Universidad Nacional Autónoma de México

Facultad de Estudios Superiores Cuautitlán

Control de acceso con reconocimiento facial utilizando el modelo de clasificación HaarCascade de OpenCV y el método de histogramas por medio del algoritmo LBPH

T E S I S

que para obtener el título de:

Ingeniero en telecomunicaciones, sistemas y electrónica

P R E S E N T A :

víctor iván rodríguez ábrego

tutor

Dr. David Tinoco Varela

Cuautitlán Izcalli, Edo. de Mex. 2024

Agradecimientos

A mi mamá, Reyna Ábrego, por toda la motivación que me dio durante este largo trayecto, por las palabras de aliento cuando más las necesité. Por ser el más grande ejemplo de persistencia, dedicación, responsabilidad y compromiso. Y sobre todo, por el gran amor incondicional que me brinda todos los días.

Gracias mamá, te quiero.

Asimismo, a mi papá, Víctor Rodríguez, por plantearme retos todos los días, impulsándome a ser mejor desde pequeño. Por todo su esfuerzo realizado para poder enfocarme por completo en este objetivo. Por enseñarme que en la vida no hay límites ni momentos para rendirnos, también por inyectarme esa chispa de aventura y atrevimiento. Gracias por darme el ejemplo y guiarme por el camino triunfador.

Gracias papá, te admiro.

Agradezco a mis tres hermanos, Yael, Luis y Omar, porque siempre serán una motivación para crecer y dar un buen ejemplo del camino correcto. Gracias por ser parte de mis proyectos y aguantar mi mal humor en esos días grises cuando todo se venía abajo.

Gracias hermanos, los quiero.

A mi novia, Mitchell, por su paciencia y apoyo en esta última etapa, por brindarme esa ayuda tan necesaria, levantarme los ánimos, darme energía para continuar y estar conmigo en las buenas y en las malas. Le agradezco por darme una nueva perspectiva de la vida, darme su mano y decirme "tú puedes".

Gracias Mitch, te amo.

A mis amigos incondicionales de la carrera, Juanma y Carlos, que con su ayuda, las pláticas, los desayunos en el comedor y todas las cosas que vivimos, hicieron una etapa increíble. Agradezco también al equipo de básquet porque con ellos olvidaba por un momento los días más pesados, recargando energía para continuar con mi objetivo.

Gracias.

A mi profesor, director de proyecto, asesor de tesis y amigo, el Dr. David Tinoco, por confiar en mí, darme el impulso necesario, orientarme y brindarme su valioso tiempo y conocimiento.

Gracias.

A mis profesores, gracias por sus enseñanzas, su tiempo y su conocimiento compartido. También a las personas que participaron directa e indirectamente en la culminación de esta meta.

Gracias.

A la UNAM y a la FES Cuautitlán por darme una gran diversidad de aprendizaje, por ser una noble institución y brindar educación del más alto nivel en un ambiente excepcional.

Gracias.

Se agradece también por su apoyo a los proyectos **PAPIIT IA102323** *y* **PAPIME PE103223** *de la UNAM.*

Resumen

Este trabajo presenta el desarrollo de un proyecto de inteligencia artificial enfocado en la rama de la visión por computadora. Se plantea un sistema capaz de controlar el acceso a ciertas zonas privadas de una edificación a través del procesamiento de imágenes en tiempo real. El proceso se lleva a cabo en 5 módulos, en los cuales cada uno desarrolla una tarea especifica para el funcionamiento del sistema, los cuales son: registro facial, entrenamiento, identificación facial, interfaz gráfica y sistema físico.

La parte fundamental del proyecto se localiza en 3 de los módulos antes mencionados (Registro facial, Entrenamiento e Identificación facial) los cuales son los destinados al desarrollo de la inteligencia artificial apoyandose en herramientas que ofrece la biblioteca de Open CV, tales cómo el modelo de clasificación estadístico HaarCascade para la identificación de rostros y el método de histogramas a través del algoritmo de texturas para el reconocimiento de rostros LBPH.

Gracias a la unión de programación estándar y paradigmas de la IA se puede ofrecer este tipo de proyectos que en la vida cotidiana facilitan tareas, reducen gastos en materiales, ofrecen mayor seguridad y acoplan las necesidades en un ambiente especifico.

Para la evaluación de resultados de este proyecto se realizaron pruebas estadísticas en conjunto de un análisis detenido para la verificación de imágenes y el correcto procesamiento de ellas. Dando bases para desarrollar en un futuro el aumento de seguridad en varios campos de la industria con la identificación facial.

Índice general

Índice de figuras

Índice de tablas

Introducción

La inteligencia artificial (IA) ha tomado mucha fuerza en los últimos años y ha llegado el punto en el que convive día a día con la gente y, aún así, falta mucho camino por seguir en el desarrollo de este nuevo paradigma de información. Es por eso que se han dado a la tarea de seguir contribuyendo en este fascinante campo de la tecnología y ampliar aún más los horizontes que se pueden cubrir.

A pesar de la amplia capacidad que tiene la IA para adaptarse a cualquier área de conocimiento, existen tareas que todavía no dan el salto tecnológico y siguen llevándose a cabo arcaicamente; se cree que es momento de seguir avanzando y actualizar esas funciones que muchas veces no son tan eficientes como se desea, ya sea porque son tareas tediosas y repetitivas, conllevan mucho tiempo realizarlas debido al procesamiento de mucha información o porque representan costos elevados debido a la producción de material innecesario.

El desarrollo de este proyecto se encarga de aumentar la eficiencia de un conjunto de tareas analizando y dándole solución a los 3 problemas antes mencionados. Se ha desarrollado un sistema de reconocimiento facial capaz de decidir si una persona es autorizada para accesar a diferentes áreas restringidas. Todas estas decisiones se llevan a cabo consultando en tiempo real el modelo preentrenado fabricado a partir de una base de datos automatizada, de igual manera, creada con datos recopilados.

Una de las ideas principales del proyecto fue fabricar módulos individuales de procesamiento para que, al final, al unirlos, trabajen con un módulo maestro. Las secciones más importantes que componen esta propuesta son: la base de datos automatizada, debido que se trata de reconocimiento facial, los datos que maneja

la base de datos están en formato de imagen, entonces se trabajó automatizando la captura de fotografías para que sea más fácil la actualización de los datos, esto provoca que se pueda modificar la robustez de la base; la siguiente sección es la de entrenamiento, esta se encarga de conjuntar la información obtenida en la base datos y transformarla en un lenguaje máquina, todo esto con el fin de ayudar a la siguiente sección. En esta se tiene una identificación facial en tiempo real, la cual se está tomando en *streaming*. Esta parte se encarga de asociar la información obtenida en vivo con los resultados anteriores para así poder tomar una decisión y mostrar un resultado final. Para lo cual se tiene la participación de un módulo capaz de recibir la última información y transformarla en decisiones físicas, para ello se realizó una simulación del funcionamiento en un circuito prueba. Por último se tiene una parte que es agradable a la vista y se encarga de darle una estructura visual y un ambiente sencillo al usuario y es la interfaz gráfica, también llamada *interfaz de usuario*.

La creación de estos submódulos fue hecha en 2 lenguajes diferentes, "Python" y "C". Puesto que Python es un lenguaje muy versátil, se decidió realizar la mayoría de los módulos ahí. Gracias a ello se pudo tomar la decisión de escoger las Biblioteca y los métodos que más beneficien, analizando la facilidad para programar, eficiencia en el procesamiento de datos y que cumplan el principal objetivo de cada submódulo. Pero también se llevó a cabo la programación en "C" ocupando el ambiente y la placa "Arduino" para la implementación del circuito.

Antecedentes

El análisis del proyecto se explicará detalladamente en los capítulos siguientes. Se profundizará en la evaluación para la toma de decisiones, el ¿por qué? y ¿para qué? del desarrollo de cada módulo. Explicando el por qué este proyecto genera un impacto importante en la industria.

Pero antes, es necesario conocer algunos de los motivantes para elegir este problema, las preguntas que acompañaron durante el desarrollo del mismo y las interrogantes a las que se dieron respuesta con el trabajo terminado.

Motivantes

Existen muchas herramientas de diferentes niveles para desarrollar cualquier aplicación de *IA*, en este caso particular se enfocará al desarrollo de visión por computadora y se cuenta con mucho material de ayuda y soporte para el desarrollo de este proyecto. Al igual que muchos colegas que desarrollaron proyectos de reconocimiento facial con base a estas bibliotecas se discutirán ventajas y desventajas, se compararán características del proyecto y se desarrollarán ideas a futuro para aplicaciones de este mismo tema.

Utilización del sistema de reconocimiento facial para preservar la seguridad ciudadana [1]

Dentro de este artículo la autora *Cristina Domingo* realiza una investigación profunda de todos los pros y contras que tiene la utilización de esta tecnología del reconocimiento facial. Como este proyecto ya esta implementado en algunas empresas y edificaciones del sector privado, busca generar una motivación para la implementación de estos sistemas en nuestra vida cotidiana al público en general en lugares públicos. Explica las ventajas que esto tiene, por lo cuál, aplicándolo al desarrollo de vida cotidiano, se pueden dar ventajas como la identificación de criminales en la vía publica, posibles atentados en eventos masivos, así como prevención de asaltos en transporte público analizando el comportamiento sospechoso por medio de la identificación facial y reconocimiento de imágenes. Estos son solo algunos ejemplos de aplicación del reconocimiento facial en lugares públicos, pero se tiene una infinidad de aplicaciones posibles. Cómo lo menciona el artículo, la prohibición de esta tecnología, lamentablemente, no es por falta de interés ni mala operación de los algoritmos, si no por el mal uso que se le puede dar, tener organizaciones en el poder que puedan corromperse y generar un mal uso es algo muy peligroso puesto que se están exponiendo bases de datos con información delicada. Y aquí es donde entran los grupos de defensa de los derechos humanos, al ser una tecnología de fácil recopilación de datos genera una angustia y por la privacidad de la

comunidad y muchos, en lugar de ver las ventajas, solo ven las desventajas y reprimen el desarrollo y crecimiento de esta gran herramienta.

Dentro de este articulo solo se trata el tema general del reconocimiento facial, pero da una diferente perspectiva a lo que nosotros los desarrolladores tenemos en la cabeza. Muchas veces solo nos encargamos de facilitar tareas al usuario, verificar algoritmos y hacer más eficientes los sistemas sin importarnos la otra parte de la comunidad. Nuestros trabajos deben de ser equilibrados, que no generen un impacto negativo a la forma de pensar pero que también sean innovadores para la parte restante de la población.

Con lo antes dicho, el sistema de identificación facial trabajado se enfoca a los lugares donde se tiene que hacer un registro previo de los usuarios, en ningún momento se recopilan datos de personas que no van a formar parte de este sistema de seguridad. Dentro del lugar de operación, el responsable debe de dar aviso al usuario que se le tomarán fotografías para generar la base de datos, es decisión de cada usuario si acepta o no, y dentro del mismo Algoritmo, en la ejecución del programa, se da aviso y se pregunta al usuario si está de acuerdo en continuar con el proceso, ya que, a partir del siguiente paso, quedará expuesto a la cámara extrayendo 200 fotogramas dentro de un video en streaming con una velocidad menor de 20 segundos.

Con el apoyo de este articulo se generan algunas ideas que se podrían implementar en el sistema desarrollado, aumentar la seguridad en la base de datos es una muy buena opción debido a que es el lugar donde se almacenan los fotogramas de los usuarios y estas quedan expuesta en la computadora de operación, debido a esto, sería conveniente proteger el acceso a esta carpeta para no poder manipular, extraer o agregar información de los usuarios.

Aplicación práctica de la visión artificial para el reconocimiento de rostros en una imagen, utilizando redes neuronales y algoritmos de reconocimiento de objetos de la biblioteca OpenCV [2]

En este proyecto se hace un reconocimiento facial, sobre una imagen cargada desde el computador con ayuda de un algoritmo diferente al utilizado en nuestro

proyecto. Ellos se encargan de simplemente poner en practica la identificación facial, realizan una etapa de entrenamiento donde ellos, para alimentar la base de datos en su directorio local, cargan una imagen que contenga al menos un rostro de la persona a almacenar, su algoritmo se encarga de buscar posibles rostros dentro del campo de visión de la imagen y de las distancias colocadas en las características de su algoritmo. Con ayuda del modelo clasificador Heigenfaces detectan todos los posibles rostros en la imagen y los almacenan, para después ejecutar su modelo de entrenamiento. Después, en la segunda parte, vuelven a cargan una imagen para ahora hacer la identificación de usuarios. Al igual que en nuestro proyecto, señalan el contorno del rostro y señalan que es un usuario conocido.

Este proyecto muestra un método y una posible aplicación diferente a los utilizados. En primer lugar, el método; ocupan un modelo pre entrenado de la biblioteca de OpenCV igual con método de clasificador de cascada (HaarCascade) pero de una clase diferente a la nuestra. Dentro del Face analytic de OpenCV hay tres algoritmos destinados a esto. Dos de reconocimiento básico (Heigenfaces y Fisherfaces) y uno más profundo LBPH (Local Binary Pattern Histograms). La diferencia de estos es el tipo de textura y recopilación de datos biométricos que analizan en los rostros de los usuarios, siendo el algoritmo Heigenfaces el más básico, en segundo lugar, con algunas modificaciones y mejoras, pero siguiendo la misma base de operación, el Fisherfaces, y al final, el LBPH. Para las características del proyecto, el que mejor se adaptó al análisis por streaming fue el algoritmo LBPH debido a que el Heigenfaces que ocupan ellos presentó una baja tasa de fiabilidad a la hora de hacer el reconocimiento, teniendo fallos al 80 %. En segundo lugar, la aplicación que recomiendan los autores es un control de presencia o de acceso en varios ámbitos educativos y empresariales. Pero debido a que es un análisis fijo, dentro de una imagen almacenada y cargada a la aplicación, esto puede tener varios puntos en contra, puesto que el acceso y el control de presencia debe de ser en tiempo real. Lo que podemos recomendar es hacer una aplicación tipo álbum fotográfico, donde pueda registrar usuarios y cada que se tome una nueva foto, la aplicación sea capaz de identificar qué personas aparecen en la fotografía, para después tener un buscador.

API para control de asistencia con reconocimiento facial usando OpenCv.JS [3]

En este artículo, los autores desarrollan una investigación para la creación de una API (Interfaz de Programación de Aplicaciones) para el control de asistencia en un salón de clases. En la primera parte de su artículo, ellos realizan una profunda investigación en la cual, por medio de investigación técnica, cualitativa y cuantitativa, generan encuestas y desarrollan unas gráficas en las cuales muestran los métodos utilizados para la toma de asistencia en el ambiente estudiantil. Lo que buscan ellos es generar una aplicación óptima donde reduzca el tiempo desperdiciado durante esta toma de asistencia, sustituyendo estos minutos perdidos con alguna actividad curricular.

El método utilizado para la identificación facial es con la biblioteca de código abierto OpenCV, el diseño de la interfaz es con *Node.js* y para la creación de base de datos es con *MySQL*, todo esto programado en el lenguaje de *JavaScript*. El método para la identificación de objetos fue a través de una red de detección de caja múltiple (SSD) que utiliza una red neuronal profunda implementada especialmente para la detección de objetos por parte de la biblioteca OpenCV. Y también a través de *TinyFaces*, algoritmo enfocado a la detección de objetos mejorado para detectar objetos pequeños no enfocados en la imagen central, este algoritmo se fabricó a partir de los problemas que tenían los algoritmos convencionales al no poder identificar objetos de diferentes tamaños, siendo así uno de sus principales características, el poder abarcar hasta los más pequeños detalles, uniendo varias configuraciones en un mismo identificador.

Las ventajas de los métodos utilizados en este proyecto son varios. Primero, hablando de interfaz, es mucho más interactiva, fácil y completa la fabricación con *NodeJs*, a diferencia del proyecto en desarrollo con *Tkinter* que también es interactiva pero no queda muy robusta para el usuario, al final se convierte en una interfaz muy simple. En segundo, la creación de base de datos con *MySQL* es mucho mejor, debido a ser una plataforma dedicada específicamente a la fabricación de esto, pues los datos

fueron almacenados en un directorio local, esto tiene mucho mejor rendimiento y mejor adaptación a una posible expansión futura. Por último, el uso de los dos métodos de identificación de objetos; son métodos destinados a un ambiente general (objetos) que, sí se pueden adecuar a la detección de rostros, pero necesitan un *Dataset* de alguna biblioteca como *TensorFlow* o *PyTorch* donde puedan extraer y realizar el modelo de entrenamiento, al final con la unión de estos dos modelos es cuando ya se puede hacer una identificación facial. A diferencia del diseño presentado, los algoritmos son destinados al análisis facial, donde se puede realizar propios conjuntos de datos y crear modelos de entrenamiento específicos para los usuarios. Aunque son muy buenos algoritmos, no son los mejores para el desarrollo del proyecto.

