 \end{bmatrix}$ generamos una matriz de 10 filas por 50 columnas, para el caso del cajón, por ultima en la esquina inferior derecha tendríamos una $f(683,1024) = 75$.

El objetivo principal de procesar una imagen es mejorar su calidad o extraer sus características para una descripción o también para una caracterización de sus componentes más notables o a su vez resaltar algún detalle que se necesite dentro de la imagen, para realizar el procesamiento se debe cumplir con los pasos de muestreo y cuantificación (Ambarjarafi, 2014). El muestreo define la resolución espacial de una imagen en la figura 5 se ejemplifica el muestreo de una imagen y la cuantificación determina el nivel de grises de la imagen en la figura 6 se puede apreciar una imagen cuantificada con diferente número de bits (Ambarjarafi, 2014).

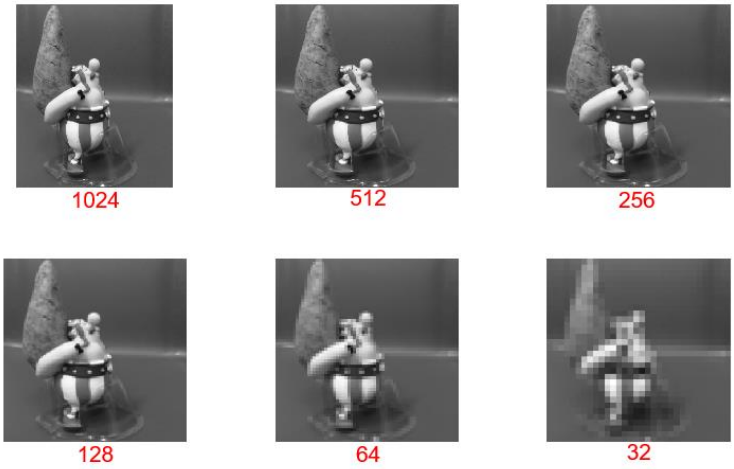


Figura 5. Muestreo de una imagen

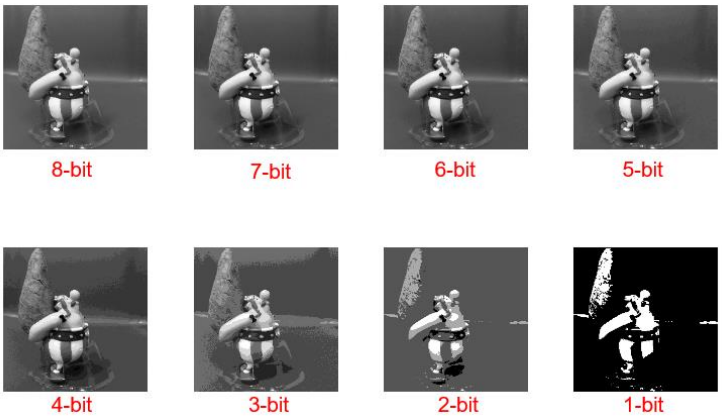


Figura 6. Cuantificación de una imagen

Hay que tomar en cuenta que cuando una imagen se cuantifica en 8 bits, tiene 255 niveles porque la intensidad de grises se maneja como se muestra en la figura 7. En donde n representa la cantidad de bits con la que se está cuantificando la imagen, mientras más bits la calidad de la imagen mejora.

Figura 7. Intensidad de grises

Una vez que la imagen esta digitalizada es importante eliminar el ruido, el fondo, etc. para lo cual se usan diferentes tipos de filtros. Por ejemplo, se puede utilizar un filtro para revertir el contraste, cuando la imagen debe está cuantificada en 8 bits, se aplica la siguiente función matemática, (Beyerer , Puente León , & Frese , 2016).

$$f'(x, y) = 255 - f(x, y)$$

Cada pixel de la imagen que se esté tratando se verá afectado por la resta, con lo que se logra eliminar el color blanco de la escala de grises de la imagen original. Para una mejor apreciación de lo que realiza f' se observa la figura 8.

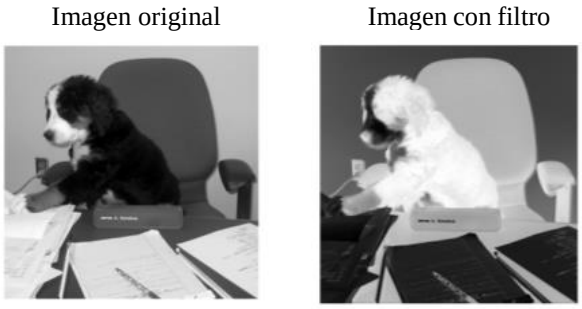


Figura 8. Filtro para revertir el contraste

Uno de los procedimientos más utilizados para realizar el filtraje de imágenes es la aplicación de máscaras que se basa en la realización de operaciones convolucionales sobre una imagen. En el siguiente ejemplo se muestra cómo actúa un filtro sobre una imagen.

Si se tiene el arreglo matricial de una imagen como el siguiente:

$$\begin{matrix} 35 & 40 & 41 & 45 & 50 \\ 40 & 40 & 42 & 46 & 52 \\ 42 & 46 & 50 & 55 & 55 \\ 48 & 52 & 56 & 58 & 60 \\ 56 & 60 & 65 & 70 & 75 \end{matrix}$$

Y una máscara:

$$\begin{matrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{matrix}$$

Se aplica la convolución a la imagen:

$$\begin{matrix} 35 & 40 & 41 & 45 & 50 \\ 40 & 40 & 42 & 46 & 52 \\ 42 & 46 & 50 & 55 & 55 \\ 48 & 52 & 56 & 58 & 60 \\ 56 & 60 & 65 & 70 & 75 \end{matrix} * \begin{matrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{matrix}$$

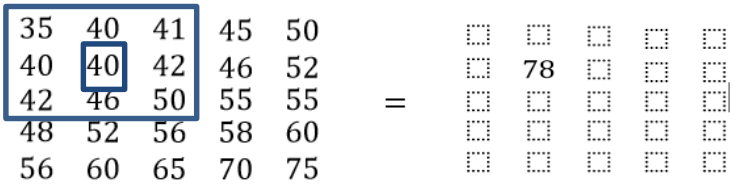
De forma general se puede considerar que una máscara tiene la forma:

$$\begin{matrix} w1 & w2 & w3 \\ (x-1, y-1) & (x-1, y) & (x-1, y+1) \\ w4 & w5 & w6 \\ (x, y-1) & (x, y) & (x, y+1) \\ w7 & w8 & w9 \\ (x+1, y-1) & (x+1, y) & (x+1, y+1) \end{matrix}$$

El centro de la máscara (w5) se mueve a lo largo y ancho de la imagen. En cada posición multiplicamos cada punto que esta contenido dentro del área que ocupa la máscara por el correspondiente coeficiente de la máscara, lo que resulta en un nuevo valor dado por la siguiente expresión.

$$f'(x,y) = w1f(x-1,y-1) + w2f(x-1,y) + w3f(x-1,y+1) + w4f(x,y-1) + w5f(x,y) + w6f(x,y+1) + w7f(x+1,y-1) + w8(x+1,y) + w9(x+1,y+1)$$

Los valores de la imagen filtrada se van generando conforme la máscara pase por cada uno de los pixeles de la imagen original. Si por ejemplo se realiza la convolución sobre el pixel seleccionando en el cuadrado azul:



Calculando el valor de $f'(x,y)$ del ejemplo:

$$f'(x,y) = -2 * 35 + -1 * 40 + 0 * 41 + -1 * 40 + 1 * 40 + 1 * 42 + 0 * 42 + 1 * 46 + 2 * 50$$

$$f'(x,y) = 78$$

Una vez con la imagen de mejor calidad se procede a hacer la segmentación, según (Domínguez Torres, 1996) esta va de la mano con otro tipo de filtros que tienen un tratamiento matemático más avanzado, entonces lo que se puede obtener al aplicar la segmentación a una imagen es, dividirle en sus partes constituyentes u objetos para poder ser manipulados, además se puede extraer las características de los objetos en los que se divide la imagen siendo esta la pauta para el reconocimiento automático de imágenes. Existen dos formas de realizar una segmentación.

- **Similitud:** Agrupa objetos de la imagen de acuerdo a una intensidad de grises predeterminada (Domínguez Torres, 1996).
- **Discontinuidad:** Agrupa objetos de la imagen en los cambios exagerados de intensidad

de grises, se usa para detectar bordes generalmente (Domínguez Torres, 1996).

Un ejemplo clásico de la segmentación de imágenes, es aplicar el filtro Sobel para detección de bordes. Se necesita de una máscara para los pixeles en x y otra para los pixeles de y, luego la suma de las dos genera una imagen con sus bordes resaltados, en la figura 9 se puede apreciar el filtro Sobel.

$$s_y = \begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$$

$$s_x = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

Si se considera una imagen como una matriz A y la nueva imagen con filtro como B, la operación con el filtro de Sobel está determinada de la siguiente forma.

$$B = s_y * A + s_x * A$$

Imagen original

Imagen con filtro Sobel

Figura 9. Filtro Sobel

Para el procesamiento de imágenes a colores se hace exactamente lo mismo, pero con máscaras que tienen una dimensión largo x ancho x 3, la diferencia erradica en que son 3 canales RGB por tanto se multiplica la máscara por un 3 al final.

2.4 Red neuronal artificial.

Una red neuronal artificial es un modelo matemático inspirado en el comportamiento biológico de las neuronas y en la estructura del cerebro, y que es utilizada para resolver un amplio rango de problemas. Debido a su flexibilidad una única red neuronal es capaz de realizar diversas tareas. (Tablada & Torres, 2015).

Es importante tomar en cuenta como se forman las redes neuronales, en primera instancia se tiene una neurona con una sola entrada, después varias neuronas con varias entradas, el siguiente paso a seguir es unir estas neuronas para formar una capa y finalmente ubicar en forma de cascada estas capas para producir una red neuronal. Entonces definir que es una neurona es clave para entender el funcionamiento de las redes neuronales. En la figura 10 se aprecia el esquema del modelo matemático de una neurona artificial.

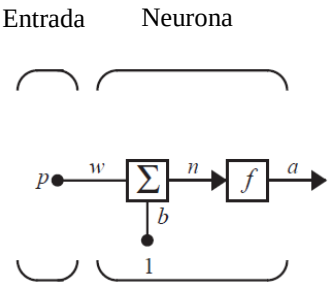


Figura 10. Neurona Artificial

Una neurona se la puede definir matemáticamente de la siguiente forma

$$a = f(wp + b)$$

Donde:

- p es una entrada.
- w es un valor ajustable escalar del peso.
- b es un valor ajustable escalar del bias.

- f es la función de activación, que produce un valor escalar de salida.

Cabe recalcar que los parámetros w y b están ajustados por alguna regla de aprendizaje para que la relación de entrada-salida de la neurona cumpla con algún objetivo específico.

Una de las funciones de activación más utilizadas es la sigmoide, matemáticamente se expresa en siguiente forma y su grafica se encuentra en la figura 11.

$$f = \frac{1}{1 + e^{-n}}$$

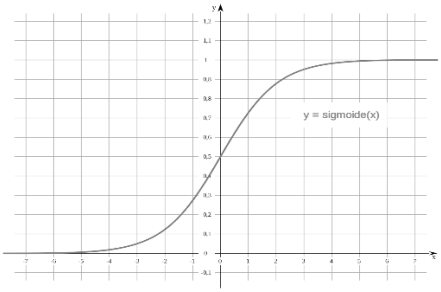


Figura 11. Función activación sigmoide

Esta función realiza la conversión del valor de entrada a uno que este en los límites de 0 a 1.

Una neurona con múltiples entradas, se la puede observar en la figura 12.

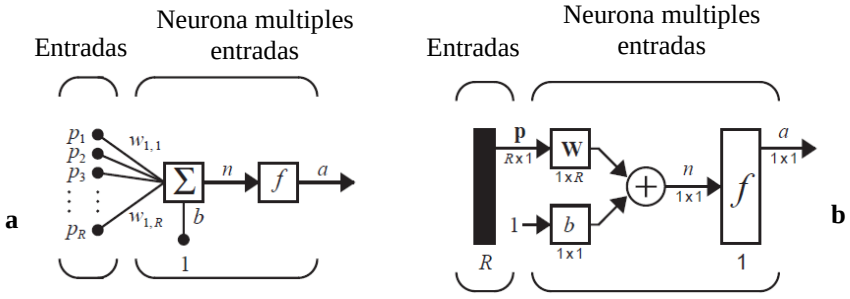


Figura 12. Neurona artificial múltiples entradas

Una neurona con varias entradas se la puede expresar matemáticamente de la siguiente forma.

$$n = w_{1,1} * p_1 + w_{1,2} * p_2 + \cdots + w_{1,R} * p_R$$

Los pesos se pueden expresar en una matriz de una fila por R columnas llamada W y las entradas están contenidas en otra matriz de una columna por R filas denominada p, como se observa en la imagen 12 b, la salida de la neurona está definida por.

$$a = f(\boldsymbol{W}\boldsymbol{p} + b)$$

Continuando con el orden para formar una red neuronal, hay que tener en cuenta que es una capa, en la figura 13 a, se puede apreciar la representación de varias neuronas en paralelo con múltiples entradas que forman una capa que se la puede colocar de forma matricial como en la figura 13 b.

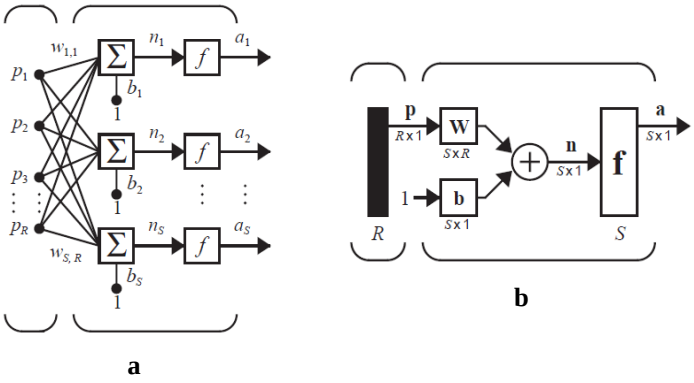


Figura 13. Capa neuronal

Por ultimo para formar una red neuronal, se pone a trabajar varias capas en forma de cascada como en la figura 14, donde existen 3 capas.