La aplicación que le dan ellos a su *API* es sumamente funcional, se concluye que utilizan las herramientas adecuadas, los métodos y algoritmos más eficientes para esta tarea, al tener pensado tomar asistencia en un salón de clases, es muy funcional tomar la fotografía del salón completo para luego identificar cada rostro en una sola imagen cargada a la aplicación, lo cual es más eficiente porque todos los alumnos están en un mismo sitio por al menos 50 minutos. En la aplicación se debe hacer el escaneo rápido y en vivo por las personas que transitan ahí para poder accesar, es por eso que no es un método que ayude al proyecto.

Sistema de bajo coste para detectar personas con mascarilla y su temperatura a través de redes neuronales [4]

Este trabajo plantea un sistema a través de redes neuronales convolucionales que detecte en tiempo real si un usuario lleva consigo cubre bocas o no, además de añadir un sensor capaz de detectar la temperatura corporal. Separan el sistema en dos módulos principales, el primero de *IA* donde es la detección de objetos y el segundo la parte analógica donde se realiza el monitoreo de temperatura.

Para la parte de clasificación de objetos, ellos utilizan un sistema de código abierto llamado *YOLO (You Only Look Once)* que se utiliza para la detección de objetos en tiempo real, funciona a través de una única red neuronal convolucional que aumenta la velocidad de procesamiento por no tiene tantos filtros. También utiliza el algoritmo

K-means, que es un algoritmo no supervisado para el agrupamiento de datos o llamado *Clustering*. Por último, la red neuronal de *MobilNet*, que simplemente es una red neuronal convolucional que ocupa muy bajos recursos, adecuada para operar con diferentes aplicaciones del genero Artificial Intelligence of Things (AIoTs).

El proceso de entrenamiento fue con una recopilación de imágenes con y sin mascarilla, se alimentó un *dataset* y se realizó el entrenamiento de la red neuronal para la clasificación ayudándose de la biblioteca de *Tensorflow*. Ya en la ejecución del sistema, la identificación fue desarrollada en tiempo real, mostrando los resultados en una interfaz sencilla señalando si el usuario tiene o no mascarilla, mostrando en un recuadro la temperatura cutánea.

Estas son las herramientas ocupadas por estos autores, puesto que su diseño fue fabricado con una característica especifica que es la de ser dedicada a un ambiente de bajos recursos. A pesar de utilizar una cámara de bajo coste, los resultados fueron válidos y tuvieron un 80 % de fiabilidad. Debido a la orientación de su aplicación, no afecta demasiado si este porcentaje reduce, por ser solo es un sistema de prevención, es muy diferente este campo de aplicación a uno por ejemplo de seguridad donde tengamos que desbloquear algún dispositivo con la identificación facial.

A diferencia del proyecto desarrollado, en este la elección de estos algoritmos y herramientas son dedicados al bajo rendimiento, por lo cual los algoritmos y la manera en que crearon el entrenamiento y los *dataset* son especiales y adecuados para esto. En todo este campo, existen bibliotecas, módulos y algoritmos destinados a diferentes niveles y el aplicado en este proyecto es uno de bajo nivel, donde se tiene que verificar paso por paso el proceso de análisis de la red neuronal, teniendo en cuenta la máxima reducción de procesos, esto hace que se revise el desarrollo del método de optimización, el descenso del gradiente, las capas de convolución de matrices, las capas de reducción o *Pooling*, etc. En el sistema desarrollado de acceso, el objetivo no es desarrollar un sistema de bajo costo, por lo cual no se busca optimizar al máximo la cantidad de recursos. Se utilizan lenguajes de programación de alto nivel y modelos preentrenados que no están optimizados al máximo, dándole un enfoque mayor alporcentaje de confiabilidad.

En la parte física del proyecto, ellos realizan un análisis de costos para montar el procesador, el termómetro y la cámara, dentro de 4 opciones que tiene, 3 son con la placa *Raspberry* y una con el módulo *MaixCube*. Debido al bajo costo de este último, deciden ocupar este módulo y hacer el prototipo en torno a este sistema físico. La diferencia con el desarrollado es que el procesamiento lo realiza la placa, en este caso, todo el proceso lo realiza la computadora para al final mandar la señal booleana a la placa *Arduino-uno*, teniendo una versión aún más económica con la placa *Arduino-nano*.

Control de acceso e integración CCTV en línea en el Edificio JC, municipio de Fusagasugá. [5]

Dentro de este trabajo de grado los autores implementan un sistema de acceso y salida con el objetivos muy parecidos. Ellos proponen un control de seguridad a través de procesos de captura de datos biométricos, aseguran que aumenta la confianza de los usuarios al ser datos únicos y aumenta significativamente la seguridad del edificio, dentro de los cuales solo consideran la huella dactilar, además de utilizar una tarjeta magnética, un *NIP* de acceso o el uso de un *TAG* para el control vehicular. En la parte de la supervisión, lo desarrollan a través de un circuito cerrado de televisión ($CCTV$) en línea, para después poder obtener monitoreo del edificio, estacionamiento y áreas comunes y así expandirlo a un software soportado en teléfonos móviles, tabletas, laptops y computadoras de escritorio.

Lo interesante de este trabajo y la parte que se debe analizar y discutir con respecto al proyecto es la diferente tecnología aplicada para un mismo objetivo. Plantean una tecnología muy efectiva en cuestión de seguridad, que actualmente se utiliza para dar acceso a cuentas bancarias, información muy importante, desbloqueo de cajas de seguridad, etc. Se puede decir con firmeza que es el control de seguridad más alto que existe, claro, aplicando varios filtros y haciendo una suma de estos datos biométricos. La seguridad no es ningún problema, pero la comodidad sí.

Considerando que el edificio JC es un complejo multifamiliar habitacional la mejor opción sería brindar, antes que nada, comodidad, esa facilidad de poder entrar a tu

domicilio sin necesidad de pasar pruebas que requieran dedicar un tiempo, más si no tienes las manos disponibles. Justo este es el caso donde la *IA* sale ganando, puesto que si se sutituye el acceso biométrico dactilar por un reconocimiento facial de *IA* se obtienen todos los beneficios de la comodidad. Se pueden realizar las mismas tareas y cumple con los objetivos del proyecto planteado, el registro de personas con la hora de llegada y salida del multifamiliar, el acceso al estacionamiento y al edificio principal, así como el monitoreo de las áreas comunes. Todo esto ocupando las mismas cámaras del circuito cerrado, ahorrando mucho económicamente en los sensores dactilares, lectores de tarjetas magnéticas, teclados para la digitación del NIP y sensores de TAG para los vehículos, ya que todas estas tareas se pueden realizar con una modificación grande en el software, en el código de operación y algoritmos de desarrollo, haciendo mucho más óptimo el sistema para el área al que va dirigido.

Después de abrir el panorama de los trabajos y proyectos realizados con algunas de las herramientas de *IA*, visión por computadora, redes neuronales y también metodologías diferentes, cumpliendo un objetivo en común, su pudo hacer un análisis de varios puntos. Primero se explicaron los objetivos, funciones y herramientas que ocupan en cada proyecto, después se hizo una comparación con nuestro proyecto, para después discutir algunas mejoras o justificar el porqué de la elección de nuestras características. Llegando a una conclusión, pudiendo así dar respuesta a las interrogantes iniciales.

Las interrogantes

¿Qué motivó a utilizar IA?

En primera instancia, la facilidad de adaptación a varias tareas. Utilizar IA es un método versátil que se puede aplicar a muchas áreas de la ingeniería, generando una reducción relativa en costos de hardware.

También, el bajo esfuerzo de la mano de obra que se tiene que realizar fue uno de los motivantes, porque se busca todo el tiempo hacer la vida de las personas más sencilla, eliminando tareas tediosas y muy repetitivas. Es por eso que la identificación

facial es mucho mejor a tener un cuaderno y que una persona de seguridad busque el nombre del usuario entre una lista de 1000 empleados para otorgar acceso. También, es mucho más óptimo tener un control automático de compuertas que esté vinculado al software de IA para que las puertas abran solas y no sea necesario que una persona este manipulando la puerta.

Por último, esta ciencia permite trabajar con muchos módulos de operación, uno de los más importantes que se lleva es la seguridad, puesto que hacia ese campo está dirigido el proyecto y una de las mejores técnicas de seguridad es utilizar datos biométricos, ahora bien, en conjunto con el reconocimiento facial, harían este sistema con seguridad de alto nivel.

Quedando así convencidos, que la IA está por encima de cualquier método ordinario, algún programa informático de programación lineal, queda muy por debajo en lo que queremos aplicar, dando mejores resultados, un código de IA.

¿Por qué se eligieron estas herramientas?

En la primera parte, un lenguaje de programación muy amplio de alto nivel, que ayudara en todas las bibliotecas orientadas a esta ciencia, un lenguaje que no limitara en versiones, módulos y que fuera de fácil manejo a la hora de programar.

La elección de las bibliotecas fue pensada todo el tiempo de código abierto, para que cualquier desarrollador pudiera tener acceso a todas las versiones. Además de ser elegidas por ser las más utilizadas en el campo, con mucha variedad de módulos y modelos preentrenados.

En muchas ocasiones estas elecciones fueron hechas a prueba y error, probando el comportamiento del sistema con el equipo de computo propio, con la cámara propia, con la placa de desarrollo en existencia y demás, todo el tiempo haciendo un análisis, registrando el comportamiento en una bitácora que ayudó al final a hacer una elección entre los métodos probados. Pero también se realizaron investigaciones previas para la elección correcta, como es el caso de la biblioteca para la interfaz gráfica, que se investigaron diversas opciones y al final se llegó a la conclusión de ocupar Tkinter que es soportada solo en PC y laptops.

¿Qué beneficios se obtienen al utilizar una tecnología de este tipo?

Uno de los objetivos de la ingeniería es adaptar la tecnología para las necesidades humanas. En pocas palabras hacerles la vida más fácil a los humanos y esto genera muchos beneficios. Ahora en conjunto con la IA aumentan exponencialmente las buenas noticias, porque al final, los que se llevan el trabajo pesado son los desarrolladores, pero ya el resultado final para los usuarios es muy satisfactorio.

Dentro de los beneficios adquiridos se obtuvo la reducción de costos. El sustituir muchos sensores necesarios para estas tareas, ya que la visión por computadora es un sensor de amplio espectro, en el cual bajo una misma toma puedes recibir mucha más información que con el uso de sensores tradicionales. Hablando solo de visión por computadora obtenemos una cantidad descomunal de información, ya dependerá de la capacidad del computador, la habilidad de los programadores y la correcta aplicación de esta tecnología en un ambiente real. Porque al final no sirve de nada tener tecnología de punta, desarrollada por los mejores, si al final no se le da la verdadera función, o simplemente no está siendo explotada en las zonas para las que fue desarrollada.

¿Qué se busca a futuro?

En primer lugar, se busca aplicar este proyecto, que no solo quede en un prototipo. Se planea montarlo en algún edificio del sector privado, en algún edificio escolar de uso administrativo o deportivo.

En la parte técnica, se busca expandir este proyecto a varios puntos de conexión, para que no solo desde una cámara se haga el acceso, si no que, desde varios puntos de operación, haya cámaras listas para la detección y todo se ejecute en un mismo procesador. En segundo lugar, se busca ampliar las características de detección, donde el sistema sea capaz de crear un registro de horario, fecha y zona en la que se identificó un usuario, para llevar un control administrativo en caso de cualquier situación que se presente y se tenga que corroborar quien estuvo en cierta zona. Como tercer punto, se planea implementar más algoritmos de detección,

aprovechando las cámaras de detección facial podemos reconocer ciertos patrones en el comportamiento, específicamente en las expresiones faciales y así poder hacer recomendaciones personalizadas con un análisis de predicción. Por último, en la parte de seguridad, se puede agregar un modelo preentrenado para la detección del uso de cubre bocas debido a que todavía en ciertos lugares está prohibida la entrada sin cubre bocas. También, dependiendo la aplicación de este sistema, se puede aumentar el nivel de seguridad, colocando otros filtros o barreras de acceso con NIP o más datos biométricos como detección de huella dactilar.

Objetivos

Objetivo general

- Diseñar un sistema de reconocimiento facial para dar acceso a usuarios a determinadas áreas restringidas de un edificio mediante diferentes herramientas de inteligencia artificial tales como machine learning, redes neuronales y sistemas inteligentes.

Objetivos particulares

- Analizar las diferentes herramientas de IA ocupadas para hacer más eficiente el sistema general mediante consultas a manuales, documentaciones y experimentaciones.
- Comparar los resultados de nuestra experimentación para comprender la importancia del uso de la IA en nuestro día mediante análisis de datos (minería de datos y creaciones de dashboards).

Capítulo 1

Marco de referencia

En este capítulo se abordará una introducción a conceptos importantes como ¿Qué es la inteligencia artificial?, ¿Cómo funciona?, sus diferentes ramas de aplicación, así como definiciones más específicas del tema, tales como: visión por computadora, redes neuronales y algunas de las herramientas más importantes en el desarrollo del proyecto como las bibliotecas, módulos y el porqué del uso de estas. Para después complementar estos conocimientos con el desarrollo central de la creación de los algoritmos.

1.0.1. Inteligencia artificial

La *IA* o inteligencia artificial es un campo de la ciencia y la ingeniería que se ocupa de la capacidad y comprensión de las máquinas o sistemas informáticos para realizar tareas que normalmente requieren inteligencia humana. En el trabajo *"Inteligencia Artificial: retos, perspectivas y papel de la Neutrosofía"* de *M. Leyva-Vázquez y F. Smarandache* se mencionan 7 ramas de investigación de la IA. [6]:

- **Aprendizaje automático:** Es el proceso de las computadoras que desarrollan el reconocimiento de patrones, tiene la capacidad de aprender y hacer predicciones basándose en datos recopilados.
- **Procesamiento del lenguaje natural (NLP):** Es una rama de la *IA* que ayuda a las computadoras a entender, interpretar y manipular el lenguaje humano, buscando la unión definitiva entre estas dos comunicaciones.

- **Sistemas expertos:** Es un sistema que emplea conocimiento humano capturado en una computadora para resolver problemas que normalmente requieran de expertos humanos. Imitan el proceso de razonamiento que los expertos realizan para resolver problemas específicos [7].

- **Visión por computadora:** Es un proceso de varias tecnologías y herramientas por el cual permiten a los equipos captar imágenes del mundo real, procesarlas y generar información a través de ellas [8].

- **Reconocimiento automático del habla:** Es un proceso de clasificación de patrones, cuyo objetivo es ordenar la señal de entrada (onda acústica) en una secuencia de patrones previamente aprendidos y almacenados en unos diccionarios de modelos acústicos y de lenguaje [9].

- **Planificación:** Proceso de encontrar diferentes secuencias de acciones que alcanzan un determinado objetivo si se ejecutan de un estado inicial. Desarrollo de la liberación que escoge y organiza acciones anticipando sus resultados y consecuencias.

- **Robótica:** Es un sistema unido entre el proceso de la creación del hardware de robots con el software de la *IA*, para tener un sistema robótico inteligente capaz de aprender y decidir.

1.0.2. Redes neuronales

Una red neuronal es un conjunto de neuronas artificiales interconectadas entre sí, capaz de realizar un análisis de información, tener un aprendizaje, retener datos y tomar decisiones. Las redes neuronales son un modelo de computación cuya estructura de capas se asemeja a la estructura interconectada de las neuronas en el cerebro, con capas de nodos conectados. Una red neuronal puede aprender de los datos, de manera que se puede entrenar para que reconozca patrones, clasifique datos y pronostique eventos futuros [10].

Por medio de este proceso de aprendizaje se puede lograr que el sistema sea capaz de aprender y tomar las decisiones para la clasificación de personas. No solo existe un tipo de red neuronal, sino que, dependiendo de varios factores se clasifican en diferentes tipos de redes.

Por número de capas: [11]

- **Monocapa:** Como el perceptrón simple, conectado directamente entre la capa de entrada y la capa de salida.
- **Multicapa:** Es el perceptrón multicapa que entre la capa de entrada y de salida tiene una o varias capas ocultas.

Por tipo de conexión: [12]

- **Recurrentes:** Con conexiones arbitrarias entre neuronas, capaces de crear ciclos y consiguiendo temporalidad y memoria.
- **No recurrentes:** Son las menos utilizadas y carecen de memoria.
- **Redes convolucionales (CNN):** Este tipo de red neuronal no se conecta con todas y cada una de las capas siguientes, sino que solo con un subgrupo de ellas, haciendo que se especialice en un conjunto más óptimo con menor número de neuronas, consiguiendo reducir el número de neuronas necesarias y la complejidad computacional necesaria para su ejecución y aumentando la eficiencia. Estas redes son ideales para el procesamiento de visión por computadora.