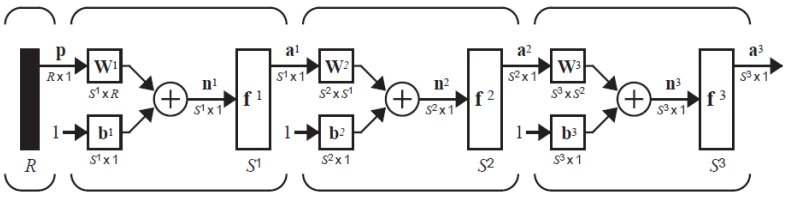


Figura 14. Red neuronal artificial

De forma matemática se puede definir una red neuronal de tres capas de la siguiente forma.

$$a^3 = f^3(W^3 f^2(W^2 f^1(W^1 p + b^1) + b^2) + b^3)$$

Para encontrar los valores de los parámetros de peso y de bias se debe entrenar la red neuronal, existen varios métodos, en este caso se analiza uno muy conocido denominado **backpropagation** que se basa en el descenso de gradiente. Tomando en cuenta que la salida de una capa es la entrada de una nueva capa, la salida de la capa que se necesite se describe de la siguiente forma.

$$a^{m+1} = f^{m+1}(W^{m+1} a^m + b^{m+1})$$

$$m = 0,1, \dots, M - 1$$

Donde M es el número de capas en la red neuronal, se considera la primera capa como en la siguiente ecuación.

$$a^0 = p$$

A la última capa su salida está dada de la siguiente forma.

$$a = a^M$$

Considerando toda esta información es momento de entrenar la red neuronal, se realiza un aprendizaje supervisado, para lo que usamos dos tipos de datos, las entradas (p) y el resultado que busca con cada una de ellas (t).

$$(p_1, t_1), (p_2, t_2), \dots., (p_Q, t_Q)$$

Cuando los datos ingresan a la red se busca minimizar el error cuadrático medio que se expresa con en F(x), por lo que los valores de los pesos van a ir cambiando hasta generar uno muy bajo.

$$F(\boldsymbol{x}) = \sum_{q=1}^{Q} e_q^2 = \sum_{q=1}^{Q} \left(t_q - a_q\right)^2$$

En caso de tener múltiples salidas la expresión es matricial, entonces se modifica

$$F(\boldsymbol{x}) = \sum_{q=1}^{Q} \boldsymbol{e}_q^T \boldsymbol{e}_q = \sum_{q=1}^{Q} \left(\boldsymbol{t}_q - \boldsymbol{a}_q\right)^T \left(\boldsymbol{t}_q - \boldsymbol{a}_q\right)$$

Expresando la ecuación de forma que se calcule F(x) en cada interacción.

$$\hat{F}(\boldsymbol{x}) = \left(t(k) - a(k)\right)^T - \left(t(k) - a(k)\right) = e^T(k) - e(k)$$

Entonces la función de **a** esta en función de **W** y **b**, las cuales se hallan usando el descenso de gradiente que pasa por cada capa de la red, de la siguiente forma. El valor m, es el número de capa de la red donde se encuentra la aplicación del descenso de gradiente.

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha \frac{\partial \hat{F}}{\partial w_{i,j}^m}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha \frac{\partial \hat{F}}{\partial b_i^m}$$

En donde α es la taza de aprendizaje. Para hallar las derivadas parciales de las expresiones anteriores se necesita utilizar la regla de la cadena, en donde n es la entrada es la entrada a la funcion F de cada neurona.

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m}$$

En donde n_i^m puede calcularse como una función de los pesos y bias de cada capa.

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m$$

Reemplazando el termino.

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1}$$

$$\frac{\partial n_i^m}{\partial b_i^m} = 1$$

Ahora si se define.

$$s_i^m = \frac{\partial \hat{F}}{\partial n_i^m}$$

Reemplazando todos los terminos

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha s_i^m a_j^{m-1}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha s_i^m$$

De forma matricial, en la figura 14 se muestra como estas ecuaciones estan representadas en la neurona artificial.

$$\boldsymbol{W}^m(k+1) = \boldsymbol{W}^m(k) - \alpha \boldsymbol{s}^m(\boldsymbol{a}^{m-1})^T$$

$$\boldsymbol{b}^m(k+1) = \boldsymbol{b}^m(k) - \alpha \boldsymbol{s}^m$$

2.5 Redes neuronales convolucionales.

Para el análisis de imágenes existe un tipo especial de red neuronal que se denomina red neuronal convolucional, su funcionamiento se basa en, que las capas asumen las entradas como imágenes por esta razón sus neuronas están organizadas en tres dimensiones: ancho, alto, profundidad (Johnson, 2018). Entonces este tipo de capas para ser más precisos son filtros que se pueden usar para pre-procesar una imagen de una cierta manera, depende de la cantidad de filtros que tenga una red neuronal convolucional poseerá la capacidad de reconocer patrones en una imagen, algunos filtros pueden detectar esquinas, círculos, cuadrados. Ahora, estos filtros simples y geométricos son lo que se observa al comienzo de una red neuronal convolucional. Existen otros filtros que son muy sofisticados y pueden buscar en una imagen gatos, perros, ojos, etc. por lo general este filtraje se encuentra en las ultimas capas de la red neuronal.

Los tipos de patrones que se buscan generalmente son:

- bordes
- formas
- texturas
- curvas
- objetos
- colores

La arquitectura de una red convolucional como se muestra en la figura 16, cuenta con las siguientes capas repartidas según el tipo de red que se esté utilizando:

- **Capa convolucional:** realiza una operación convolucional a la imagen para encontrar rasgos característicos, una vez que la imagen de entrada pasa por una capa

convolucional el resultado es un tensor que es un mapa de características de la imagen, si por ejemplo a la entrada se tiene una imagen dada por su largo x ancho x 3 pasa por una sola capa convolucional que contiene un filtro con su respectivo de largo1 x ancho1 x 3, el tensor estaría definido por largo1 x ancho1 x 1, el volumen del tensor depende de la cantidad de capas convolucionales que atraviese la imagen.

- **Función lineal rectificada (ReLU):** es una función de activación parecida a la de una red neuronal normal, lo que realiza ReLU es a los valores negativos los hace 0 y con los valores positivos hace una función lineal y no cambia el volumen de la entrada.

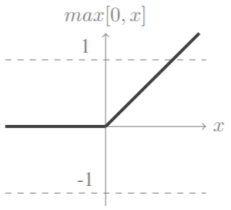


Figura 15. Función ReLU

- **Pool:** esta capa realiza una operación de re-muestreo a lo largo de las dimensiones espaciales (ancho, largo), para reducir su espacio, entonces si se tiene una entrada de 128x128x4 se pasa por una capa Pool se tiene una 32x32x4.

- **Capa conectada completamente (FCL):** esta capa se encarga de aprender los pesos W, tal cual como en las redes neuronales normales, funciona re-modelando la entrada de datos que tiene a un vector por ejemplo si un tensor es de 128x128x5 se hace un vector de 1x (32x32x4) = 1x4096, entonces cada neurona en esta capa está relacionada con 4096 pesos y hay que agregarle un bias para cada neurona. Se entrena con el mismo principio de descenso de gradiente que una red neuronal normal.

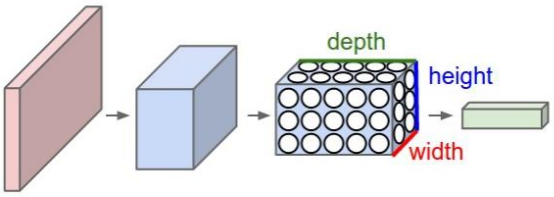


Figura 16. Red neuronal convolucional

Como un ejemplo ilustrativo del funcionamiento de una capa convolucional se define las siguientes matrices que contienen las capas RGB de una imagen:

Capa R

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	0	1	0	0	0
0	0	2	2	1	1	0
0	0	1	2	2	0	0
0	0	1	0	1	2	0
0	0	0	0	0	0	0

Capa G

0	0	0	0	0	0	0
0	0	2	1	2	1	0
0	0	0	2	0	2	0
0	1	2	0	1	0	0
0	0	1	0	1	0	0
0	1	1	2	0	2	0
0	0	0	0	0	0	0

Capa R

0	0	0	0	0	0	0
0	0	2	1	1	1	0
0	0	1	2	2	1	0
0	1	1	0	1	1	0
0	0	0	2	2	1	0
0	1	0	2	2	2	0
0	0	0	0	0	0	0

Se observa que los bordes de las matrices existen únicamente ceros, esta es una técnica que se llama relleno de cero que permite controlar el tamaño de salida después del filtro. La imagen va a pasar por una capa convolucional definida a continuación:

Filtro R

−1	−1	−1
0	−1	−1
0	1	−1

Filtro G

1	1	1
0	1	−1
−1	0	0

Filtro B

0	1	1
0	0	1
1	0	0

Se procede a realizar la operación convolucional a lo largo y ancho de la imagen de entrada:

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	0	1	0	0	0
0	0	2	2	1	1	0
0	0	1	2	2	0	0
0	0	1	0	1	2	0
0	0	0	0	0	0	0

*

−1	−1	−1
0	−1	−1
0	1	−1

$$\begin{matrix}0&0&0&0&0&0&0\\0&0&2&1&2&1&0\\0&0&0&2&0&2&0\\0&1&2&0&1&0&0\\0&0&1&0&1&0&0\\0&1&1&2&0&2&0\\0&0&0&0&0&0&0\end{matrix} * \begin{matrix}1&1&1\\0&1&-1\\-1&0&0\end{matrix}$$

$$\begin{matrix}0&0&0&0&0&0&0\\0&0&2&1&1&1&0\\0&0&1&2&2&1&0\\0&1&1&0&1&1&0\\0&0&0&2&2&1&0\\0&1&0&2&2&2&0\\0&0&0&0&0&0&0\end{matrix} * \begin{matrix}0&1&1\\0&0&1\\1&0&0\end{matrix}$$

El tensor de salida está definido por un mapa de características de 3x3x1:

$$\begin{matrix}3&3&4\\2&2&4\\2&7&1\end{matrix}$$

Para entrenar una red neuronal convolucional se necesita tener tres componentes.

- **Conjunto de entrenamiento:** sirve para modificar los pesos de la red, así como para modificar sus parámetros como el bias, se utiliza un número bastante amplio de datos para realizar un entrenamiento en el caso de las redes convolucionales son imágenes.
- **Conjunto de validación:** permite comparar el desempeño de diferentes arquitecturas de red como lo son ResNet, VGG16, GoogLenet, entre otras todo depende del caso que se esté tratando como detección de objetos o clasificación de imágenes.
- **Conjunto de prueba:** evalúa el desempeño de la red con datos totalmente nuevos en el caso de las redes convolucionales son nuevas imágenes.

2.6 Distancia Euclidiana

También conocido como norma L2, calcula la distancia entre dos puntos en el espacio Euclidiano, si se tiene los puntos $p = (p_1, p_2, .., p_n)$ y $q = (q_1, q_2, .., q_n)$, cabe recalcar que la operación anterior se realiza para un espacio Euclidiano de n-dimensiones, con la siguiente formula se halla la distancia pq.

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

Una forma de extraer las características de un rostro es medir algunas de sus distancias por ejemplo desde el centro de la nariz al ojo izquierdo, desde el centro de la boca al centro de la nariz, etc. como se puede observar en la figura 2, pueden existir varios puntos que mediante la fórmula $d(p,q)$ se computa un solo resultado que caracteriza a la cara (Schroff, Dmitry , & Philbin, 2015).

2.7 Clasificadores.

Dentro de los algoritmos de clasificación existen varios, uno de ellos muy utilizado por su alto grado de precisión es el Support Vector Machine (SVM), es una técnica de machine learning sirve tanto para la clasificación como para la regresión, se basa en un aprendizaje supervisado. Una de las ventajas de las SVM es predecir el comportamiento de un sistema que se compone de múltiples variables (Alonso, 2007).

Para aclarar el funcionamiento del SVM se realiza un ejemplo, si se tiene puntos distribuidos en un plano x-y como en la figura 17, estos son datos etiquetados para el caso son cuadrados rojos y círculos azules.

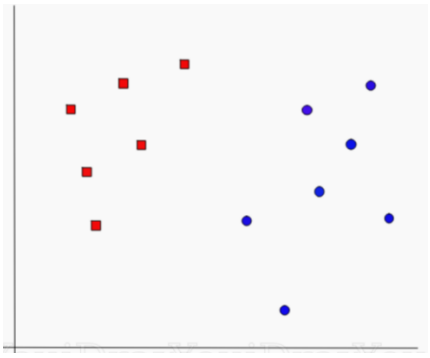


Figura 17. Datos de entrenamiento de un SVM

La idea del SVM es generar una línea para separar estos datos etiquetados de una forma óptima, esto quiere decir que se crea el máximo margen de distancia entre las clases, se toma como referencia para realizar esta tarea los puntos más cercanos entre cada grupo que se denominan vectores de soporte, como se observa en la figura 18. Separar las clases sirve para el momento que llegue un nuevo valor de **x** el SVM pueda predecir a que clase pertenece para el ejemplo cuadrados rojos o a los círculos azules.

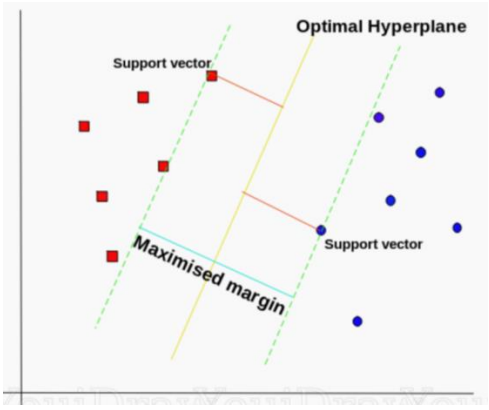


Figura 18. Separación de datos

Cabe recalcar que, si tenemos datos como en la figura 18, se genera una línea de separación, si los datos de entrenamiento generan una superficie en tres dimensiones estos estarían separados por un plano, en el caso de que existan un número n-simo de variables de entrenamiento se produce un hiperplano (se le denomina así para cualquier caso) de separación.

Un hiperplano viene dado por la siguiente ecuación.

$$g(x) = w \cdot x + b$$

La clasificación en el momento que un nuevo dato ingresa al SVM está definida por:

$$g(x) \geq 1 \ \ \forall \epsilon \ Cuadrados \ Rojos$$

$$g(x) \leq -1 \ \ \forall \epsilon \ Circulos \ Azules$$

Donde:

- w: es un vector de pesos
- x: es un vector de entradas.
- b: es un bias.

Existen casos en dónde los datos no son linealmente separables como se observa en la figura 19 y se tienen que usar algunos artificios matemáticos conocidos como kernels para poder separar los datos mediante un hiperplano (Alonso, 2007).

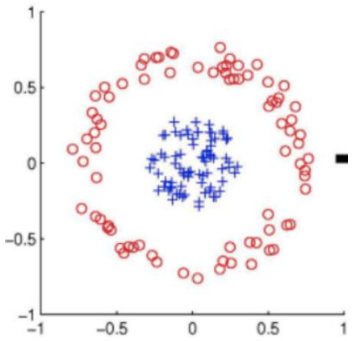


Figura 19. Datos linealmente inseparables

Para el caso de la figura 19, se debe usar un kernel para mapear de forma no lineal los datos de entrada a un espacio de alta dimensión, donde se vuelven linealmente separables, como se observa en la figura 20.

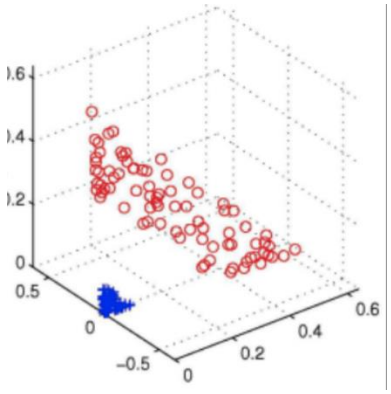


Figura 20. Datos separados haciendo uso de un Kernel

Retomando el ejemplo de la sección 2.6, se puede clasificar a que persona pertenece una cara, si se tienen como dato para entrenar una SVM la distancia euclidiana. En referencia a la figura 18 suponiendo que los cuadrados rojos son las distancias euclidianas de una persona y los círculos azules los de otra persona se puede clasificar que persona es si se tiene una nueva entrada gracias a este algoritmo de machine learning.

2.8 Wiegand.

Para leer las tarjetas RFID, se utiliza el protocolo Wiegand se compone de líneas de alimentación, datos y señalización esta se utiliza para manejar los leds de la lectora. Para los datos usa dos líneas Data 0 y Data 1, los cuales envían un 1 lógico colocando un pulso negativo en la línea Data 1 siguiendo el mismo procedimiento para la línea Data 0 como se puede apreciar en la figura 21. Las tarjetas se codifican en formato Wiegand xx, por tanto, para que puedan ser captadas por la lectora esta debe tener un protocolo de comunicación Wiegand xx. Uno de los formatos Wiegand más usado es el de 26 bits, en la tabla 2 se puede observar cómo se distribuyen los bits para un formato Wiegand 26.

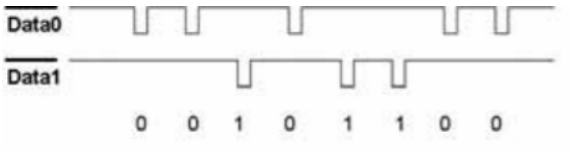


Figura 21. Envió de datos

Tabla 2
Wiegand 26.

	1 bit	**2-9 bits**	**10-25 bits**	**26 bit**
Wiegand 26	Paridad par de los próximos 12 bits.	Facility Code.	Identificación.	Paridad impar de los 12 bits anteriores a este.
Numero	0-1	0-255	0-65535	0-1

Por ejemplo, se tiene 1 10101110 0101 100110100110 1, donde el primer bit es 1 porque en los 12 subrayados existen siete unos por lo que paridad par es 1, el facility code del ejemplo seria 10101110 0 = 174, el código de identificación es 101 100110100110 = 22950 y por último la paridad impar se analiza en los bits que están con negrilla y curvos en donde hay seis unos por lo que la paridad impar es 1 (Corporation H. , 2006).

2.9 Python.

Lenguaje de programación creado por Guiado Van Rossem, de fácil aprendizaje, cuenta con estructuras de datos eficientes y de alto nivel, tiene un enfoque de programación orientada a objetos, además su sintaxis es muy accesible para usuarios nuevos, se utiliza un lenguaje C o C++. Se puede instalar un gran número de librerías para usar diferentes funcionalidades como la visión artificial, manejo de matrices, etc. La curva de aprendizaje del lenguaje es corta (Rossum, 2009).

2.10 OpenCV.

Librería abierta para la visión por computador, tiene como componente un sin número de algoritmos encapsulados para procesar imágenes de forma accesible para los usuarios. Dentro de esta librería se puede encontrar algoritmos de machine learning, filtraje de imágenes, entre muchos otros, listos para ser utilizados. Está disponible para diferentes plataformas como Windows, Linux, OS X, Android, iOS, por nombrar las más importantes (Mordvintsev, 2017). Se compone de 5 elementos claves los que se muestran en la figura 22, que ayudan para tratar imágenes es importante saber que OpenCV está enfocado a realizar el procesamiento con imágenes en tiempo real.

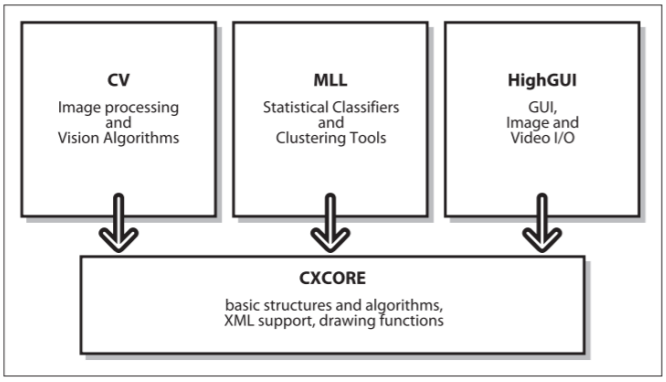


Figura 22. Componentes OpenCV

2.11 Xeoma.

Xeoma es software libre para poder acceder a cámaras de video-vigilancia analógicas, IP, web, etc. tiene un entorno muy amigable al usuario, permite la detección de las cámaras que estén en la red de la computadora donde se encuentra corriendo Xeoma con un procedimiento muy accesible para el usuario.

CAPITULO III

DESARROLLO DEL PROTOTIPO

3.1 Análisis

El proyecto, como primera parte tiene que contar con un **registro de personal** la segunda parte engloba a un **reconocimiento facial** más la **lectura de tarjetas RFID** que a fin de cuentas es el control de acceso biométrico. Con estas características nombradas el prototipo puede contar con la ventaja de usar dos métodos de verificación, buscando un aumento de la fiabilidad.

3.2 Registro de personal

3.2.1 Descripción general del proceso

En la primera parte del trabajo es importante recaudar los datos de los diferentes usuarios que van a utilizar el sistema. Las variables a registrar de cada usuario son las que se indican en la tabla 3. Estos datos se pueden obtener mediante una interfaz gráfica, que consta de las siguientes pantallas una en donde se ingresa los datos personales del usuario y otra en donde se puede capturar fotos de la persona, en la figura 23 se observa el diagrama de componentes del programa.

Tabla 3
Variables a considerar para el registro de personal.

Numero	Variable
1	Nombre Completo
2	Cédula de identidad
3	Edad
4	Cargo que desempeña
5	Número de tarjeta RFID
6	Fotos de la cara del usuario en diferentes ángulos

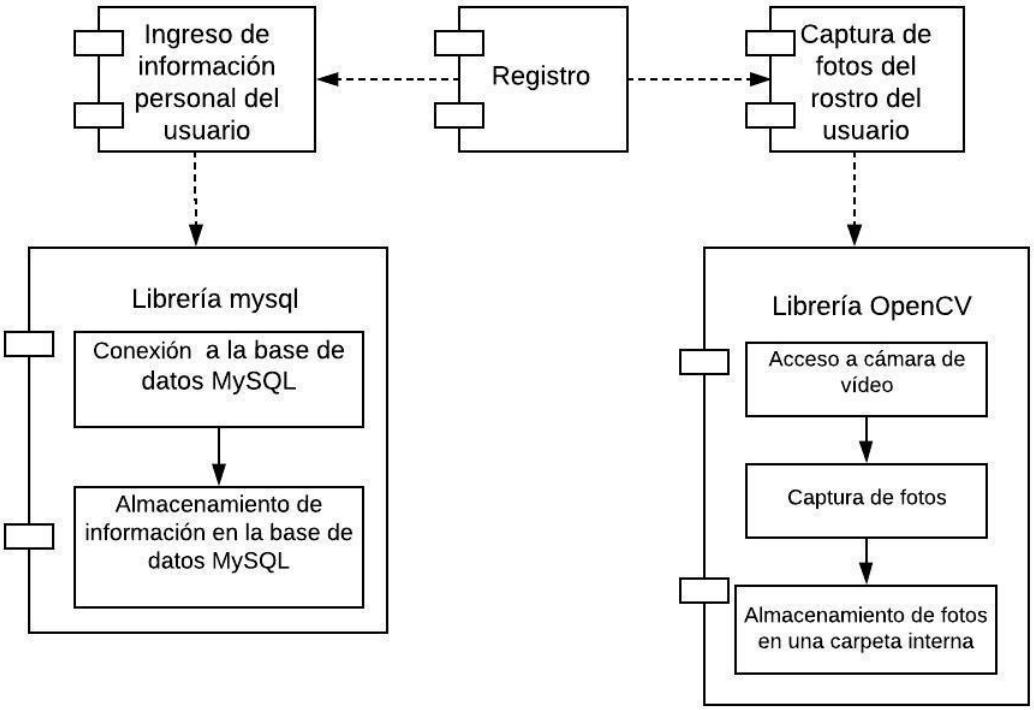


Figura 23. Diagrama de componentes Registro

3.2.2 Material necesario

Para esta sección, los componentes tanto de hardware y software, se detallan en la tabla 4.

Tabla 4
Material necesario para el registro de personal.

Material	Función
Python	Lenguaje de programación
OpenCV	Librería para el tratamiento de imágenes con diferentes métodos para el lenguaje de programación Python.
TKinter	Liberia para crear interfaces de usuario GUI con lenguaje de programación Python.
Cámara IP	Se utiliza para la captura de fotografías en diferentes ángulos la cara de la persona.
Xeoma	Programa con el cual podemos acceder a la dirección rstp (real time streaming protocol), o

	html (hipertext Transfer Protocol), de una cámara IP
Servidor local	Se utiliza para cargar los diferentes registros en una base de datos MySQL.
MySQL	Base de datos para grabar las variables que se consideran para el registro de personal.
Sistema operativo Linux	El desarrollo del prototipo se los trabajo bajo este SO.

Características técnicas de los materiales: el software y hardware que se utiliza en esta sección es muy importante porque es el mismo para el de las siguientes secciones.

Tabla 5
Características técnicas del software.

Software	Característica
SO Linux	Distribución Ubuntu 18.10
Python	Versión Python 3.3
Librería OpenCV	Versión 3.0
Librería mysql	Versión 2.4
Xeoma	Versión gratuita
Servidor local	XAMPP 16.04 para Linux versión gratuita
MySQL	Versión 5.0.12

Tabla 6
Características técnicas del hardware.

Hardware	Característica
Cámara IP, HikVision	Cámara tipo domo. Resolución de imagen 960P color CMOS, 1.3 megapíxeles, 1280X960.

Soporta protocolo ONVIF.
Soporta hasta 45 FPS.

3.2.3 Funcionamiento

Dentro de la pantalla de ingreso de información personal se llenan los campos requeridos, todos obligatorios, el último dato necesario son las fotografías de la cara de la persona que se está registrando, para esto se toman cinco fotos con el programa, por lo que debemos tener conectada una cámara IP, se tiene una separación de un tiempo moderado para cada captura de fotos, así el usuario podrá cambiar de posición con lo que se gana diferentes ángulos de la cara. Además de las cinco fotos se debe solicitar al usuario una cantidad de diez fotos extras, esta se las puede ingresar de forma manual, para mejorar la calidad de la base de datos de fotografías.

3.2.4 Procedimiento

Como se indica en el diagrama de componentes se necesita acceder a una cámara para poder capturar las imágenes de la persona que se está registrando, se usa una cámara IP de acuerdo a la descripción de materiales de la tabla 4, el primer paso para que OpenCV pueda leer las imágenes es tener la dirección IP de la cámara, el programa Xeoma ayuda para conseguir la misma, de manera automática tan solo poniendo en marcha el software, basta que la cámara IP esté conectada a la misma red donde se ejecuta Xeoma. En la figura 24 se puede apreciar al programa Xeoma trasmitiendo la señal de la cámara IP.

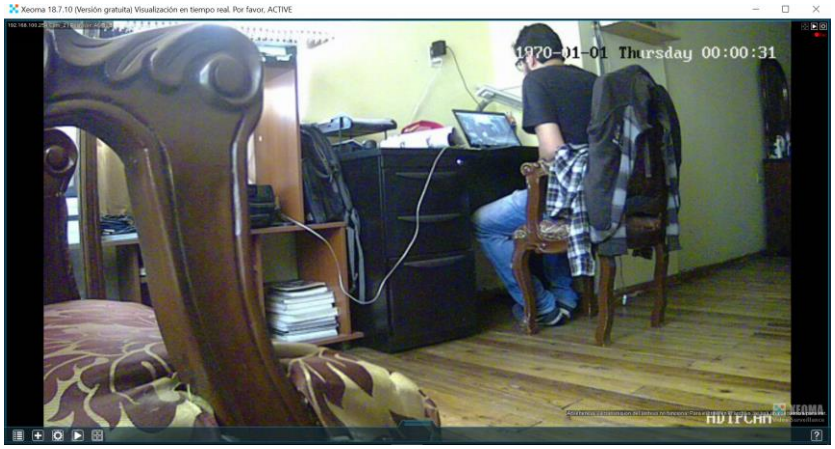


Figura 24. Xeoma trasmitiendo cámara IP

Este programa ayuda a encontrar la dirección de real time streaming protocol o más conocida como rtsp que es un protocolo para la trasmisión de información en tiempo real, para hallar la misma en Xeoma hay que dar clic en la parte inferior izquierda en herramientas, donde aparece una pestaña, en la cual se debe dar un clic en la parte de Parámetros de flujo de vista previa es ahí donde se encuentra la dirección rtsp como se observa en la figura 25, que será utilizada por OpenCV.

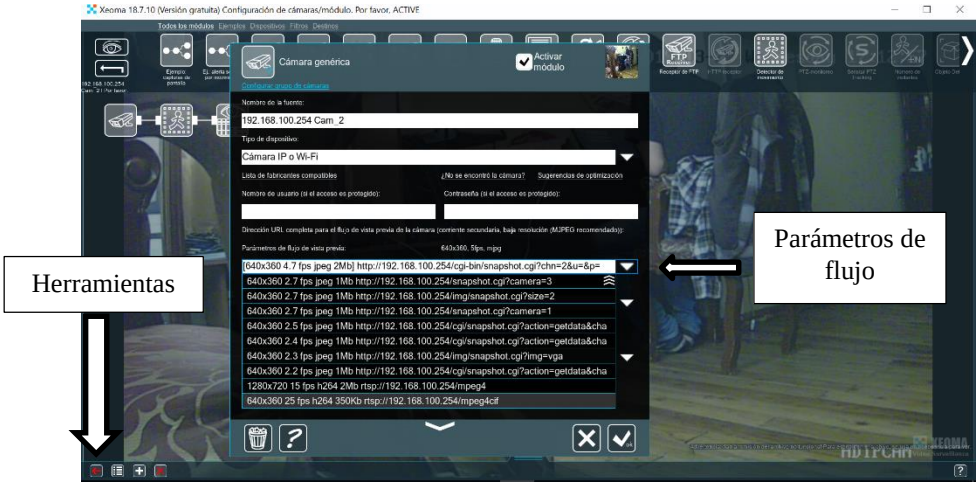


Figura 25. Dirección rtsp

Una vez con la dirección rtsp, hay que llamar a la cámara en Python para lo cual se debe en el script importar la librería OpenCV con el siguiente comando.

```
import cv2
```

Para llamar a la cámara en el script se necesita de un comando de la librería OpenCV.

```
cap = cv2.VideoCapture('rtsp://192.168.100.254/mpeg4cif')
ret, frame = cap.read()
```

La variable de la cámara es frame, cada vez que se le quiera mostrar la cámara se necesita usar el siguiente comando.

```
cv2.imshow('frame',frame)
```

Para guardar las imágenes del registro se necesita de una dirección en este caso es la carpeta con el nombre de los usuarios que se van registrando.

```
img = cv2.imread('C:\Documentos\FR\Israel','1.jpj',frame,)
```

Para subir la información a MySQL, que se necesita por cada usuario referenciada en la tabla 3 el primer paso es habilitar los servicios de XAMPP para Linux, los comandos que se utilizan en la terminal de Linux se los puede observar en la figura 26.

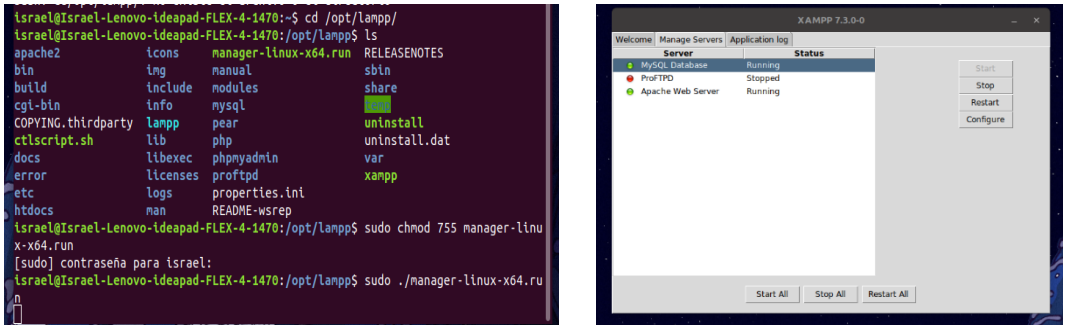


Figura 26. Servicios MySQL

Una vez con los servicios activos se debe utilizar un comando SQL para grabar los datos, cabe recalcar que la base de datos ya debe estar creada de forma manual. Esto quiere decir que exista una tabla con sus respectivos campos a donde se puede apuntar desde Python. En la figura 27 se puede observar que dentro de la tabla usuarios existen los campos referenciados en la tabla 3, menos uno el campo que tiene que ver con la foto.