Por grado de conexiones:

- **Totalmente conectadas.**
- **Parcialmente conectadas.**

Por tipo de aprendizaje:

- **Supervisado:** [13]

- **Aprendizaje por corrección de error:** Se hace una comparación entre los valores de entrada y de salida, buscando la mínima diferencia entre la respuesta de salida y la deseada.
- **Aprendizaje por refuerzo:** El sistema arroja una señal de refuerzo entre "Éxito" y "Fracaso", para que así el diseñador pueda modificar los pesos y tener una respuesta deseada.
- **Aprendizaje estocástico:** Se realizan cambios aleatorios en los pesos de la red y se analiza la salida obtenida con la salida deseada.

- **No supervisado:** [14]
  - **Aprendizaje Hebbiano:** Permite medir la familiaridad o extraer las características de los datos de entrada.
  - **Aprendizaje competitivo y comparativo:** Permite hacer clasificaciones de los datos de entrada.

Después de dar un análisis teórico a los diferentes tipos de redes neuronales, entender el funcionamiento, características, ventajas y desventajas, se destaca que las redes neuronales convolucionales son de las mejores opciones para manejar el procesamiento de imágenes, aunque en este proyecto se utiliza un método diferente por medio de clasificadores en cascada para el entrenamiento y el algoritmo estadístico de histogramas para la identificación de rostros, es importante detallar que las redes neuronales convoluciones son muy utilizadas para estas tareas. Dentro de la visión por computadora se busca que el sistema sea capaz de comprender e identificar qué tipo de imágenes se está observando, de igual manera que se hace al observar un objeto y lograr diferenciarlo entre una persona, una prenda de vestir o una mascota. Al final se busca un proceso similar, por ejemplo, para poder identificar qué es un perro, es necesario obtener datos de entrada como saber que tiene 4 patas, tiene abundante pelo, tiene orejas, dientes, etc. Pero eso no es todo, ya que con esa información sería fácil confundirse entre un perro y algún otro animal, entonces es necesario enriquecer la base de datos, la base de conocimientos donde se alimentará con ejemplos de lo que

realmente es un perro, dónde la computadora sea capaz de analizar pixel por pixel, dependiendo el método o algoritmo utilizado, la forma física de un perro. Obtener ejemplos de calidad, sin ruido, es importante para enriquecer el entrenamiento. Ya con ejemplos reales de cómo es, guiamos a nuestro modelo de entrenamiento a tener una clasificación correcta de este animal. En el caso de una red convolucional, tenemos la ventaja de cuando se necesite realizar un nuevo aprendizaje de un objeto diferente, la propia topografía de la red va a orillar al entrenamiento a tomar un camino diferente y más específico en una próxima consulta.

En la figura 1.1, se observa un ejemplo simplificado de como la red obtiene características especiales de la imagen, valores estandarizados entre 0 y 1, donde primero se realiza un redimensionamiento a la imagen para poder sacar una matriz de tamaño especifico, para después ir reduciendo el tamaño de los filtros e ir analizando las secciones requeridas de la imagen, en este caso la bicicleta. Es importante mencionar cómo es que funcionan estas redes debido a la gran popularidad de estas y para el apoyo correcto de futuros proyectos de visión por computadora.

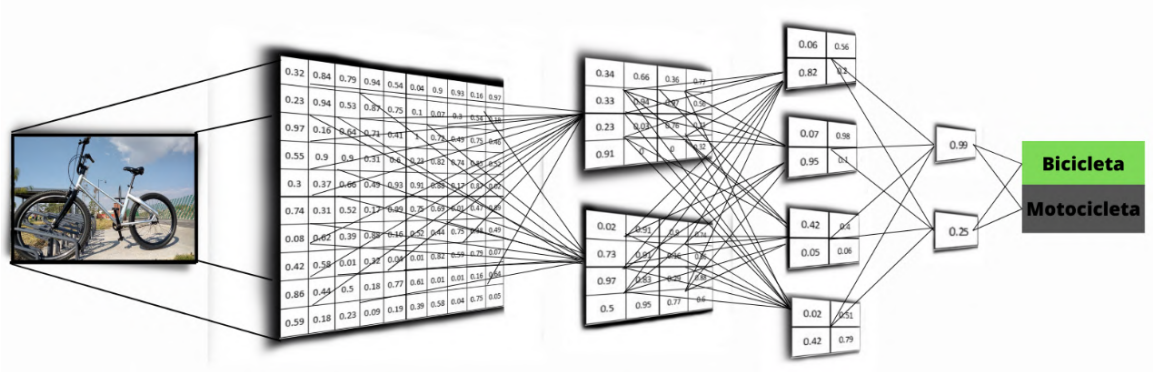


Figura 1.1: Ejemplo de cómo operan los filtros en una RNC. [15]

1.0.3. Reconocimiento de imágenes

El reconocimiento de imágenes es un campo dentro de la rama de visión por computadora que ofrece la IA. Este campo ofrece un proceso de entendimiento a los computadores para interpretar, identificar y extraer características del mundo visual, gracias a esto, las maquinas pueden lograr clasificar, ordenar y reaccionar ante los

estímulos visuales.

Este proceso funciona extrayendo características individuales de una imagen o de un video. Dentro del portal de SAS Insights [16] hacen una analogía del funcionamiento del reconocimiento de imágenes muy interactiva:

"Los ordenadores reúnen imágenes visuales de la misma forma en la que se puede encajar un puzzle. Piensa en cómo se forma un puzzle. Necesitas juntar todas esas piezas para crear una imagen. Distinguen muchas partes diferentes de la imagen, identifican los bordes y seguidamente moldean los subcomponentes. Mediante el filtrado, pueden juntar todas las piezas de una imagen, prácticamente de la misma forma en la que harías un puzzle."

"El ordenador no cuenta con una imagen final de referencia en la caja del puzzle, sino que suele disponer de cientos o miles de imágenes relacionadas para entrenarse en el reconocimiento de objetos concretos."

Con esta referencia de la explicación más aterrizada al ambiente cotidiano, se puede entender el funcionamiento de este campo. Entonces, no es necesario llenar una base de datos para formular un grupo de imágenes positivas con características especiales de un objeto, simplemente cubrir esos grupos de imágenes para entrenamiento con ejemplos del objeto a identificar o características a extraer. Si se requiere el análisis para identificar autos dentro de un video de una cámara de seguridad, la base de datos para el entrenamiento del modelo de machine learning se alimentará de millones de fotografías de diferentes tipos de autos y no características particulares de los autos, como por ejemplo mil fotos de llantas, mil fotos de puestas, mil fotos de parabrisas, etc.

El reconocimiento de imágenes, como se muestra en la imagen 1.2, sigue un proceso de 3 pasos para llegar a su objetivo :

Figura 1.2: Procedimiento para el reconocimiento de imágenes. [17]

La obtención de la imagen es en donde el computador adquiere el fotograma donde

se encuentra ubicado el objeto a identificar.

En el procesamiento de la imagen entran en acción los algoritmos de aprendizaje profundo, entrenando estos modelos con miles de imágenes tanto positivas como negativas, para separar características.

Por ultimo, la comprensión de la imagen es el paso en el que el computador ya cuenta con las características de cada objeto para hacer la comparación y poder identificar y clasificar.

Existe una gran cantidad de ventajas al utilizar el reconocimiento e identificación de imágenes que se fueron implementando conforme avanzó este campo. Se pudo identificar que es mejor el sustituir tareas humanas con este procesamiento de imágenes ya que es mas eficiente y se ha comprobado que se hace con menor tasa de error, sin mencionar que es mas adaptable a cualquier espacio y que reduce costos.

Algunas de las aplicaciones más comunes enfocadas al desarrollo de *deep learning* son mencionadas por los desarrolladores de *MathWorks* en su articulo *"¿Qué es el reconocimiento de imagenes?"*, las cuales son [18]:

- ***Inspección visual:*** *Proceso de identificación de elementos o características tales como piezas defectuosas o no defectuosas en la fabricación, esto permite inspeccionar rápidamente miles de piezas en una cadena de montaje.*
- ***Clasificación de imágenes:*** *Categorización de imágenes en función de su contenido, esto resulta especialmente útil en aplicaciones tales como recuperación de imágenes y sistemas de recomendación en comercio electrónico.*
- ***Conducción autónoma:*** *Capacidad de reconocer una señal de stop o un peatón en una imagen, esto es fundamental en las aplicaciones de conducción autónoma.*
- ***Robótica:*** *Reconocimiento de imágenes utilizado por robots para mejorar la navegación autónoma e identificar ubicaciones y objetos en su ruta.*

Según *SAS institute*, existen diferentes tipos de reconocimiento de imágenes que cumplen con diferentes objetivos de la tabla 1.1.

Tipos de reconocimiento de imágenes por su tipo de características.	
Segmentación de la imagen	Separa la imagen en secciones para analizarlas por separado.
Detección de objetos	Identifica un objeto en especifico.
Reconocimiento facial	Detecta un objeto en especifico (rostro) pero además logra identificar entre personas en particular.
Detección de bordes	Identifica el borde exterior de una objeto.
Detección de patrones	Reconoce formas, colores y características que se repiten en una imagen.
Clasificación de imágenes	Agrupa imágenes a una categoría dependiendo del contenido de la imagen.
Segmentación de rasgos	Detecta patrones para clasificar en categorias.

Tabla 1.1: Tipos de reconocimiento de imágenes por sus características de identificación

1.0.4. Machine learning

El *machine learning (ML)* o aprendizaje automático, es una técnica de análisis de datos que enseña a los ordenadores a hacer lo que resulta natural para las personas y los animales: aprender de la experiencia. Los algoritmos de aprendizaje automático emplean métodos de cálculo para aprender información directamente de los datos sin depender de una ecuación predeterminada como modelo. Los algoritmos mejoran su rendimiento de forma adaptativa a medida que aumenta el número de muestras disponibles para el aprendizaje [19].

Con el aumento de la cantidad de big data (véase sección 1.0.6), el *machine learning* tiene la capacidad de solucionar problemas en áreas como:

- **Finanzas:** Para la calificación crediticia y el *Trading* algorítmico.
- **Procesamiento de imágenes:** Cualquier tarea relacionada al reconocimiento facial y la detección de objetos.
- **Biología:** Funciona en procesos como detectar tumores o en la medicina para

descubrir nuevos fármacos y en el análisis del ADN.

- **Mecánica automotriz y aeroespacial:** Útil para el mantenimiento y supervisión predictiva.
- **Procesamiento del lenguaje natural:** Para aplicaciones del lenguaje así como el reconocimiento de voz.

El aprendizaje automático se enfoca principalmente en dos puntos principales: el aprendizaje supervisado, que implica entrenar un modelo utilizando ejemplos etiquetados para predecir resultados futuros, y el aprendizaje no supervisado, que descubre patrones o estructuras en los datos sin utilizar etiquetas previas. Algunas veces se llega a emplear una tercera técnica que es el aprendizaje reforzado que se ocupa para la toma en tiempo real de decisiones, para los juegos con *IA* y para la navegación robótica como se puede observar en la fig. 1.3

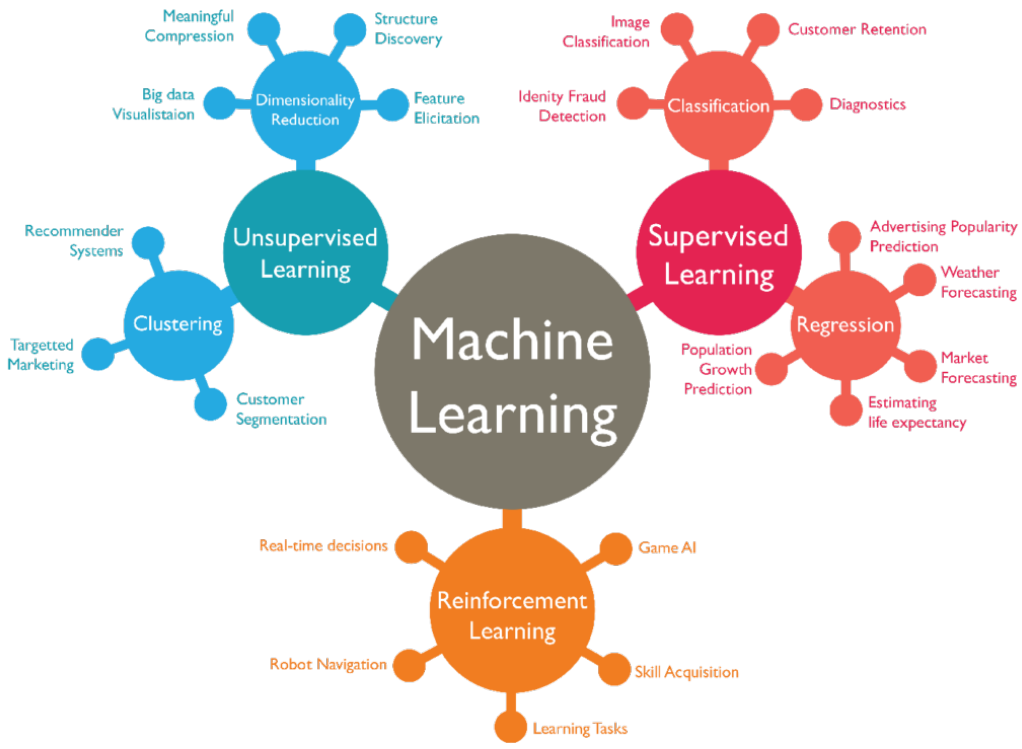


Figura 1.3: Esquema general del machine learning. [20]

1.0.5. Deep learning

Deep learning es un subconjunto del *machine learning* (que a su vez es parte de la inteligencia artificial fig. 1.4) donde las redes neuronales, que son algoritmos inspirados en cómo funciona el cerebro humano, aprenden de grandes cantidades de datos [21].

Los algoritmos de aprendizaje profundo llevan a cabo tareas de manera iterativa, lo que conduce a mejoras graduales en los resultados a través de *Deep Layers*. Este enfoque facilita un proceso de aprendizaje progresivo. Estos algoritmos forman parte de una categoría más amplia de técnicas de *machine learning* que se basan en redes neuronales.

El deep learning ha tenido un impacto significativo en diversas industrias. Puede aplicarse para:

- Analizar imágenes de manera avanzada.
- En la investigación para el descubrimiento de medicinas.
- Predecir problemas de salud y síntomas de enfermedades.
- Acelerar el conocimiento mediante la secuenciación genómica.
- Ayudar a los vehículos autónomos a adaptarse a las condiciones cambiantes en el transporte.
- Para proteger infraestructuras críticas.

1.0.6. Big data

El *big data* (datos masivos) es el término que describe un gran volumen de datos, el cual crece de manera exponencial con el paso del tiempo.

El *big data* está formado por conjuntos de datos de gran tamaño y más complejos, procedentes de nuevas fuentes de datos no estructurados. Estos conjuntos de datos

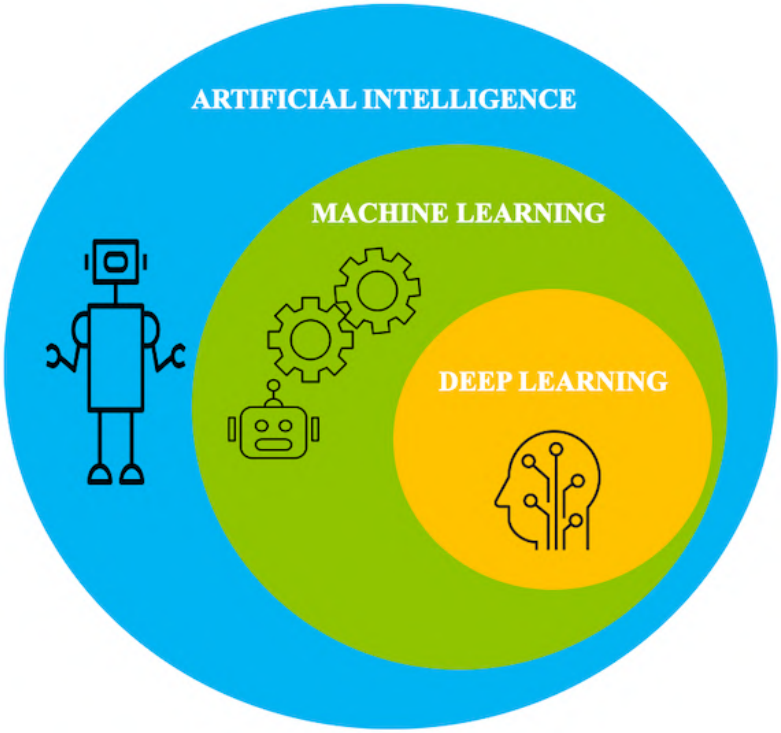


Figura 1.4: Deep learning, subconjunto del ML y de la IA. [22]

son tan voluminosos que es necesario el uso de software no convencional para el procesamiento de datos, debido a que no cualquier computadora puede gestionarlos.