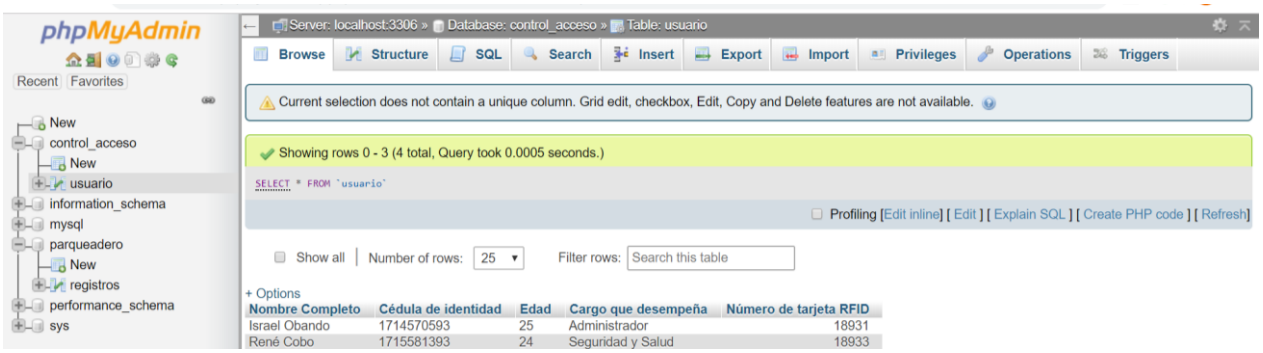


Figura 27. Base de datos registro

Para grabar los datos directamente desde Python se necesita importar la librería para conectarse con MySQL.

```
import mysql.connector
```

Se debe crear un conector a la base de datos, este tiene que ver con el puerto que este asignado el servidor MySQL.

```
mydb=mysql.connector.connect(user="root",password="root",host="localhost",port="3306",database="control_acceso")
mycursor = mydb.cursor()
```

Para capturar los datos en la base de datos.

```
sql = "INSERT INTO usuarios(Nobre_Completo" \
      ", Cédula_identidad, Edad,Cargo_que_desepeña " \
      "Número_de_tarjeta_RFID) VALUES (%s, %s, %s, %s, %s)"
val = (name,ci,age,charge,RFID)
```

3.3 Control de Acceso biométrico

3.3.1 Descripción general del proceso

Para desarrollar el prototipo primero se realiza el reconocimiento facial el cual cuenta con varias etapas que se detallan en la siguiente sección, el segundo paso a seguir es tomar la lectura de la tarjeta RFID y como último paso se realiza la integración de los dos programas en uno que realiza el control de acceso con la verificación de estos dos datos antes nombrados.

3.4 Reconocimiento facial.

3.4.1 Descripción general del proceso.

El reconocimiento facial en esta sección se centra en entrenar un algoritmo de machine learning con la base de datos de fotografías que se obtiene en el registro, para esto se hace uso de dos programas referenciados en la tabla 7. Los pasos que se llevan a cabo en esta sección se resumen en la figura 28.

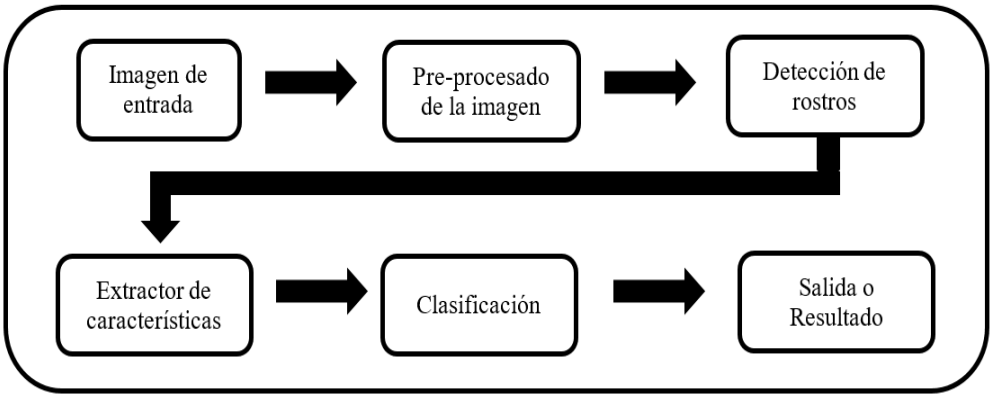


Figura 28. Reconocimiento Facial Diagrama de Bloques

Tabla 7
Estructura del programa de reconocimiento facial.

Programa	Entrada	Salida
extractor_características	Imágenes de la base de datos	Archivo denominado **output**, con las características de cada usuario definida por vectores
entrenar_modelo	Archivo **output**	Archivo denominado **output1,** entrenamiento del SVM.

3.4.2 Material necesario.

En este caso se utilizan los mismos materiales que en la **tabla 4** exceptuando el servidor local y MySQL .

Características técnicas del material.

Las características de los materiales ya se ven expresadas en la **tabla 5 y 6** exceptuando el servidor local y MySQL.

3.4.3 Funcionamiento

Lo primero que se debe hacer es crear una base de datos con un mínimo de quince fotografías de las personas que van a usar el sistema, el siguiente paso es almacenar en carpetas que contengan el nombre de cada uno de los usuarios. Es importante crear una carpeta para un usuario desconocido, la cual contiene caras de diferentes personas que no vayan a tener acceso al lugar donde se coloque el control acceso. Una vez con una base de datos fiable se procede a ejecutar el programa llamado **extractor_características**, la función principal de este algoritmo es ir a la base de datos de fotografías, pre-procesar las imágenes, buscar un rostro en cada una de las fotos, extraer los vectores característicos de cada una de las caras y por ultimo como

salida se tiene un archivo que contiene toda la información del aspecto de cada individuo contenida en un vector de 128 elementos, llamado **output**.

El siguiente paso a tomar es entrenar un modelo de machine learning, en este caso se usa un Support Vector Machine (SVM), para lograr el objetivo de esta sección se debe ejecutar el programa llamado **entrenar_modelo** este ayuda a reconocer a realizar una clasificación de las personas de acuerdo a sus vectores característicos mediante el archivo **output,** con el algoritmo extractor_características. Se inicia con la carga del archivo denominado **output**, como segundo punto se debe entrenar el SVM dotándole de la información suficiente de que datos pertenecen a que persona con respecto al archivo **output**, para finalizar el programa entrega un archivo que contiene la información del SVM ya entrenado llamado **output1**.

3.4.5 Programa extractor_características.

En la figura 29 se observa un diagrama de componentes del programa extractor_características, en donde se identifican los pasos que se realiza en el script para extraer las características de los rostros de la base de datos.

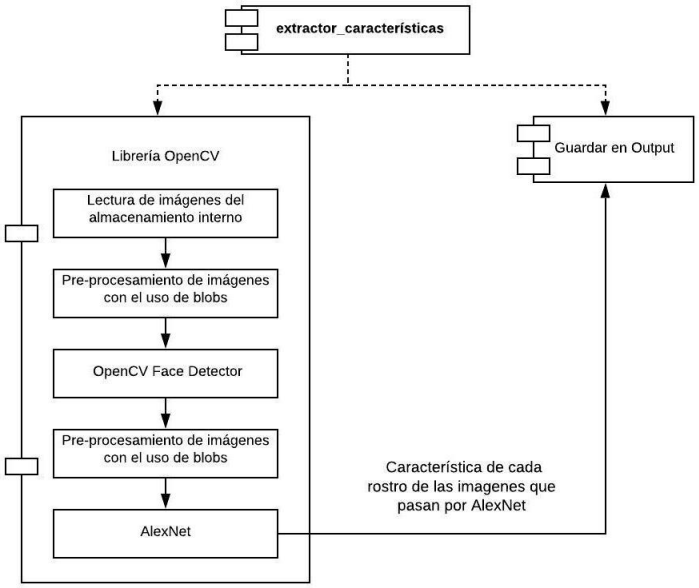


Figura 29. Diagrama de componentes extractor_características

3.4.5.1 Funcionamiento.

Hay que tener en cuenta que para realizar la detección de rostros y la extracción de características se usan redes neuronales pre-entrenadas, la razón emplear este tipo de redes es porque el tiempo de respuesta, así como el procesamiento de nuestro PC es menor (Rosebrock, 2018). Hacer uso de una red neuronal pre-entrenada es simplemente volver a utilizar los pesos de redes neuronales sofisticadas que tienen buenos resultados y normalmente son realizadas por centros de investigación como Google o por universidades, para generar clasificadores de un mejor performance. Una vez que se selecciona una red pre-entrenada, se toman en cuenta los parámetros que necesita de entrada, respetar estos es de gran importancia porque ayuda a que la respuesta a la salida de la red neuronal no tenga muchos errores.

Para el caso de la **detección de rostros**, la red neuronal pre-entrenada es una provista por un modelo Caffe deep learning, cuando se utiliza esta herramienta los valores de entrada que necesita de la imagen a procesar son los de la tabla 8.

Tabla 8
Parámetros del modelo OpenCV Face Detector

Modelo	Escala	Tamaño (largo x ancho) Pixeles	Resta de media	Orden de canales
OpenCV Face Detector	1	300x300	104 – 177 - 123	BGR

Fuente: (Kurtaev & Alekhin, 2018)

Para el caso de **la extracción de características,** la red neuronal pre-entrenada es una provista por un modelo de Torch llamado, openface_nn4.small2.v1.t7, los valores de entrada preestablecidos para esta red neuronal pre entrenada son los de la tabla 9.

Tabla 9
Parámetros del modelo OpenCV Face Detector

Modelo	Escala	Tamaño (largo x ancho)	Resta de media	Orden de canales
AlexNet	1.0 / 255	96x96	0 – 0 - 0	RGB

Fuente: (Kurtaev & Alekhin, 2018)

Con el conocimiento previo de los parámetros que se necesitan para cada red neuronal pre-entrenada, el siguiente paso es hacer el pre-procesamiento de las imágenes de la base de datos de los usuarios del control de accesos para estandarizarlas de acuerdo a la red pre-entrenada que se vaya a utilizar.

Pre-procesado

Una técnica muy utilizada para el procesamiento digital de imágenes es la representación BLOB (Binary Large Objects), esta ayuda a disminuir los efectos que tiene la luz sobre una fotografía (Rosebrock, 2018). Con la ayuda de la librería OpenCV, se consigue la extracción de un BLOB de una manera simple únicamente utilizando la siguiente línea de código.

```
cv2.dnn.blobFromImage
```

Al utilizar este comando, se necesitan 4 parámetros la imagen, el factor de escalamiento, el tamaño de la imagen, los valores de la media y el tipo de canal a continuación se muestra como se utiliza esta instrucción.

```
blob = cv2.dnn.blobFromImage(image, scalefactor=1.0, size, mean, swapRB=True)
```

Al utilizar un cv2.dnn.blobFromImage se transforma la imagen a un formato estándar para que ingrese a una red neural pre entrenada y realiza las acciones detalladas a continuación.

- **Mean subtraction:** realiza una operación de resta entre los valores pre-establecidos de mean que para el caso de las redes pre-entrenadas que se utiliza se los puede encontrar en la tabla 8 y 9 respectivamente en la columna de Resta de media ($R_{mean}, G_{mean}, B_{mean}$), de la cantidad promedio de RGB de una imagen ($\mu R, \mu G, \mu B$), matemáticamente hablando se ejecuta lo siguiente.

$$R = R_{mean} - \mu R$$

$$G = G_{mean} - \mu G$$

$$B = B_{mean} - \mu B$$

- **Scaling:** Normaliza los valores de RGB, dividiéndolos para una constante sigma para el caso de las redes neuronales pre-entrenadas se referencia a las tablas 8 y 9 donde se encuentra el valor de sigma en la columna de Escala.

$$R = \frac{(R_{mean} - \mu R)}{\sigma}$$

$$G = \frac{(G_{mean} - \mu G)}{\sigma}$$

$$B = \frac{(B_{mean} - \mu B)}{\sigma}$$

- **Size:** El tamaño de la imagen que está ingresando en cuando a su ancho y largo de a acuerdo a sus pixeles.
- **SwapRB:** las imágenes que se ingresan tienen un canal RGB, si es necesario cambiarle a uno BGR, se usa un True, de no ser necesario se usa un False.

Para la detección de caras hay que generar un blob que use los parámetros de la **tabla 7.**

```
imageBlob = cv2.dnn.blobFromImage(
    cv2.resize(image, (300, 300)), 1.0, (300, 300),
    (104.0, 177.0, 123.0), swapRB=False, crop=False)
```

Para la extracción de características hay que generar un blob que se acople a los parámetros de la **tabla 8.**

```
faceBlob = cv2.dnn.blobFromImage(face, 1.0 / 255,
    (96, 96), (0, 0, 0), swapRB=True, crop=False)
```

Detección

Para detectar rostros en una imagen se usa una red neuronal pre-entrenada, que lleva el nombre de OpenCV Face Detector, que se basa en un detector de objetos llamado Single Shot Detector (SSD), este por su parte tiene una arquitectura que en sus primeras capas se la puede variar usando diferentes modelos para extraer las características de las imágenes, para este modelo en específico se usa como base ResNet. Este tipo de redes neuronales después de entrenarlas y probarles se puede generar una especie de caja negra en donde se reciben imágenes de un cierto formato y a la salida se obtendrá el resultado para el que fueron creadas, para el caso de OpenCv Face Detector se utiliza Caffe.

Single Shot Detector (SSD)

Es una red neuronal convolucional (CNN) para detectar objetos en una imagen, la SSD cumple con el siguiente procedimiento para realizar la detección:

- Primero se debe entrenar la red, para realizar esta tarea se necesita de datos etiquetados, esto quiere decir que de un conjunto de imágenes para entrenamiento se debe seleccionar la región de interés (ROI) de cada una de estas, a la ROI seleccionada se le asigna un nombre especifico. Por ejemplo, en la figura 30 se etiqueta la imagen de una cara con el nombre del sujeto y las coordenadas en x e y del rectángulo que encierra a la persona.

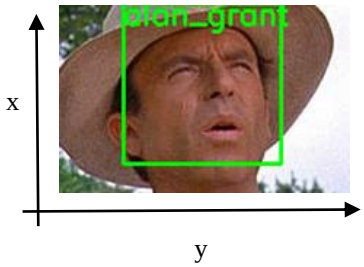


Figura 30. Etiquetado de una imagen

- Una vez que la red se entrena, al momento que se ponga a prueba la SSD con una nueva imagen, en las primeras capas de la CNN se extraen las características como bordes, formas, etc. resultando en un mapa de características, el siguiente paso que realiza es un barrido por todos los valores del mapa, seleccionando regiones de interés y encerrando estas en cuadros delimitadores, como se observa en la figura 31 los cuadros azules son las regiones de interés.

Figura 31. ROI

- El siguiente paso es extraer las características de los cuadros delimitadores, la red evalúa si existen objetos parecidos a los de su entrenamiento puntuando los cuadros con valores de porcentaje (0%-100%).
- Como último paso los cuadros delimitadores superpuestos se combinan en un solo cuadro delimitador de acuerdo a la puntuación que obtiene la SSD como se puede observar en la figura 32.

Figura 32. Detección de objetos.

Cabe recalcar que en las primeras capas de la red SSD se usa arquitecturas estándar. Para OpenCV Face Detector se usa ResNet (Deep Residual Learning for Image Recognition) como base, con la que se extraen características importantes de la imagen, en el caso de la detección de caras se enfoca en los bordes y las figuras que la imagen contengan (Liu, y otros, 2016).

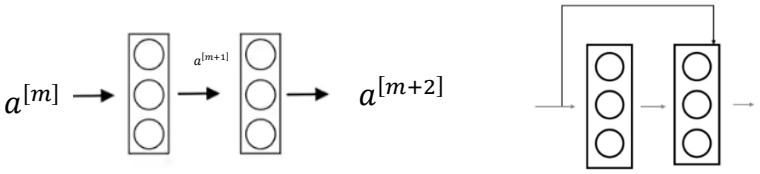
Las imágenes utilizadas en el entrenamiento de la red para la detección de caras están contenidas en una base de datos de acceso libre llamada Labeled Face in the Wild (LFW) que cuenta con alrededor de 13233 imágenes correctamente etiquetadas. Es una gran ventaja usar estas imágenes por que cumplen con los requerimientos del SSD, la base de datos usa un cuadro que encierra la cara de las personas en miles de fotografías, en la figura 33 se puede apreciar un ejemplo de las imágenes que se encuentran guardadas en LBF (Huang, Mattar, Berg, & Learned-Miller, 2015). OpenCV Face Detector para su entrenamiento usa 1500 imágenes para

entrenar, 500 usa para validar y 927 usa para hacer pruebas (Huang, Mattar, Berg, & Learned-Miller, 2015).

Figura 33. Labeled Face in the Wild

ResNet (Deep Residual Learning for Image Recognition)

Es una CNN, que se la puede usar para la extracción de características, clasificación o la detección de objetos en una imagen dependiendo de su arquitectura y entrenamiento. La principal característica de una ResNet es mitigar el error de entrenamiento que sufren las CNN, usando un salto de conexión entre las capas que componen una red, como se observa en la figura 34.



Sin salto de conexión Salto de conexión

Figura 34. Salto de conexión

El fundamento de ResNet es usar un artificio matemático para aproximar el resultado que se obtiene cuando no existe un salto de conexión. Como ejemplo ilustrativo, en la figura 34 para obtener al resultado $a^{[m+2]}$ se sigue el proceso que se muestra en la figura 35.

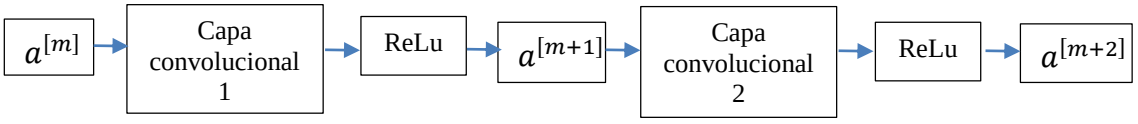


Figura 35. Procedimiento sin salto de conexión

La salida de la capa convolucional 1 tiene una ecuación de la siguiente forma.

$$n^{[m+1]} = W^{[m+1]}a^{[m]} + b^{[m+1]}$$

Luego entra a una capa ReLu donde se aplica una función de activación para normalizar los valores que se tengan en $z^{[l+1]}$.

$$a^{[m+1]} = g(n^{[m+1]})$$

A continuación, este resultado ingresa a la capa convolucional 2 en donde se genera la salida.

$$n^{[m+2]} = W^{[m+2]}a^{[m+1]} + b^{[m+2]}$$

Por ultimo ingresa a una capa ReLu $z^{[l+2]}$, resultando.

$$a^{[m+2]} = g(n^{[m+2]})$$

Al momento de aplicar un bloque residual el resultado se refleja a continuación.

$$a^{[m+2]} = g(z^{[m+2]} + a^{[m]})$$

Específicamente ResNet se usa como una red base en la arquitectura SSD para extraer características de una fotografía. Esta red base se usa para realizar una transferencia de aprendizaje (transfer learning), esto significa que, ResNet aprende características importantes

de una imagen y envía un mapa de características a las siguientes capas que estén fuera de la ResNet de tal forma que se haga mucho más eficiente el paso de información por las capas restantes del SSD, con lo que se gana tiempo de procesamiento de la CPU, además la cantidad de datos que se requieren para entrenar una red para la detección de objetos se reduce. La idea general de la transferencia de aprendizaje es utilizar los conocimientos aprendidos de las tareas para las que se dispone de una gran cantidad de datos etiquetados en configuraciones donde solo hay pocos datos etiquetados. Crear datos etiquetados es costoso, por lo que es clave aprovechar de manera óptima los conjuntos de datos existentes (He, Zhang, Ren, & Sun, 2014).

Procedimiento.

En este caso vamos a ingresar imágenes con el formato de referencia de la tabla 8, de las cuales después de cruzar por la red neuronal se espera obtener un mapa de características en donde se encuentre aislada la cara de todo el resto de la imagen. Se usan dos archivos para hacer correr la red, uno contiene toda la arquitectura de la red como las capas convolucionales, las de pool, las de activación y las de conexión total, así como la red base que se va a utilizar, en este caso ResNet-10 y el otro es un fichero de Caffe donde se alberga los pesos y bias de la CNN para la detección de rostros. En la figura 36 se aprecia cómo está formada la red neuronal convolucional para la detección de rostros. En la tabla 10 se aprecia los archivos nombrados que son de uso libre. Estos dos archivos se los puede descargar ingresando a la página web: https://github.com/opencv/opencv/tree/master/samples/dnn/face_detector.

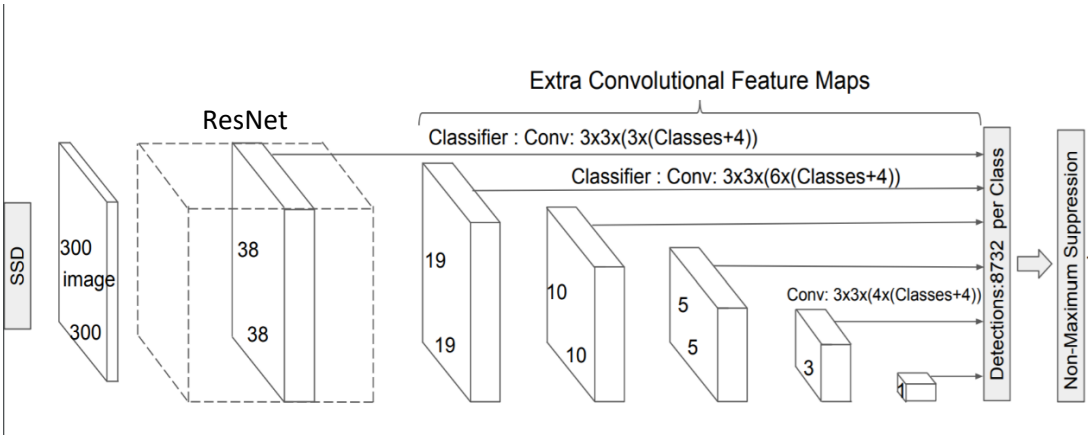


Figura 36. SSD arquitectura

Tabla 10
Red neuronal pre-entrenada SSD

Archivo	Descripción
res10_300x300_ssd_iter_140000.caffemodel	Modelo Caffe de una Red neuronal pre entrenada, para la detección de rostros
deploy.prototxt	Arquitectura de la red pre entrenada

En Python lo que se debe cargar es el modelo Caffe, de la siguiente forma. Primero se necesita dotarle al programa de la dirección en donde están guardados los archivos referenciados en la tabla 10.

```
ap.add_argument("-d", "--detector", required=True,
    help="path to OpenCV's deep learning face detector")
```

Una vez que Python sabe dónde se encuentran, se debe buscar con un nombre especifico a los archivos.

```
protoPath = os.path.sep.join([args["detector"], "deploy.prototxt"])
modelPath = os.path.sep.join([args["detector"],
    "res10_300x300_ssd_iter_140000.caffemodel"])
```

Una vez cargado el modelo, se debe leer la red convolucional con su respectiva arquitectura y pesos, por lo tanto, se usa el comando cv2.dnn.

```
detector = cv2.dnn.readNetFromCaffe(protoPath, modelPath)
```

Con el modelo detector, ya se le puede ingresar las imágenes pre-procesadas como lo indica la tabla 8.

```
detector.setInput(imageBlob)
detections = detector.forward()
```

Por cada imagen que pase por la red neuronal se genera un cuadro donde se encierra el objeto que tenga la mayor probabilidad de ser una cara. Como siguiente y último paso se debe discriminar todo lo que no esté dentro del nombrado cuadro y generar una nueva imagen.

```
box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
(startX, startY, endX, endY) = box.astype("int")
face = image[startY:endY, startX:endX]
```

Extractor de características

Para extraer las características de una cara y asociarle a su correspondiente nombre se utiliza una red neuronal convolucional pre-entrenada, llamada FaceNet que computa 128 medidas de una cara, la entrada a la red es una imagen que contenga un cuadro en donde se encuentre la cara. La técnica que se utiliza para calcular estas medidas se llama deep metric learning que consiste en generar un vector de características reales en lugar de realizar cuadros de delimitación.

Esta red neuronal fue creada con el único propósito de reconocer personas mediante sus rostros,

sigue el siguiente proceso, la imagen de la cara que ingresa a la red pasa por una CNN el resultado son las 128 medidas que se normalizan mediante la función L2 de la cual se obtiene la incrustación (embedding) de cada cara y por ultimo usa una función de perdida llamada triplet, para el caso del proyecto basta con obtener la incrustación de cada cara. La red neuronal pre entrenada es un modelo de PyTorch que es una herramienta que permite generar redes neuronales al igual que Caffe (Schroff, Dmitry , & Philbin, 2015).

Procedimiento

En este caso vamos a ingresar imágenes con el formato de referencia de la tabla 9, en donde después de cruzar por la red neuronal se espera obtener la incrustación de cada cara. Se usa solo un archivo para poner en marcha la red que contiene toda la información necesaria como lo son la arquitectura de la red y sus diferentes parámetros. En la figura 37 se puede apreciar cómo está compuesta la red convolucional. En la tabla 11 se destaca el nombre del archivo a utilizar que es de acceso libre. Este archivo se lo puede descargar de la página web:

https://github.com/opencv/opencv/tree/master/samples/dnn/face_detector.

Figura 37. FaceNet arquitectura

Tabla 11

Red neuronal pre-entrenada FaceNet.

Archivo	Descripción
openface_nn4.small2.v1.t7	Modelo de PyTorch de una red pre-entrenada para computo de los embeddings de cada cara

En Python lo que se debe cargar es el modelo PyTorch, de la siguiente forma. Es recomendable que en donde se guarda el Script se guarde también el modelo de PyTorch para que así se vuelva fácil de encontrar el archivo referenciado en la tabla 11, con el siguiente comando solo se debe poner el nombre del archivo.

```
ap.add_argument("-m", "--embedding-model", required=True,
    help="path to OpenCV's deep learning face embedding model")
```

Una vez que Python ha cargado el modelo, se debe generar la red neuronal.

```
embedder = cv2.dnn.readNetFromTorch(args["embedding_model"])
```

Con el modelo embedder, ya se le puede ingresar las imágenes pre-procesadas como lo indica la tabla 9.

```
embedder.setInput(faceBlob)
vec = embedder.forward()
```

Por cada imagen que pase por la red neuronal se genera un vector que debe ir acompañado del nombre de la persona de la imagen, por tanto, hay que referenciar como se cargan las imágenes de la base de datos porque como se explicó previamente las imágenes están dentro de una carpeta con el nombre de cada persona que va a usar el sistema. Hay que ingresar el nombre de la carpeta donde está contenida las imágenes de cada individuo dentro de otra carpeta.

```
ap.add_argument("-i", "--dataset", required=True,
    help="path to input directory of faces + images")
```

Ahora hay que averiguar cuantas carpetas contienen imágenes de las personas y hacer una lista.

```
imagePaths = list(paths.list_images(args["dataset"]))
```

Para saber a qué persona pertenece cada imagen finalmente se realiza un Split de la dirección donde están guardadas las carpetas con las imágenes, por ejemplo, la dirección en el caso de la carpeta con fotos de Israel es **/tesis/face_reco/dataset/Israel** en donde solo es de interés la última parte de la dirección porque es el nombre de la persona mediante la siguiente instrucción se logra guardar los nombres de cada una de las personas registradas en la base de datos.

```
name = imagePath.split(os.path.sep)[-2]
```

Como penúltimo paso hay que llenar una lista con el nombre de la persona y otra con los vectores.

```
knownNames.append(name)
knownEmbeddings.append(vec.flatten())
```

Por ultimo hay que escribir el archivo output referenciado en la tabla 7.

```
data = {"embeddings": knownEmbeddings, "names": knownNames}
f = open(args["embeddings"], "wb")
f.write(pickle.dumps(data))
f.close()
```

Este procedimiento se lo realiza dentro de un bucle en donde se pueden pasar todas las imágenes de la base de datos.

3.4.6 Programa entrenar_modelo

En la figura 38 se observa un diagrama de componentes del programa entrenar_modelo, en donde se identifican los pasos que se realiza en el script. La clasificación se realiza mediante una Maquina de Soporte de Vectores (Support Vector Machine / SVM), en donde los datos de

entrenamiento es la incrustación debidamente etiquetadas con cada nombre en el archivo output, este procedimiento se lo realiza en un script nuevo llamado entrenar_modelo referenciado en la tabla 7.

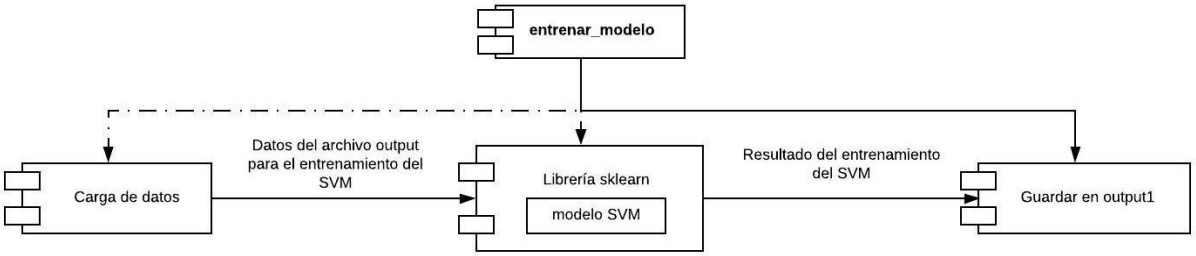


Figura 38. Diagrama de componentes entrenar_modelo

3.4.6.1 Funcionamiento

En este caso se necesita una librería de nombre **scikit-learn** para llamar a un SVM, esta no viene instalada en el paquete normal de Python por tanto se la debe descargar, una vez que se tenga esta librería habilitada se procede importar la librería.

```
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
```

Se debe dotar al programa la dirección del archivo output.

```
ap.add_argument("-e", "--embeddings", required=True,
    help="path to serialized db of facial embeddings")
```

Como siguiente paso se debe cargar el archivo output, para entrenar al SVM.

```
data = pickle.loads(open(args["embeddings"], "rb").read())
```

Los datos etiquetados con los nombres es una de las salidas del SVM.

```
le = LabelEncoder()
labels = le.fit_transform(data["names"])
```

Es momento de entrenar el modelo de SVM.

```
recognizer = SVC(C=1.0, kernel="linear", probability=True)
recognizer.fit(data["embeddings"], labels)
```

Hay que escribir el archivo output1 que contenga el modelo del SVM para poder reutilizarlo en otro Script.

```
f = open(args["recognizer"], "wb")
f.write(pickle.dumps(recognizer))
f.close()
```

Con el modelo de SVM entrenado se procede a cerrar el programa de entrenar_modelo.