Según el portal de internet de la Universidad Católica de San Pablo el análisis de grandes volúmenes de datos se distingue por tres atributos principales [23]:

*"**Volumen:** pueden ser datos de valor desconocido, como los Feeds de datos de Twitter, flujos de clics en una página web o en una aplicación para móviles, o equipos con sensores habilitados. Esto puede significar decenas de terabytes de datos, incluso, cientos de petabytes."*

*"**Velocidad:** la velocidad es el ritmo en el que se reciben y procesan los datos. Algunos productos inteligentes habilitados para internet funcionan en tiempo real y requieren una evaluación y actuación en tiempo real lo que produce una captura de datos elevada a una velocidad considerable."*

*"**Variedad:** la variedad hace referencia a los diversos tipos de datos disponibles. Los datos convencionales son estructurados y pueden organizarse claramente en una base de datos relacional. Con el incremento de datos masivos, estos vienen en nuevos*

tipos de datos no estructurados. Los tipos de datos semi y no estructurados, como textos, audios y videos, requieren un preprocesamiento adicional para deducir su significado y ser compatibles con los Metadatos."

Dentro de la tabla 1.2, se puede observar los 3 tipos de datos que se manejan en el *big data* con una breve definición y composición de cada uno de ellos.

Tipos de datos en el big data	
Estructurados:	Cualquier dato que se pueda almacenar, acceder y procesar en formato fijo.
No estructurados:	Son cualquier dato en formato desconocido o datos heterogéneos que contienen una combinación de archivos de texto simples, imágenes, vídeos, entre otros.
Semiestructurados:	Contienen una combinación de datos estructurados y no estructurados. Por lo general, estos datos tienen un formato definido, pero pueden ser difíciles de entender para el usuario sin la aplicación de reglas complejas que especifiquen cómo interpretar cada parte de la información Un ejemplo de un dato semiestructurado es uno representado en un archivo XML.

Tabla 1.2: Tipos de datos en el big data

Algunos de los beneficios del *big data* son que permite obtener respuestas más completas, ya que tiene mayor cantidad de información, esa disponibilidad de respuestas más completas significa un mayor número de fiabilidad, lo que implica un enfoque distinto a la hora de abordar problemas. Algunas de las aplicaciones del *big data* son [24]:

- Desarrollo de productos.
- Mantenimiento predictivo.
- Experiencia de cliente.
- Fraudes.
- Eficiencia operativa.

- Innovación.

El big data es una herramienta muy importante dentro de la visión por computadora y el analisis de imágenes, ya que dentro de estas ramas de la IA es común que se analicen grandes cantidades de información y al ser bases de datos con bancos de imágenes, el tamaño de estas ocupa mas espacio que una base de datos con datos simples y estructurados. Aunque para este proyecto no es necesario recurrir al big data, ya que en esta ocasión, por el enfoque del proyecto realizado, es suficiente desarrollar el entrenamiento con una poca cantidad de datos debido al enfoque y objetivo de operación. Si se planea expandir la implementación a un nivel de operación mayor, es necesario contar con esta información.

1.1. Marco teórico

1.1.1. HaarCascade

Es un modelo de clasificación de objetos creado por Paul Viola y Michael Jones en 2001 de baja demanda computacional para la epoca actual a diferencia de las redes neuronales convolucionales y se basa en los niveles de intensidad que presentan los pixeles de una imagen.

Este clasificador es un modelo pre entrenado de la biblioteca OpenCV que sigue el método de clasificación en cascada por medio de árboles de clasificación. La técnica de cascada se basa en la unión secuencial de varios clasificadores débiles, donde cada uno analiza una sección distinta de una imagen o fotograma en el caso de un vídeo. Estos clasificadores son considerados débiles porque tienden a tener una alta tasa de falsos positivos. Sin embargo, al combinar los resultados de todos los clasificadores, en conjunto se vuelven muy eficaces. El algoritmo utiliza ventanas del mismo tamaño con 2, 3 y 4 rectángulos de igual dimensión como se ilustra en la figura 1.5. Tiempo despues, se agregó un nuevo formato de dos regiones para poder identificar centros y contornos como se muestra en la figura 1.6. En cada una de esas ventanas aplica la función Haar que se calcula como la suma de los píxeles que se

encuentran dentro de los rectángulos blancos y se resta con la suma de los píxeles del rectángulo sombreado. Esto permite encontrar las características del objeto de análisis comparando el resultado en cada una de las imágenes positivas con las imágenes negativas y encontrar similitudes en todas las imágenes positivas. En el momento del entrenamiento se necesita contar con un dataset de imágenes positivas, en el cual se encuentra el objeto que se quiere detectar y un dataset de imágenes negativas donde no se encuentra el objeto en cuestión [25].

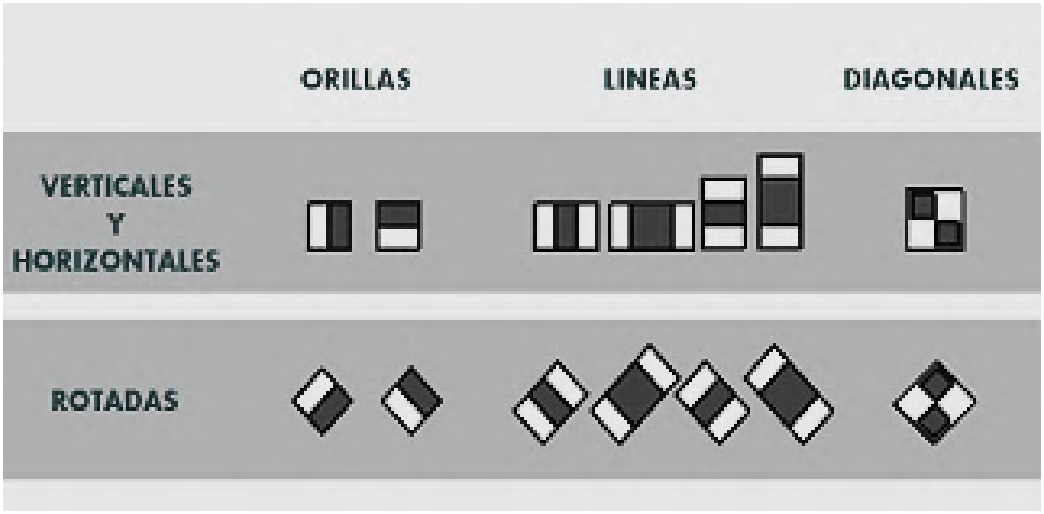


Figura 1.5: Filtros digitales tipo haar. [26]

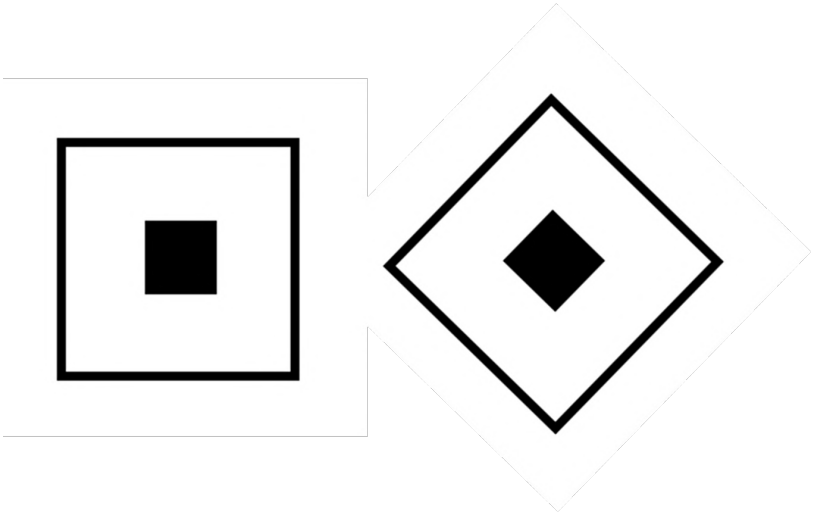


Figura 1.6: Filtros digitales en cascada para centros y contornos. [27]

Los autores de este modelo lo describen como un clasificador debil, pero ¿cúal es el

motivo de esto? La respuesta esta en la forma en que opera la detección en la imagen, poniendo de ejemplo que se quiere identificar una oveja negra dentro de una fotografía con muchas ovejas blancas en un campo, el modelo realiza un escaneo con los 4 tipos de ventanas mencionadas anteriormente en todas sus direcciones dando un numero determinado de características de toda la imagen de por ejemplo 20x20 pixeles. El nivel de procesamiento computacional es altamente demandante y en este punto, el modelo es muy debil por arrojar muchos falsos positivos extrayendo mas de 15mil características. Dentro de este primer filtrado se busca posicionar una ventana la cual identifique a esa oveja negra colocando un recuadro negro y junto a el cuadros blancos, así es como se identificaría a la oveja, pero con un primer filtro no es suficiente debido a que puede haber 15mil recuadros negros apuntando a árboles, arbustos, piedras, sombras y demás. Es por eso que se ejecuta un proceso llamado *Adaboost* el cual busca la reducción de extracción de características detallando y aumentando la calidad en la fase de entrenamiento, alimentando el banco de imágenes con verdaderos positivos o características de esa oveja negra.

Una vez reducido el proceso computacional es mas facil y mas rapido ejecutar mas procesos de filtrado, de ahí el nombre de clasificadores en cascada. Aquí es donde se vuelve muy fuerte el modelo de clasificación y bajan las tasas de error, ya que se aplican mas filtrados enfoncandose solamente en las areas de atención o donde se extrajeron mas características en la ronda anterior, así hasta encontrar a esa oveja negra.

Actualmente se enriqueció su desarrollo creando modelos pre entrenados para diferentes tipos de detección, como por ejemplo:

- **FrontalFace:** Para detectar rostros de frente.
- **FullBody:** Detecta personas de cuerpo completo.
- **Eye:** Ayuda a ubicar los ojos en las imágenes.
- **ProfileFaces:** Identifica rostros vistos de perfil.
- **Smile:** Detecta sonrisas.

- **EyeGlasses:** Puede detectar el uso de gafas.

Ademas de poder adaptar este modelo a cualquier tarea en particular.

1.1.2. LBPH (*Local binary patterns histograms*)

El método de histogramas de patrones binarios locales es un operador de descripción visual diseñado para etiquetar pixeles y otorgar una textura a la imagen. Es un método mejorado de sus antecesores para reconocimiento facial (EigenFaces y FisherFaces) con el fin de eliminar los problemas en la descripción de características de un rostro.

Para formar la descripción o textura general de la imagen, se realiza un recorrido en la imagen ya procesada, es decir, alineada, en escala de grises y redimencionada y se le aplica el operador LBPH. Este recorrido se realiza con una ventana enjaulando una matriz de dimenciones 3x3 pixeles como se muestra en la figura 1.7 donde a cada pixel extrae una caracteristica y lo transforma a un valor dependiendo el tono a la graduación de grises de 0 a 255.

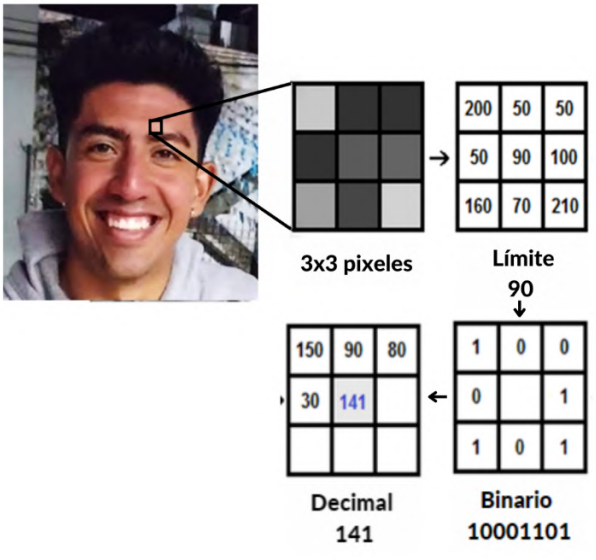


Figura 1.7: Ejemplo de obtención de parámetros del algoritmo LBPH. [28]

Estos pixeles se separan en dos, el central y los 8 adyacentes los cuales van a ser

comparados con el pixel central, la comparación se realiza con un circulo en diección horaria de forma ordenada a partir de la esquina superior izquierda para estandarizar el valor a binario, 1 o 0 dependiendo el valor del pixel comparado como se muestra en la figura 1.8.

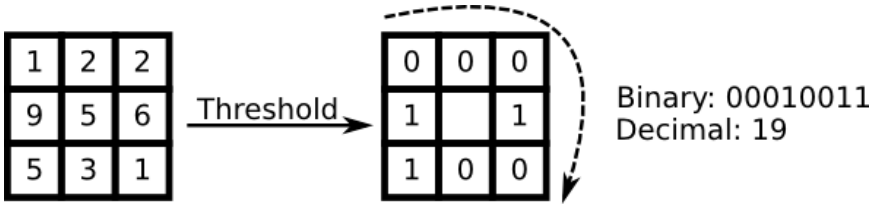


Figura 1.8: Estandarización de características LBPH. [29]

Para al final regresar ese numero binario a decimal y hacer las operaciones requeridas con todas las matrices estandarizadas en la imagen. Obteniendo un histograma de todas las etiquetas y formar una descripción de la textura final.

Existe algunas ventajas de este operador debido a su poder discriminativo y con un bajo costo en los procesos computacionales, el operador de textura LBPH se ha convertido en una buena opción en diversas aplicaciones. Por su procesamiento en los modelos estadísticos y estructurales tradicionalmente del análisis de texturas. La ventaja mas importante de este modelo discriminante de texturas es la eficiencia y buena tasa de respuesta a los cambios de graduaciones en la escala de grises que son provocados por cambios en la iluminación. Dentro de la imagen 1.9 de los desarrolladores de OpenCV se observa la respuesta del operador a diferentes condiciones de luz.

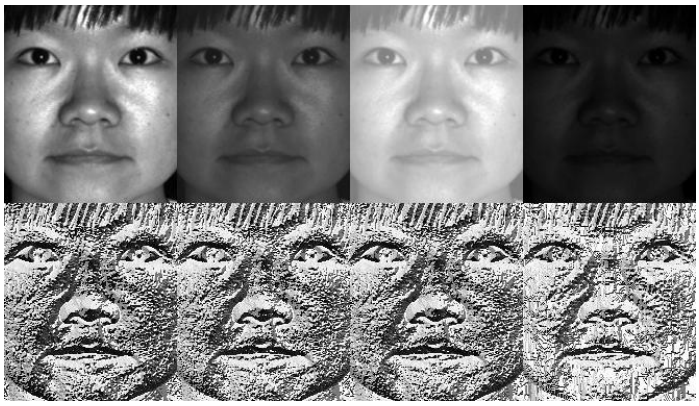


Figura 1.9: Muestra de la aplicación de textura del algoritmo LBPH. [30]

Capítulo 2

Estado del arte

Para el desarrollo de este capítulo, fue necesario tocar algunas de las ramas que ofrece la inteligencia artificial, debido a que es una ciencia amplia y versátil que tiene la facilidad de ser aplicada en cualquier área de aprendizaje y/o conocimiento, lo cual permite desarrollar la idea central del desarrollo de la visión por computadora y el análisis de imágenes.

Antes de entrar al tema, se tienen que comprender el funcionamiento de herramientas importantes que fueron necesarias para el desarrollo, conceptos teóricos y ejemplos del funcionamiento que están detrás de la programación, todo esto ayudara a darle sentido y dirección a este tema tan extenso.

2.1. La inteligencia artificial

A lo largo de la historia muchos autores han querido dar su definición de inteligencia artificial y aunque todos van en la misma dirección, no existe una definición específica y neutral para esta ciencia. Uno de los personajes más importantes es el científico inglés *Alan Turing* que fue el primero en cuestionarse si las máquinas podían pensar, lo hizo dentro de su artículo *"Computing machinery and intelligence"* [31]. Fue pionero de un nuevo pensamiento hacia la tecnología de la década de los 50's, sin saber que ese nuevo pensamiento hoy en día estaría presente conviviendo día a día en todos los ámbitos de la vida cotidiana, dando nacimiento a

una nueva rama de la ciencia.

Pero no fue hasta 1956 que el término *"Inteligencia artificial"* surgió en una conferencia en la universidad de Stanford por el científico *John McCarthy* [32], dando su definición especial:

"La ciencia e ingenio de hacer maquinas inteligentes"

Tiempo después, diferentes científicos, ingenieros y técnicos dieron varias definiciones de *IA* como *H. Winston* que dice:

"La Inteligencia Artificial es el estudio de las ideas que permiten ser inteligentes a los ordenadores"

Una definición más apegada a la práctica es la del autor *Marvin Minsky*, pionero de la *IA*, que nos dice lo siguiente:

"La Inteligencia Artificial es la ciencia de construir máquinas para que hagan cosas que, si las hiciéramos los humanos, requerirían inteligencia"

Y una definición más completa es la que aparece en *Encyclopedia of artificial intelligence* [33]:

"La IA es un campo de la ciencia y la ingeniería que se ocupa de la comprensión, desde un punto de vista informático, de lo que se denomina comúnmente comportamiento inteligente. También se ocupa de la creación de artefactos que exhiben este comportamiento"

Con la creación de este proyecto, las clases sobre el tema, los profesores, los cursos y demás preparación para el desarrollo de esta idea, lleva a pensar que la inteligencia artificial es una ciencia multidisciplinaria, que busca realizar procesos inteligentes que tienen los seres vivos, aplicándolos a computadores para poder sustituir tareas y facilitar la vida de los seres humanos.

Analizando la última definición se puede notar que cumple con varias ramas que tiene la *IA*, se busca automatizar procesos, tener un aprendizaje automático, hacer óptimo un sistema recurriendo a varios algoritmos, sustituir la mano de obra con la robótica y poder tener una conversación con un simple asistente virtual. Así mismo, existe una infinidad de aplicaciones, y al final, todos los autores tienen claro el sentido y dirección de la *IA*.

Con estas definiciones, se tiene claro qué es y a qué se dedica la *IA*, ahora, es momento de ver en qué áreas se puede aplicar.

Dentro de este basto mundo, se difiere un poco en la manera en que la *IA* es clasificada y sus principales ramas de operación, dependiendo de los autores que escriban de ella, varía el número de ramas, pero al final todas cumplen los mismos objetivos y engloban las intenciones modernas de la *IA*, es por eso que se decidió usar como guía la clasificación hecha por el doctor *Maikel Leyva-Vázquez* que divide la inteligencia artificial en 7 ramas principales [6], las cuales son: aprendizaje automático, procesamiento del lenguaje natural, sistemas expertos, visión por computadora, reconocimiento automático del habla, planificación y robótica. (figura. 2.1)

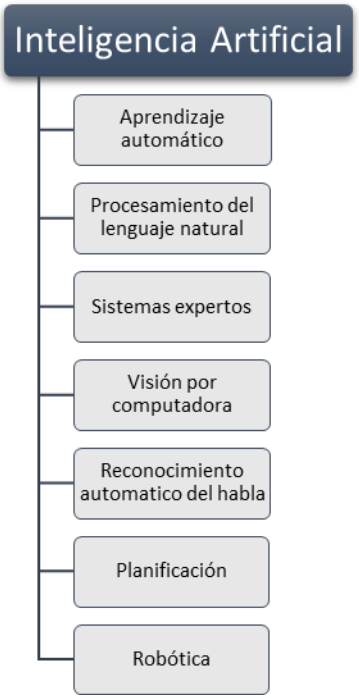


Figura 2.1: Ramas de la inteligencia artificial. [34]

2.2. La visión por computadora

La **visión por computadora** es una rama importante de la *IA* porque su objetivo es emular el sentido humano con gran cantidad de entrada de información, la vista. Debido a los abundantes datos que los humanos pueden recibir a través de este

sentido, se busca realizar procesos similares en la computación. Existen procesos fundamentales dentro de la visión por computadora, uno de ellos es el reconocimiento de objetos, que es la actividad principal de esta rama y va de la mano con la identificación y clasificación de objetos. Para poder llevar a cabo tareas específicas de esta rama es necesario primero enseñar a la computadora a identificar objetos con ayuda de diferentes tipos de redes neuronales o algoritmos específicos para su aplicación. Muchas bibliotecas de uso libre cuentan con modelos de diferentes tipos, algunos de ellos preentrenados para poner en práctica su funcionamiento; con otros, se tiene que manipular algunos de sus parámetros para adecuarlos a las necesidades, también existen métodos y comandos específicos para que se pueda crear un modelo particular para el proyecto, programándolos con características especiales, analizando su rendimiento dependiendo de la computadora y también de la cantidad de datos a entrenar.

Bibliotecas dedicadas a la visión por computadora como *OpenCV* [35] cuentan con un gran número de algoritmos para esta aplicación, de diferentes niveles desde básicos y de fácil manejo, hasta más complejos y dedicados específicamente al desarrollo de la investigación. Los algoritmos *LBPH, Heigenfaces y Fisherfaces* son de los más útiles y con mejores resultados dentro del *face analytic*, con algunas diferencias entre sí, pero cada uno con su toque de funcionamiento particular, ideales para poner a prueba y dependiendo de las necesidades a trabajar con un algoritmo de estas características. Dentro de esta biblioteca también se cuenta con muchos conjuntos de datos para alimentar y crear tus propios modelos de entrenamiento, o bien, con modelos preentrenados dónde ya solo se necesita cargarlos en los Script y adecuar el algoritmo para la ejecución correcta. Modelos tipo cascada como todos los *HaarCascade (FrontalFace, Fullbody, Eye, ProfileFace, Smile, EyeGlasses)* [36] son algunos de los que se pueden ocupar para no partir desde cero.

2.3. El proyecto propuesto

Se plantea un sistema de control de accesos utilizando identificación de rostros, este sistema funciona a través de 5 módulos principales que realizan las funciones de identificación de rostros, registrar a los usuarios en una base de datos, entrenar el algoritmo con los datos adquiridos, hacer un análisis del rostro del usuario en tiempo real y mostrar una respuesta dependiendo de los resultados del análisis en el circuito de operación. A grandes rasgos, este es el resumen de operación del proyecto y se puede identificar que la mayoría de las tareas requieren inteligencia artificial, pero ¿qué rama de la *IA* se va a utilizar?

Con este sistema de control de acceso con identificación facial no solo se abarca una rama, si no que se emplearan 3, todas van ligadas y trabajan en conjunto: visión por computadora, aprendizaje automático y robótica. Para realizar el proceso de reconocimiento e identificación es necesario la visión por computadora ya que se tendrá un usuario frente a una *webcam* intentando hacer el registro y autenticación, lo que hará la computadora será identificar a través del vídeo en *streaming* los posibles rostros humanos y con ello realizar comparaciones con los datos de entrenamiento. El registro de cada usuario se hace de manera automática almacenando los rostros del usuario en carpetas personales, dándole una etiqueta en el nombre introducido por el usuario, esta información va de la mano con el proceso de entrenamiento que va a ligar la información de cada usuario con su tipo de rostro y su nombre, realizando un aprendizaje automático no supervisado por medio de clasificadores en cascada con arboles de decisión. A pesar de que este aprendizaje se logra con datos no etiquetados como son los rostros a la hora de hacer el registro, ya que el sistema en ese momento aún no es capaz de suprimir rostros vecinos, se interpreta que es no supervisado porque en el momento de clasificar los rostros en las carpetas de cada usuario se le da una identidad a cada imagen almacenada con una etiqueta única, lo cual, después de realizar este proceso de clasificación se convierte en un sistema que aprende por sí solo, sin necesidad de alguna interacción humana. Por último, la sección de la robótica se desarrolla en el módulo del circuito que maneja el acceso ya que se quiere controlar

un sistema de entrada y salida de personas, al implementar un sistema físico con motores, compuertas y sensores que toma decisiones a partir de algoritmos de *IA*.

Capítulo 3

Diseño metodológico y procedimiento experimental

El proyecto se desarrolló mediante un *top-down* con módulos independientes derivados de un módulo maestro como programa principal. Los módulos son: cuatro independientes **(registro facial, entrenamiento, identificación facial y sistema físico)** y un módulo maestro **interfaz gráfica**.

El primer paso fue desarrollar el diagrama de bloques general del sistema (figura 3.1). Para poder identificar qué tareas son necesarias cubrir, así como los procesos más importantes dentro de la unidad de procesamiento, ¿dónde se tiene interacción con la unidad de procesamiento? ¿cuáles son las tareas que ejecutará la computadora?.

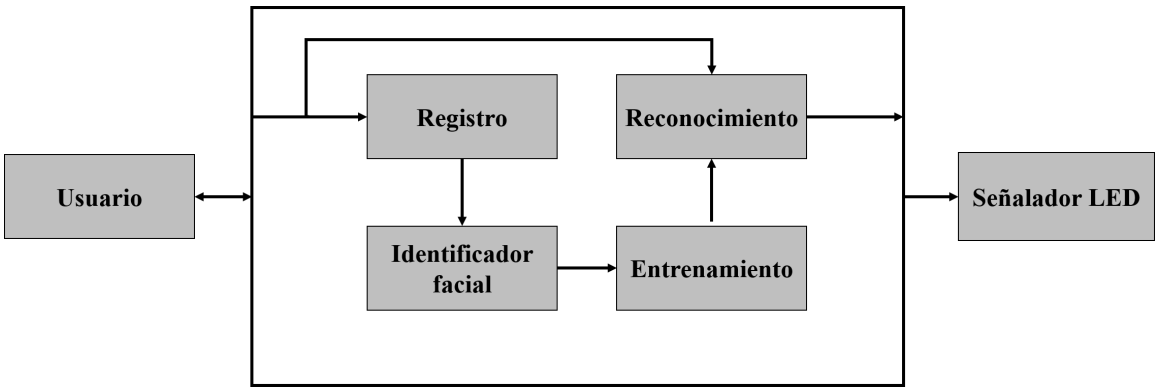


Figura 3.1: Diagrama del funcionamiento general. [37]

En segundo lugar se desarrolló un diagrama de flujo para los módulos

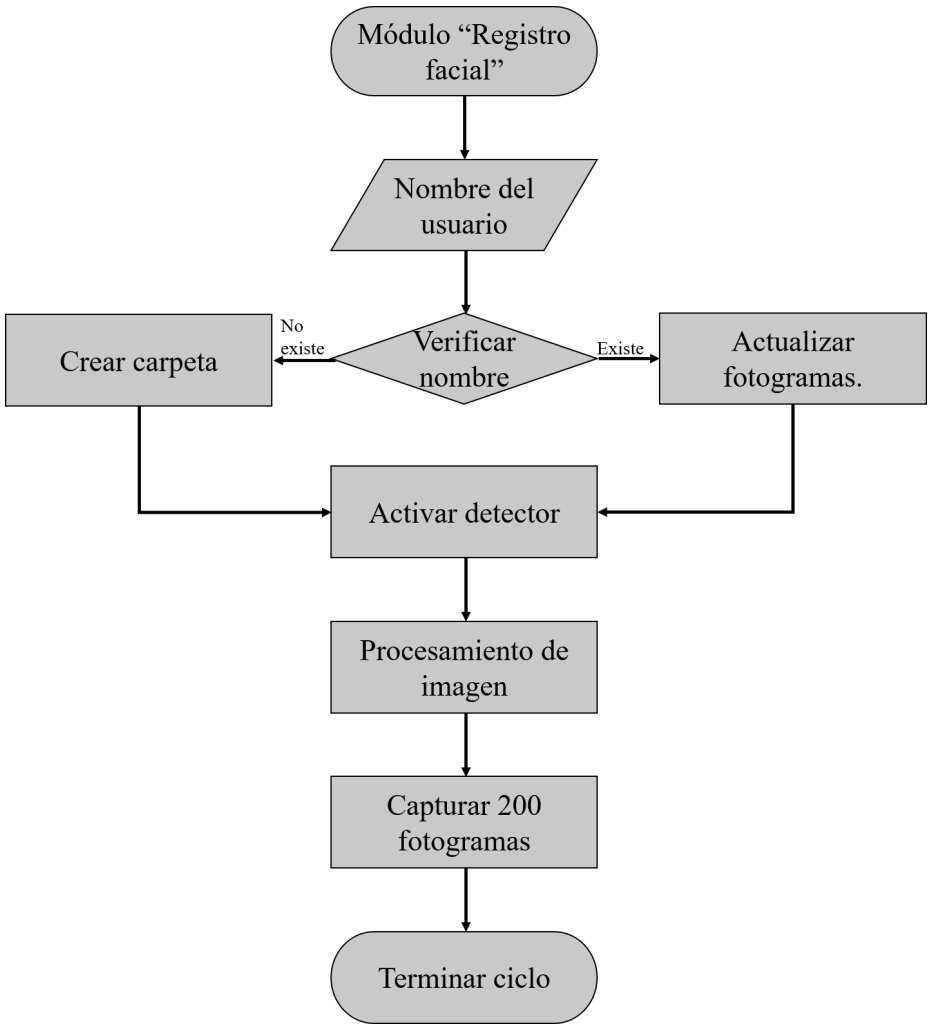


Figura 3.2: Diagrama de flujo del funcionamiento del módulo individual "Registro facial".

[38]

independientes: registro facial, entrenamiento e identificación facial. (Figuras 3.2, 3.3 y 3.4).

El desarrollo de programación de estos tres módulos fue simultaneo. Se tomó la decisión de este procedimiento debido a que era necesario observar el comportamiento del algoritmo, para que entre prueba y error se modificaran los parámetros y realizar un ajuste en el código. Esta etapa fue la más lenta, pues sé dejaron los algoritmos en su máxima capacidad de funcionamiento. Además, fue el punto clave para modificar y ajustar los parámetros. Asimismo, los siguientes pasos consisten en hacer una interfaz fácil de manipular e implementarlo al funcionamiento físico.

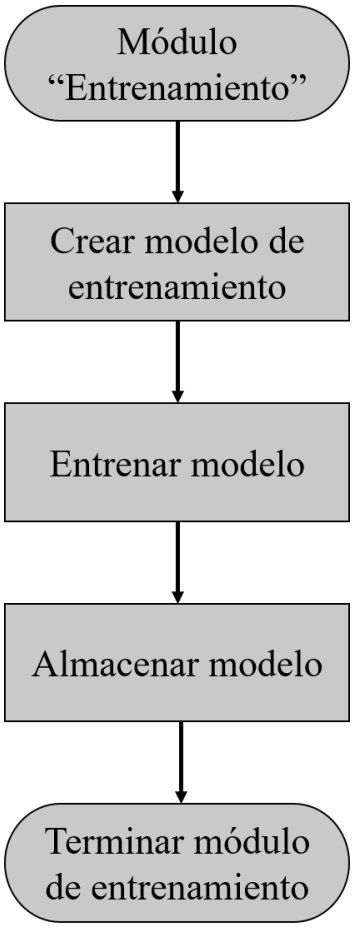


Figura 3.3: Diagrama de flujo del funcionamiento del módulo individual “Entrenamiento”.

[39]

El trabajo de programación de estos tres *scripts* fue directamente desarrollado en la consola de *Visual Studio Code*, con una cantidad de pruebas que más adelante en el capítulo 4 se analizaran detalladamente.

Este primer desarrollo era primitivo, los tres módulos aún no estaban ligados entre sí, simplemente se mandaban a llamar a través de las instrucciones en consola, desplazándose entre *scripts* para que se corrieran por separado.

Como siguiente paso, se trabajó la unión de los 3 módulos independientes con el módulo maestro “interfaz gráfica”. Se creó el diagrama de flujo detallado del funcionamiento general del sistema para ordenar las funciones de los botones, ya que dentro de estos se integrarán los diferentes submódulos. (Figura 3.5).