3.5 Lectura de tarjetas RFID.

3.5.1 Descripción general del proceso.

En esta sección se analiza el hardware y software necesario para obtener los diferentes números de tarjetas.

3.5.2 Material Necesario.

Para capturar la información de las tarjetas RFID se necesita un dispositivo al cual se le pueda conectar una lectora, con esta premisa se usa un Arduino Mega 2560 que brinda las suficientes prestaciones, en la tabla 12 se puede apreciar sus características técnicas.

Tabla 12
Características Técnicas Arduino Mega 2560

Característica	Descripción
Microcontrolador	ATmega2560
Voltaje de operación	5 [V]
Voltaje de entrada (recomendado)	7-12 [V]
Voltaje de entrada (límite)	6-20 [V]
Pines digitales de E/S	54 (14 de estos pueden proveer una salida PWM)
Pines análogos de entrada	16
Corriente DC por pin de E/S	40 [mA]
Corriente DC del pin de 3.3 [V]	50 [mA]
Interrupciones	6 interrupciones pines digitales [2,3,21,20,19,18]

Fuente: (González, 2018)

La distribución de pines, microcontrolador y demás características del Arduino Mega2560, se pueden observar en la figura 39.

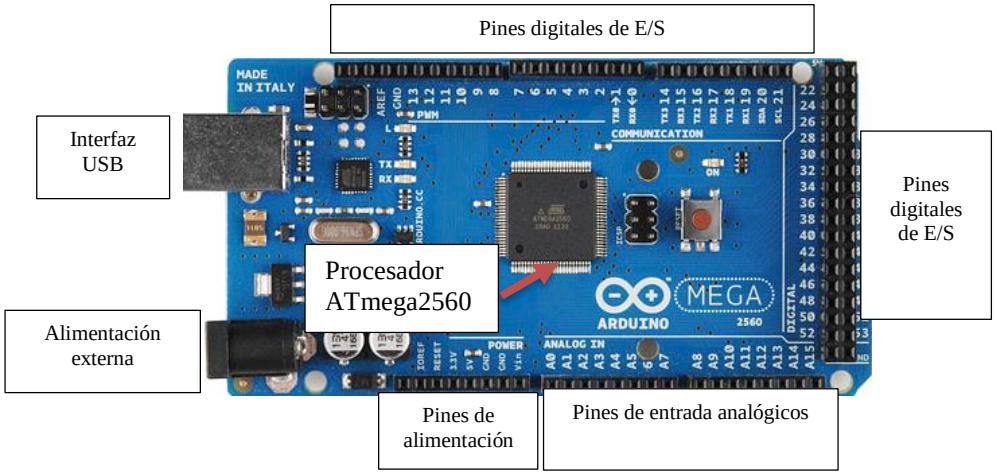


Figura 39. Arduino Mega 2560

La lectora Wiegand de la marca ROSSLARE modelo AY-K12, es muy usada en el mercado comercial se referencia sus características técnicas en la tabla 13.

Tabla 13
Características Técnicas AY-K12

Característica	Descripción
Protocolo	Wiegand 26
Voltaje de operación	5 [V]
Voltaje de entrada (recomendado)	5-16 [V]
Distancia de lectura	80 [mm]
Dimensiones	80 [mm]x 40 [mm]x 12,8 [mm]
Pines de conexión	Rojo: 5-16 [V] Negro: Tierra Blanco: Data1 Verde: Data0

Fuente: (SIASA, 2018)

3.5.3 Funcionamiento.

Se debe realizar un programa en el software de programación para arduino, en donde se recibirá los números de las tarjetas que pasen por la lectora. Para leer el Data0 y Data1 se utiliza una interrupción de flanco negativo para este caso los pines son 2 y 3.

```
pinMode(2, INPUT);      // DATA0 (INT0)
pinMode(3, INPUT);      // DATA1 (INT1)

attachInterrupt(0, ISR_INT0, FALLING);
attachInterrupt(1, ISR_INT1, FALLING);
```

Hay que verificar si existen 26 bits en la lectura y tomar los bits del facility code y los de

identificación como se los referencia en tabla 2, con el operador <<= se trasforma en decimal el vector de bits.

```
else if (bitCount == 26) {
  for (i=1; i<9; i++) {
     facilityCode <<=1;
     facilityCode |= databits[i];
  }
  for (i=9; i<25; i++) {
     cardCode <<=1;
     cardCode |= databits[i];
  }
```

Por ultimo para leer en Python los datos se necesita imprimir por el serial la lectura.

```
Serial.print(cardCode);
```

Para leer los datos en Python se necesita exportar la librería serial.

```
import serial
```

Abrir una comunicación con Arduino.

```
arduino = serial.Serial('COM10', 9600)
```

Leer el dato se hace uso de un comando de la librería serial

```
rawString = arduino.readline()
```

En el sistema operativo Linux, para poder leer el puerto serial se necesita dotarle de permisos al SO, los comandos que se utilizan en la terminal de Linux se observan en la figura 40.

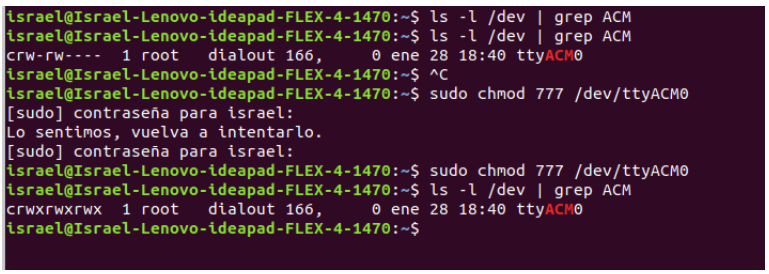


Figura 40. Permisos puerto serial

3.6 Integración

3.6.1 Descripción general del proceso.

El objetivo de esta sección es mediante un reconocimiento de rostros en tiempo real más la lectura de tarjetas RFID controlar el acceso de las personas. En la figura 42 el diagrama de componentes de la integración.

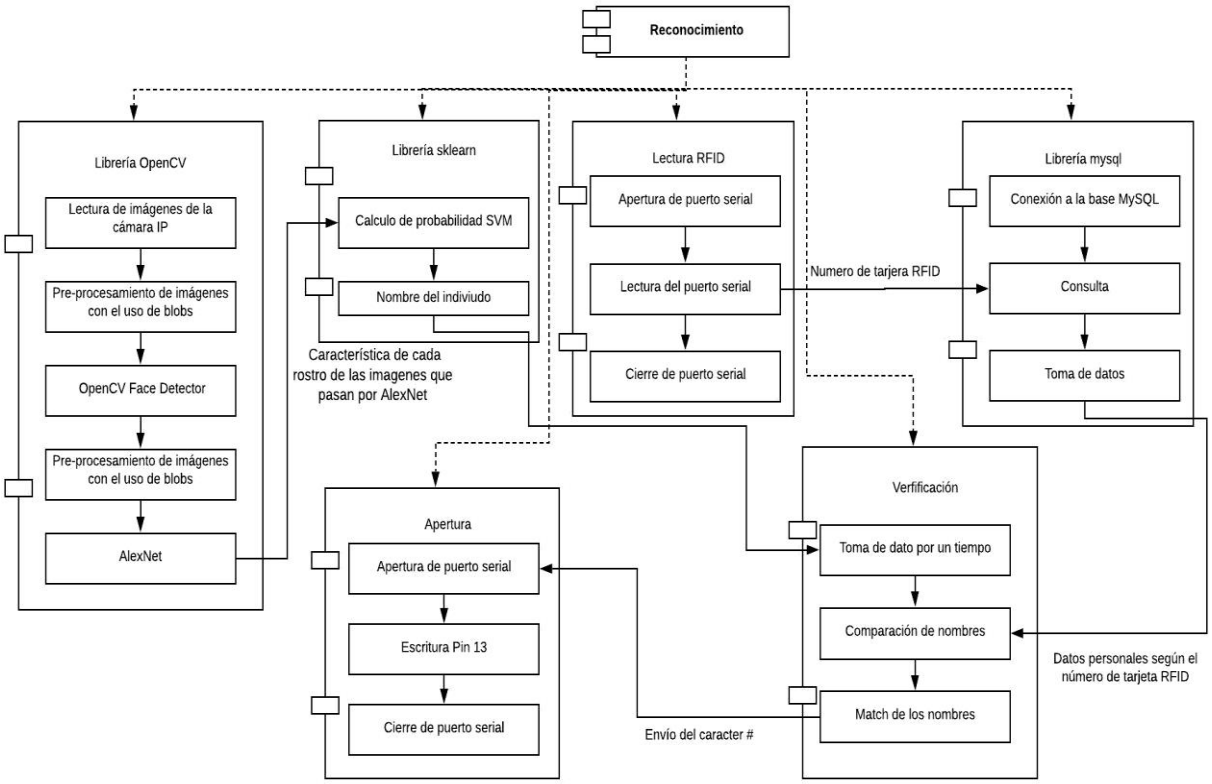


Figura 41. Diagrama de componentes integración

3.6.3 Material necesario.

Para realizar el control de acceso se necesita tanto el hardware como software que se utiliza en las secciones anteriores, por lo que no es necesario volver a describirlo y se puede resumir los componentes en el diagrama de bloques de la figura 43.

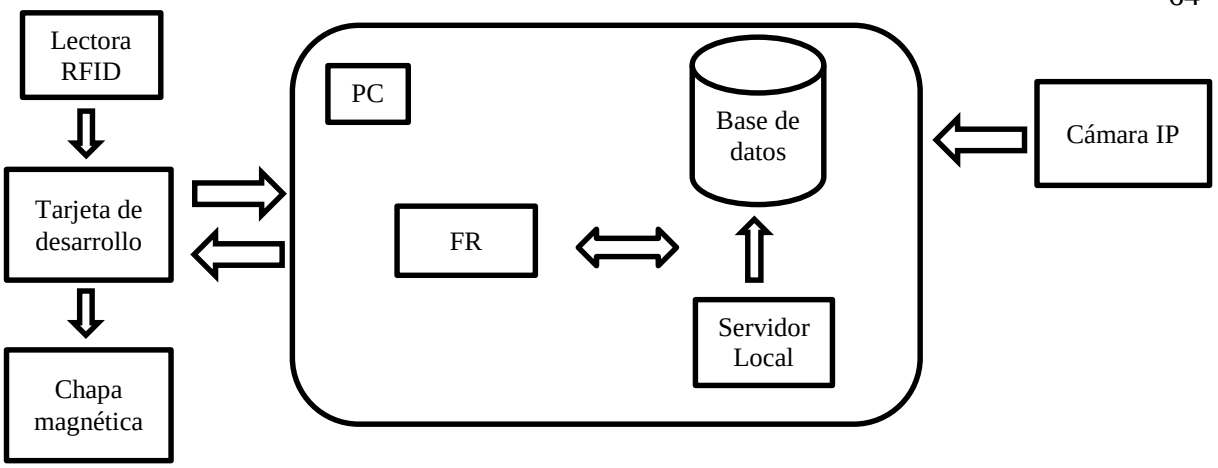


Figura 42. Diagrama de bloques

3.6.3 Funcionamiento.

Las condiciones para que una persona puede ingresar o no, se basa en que, primero debe pasar la tarjeta RFID por la lectora, al mismo tiempo tiene que estar frente a la cámara por 5 segundos, en el transcurso de este tiempo se consulta el número de tarjeta en la base de datos y además se toma la salida del SVM que es la probabilidad de que sea alguna cara con las que fue entrenada, esta se representa con dos variables una que contiene el nombre de a quién pertenece el rostro y otra que contiene la probabilidad. Si el nombre de la cara está asociado a la tarjeta que recibió el lector simplemente se activa una salida de la tarjeta de desarrollo que prende un relé que da paso a la apertura de una chapa magnética. Hay que recalcar que el programa está recibiendo imágenes de la cámara IP todo el tiempo, esto implica que si una persona se pone en frente de la cámara se va a procesar el reconocimiento, pero la verificación de los datos solo se da una vez que la tarjeta es leída. Como en la sesión de registro, se necesita ingresar a la cámara IP, pero en este caso no solo para tomar una foto al contrario hay que trasmitir un video. Para poder ver el vídeo en tiempo real en Python se necesita una rutina en donde se tome en cuenta un bucle infinito que va a estar leyendo todo el tiempo los frames per second (FPS) de la cámara.

Como en la sesión de registro se agrega la dirección rstp de la cámara al programa, luego con la variable frame dentro de un bucle while se empieza a recibir la imagen de la cámara.

```
cap = cv2.VideoCapture('rtsp://192.168.100.254/mpeg4cif')
while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()
```

En la variable frame está contenida una imagen que se refresca cada cierto número de segundos dependiendo la cámara que se use para la HikVision son 45 imágenes por segundo, lo que pasa con la imagen es lo que se trató en las secciones anteriores este frame se pre-procesa, ingresa en la red neuronal para buscar caras en la imagen, si encuentra una cara entra a otra red neuronal para sacar sus características y luego al clasificador que indica a quien pertenece esa cara, todo esto en tiempo real.

Una vez que se detecta una tarjeta RFID, se toma el dato de la misma y se consulta en la base de datos de MySQL a quien pertenece esta tarjeta.

```
RFID = rawString[0]
sql = ("SELECT * FROM usuario WHERE RFID='" + RFID + "")
mycursor.execute(sql)
row = mycursor.fetchall()
name3 = row[0]
```

La toma de la salida del SVM del nombre se va a repetir tantas veces como se reciba los FPS, esto se guarda en un vector en donde se cuentan los nombres que más se repitieron y se realiza una media. por lo tanto, para 5 segundos hay que tener claro que ingresaran un numero de imágenes representados en la siguiente ecuación.

$$n_i = 45\left[\frac{frames}{s}\right] \times 5[s]$$

$$n_i = 225\ frames$$

Ahora se busca en el vector de los nombres cual es el que más se repite y se cuenta las veces se repite.

```
if cont  <= 225:
    listofzeros.insert(cont, name1)
    #print(listofzeros)
    cont=cont+1
    if cont == 225:
        acceso1.update(listofzeros)
        #acceso1 = mode(acceso1)
        acceso = listofzeros.count('unknown')
        acceso = acceso/100
        print(listofzeros)
        print(acceso)
        acceso1 = acceso1.most_common(1)
        print(acceso1)
```

Se divide para el número total de frames, se obtiene un dato estadístico si se obtiene más del 90% de probabilidad se pasa a verificar nombre que más se repite no sea el de la persona desconocida en caso contrario se abre la puerta. Se busca si el nombre es desconocido para negar el acceso en caso contrario si no es desconocido y es igual al nombre de la consulta en MySQL se da paso a la abertura.

```
    if acceso >= 0.90:
        print('no abrir')
        listofzeros = [ ]
        aux = 101
        fps.stop()
        cv2.destroyAllWindows()
        frame = vs.stop()
        rawString = unalista
    elif name1==name3 :
        print('abrir')
```

CAPITULO IV

PRUEBAS Y RESULTADOS

4.1 Análisis

Para completar esta sección se ha propuesto 3 pruebas básicas en donde se verifica que tan bueno es el reconocimiento facial, dejando de lado por un momento el control de acceso puesto que el sistema se vuelve robusto si el reconocimiento de rostros funciona de manera correcta. Es importante antes de las pruebas tener en claro la cantidad de individuos que existen en la base de datos y cuantas fotos se tienen de cada uno. Hay que agregar a la base de datos una carpeta en donde se ingresen fotos de gente desconocida porque así las personas que no estén registradas en la base de datos, su probabilidad de que se lo relacione con alguien que este registrado disminuye. Por lo que en la tabla 14 se aprecian estas variables.