Una vez que se obtuvo el correcto funcionamiento del módulo maestro se desarrolló

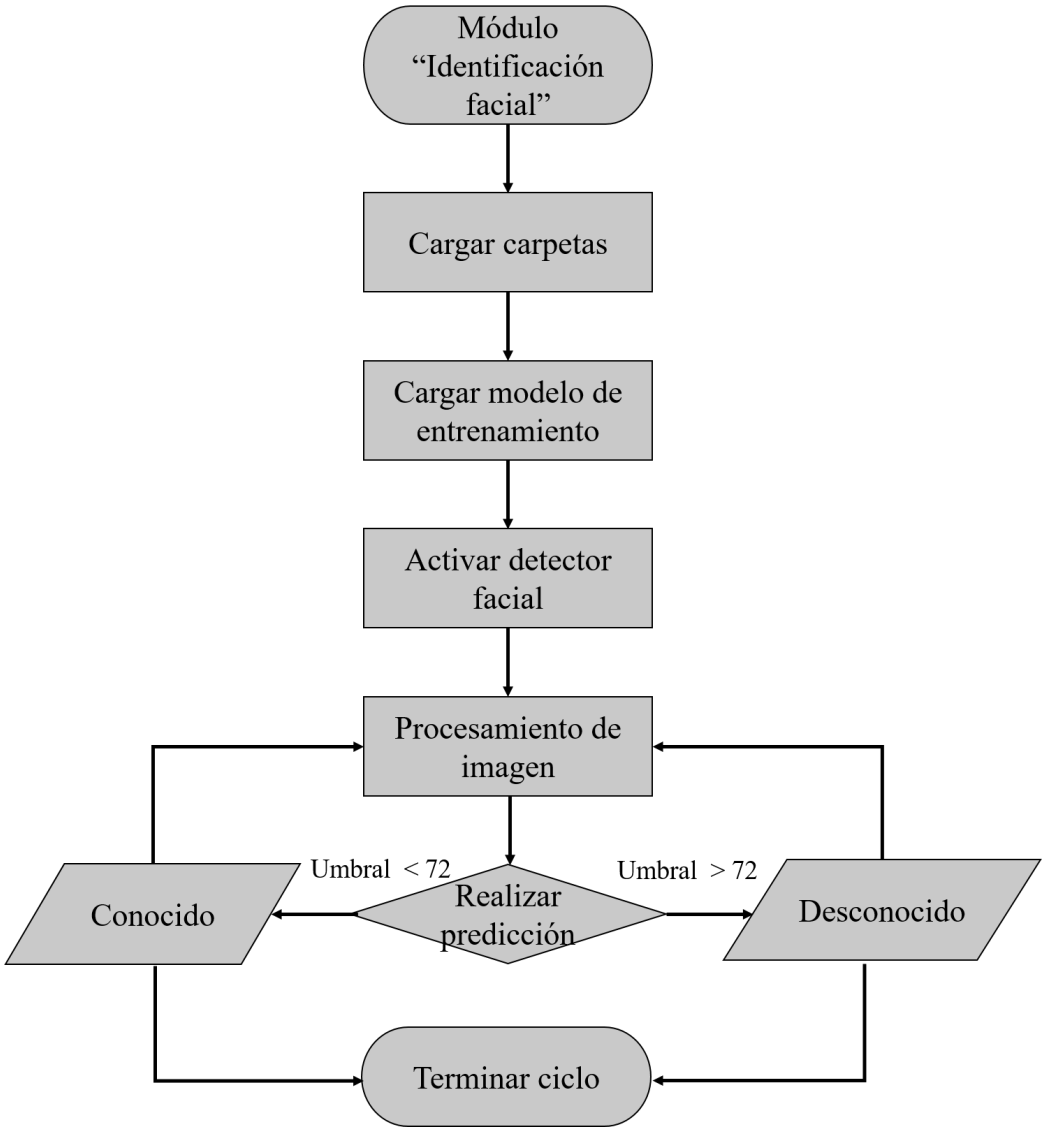


Figura 3.4: Diagrama de flujo del funcionamiento del módulo individual "Identificación facial".

[40]

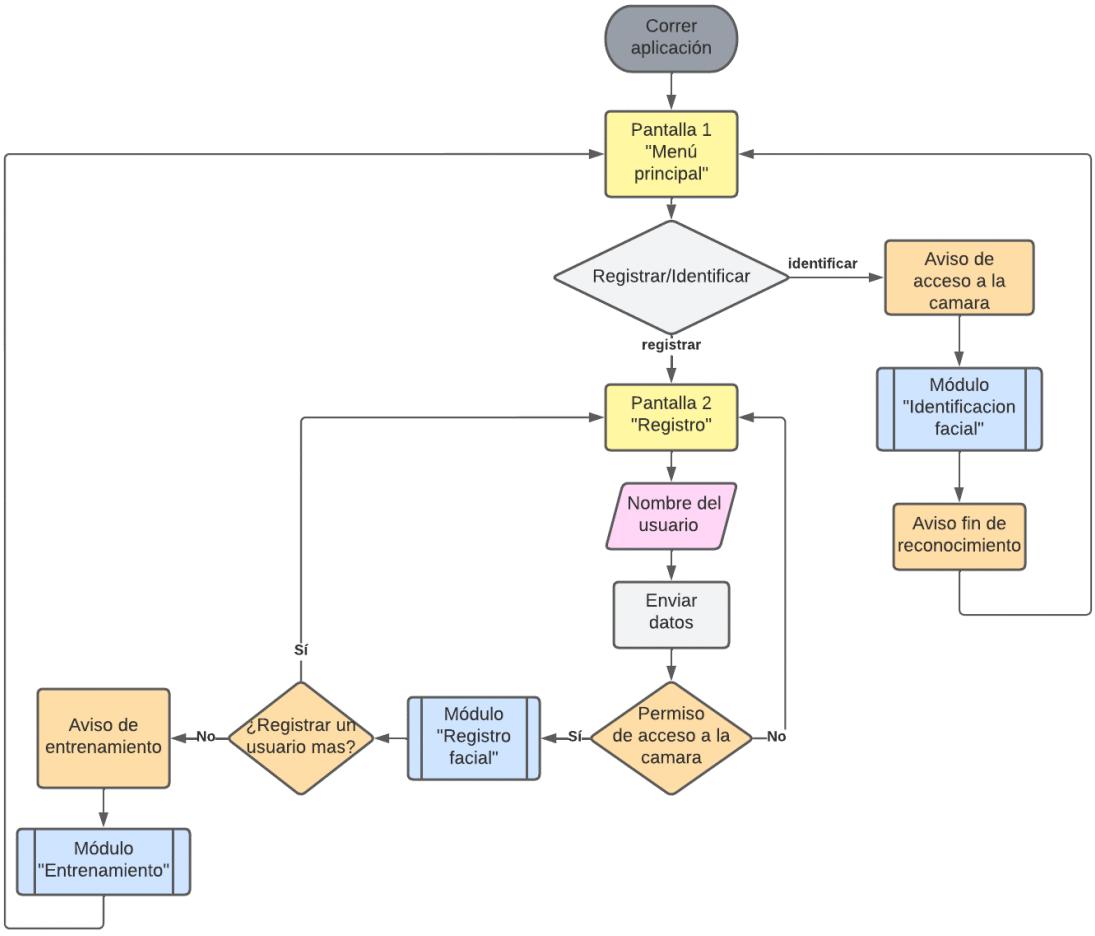


Figura 3.5: Diagrama de flujo del funcionamiento general del sistema. [41]

el ultimo módulo independiente llamado “sistema físico”, con ayuda del diagrama de bloques de la figura 3.6. analizando primero el objetivo de dar y denegar el acceso a los usuarios. Se desarrolló el circuito prototipo y se agregaron las bibliotecas necesarias en el código para la conexión entre el *arduino* y *python*.

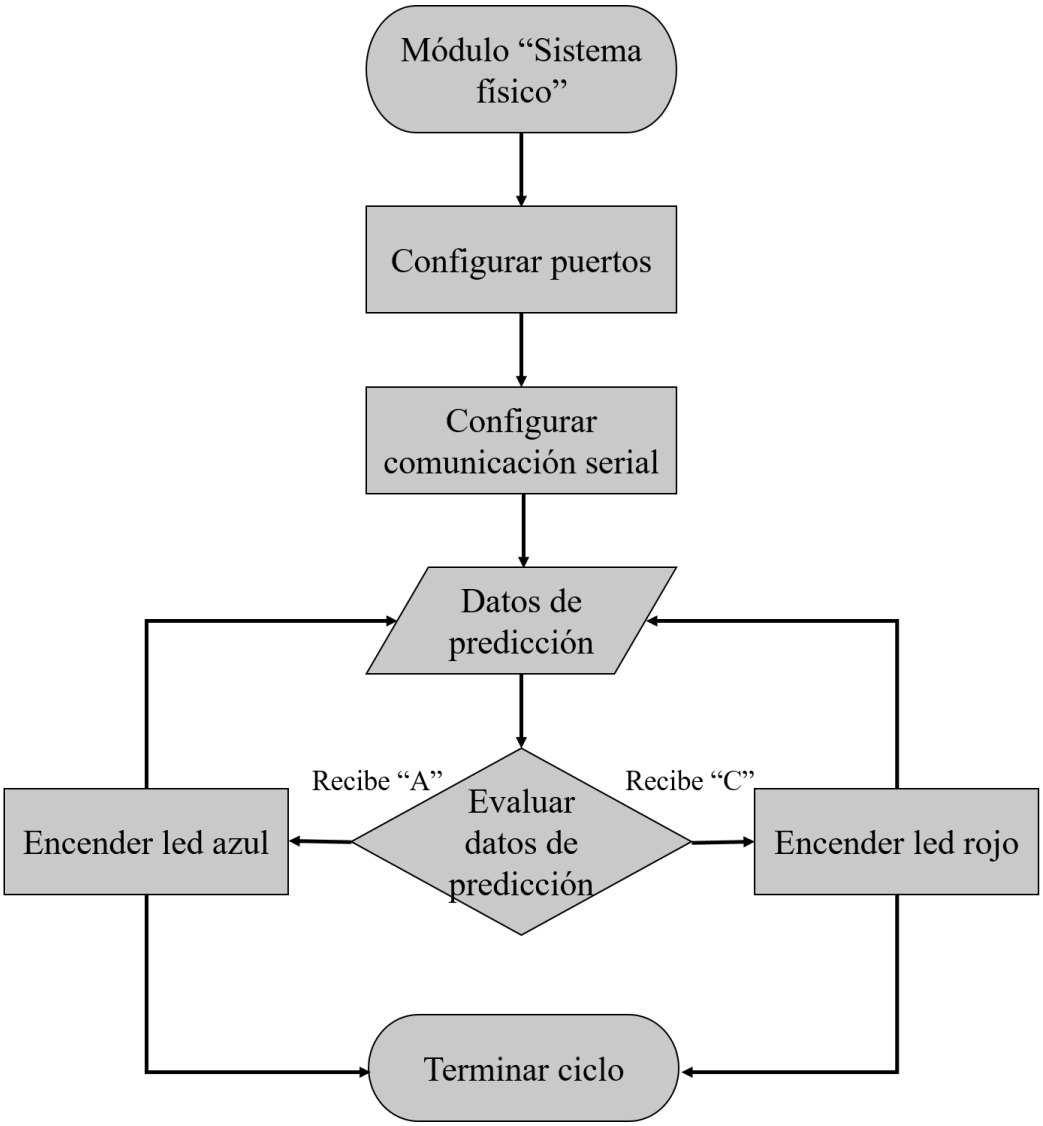


Figura 3.6: Diagrama de flujo del funcionamiento del módulo individual "Sistema físico". [42]

3.1. Primer módulo "Registro facial"

Este primer módulo tiene la función de crear un registro del usuario, primeramente solicita el nombre con el que se quiere ingresar a la persona. El siguiente paso es verificar si el nombre de usuario introducido existe dentro de la base de datos. Una vez verificado hay dos caminos. Si es un nombre nuevo, el programa tiene la instrucción de crear una carpeta con el nombre del usuario dónde se almacenarán los fotogramas. Si el nombre de usuario ya existe, el programa actualizará los fotogramas dentro de la carpeta. En la figura 3.7 se observan las instrucciones para automatizar este procedimiento.

```
#------------Crear carpeta-----------

nombre=entryNombre.get()

rutaBase =
'C:/Users/Pavilion/Documents/Ivan/Proyecto SS/Base_de_datos_LBP
H'
carpetaPersonal = rutaBase + '/' + nombre

if not os.path.exists(carpetaPersonal):
    print('Carpeta creada: ',carpetaPersonal)
    os.makedirs(carpetaPersonal)
```

Figura 3.7: Instrucciones para la creación de carpetas automáticas. [43]

Después de la creación de la carpeta con el nombre del usuario, el *script* se encarga de hacer una visualización a través de la cámara configurada dentro del código, en este caso se esta utilizando una laptop marca Lenovo modelo Thinkpad T450 con cámara web HD de 720p. La visualización que se obtendrá es la función de una cámara de video que sea capaz de distinguir entre todos los objetos mostrados en el fotograma los rostros de las personas. Para ello se decide ocupar la biblioteca de *OPEN CV* [44].

Esta biblioteca es una herramienta de código abierto que proporciona ayuda con el manejo de imágenes, procesamiento de video, fotografía computacional entre otras cosas.

Existe una diferencia entre reconocimiento facial y detección facial, en este proyecto es necesario llevar a cabo estas dos funciones, la detección facial en el primer módulo y el reconocimiento facial en el tercer módulo. Esta primera parte se apoya con el algoritmo *haarcascade-frontalface* [36] [45]. Este algoritmo se encarga de la detección de objetos mediante clasificadores en cascada, el objetivo principal de este método es identificar niveles de intensidad en los pixeles de la imagen y su eficacia esta dada por entrenar con un alto conjunto de imágenes positivas (imágenes con caras) e imágenes negativas (imágenes sin caras).

Dentro de la figura 3.8 se observa cómo operan estos clasificadores, se muestran las formas en las que se van superponiendo los filtros para una detección entre 1 y 0, 1 si se detecta textura (parte negra), 0 si no se detecta textura (parte blanca).

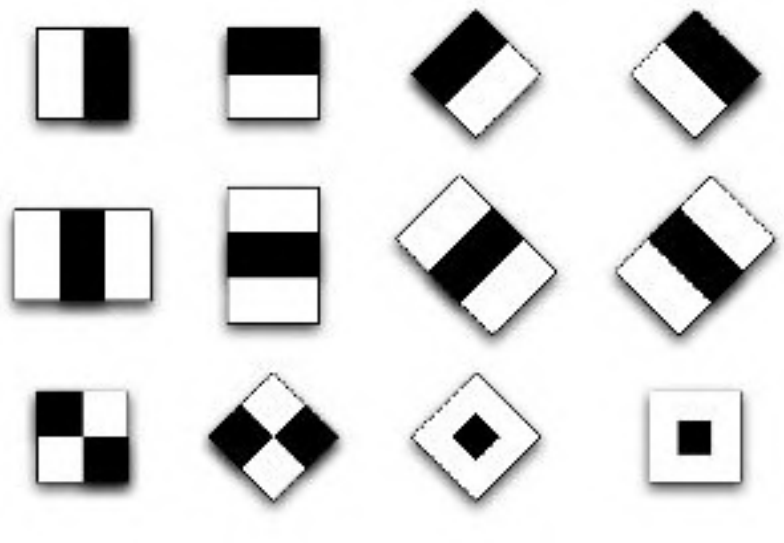


Figura 3.8: Ejemplo de clasificadores tipo Haar. [46]

Para poder realizar bien la detección de rostros es necesario darles un trato especial a las imágenes, en este caso a los fotogramas que se verán en el *frame*. Se tienen que hacer unas pequeñas modificaciones a la visualización, estas son: cambiarla a escala de grises y redimensionar la ventana.

Este algoritmo preentrenado cuenta con esas imágenes positivas y negativas, ya

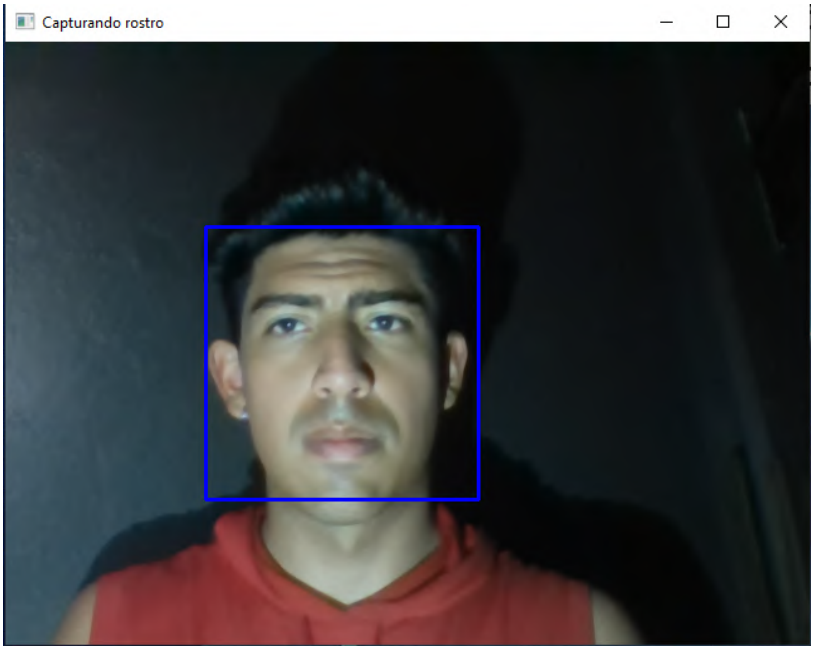


Figura 3.9: Captura de fotografías. [47]

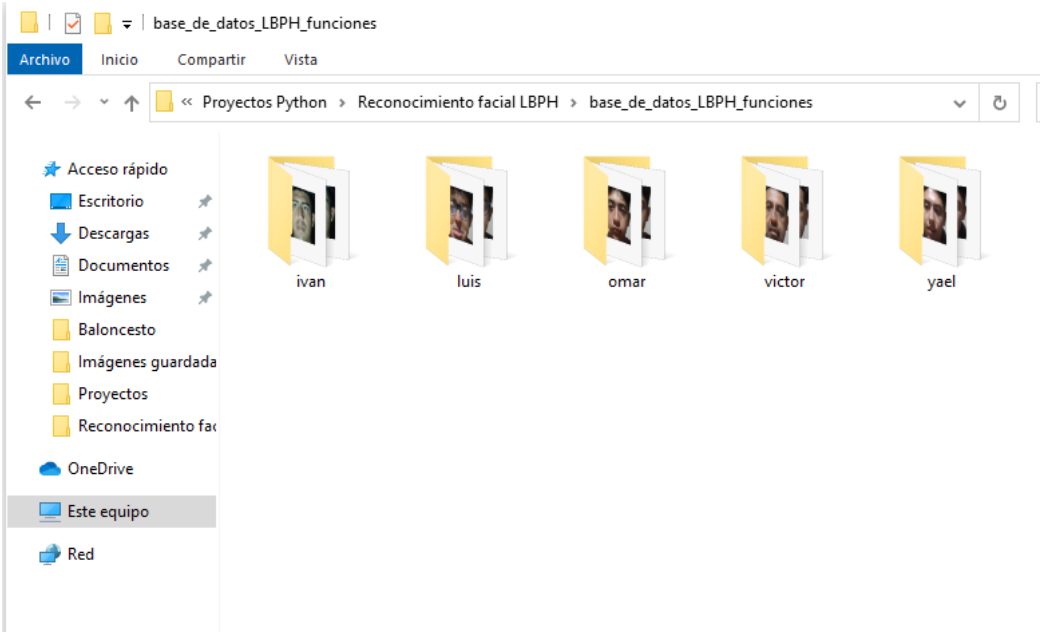


Figura 3.10: Carpetas creadas. [48]

no es necesario realizar este procedimiento por lo cual se pasara a registrar a los usuarios en la base de datos creada a partir de las carpetas. Para ello se tomaran 200 fotografías del usuario como se muestra en la figura 3.9 y se almacenaran en su carpeta correspondiente como en la figura 3.10. La toma fotogramas fue puesta a prueba dentro del capitulo 4 "Pruebas y resultados" con lo cual se concluye en tomar 200 fotogramas en un tiempo de 16 segundos.