Tabla 14
Registro de fotos.

Individuos	**Cantidad de fotos**
Karen Zurita	24
René Cobo	12
Cami Jarrín	18
Naty Oliva	19
Israel Obando	23
Desconocidos	8

La medición para saber que tanto acierta el sistema de reconocimiento se lo va a relacionar al tiempo y a la cantidad de veces que se repite el nombre, esto en base a que el sistema está configurado para mediante un tiempo determinar por medio de estadística si la persona está registrada en la base de datos o no. En la tabla 15 se detalla las pruebas que se van a realizar.

Tabla 15
Pruebas propuestas.

Prueba	**Detalle**
P1-Prueba de distancia.	Distancia variable, con un ángulo fijo.
P2-Prueba de Luz.	Afectando la luz de la escena, a diferentes distancias.
P4-Diferente cámara.	Una cámara totalmente diferente a la HiKVision.

4.2 Prueba de distancia (P1).

Para realizar esta prueba se tomas en cuenta algunas variables que se detallan en la tabla 16.

Tabla 16
Prueba 1.

Prueba 1	**Detalle**
Distancia 1	50 cm – sin movimiento – luz natural
Distancia 2	80 cm – sin movimiento - luz natural
Distancia 3	1.20 m – sin movimiento – luz natural
Distancia 4	1.20 m – con movimiento – luz natural

La figura 44 es un ejemplo tomado a Distancia 1, el sujeto de prueba es Naty Oliva. El usuario debe mantener la cara firme o moviéndose de un lado a otro hacia la cámara por 5 segundos, se genera un vector de 225 espacios en donde se encuentra el nombre de la persona que más se repite como se ve en la figura 45. El método esta aplicado a todos los usuarios y a uno desconocido de la base de datos en las mismas condiciones de escenario.

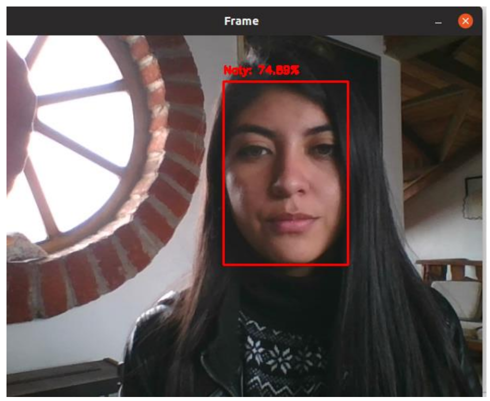


Figura 43. Resultados obtenidos P1-D1

Figura 44. Resultados obtenidos P1-D1

Para la Prueba 1 se tiene un total de 24 datos, que están tabulados en la tabla 17.

Tabla 17
Resultados prueba 1.

	Distancia 1	**Distancia 2**	**Distancia 3**	**Distancia 4**
Naty	100	96	93	77
Cami	98	95	90	75
René	90	87	82	54
Karen	100	94	90	80
Israel	95	90	88	72
Desconocido	90	88	85	50

A una corta distancia los resultados de la prueba son bastante alentadores, la probabilidad de que la persona que esté en frente de la cámara sea la misma que reconoce el sistema es bastante alta y no disminuye del 90 por ciento. La distancia del sujeto de prueba, así como el movimiento de su cara afecta de manera notable al reconocimiento facial decayendo la calidad hasta en un 40 por ciento en la figura 47 se nota este cambio brusco.

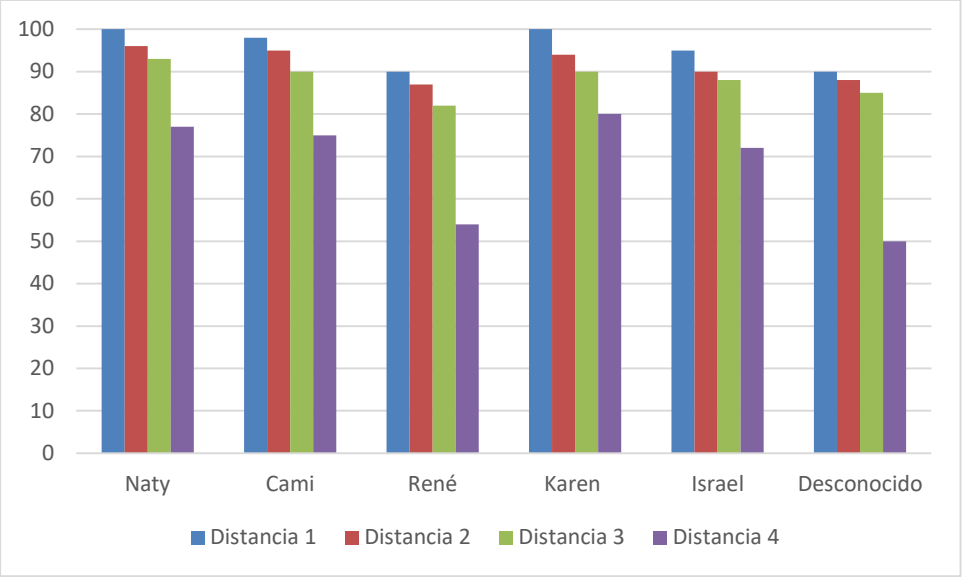


Figura 45. Resultados P1

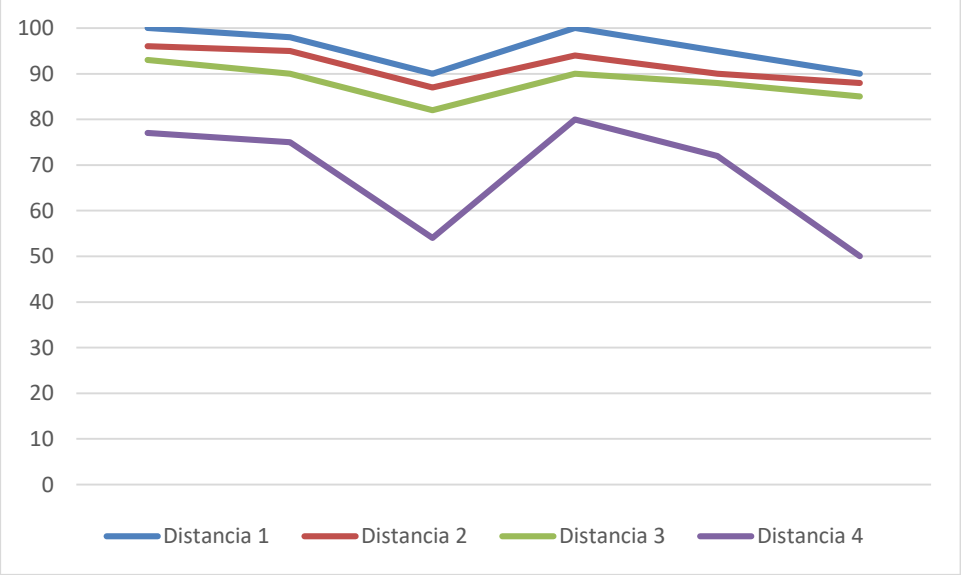


Figura 46. Porcentaje de reconocimiento prueba de distancia

4.3 Prueba de luz (P2).

Para realizar esta prueba se toma en cuenta algunas variables que se detallan en la tabla 18.

Tabla 18
Prueba 2.

Prueba 2	Detalle
Luz 1	50 cm – sin movimiento – poca luz
Luz 2	80 cm – sin movimiento – a contra luz
Luz 3	1.20 m – sin movimiento – luz incandescente
Luz 4	1.20 m – con movimiento – luz incandescente

El método esta aplicado a todos los usuarios y a uno desconocido de la base de datos en diferentes escenarios. En la figura 48 se tiene un ejemplo con poca luz, el sujeto de prueba es Israel Obando, la luz esta indirecta al rostro en un cuarto oscuro. En la figura 49 es un ejemplo tomado a Luz 2, el sujeto de prueba es Naty Oliva, se observa como a contra luz el rostro tiene una acentuación de claridad importante, es un detalle que a todos los participantes de la prueba les ocurrió. En la figura 50 es un ejemplo tomado a Luz 3, el sujeto de prueba es René Cobo, en la habitación existen dos focos incandescentes de 100 w cada uno. En la figura 51 se observa como el usuario no mantiene la cara fija a la cámara.

Figura 47. Prueba de Luz 1

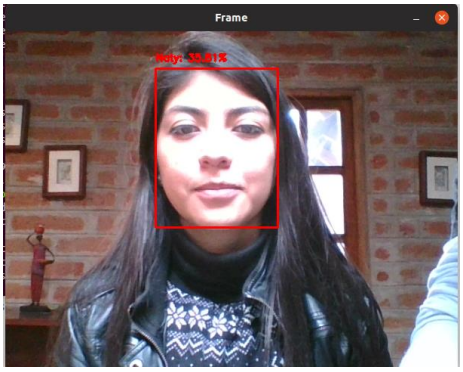


Figura 48. Prueba de Luz 2

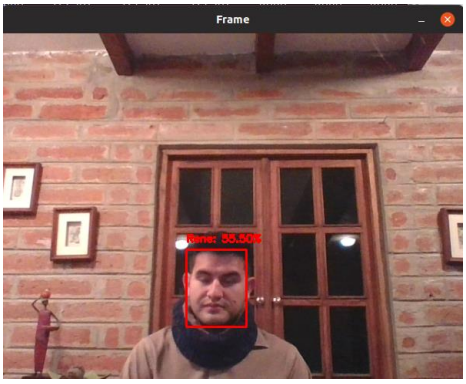


Figura 49. Prueba de Luz 3

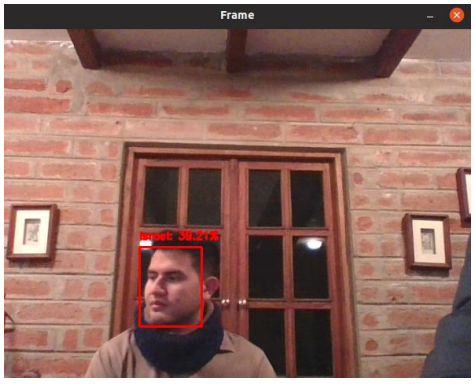


Figura 50. Prueba de Luz 4

Para la Prueba 2 se tiene un total de 24 datos, que están tabulados en la tabla 19.

Tabla 19
Resultados prueba 2.

	Luz 1	**Luz 2**	**Luz 3**	**Luz 4**
Naty	85	95	90	86
Cami	84	95	90	85
René	75	90	88	80
Karen	82	96	92	85
Israel	80	94	90	83
Desconocido	70	90	85	70

A una calidad de luz bastante baja los resultados no son del todo malos a una distancia corta de la cámara. A contra luz a pesar de que a los sujetos de prueba se les marcaba la cara prácticamente de otro color al suyo natural la prueba es bastante alentadora los porcentajes no bajan del 90 por ciento. Con una buena iluminación a pesar de la distancia los resultados siguen estando bastante acertados y no se produce una baja de calidad en el reconocimiento, contrastando con la información de los resultados obtenidos en la prueba 1. En la figura 53 se puede observar que no hay una pérdida abrupta de calidad del sistema.

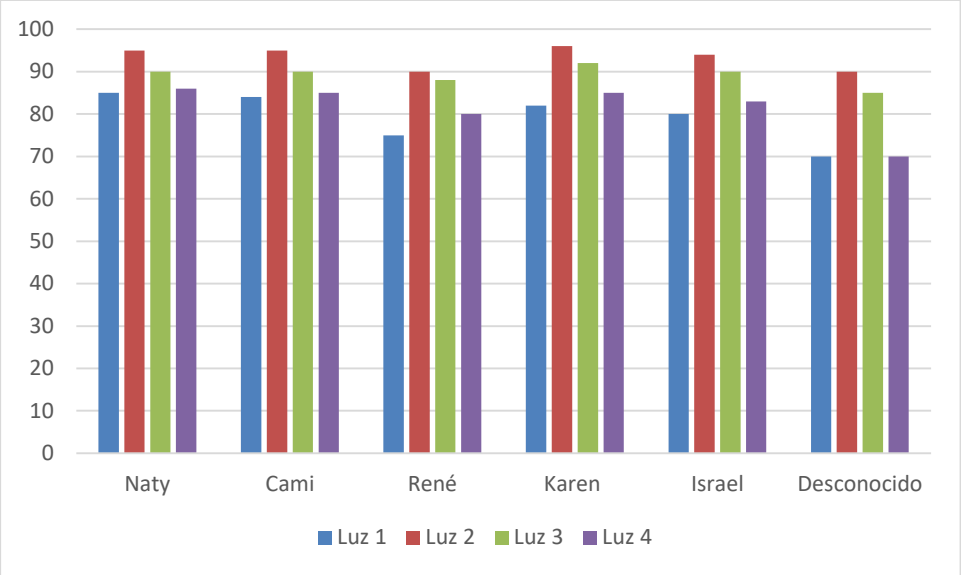


Figura 51. Resultados P2

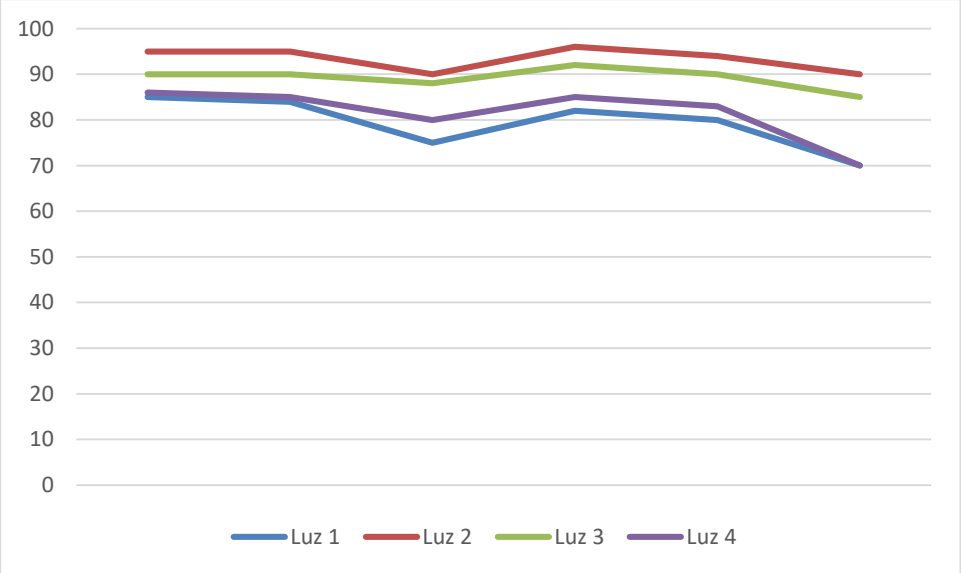


Figura 52. Porcentaje de reconocimiento prueba de luz

4.4 Prueba con una cámara diferente (P3).

Para este caso se considera una cámara con una calidad un poco más baja que la HikVison, la cámara que se usa es una web tiene 720 P calidad de imagen, además trasmite a 20 FPS, por lo que se vuelve un tanto diferente el cálculo de la probabilidad, el vector que se obtiene en las anteriores pruebas tiene 225 lugares para 5 segundos, en este caso los FPS disminuyen por lo que se llena solo 100 lugares. Para realizar esta prueba se tomas en cuenta algunas variables que se detallan en la tabla 20.

Tabla 20
Prueba 3.