Estas imágenes deben de tener el mismo tamaño, entonces también se le aplica un redimensionamiento a 150x150, se crea una copia de lo que está dentro del recuadro delimitador y ese fotograma se almacena en la carpeta, también se crea un contador para darle un nombre de identificación a cada fotograma almacenado (figura 3.12) y esto se realiza con las líneas del código de la figura 3.11.

```
while True:
    (ret,ventana) = camara.read()

    if ret==False: break
    ventana = cv2.flip(ventana,1)
    ventana =  imutils.resize(ventana, width=640)
    gray= cv2.cvtColor(ventana, cv2.COLOR_BGR2GRAY)
    auximagen = ventana.copy()

    caras = modeloClasificador.detectMultiScale(gray, 1.1, 15)


    for(x,y,w,h) in caras:
        cv2.rectangle(ventana, (x,y),(x+w, y+h), (255,0,0),2)
        rostro = auximagen[y:y+h,x:x+w]
        rostro = cv2.resize(rostro,(150,150),interpolation=cv2.INTER_CUBIC)
        cv2.imwrite(carpetaPersonal + '/rostro_{}.jpg'.format(count),rostro)
        count = count + 1
    cv2.imshow('Capturando rostro', ventana)

    key= cv2.waitKey(10)
    if key == 27 or count >=100:
        break
camara.release()
cv2.destroyAllWindows()
```

Figura 3.11: Líneas de código para la configuración de fotogramas. [49]

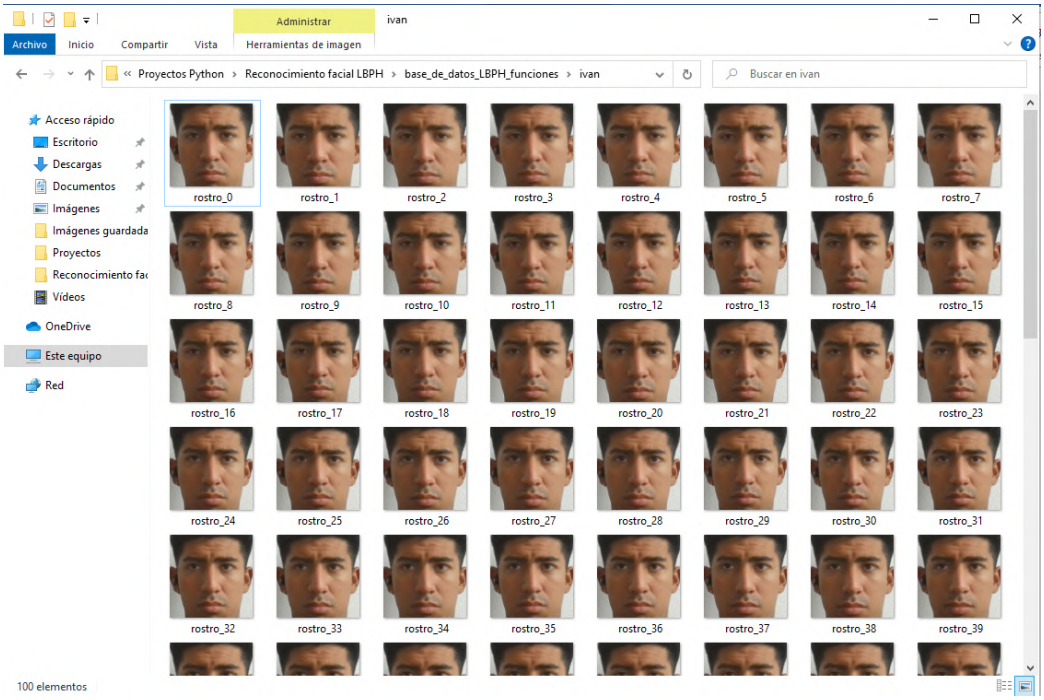


Figura 3.12: Visualización de los fotogramas almacenados con una etiqueta de identificación especial. [50]

Con esto termina la configuración de la base de datos y registro de usuario. Una vez almacenados los rostros de cada usuario queda lista la base de datos, ahora se tiene esa información disponible para poder llevarla al siguiente proceso el cual se encapsuló en el segundo módulo.

3.2. Segundo módulo "Entrenamiento"

El siguiente proceso que se llevará a cabo es el entrenamiento de la base de datos, es decir, se necesita asociar matemáticamente la información obtenida, en este caso las imágenes, con un valor único que pueda diferenciarlo. Dentro del proceso de entrenamiento se realizan varias iteraciones primero para relacionar todas las fotos de una carpeta a un usuario, una vez que el sistema puede relacionar las fotos con cada usuario se le coloca un diferenciador a cada usuario para poder distinguirlos computacionalmente. Es por eso que primero se necesitan unas líneas de código, como se muestra en la figura 3.13, para darle etiquetas especiales y el algoritmo pueda entender la relación de los usuarios con sus fotografías. Para ello se requirió el

apoyo del código del blog de *Gabriela Solano* [51].

```
rutaBase =
'C:/Users/Pavilion/Documents/Ivan/Proyecto SS/Base_de_datos_LBPH'

usRegistrados = os.listdir(rutaBase)

etiquetas = []
datosRostros = []
contador = 0

for nombreDir in usRegistrados:
    carpetaPersonal = rutaBase + '/' + nombreDir
    for fileName in os.listdir(carpetaPersonal):
        etiquetas.append(contador)
        datosRostros.append(cv2.imread(carpetaPersonal+'/'+
fileName,0))
        image = cv2.imread(carpetaPersonal+'/'+fileName,0)
    contador = contador + 1
```

Figura 3.13: Líneas de código para la configuración de las etiquetas identificadoras. [52]

Como se puede observar, primero es necesario indicarle al programa en donde están ubicadas las carpetas, después se relacionan las variables *etiquetas, datosRostros y contador* con formato de lista para almacenar las etiquetas, así como los nombres de las fotografías, esto es para registrar con un identificador único a cada usuario y después, cuando se lleve al entrenamiento, éste sea capaz de asociarlas sin ningún problema y sin ningún error.

Para el proceso de entrenamiento, dentro del aprendizaje automático o machine learning existe un comando llamado **train**, que es capaz de realizar un proceso de entrenamiento automático sin necesidad de hacer los cálculos uno por uno, es decir, realiza un conjunto de pasos para que el modelo pueda entender y clasificar imágenes, en este caso los rostros de los usuarios asociarlos con su nombre correspondiente. Como

el entrenamiento es enfocado a las imágenes, es necesario extraerlo de la biblioteca Open CV [44]. Se puede ver la sintaxis del comando en el siguiente código de la figura 3.14.

```
### comienzo del entrenamiento del modelo

reconocedor_facial = cv2.face.LBPHFaceRecognizer_create()

reconocedor_facial.train(datosRostros, np.array(etiquetas))

reconocedor_facial.write(
'C:/Users/Pavilion/Documents/Ivan/Proyecto SS/entrenamiento_fac
ial_LBPH2.xml'
)
```

Figura 3.14: Líneas de código para el entrenamiento. [53]

Aunque es breve el código para realizar el entrenamiento, se destacan tres lineas, debido a que son el fundamento para crear el modelo de entrenamiento. Lo primero que se debe resaltar es la creación de una variable donde se almacenará la creación del modelo de entrenamiento. Lo segundo a resaltar es la siguiente instrucción del código, ahí es donde se realiza el proceso de entrenamiento con el comando *train*, asociando el modelo creado junto con la variable donde se almacenaron las etiquetas de los usuarios. Al final de este bloque, en la última línea de código, es necesario almacenar el modelo de entrenamiento, esto para que no se tenga que entrenar cada vez que se quiera identificar algún rostro. Se carga el modelo de entrenamiento con nombre ‘‘`entrenamiento_facial_LBPH.xml`’’ con formato `.xml` a la variable creada que se reservó para almacenar el modelo.

Un archivo con la extensión **.xml** (Extensible Markup Language) consiste en un formato que se utiliza para almacenar y organizar datos de manera estructurada y semiestructurada que utiliza una serie de etiquetas personalizadas con la finalidad de

describir tanto la estructura como otras características del documento.

3.3. Tercer módulo "Identificación facial"

En el tercer módulo se encuentra la identificación facial y este será el encargado de reconocer entre un usuario y otro. El proceso se lleva a cabo de igual manera a través de la cámara frontal de la laptop abriendo una ventana y mostrando los rostros que el sistema puede reconocer, estos rostros son los que están dados de alta en el *dataset* o la base de datos inicial y de igual forma son los que ya pasaron por el proceso de entrenamiento. Este módulo también tiene la capacidad de colocar un mensaje en pantalla con el nombre del usuario reconocido o si es una persona desconocida. Pero estos procesos ¿cómo es que se ejecutan? Pues la biblioteca de *Open CV* sigue siendo la mejor opción por su cantidad de módulos ya que sigue siendo procesamiento de imágenes y cuenta con muchas herramientas adecuadas para este proyecto.

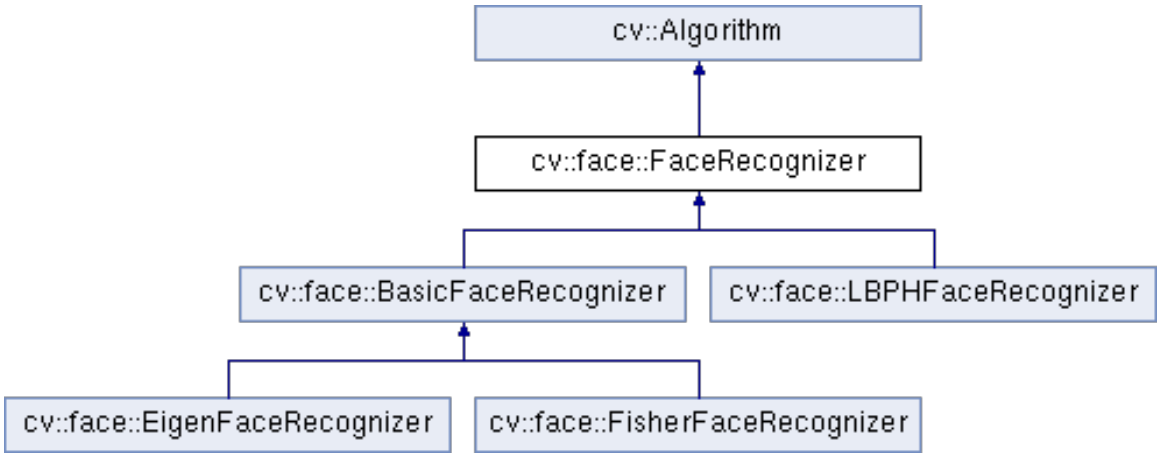


Figura 3.15: Diagrama de clases de detección facial dentro del módulo de FaceRecognizer. [54]

Dentro de su repertorio de herramientas cuenta con un módulo llamado *Faces Analysis* en el que se puede encontrar una clase llamada *LBPHFaceRecognizer* (Local Binary Paterns Histrograms) [55]. Para escoger esta clase se realizaron pruebas y se observó que es más eficiente el *LBPH* a comparación del *Fisher faces* o el *Heigen faces* que forman parte del módulo básico del *Face Analysis* (como se puede observar en la figura 3.15), por lo cual se decidió ocupar la clase *LBPH* por tener un desarrollo más

avanzado y presentar una mejor respuesta al bajar la tasa de error. Se realizaron las mismas comparaciones con la identificación facial para observar el comportamiento de este modelo *Fisher faces* dando como resultado una confusión entre el mismo usuario, aumentando la cantidad de tiempo que se muestra una persona desconocida con un usuario que si estaba registrado y almacenado correctamente en la base de datos. Por este motivo se decide continuar con el algoritmo avanzado LBPH.

Con el procedimiento del algoritmo *LBPH*, es posible que la computadora pueda entender de qué usuario se trata y quien o quienes son los que están al otro lado de la pantalla por medio de datos probabilísticos. El sistema da un valor numérico y este es el que consulta en el modelo de entrenamiento para poder comparar y dar una respuesta acertada. Cabe aclarar que el sistema trabaja en un rango de operación para reducir errores de predicción. Más adelante, en la sección 4 de análisis de resultados, se tratarán con detalle estos parámetros y rangos de operación para que el sistema sea lo más eficiente posible.

Al iniciar con este módulo se necesita indicarle al programa la dirección de memoria de donde se tomarán los nombres de los usuarios para poder mostrarlos en pantalla y se modifican para tenerlos disponibles en una lista. Dentro de la línea 1 y 2 del código de la figura 3.16 se observa la ruta de la base de datos y el orden para la lista con los nombres de los usuarios.

En las siguientes líneas del código, se crea una nueva variable igual que en la del modelo de entrenamiento, ahora de nombre `reconocedor_facial`, donde se crea el modelo de predicción LBPH. Una línea abajo, es llamado el modelo de entrenamiento creado previamente con la extensión `.xml`, por último, se configura la cámara de la laptop y se carga el modelo de detección de rostros para saber que es un rostro y que no lo es, porque esto no lo sabe el sistema y es necesario volver a darle la instrucción para que realice las predicciones de acuerdo al resultado de ese modelo de detección, es decir, con solo las imágenes positivas y no con falsos positivos.

Después es necesario configurar algunos parámetros para la lectura correcta de la imagen dentro del *frame*, estas modificaciones se hacen en el código de la figura 3.17 y son muy parecidas a la configuración del primer módulo cuando se crea el algoritmo

```
rutaBase =
'C:/Users/Pavilion/Documents/Ivan/Proyecto SS/Base_de_datos_LBPH'
carpetas_us= os.listdir(rutaBase)

reconocedor_facial= cv2.face.LBPHFaceRecognizer_create()

reconocedor_facial.read('entrenamiento_facial_LBPH.xml')

camara= cv2.VideoCapture(0)

modeloClasificador = cv2.CascadeClassifier(cv2.data.haarcascades+
'haarcascade_frontalface_default.xml')
```

Figura 3.16: Preparación para la identificación facial. [56]

para tomar las fotografías automáticamente.

En esta sección del código, representada en la figura 3.17, se describe la configuración del *frame*. Se destaca la línea número 11, donde se almacenará el resultado de la operación matemática de predicción. Esta operación implica la comparación entre el modelo de entrenamiento y el análisis en tiempo real de la cámara.

La información se guarda dentro de la variable **result** y se le ejecuta el comando **predict**, dicho comando realiza la comparación entre la variable **rostro** (que es la imagen del rostro en el *frame* en tiempo real) y el **reconocedor_facial** que es el modelo de entrenamiento.