Prueba 3	**Detalle**
P3.1	50 cm – con movimiento – luz natural
P3.2	80 cm – con movimiento – luz incandescente
P3.3	1.20 m – sin movimiento – luz incandescente
P3.4	1.20 m – con movimiento – luz incandescente

El método esta aplicado a todos los usuarios y a uno desconocido de la base de datos en diferentes escenarios. En la figura 54 el sujeto de prueba es Israel Obando. En la figura 55 el sujeto de prueba es Cami Jarrín

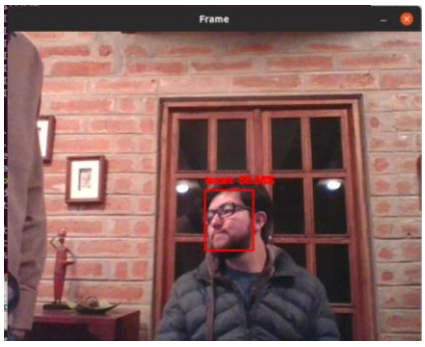


Figura 53. Ejemplo P3.2

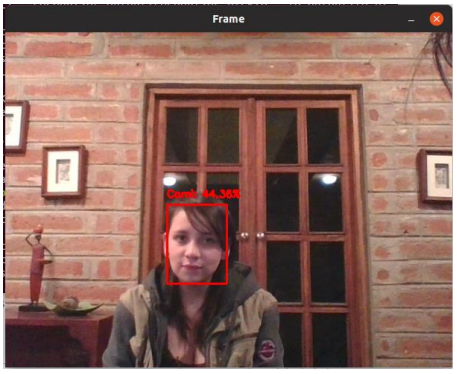


Figura 54. Ejemplo P3.3

Para la Prueba 2 se tiene un total de 24 datos, que están tabulados en la tabla 21.

Tabla 21
Resultados prueba 3.

	P3.1	**P.3.2**	**P3.3**	**P3.4**
Naty	92	96	88	80
Cami	94	94	91	82
René	90	92	85	77
Karen	95	96	93	82
Israel	93	94	91	81
Desconocido	88	90	80	70

A pesar que la cámara tiene un poco menos de calidad los resultados no son malos y se parecen mucho a los resultados de las pruebas anteriores la calidad no disminuye bruscamente, como se puede apreciar en las figuras 56 y 57.

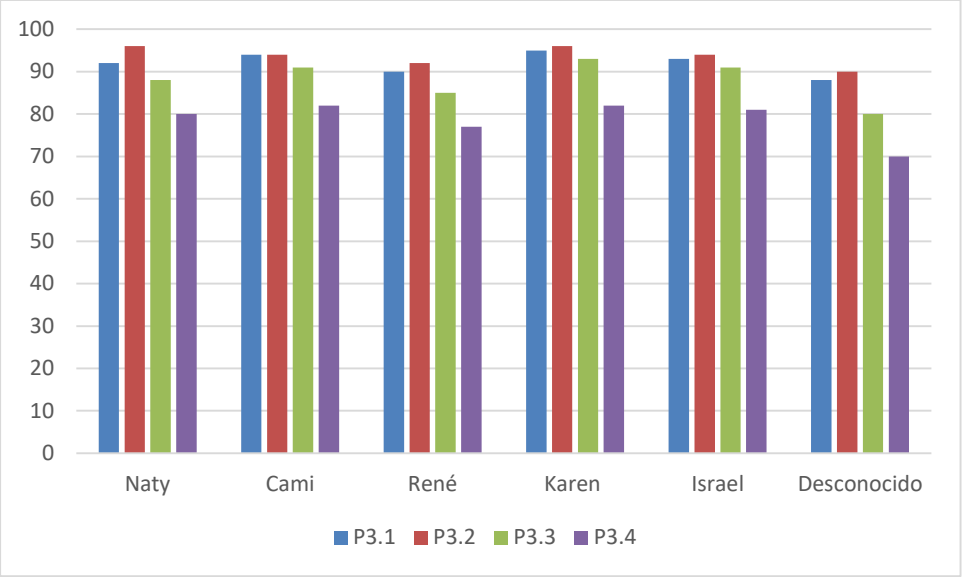


Figura 55. Resultados P3

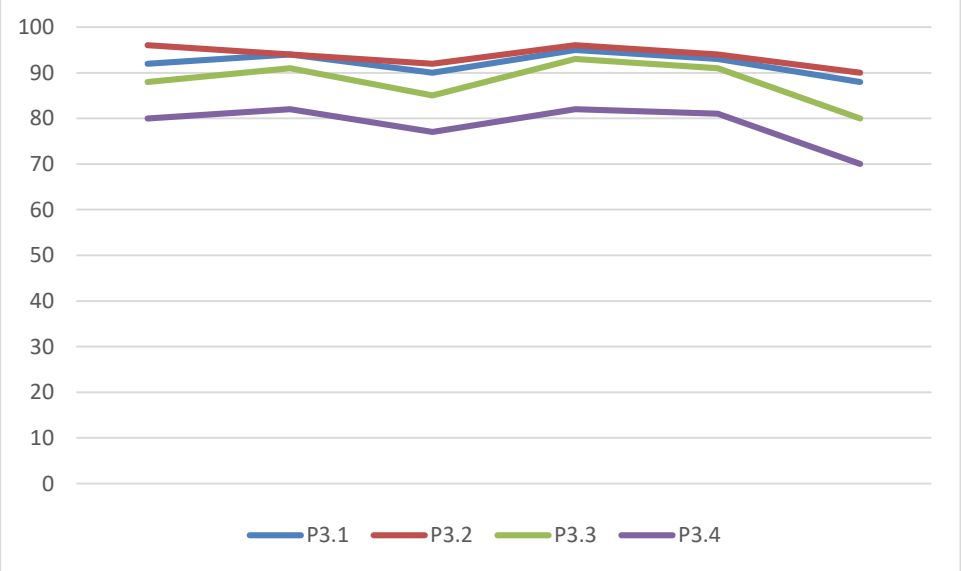


Figura 56. Porcentaje de reconocimiento P3

4.5 Escenario óptimo.

Una vez con los resultados, es notable que cuando el sujeto de prueba está más lejos de la cámara y además está en movimiento la calidad del reconocimiento es baja no se llega al 90 por ciento en ninguno de los casos, el valor promedio es de 76.05 por ciento, por lo que se debe descartar elegir un rango que llegue al 1.20 metros, entonces en distancia se tiene un rango bastante acertado desde los 50 cm hasta los 80 cm, según las pruebas realizadas por lo que hay un promedio de acierto del 88.94 por ciento que es un porcentaje bastante alentador a los 50 cm y de 92.88% a 80 cm. En cuanto a las pruebas de luz y con otra cámara no existen muchos problemas, pero considerar una puesta del sistema con una iluminación moderada mejoraría la calidad del reconocimiento, por lo que el escenario optimo se sumarian estos dos conceptos para todo el tiempo tener un buen reconocimiento facial.

CAPITULO IV

CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- En un escenario óptico como el mencionado en la sección 4.5, el sistema para controlar accesos resultaría exitoso puesto que según los porcentajes acertaría 9 de cada 10 personas.
- Es un sistema de bajo costo económico puesto que cuenta con la ventaja de funcionar con cámaras que normalmente se usan para video-vigilancia o cualquier cámara genérica, se ahorraría en costos puesto que los sistemas de reconocimiento facial en el mercado necesitan una cámara especifica que es provista por el fabricante razón por la cual los costos se elevan.
- Se puede observar que con los sujetos de prueba **desconocidos** se tiene algunos problemas en la detección puesto que su índice de efectividad es bajo en comparación con los demás participantes de la prueba, esto se debe a que el número de fotos para personas desconocidos es muy reducido y las personas que se probaron como desconocidas no formaban parte de la base de datos de personas desconocidas por eso existe un error considerable.
- Los falsos positivos que se producen con los usuarios desconocidos y si el sistema no trabaja en un escenario optimo, son causa de no tener más de 128 medidas del rostro de cada persona.
- El sistema puede trabajar con poca luz, luz natural y luz artificial, incluso si se le pone a trabajar en una condición un tanto extrema como lo es con la cámara a contra luz no tiene ningún problema en realizar el reconocimiento facial, esto se da gracias a que el

sistema usa una red convolucional pre-entrenada con una la técnica deep metric learning que mitiga las condiciones en las que se encuentre el rostro que se detecta y únicamente se foca en analizar sus medidas.

- Gracias a redes neuronales pre-entrenadas a base de una gran cantidad de datos se puede reducir el tiempo de desarrollo de sistemas de visión por computador.
- Utilizar librerías como OpenCV para reducir el esfuerzo computacional es de gran ayuda, se evitan un sin número de operaciones que lo único que causan es volver a un sistema de visión artificial lento.

5.2 Recomendaciones

- Para cada usuario se debe ingresar por lo menos 15 fotografías como se puede observar en los resultados con una base de datos compuesta por este número de fotos o más se mejora la calidad del reconocimiento facial.
- La luz incandescente es una buena opción para iluminar el escenario en donde se ponga a prueba el sistema sin embargo por efectos del calentamiento global se pueden obviar este tipo de bombillas y usar unas más a fines con el ambiente como las luces leds sin problema.
- En el caso de usar cámaras IP se debe procurar que soporten el protocolo ONVIF porque así se puede acceder a una dirección rtsp de forma automática gracias a XEOMA.
- Tomar en cuenta la comunicación serial que se realiza con el arduino desde Python, utilizar un timeout, de lo contrario el programa de reconocimiento facial entra en bucle de lectura infinito del puerto serial.
- Estudiar la visión artificial es de gran impacto puesto que la revolución industrial 4 que ya se la está viviendo está ligada a este tema en varios campos de la automatización.
- Como trabajos futuros, es clave atacar la reducción de falsos positivos aumentando la cantidad de medidas que se usan para caracterizar los rostros de los usuarios.

BIBLIOGRAFÍA

Constante Prócel, P., Gordón Garcés, A., Chang Tortolero., O., Pruna Panchi, E., Escobar Anchaguano, I., & Acuña Coello, F. (2016). Artificial Vision Techniques for Strawberry's Industrial Classification. *IEEE Latin America Transactions, 14*, 2576 - 2581.

He, K., Zhang, X., Ren, S., & Sun, J. (2014). Deep Residual Learning for Image Recognition. *Microsoft Research*.

Kurtaev, D., & Alekhin, A. (15 de Marzo de 2018). *GitHub*. Obtenido de OpenCV deep learning module samples: https://github.com/opencv/opencv/blob/master/samples/dnn/README.md

Zhang, B., Gao, Y., & Zhao, S. (2010). Local Derivative Pattern Versus Local Binary. *IEEE TRANSACTIONS ON IMAGE PROCESSING, 19*.

Alonso, E. G. (2007). *Aplicación de las máquinas de soporte vectorial.* Madrid: UNIVERSIDAD PONTIFICIA COMILLAS.

Ambarjarafi, G. (24 de Septiembre de 2014). *University of Tartu.* Obtenido de Digital Image Processing: https://sisu.ut.ee/imageprocessing/book/1

Ballantyne, M., Boyer, R., & Hines, L. (1996). Woody bledsoe: His life and legacy. *AI Magazine*.

Bertozzi, M., Broggi, A., Cellario, M., Fascioli, A., Lombardi, P., & Porta, M. (2002). Artificial Vision in Road Vehicles. *Proceedings of the IEEE*.

Beyerer , J., Puente León , F., & Frese , C. (2016). *Machine Vision.* Berlin: Springer.

Caplan, J., & Torpey, J. (2018). *Documenting Individual Identity: The Development of State Practices in the Modern World.* New Jersey: Princeton University Press.

Corporation, H. (2006). *Understanding Card Data Formats.*

Corporation, N. (2018). *Developer NVIDIA.* (NVIDIA Corporation) Recuperado el 24 de Octubre de 2018, de https://developer.nvidia.com/discover/convolutional-neural-network

Deshpande, N., & Ravishankar. (2010). Face Detection and Recognition using Viola-Jones algorithm and fusion of LDA and ANN. *IOSR Journal of Computer Engineering, 18*.

Domínguez Torres, A. (1996). Procesamiento digital de imágenes. *Perfiles Educativos*, 72.

González, A. G. (20 de Marzo de 2018). *Panama Hitek* . Obtenido de Arduino Mega: Características y Capacidades: http://panamahitek.com/arduino-mega-caracteristicas-capacidades-y-donde-conseguirlo-en-panama/

Hjelmas, E., & Kee Low, B. (2001). Face Detection: A Survey. *Academic Press*, 35.

Huang, G., Mattar, M., Berg, T., & Learned-Miller, E. (2015). Labeled Faces in the Wild: A Database for.

Jain, A., Bolle, R., & Pankanti, S. (2006). *Biometrics Personal Identification in Network Society.* New York: Springer.

Jain, A., Flynn, P., & Ross, A. (2008). *Handbook of Biometrics.* New York: Springer.

Johnson, J. (03 de 16 de 2018). *Convolutional Neural Networks (CNNs / ConvNets)*. Obtenido de Stanford University: http://cs231n.github.io/convolutional-networks/?fbclid=IwAR3R4v99sG6wZu-ls7TOI1Jktqzm23HpXtLhSB7d5g-GW9-d7Wywve5Zqiw

Kumar Basu, J., Bhattacharyya, D., & Kim, T.-h. (2010). Use of Artificial Neural Network in Pattern Recognition. *International Journal of Software Engineering and Its Applications, 4*, 23-34.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. (2016). SSD: Single Shot MultiBox Detector. *Lectures Notes in Computer Science* , 21-37.

Mallick, S. (2016). *Learn OpenCV*. Recuperado el 24 de Octubre de 2018, de https://www.learnopencv.com/histogram-of-oriented-gradients/

Marques, I. (2010). *Face Recognition Algorithms.* Universidad del País Vasco.

Mordvintsev, A. (2017). *OpenCV-Python Tutorials Documentation Release 1*.

Pérez Grassi, A., & Puente León, F. (2006). Visión Artificial. *AIDIMA*, 26-27.

Ponti, M., Ribeiro, L., & Nazare, T. (2016). Everything you wanted to know about Deep Learning for Computer Vision but were. *CVSSP – University of Surrey*, 1-25.

Rodríguez Hernández, J., & Duque Méndez, N. (2015). Reconocimiento de expresiones faciales para interacción con el computador COMPUTADOR. *Jovenes en la Ciencia, 3*, 62-66.

Rosebrock, A. (2018). *Deep Learning for Computer Vision with Python.* New York: pyimagesearch.

Rossum, G. v. (2009). *Python Tutorial.*

Sánchez, A. A. (s.f.). Clasificadores Débiles . En *Uso de características no lineales para identificar llantos de recién nacidos con un conjunto clasificador* (pág. 2015). Universidad de las Américas Puebla.

Schroff, F., Dmitry , K., & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* .

SIASA. (20 de Marzo de 2018). *Productos de Seguridad*. Obtenido de https://www.siasa.com/producto.php?prod=0500005

Sobrado Malpartida, E., & Tafur Sotelo, J. (2011). *Ssistma de visión artificial para el reconocimiento y manipulación de objetos utilizando un brazo robot* . Lima: Pontificia Universidad Católica del Perú.

Vilda, C. C. (2006). *Biomteria. Reconocimiento Facial Mediante Fusión 2D Y 3D* . Madrid: DYKINSON.

Viola, J., & Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple. *In Proc. of IEEE Conference on*.

Woodward, J., John, D., Horn, C., Gatune, J., & Thomas, A. (2003). *Biometrics: A Look at Facial Recognition.* Santa Monica: Defense technical information center.