El valor de esta predicción es almacenado dentro de la variable **result** con formato de tupla, es decir, este dato contiene dos valores numéricos, el primero es un identificador de posición que utiliza el comando **predict** para ubicar el resultado de su operación; el segundo valor numérico es el resultado de la operación interna entre comparar el rostro en tiempo real y los datos del modelo preentrenado, este segundo dato maneja un rango de valores para estandarizar los resultados, entre mas

```
while True:
    (ret,ventana) = camara.read()
    if ret==False: break
    ventana = cv2.flip(ventana,1)
    gray= cv2.cvtColor(ventana, cv2.COLOR_BGR2GRAY)
    auximagen = gray.copy()
    caras = modeloClasificador.detectMultiScale(gray,1.1,15)
    for(x,y,w,h) in caras:
        rostro = auximagen[y:y+h,x:x+w]
        rostro = cv2.resize(rostro,(150,150),interpolation=
cv2.INTER_CUBIC)
        result = reconocedor_facial.predict(rostro)
        cv2.putText(ventana,'{}'.format(result),(x,y-3),1,1.3,(
255,0,0),1,cv2.LINE_AA)
```

Figura 3.17: Configuración del frame para la identificación. [57]

cerca de las 0 unidades esté el resultado de la operación, hay más coincidencias en la comparación del modelo preentrenado y el rostro; en cambio, si el resultado de la operación se aleja de las 0 unidades, existen menos coincidencias.

Con el resultado que arroja la predicción se toman las decisiones para la condición de elección y es aquí donde se escriben los parámetros correctos para que el sistema pueda decidir si el usuario es uno de los que están dentro de la base de datos o es un usuario desconocido (figura 3.18).

Después de un análisis experimental detallado con gráficas y varias sesiones de prueba que se explica dentro del capitulo 4.3 (Pruebas y resultados - Tercer análisis, "Result", se llegó a la conclusión que el índice adecuado es de 72 unidades. Si el valor dentro de la variable **result** es menor a 72 unidades es un usuario conocido, entonces se coloca dentro del recuadro delimitador el nombre del usuario (figura 3.19), de lo contrario, si el índice es mayor a 72 unidades, el algoritmo anunciara que es un usuario desconocido (figura 3.20).

```
if result[1] < 72:
    cv2.putText(ventana,'{}'.format(carpetas_us[result[0]]),(x,y-25),
2,1.1,(0,255,0),1,cv2.LINE_AA)
    cv2.rectangle(ventana, (x,y),(x+w,y+h),(0,255,0),2)
    ser.write(b'A')
else:
    cv2.putText(ventana,'Desconocido',(x,y-20),2,0.8,(0,0,255),1
,cv2.LINE_AA)
    cv2.rectangle(ventana, (x,y),(x+w,y+h),(0,0,255),2)
    ser.write(b'C')
```

Figura 3.18: Parametros de decisión conocido/desconocido. [58]

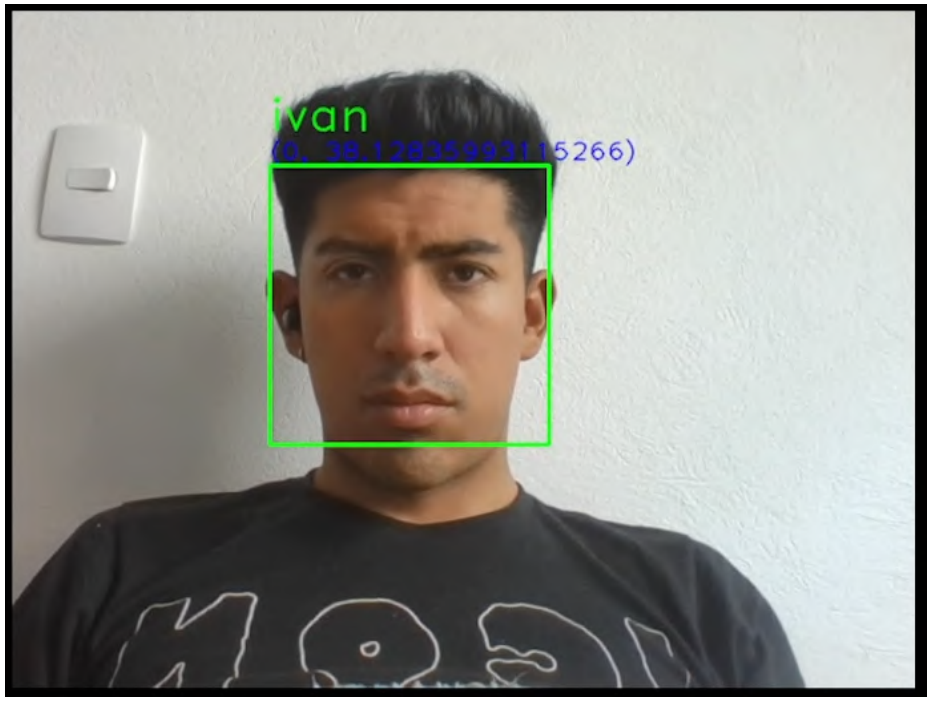


Figura 3.19: Identificación persona conocida. [59]

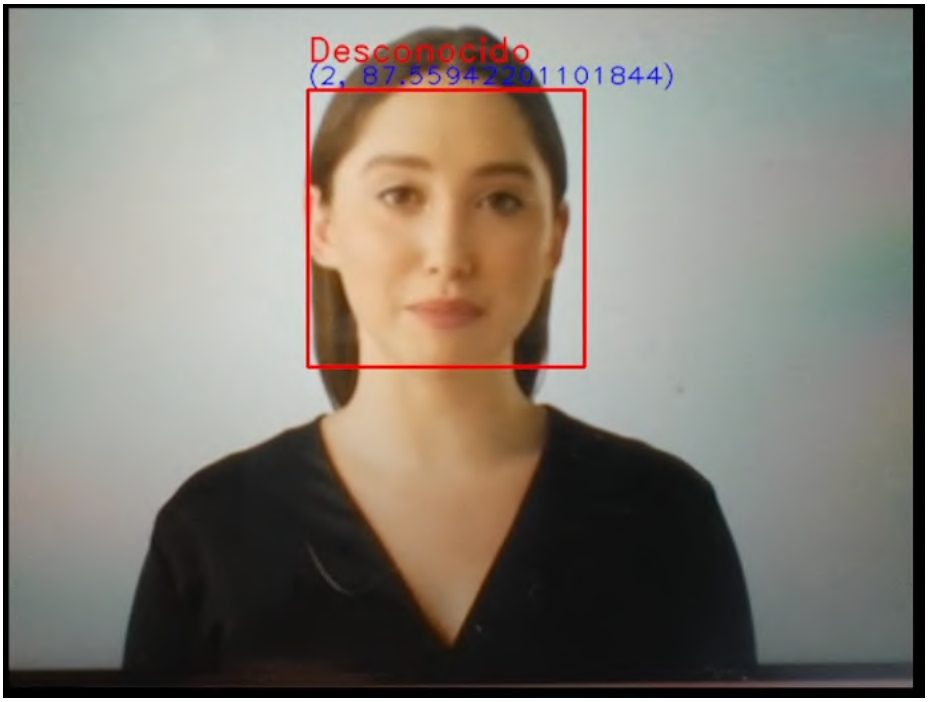


Figura 3.20: Identificación persona desconocida. [60]

3.4. Cuarto modulo "Interfaz gráfica"

Siempre es más fácil trabajar en un ambiente agradable, más estético e interactivo que en una simple ventana de comandos. Es por eso que este proyecto cuenta con la sección de interfaz gráfica. Este módulo se fabricó con una biblioteca muy amigable llamada *TKinter (Tk)*.

Tk es una herramienta para desarrollar aplicaciones de escritorio multiplataforma, esto es, aplicaciones nativas con una interfaz gráfica para sistemas operativos *Windows*, *Linux*, *Mac* y otros. Es una de las bibliotecas más sencillas de utilizar y se decidió emplear debido a la falta de experiencia en la programación *frontend*, por cuestiones técnicas también es necesario reducir el uso de procesamiento dedicado a la interfaz y otorgar una mayor parte para los procesos de análisis de imágenes [61].

Para la interfaz gráfica se programó una pantalla principal (figura 3.21), dónde se presenta un menú simple con dos opciones, la de registrar un usuario y la de realizar la identificación del usuario. Dependiendo las opciones elegidas, dirige a una nueva ventana, ya sea la del registro (figura 3.22) o algunas ventanas emergentes de avisos

e información como muestra la figura 3.23.

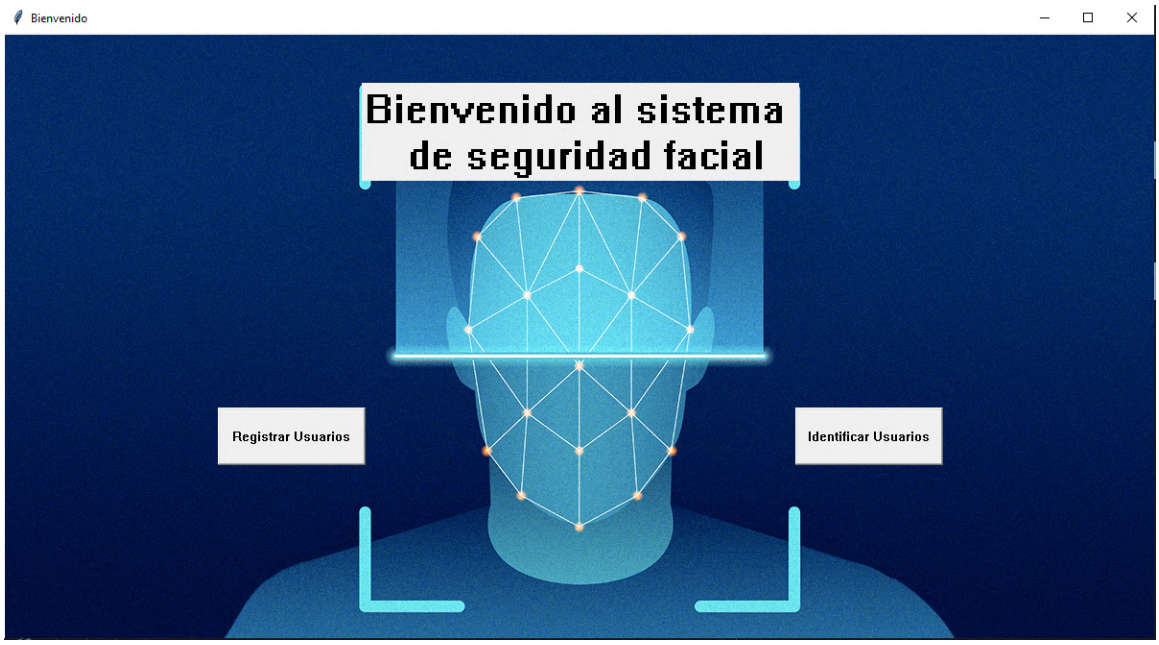


Figura 3.21: Ventana principal. [62]

La programación de las ventanas se desarrolló para funcionar con botones, lo cual lo hace interactivo. Para ello es necesario encapsular funciones a cada uno de estos.

Figura 3.22: Ventana de registro de usuarios. [63]

El desarrollo de la programación para la interfaz gráfica abarco funciones importantes, puesto que fueron implementados los módulos independientes, de los

cuales se pueden destacar algunas de las funciones que se tomaron en cuenta con ayuda del diagrama de flujo de la figura 3.5 presentado al principio de este capitulo:

- Pantallas de menú con entradas de datos.
- Botones para activar los módulos independientes (Registro facial, entrenamiento e identificación facial).
- Ventanas emergentes con avisos solicitando autorización al usuario.

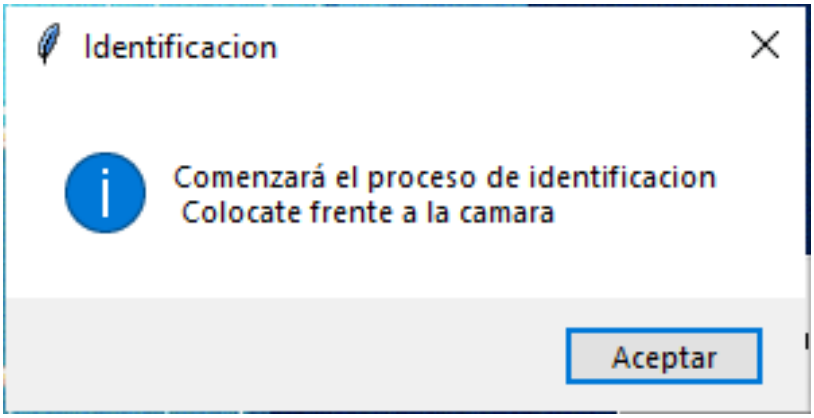


Figura 3.23: Ventana emergente de aviso al proceso de identificación facial. [64]

3.5. Quinto módulo "Sistema físico"

Este proyecto se implementó un sistema físico con una placa de desarrollo *Arduino UNO*, con el objetivo de controlar una serie de motores simulando puertas automáticas de edificios o laboratorios. Como prototipo, se realiza una implementación de LED´s para una muestra rápida. El circuito se fabrico con dos leds de 5mm color rojo y azul con una resistencia de 100 Ohms.

Gracias a este sistema se puede decidir, de forma inteligente, si un usuario tiene acceso a esa área o no. Tratando de sustituir las placass magnéticas o credenciales con un acceso supervisado, se desarrolló esta idea para automatizar aún más ese sistema.

Se desarrolló en la placa *Arduino UNO* porque el control de las puertas no requiere de mucha capacidad de procesamiento, sumando a ellos la simplicidad en

la programación en el lenguaje y el *IDE*. Esto se diseñó de manera que los cálculos los realice la computadora, todo el proceso se realizó en *Python* y dentro del algoritmo en la sección de decisión del código de la figura 3.18 se toma la lectura de manera serial para mandarla a la placa *Arduino UNO*, así, a la placa solo llegan las lecturas *booleanas* de 1 o 0.

```
import serial
import time
#--Configurando la comunicacion serial con el arduino-
ser=serial.Serial('COM3',9600,timeout=1)
time.sleep(2)
ser.write(b'A')
ser.write(b'C')
ser.close()
```

Figura 3.24: Líneas de código para configurar la comunicación serial Python-Arduino. [65]

Dentro de las líneas mostradas en el código de la figura 3.24 con ayuda de la documentación de la biblioteca *pySerial* [66] se desarrolla la configuración de la comunicación serial. Las primeras dos son las bibliotecas que se tiene que importar, la 4ta línea es la configuración donde se da la instrucción de comunicación serial a través del puerto `COM3`, esta información se muestra cuando se conecta la placa a la computadora junto con el parámetro de `9600`, este parámetro es una velocidad de comunicación serial adecuada para Arduino, con esta velocidad de transferencia reduce al máximo errores de comunicación, es por eso que necesario igualarlo con el mostrado en el ambiente de Arduino, en este caso a 9600 bps. Por último el `timeout` se colocará en 1. Después se asigna un *delay* para esperar a que el algoritmo funcione y no tome datos basura de arranque. Las siguientes dos líneas deben de estar incluidas

dentro de la condición del módulo anterior de reconocimiento: si es una persona conocida con índice *result* menor a 72 unidades se enviará una "A", en cambio si el índice es mayor a 72 unidades se enviará una "C". Al final se cierra la comunicación serial.

Para poder lograr una comunicación serial correcta entre *Python* y *Arduino* se necesita un emisor y un receptor, el emisor se trabajó con el bloque anterior en *Python*. Ahora se hablará de la sección del receptor mostrado en el código de la figura 3.25 y del circuito mostrado en la figura 3.26. La primera parte del código es una configuración para poder manipular los *LEDs*, la parte importante es dentro del *void loop* donde se le dan instrucciones para la lectura de la comunicación serial para poder recibir las variables "C" y "A" para después, realizar la toma de decisión con cada una de las variables, si se recibe una "C" el *LED* rojo encenderá y el azul permanecerá apagado indicando tener un usuario desconocido, al contrario de si se recibe una letra "A", el *LED* rojo se apagará y el azul permanecerá encendido indicando haber identificado un usuario conocido.

Script Circuito reconocimiento facial | Arduino IDE 2.0.1

File Edit Sketch Tools Help

Arduino Uno

Script Circuito reconocimiento facial.ino

```
1   int led_red = 10;
2   int led_blue = 11;
3   int option;
4
5   void setup() {
6
7     Serial.begin(9600);
8     pinMode(led_red, OUTPUT);
9     pinMode(led_blue, OUTPUT);
10  }
11
12  void loop() {
13    if (Serial.available() > 0) {
14      option = Serial.read();
15      Serial.println(option);
16      if (option == 'C') {
17        digitalWrite(led_red, HIGH);
18        digitalWrite(led_blue, LOW);
19      }
20      if (option == 'A') {
21        digitalWrite(led_red, LOW);
22        digitalWrite(led_blue, HIGH);
23      }
24    }
25  }
```

Figura 3.25: Script para control de LEDs. [67]

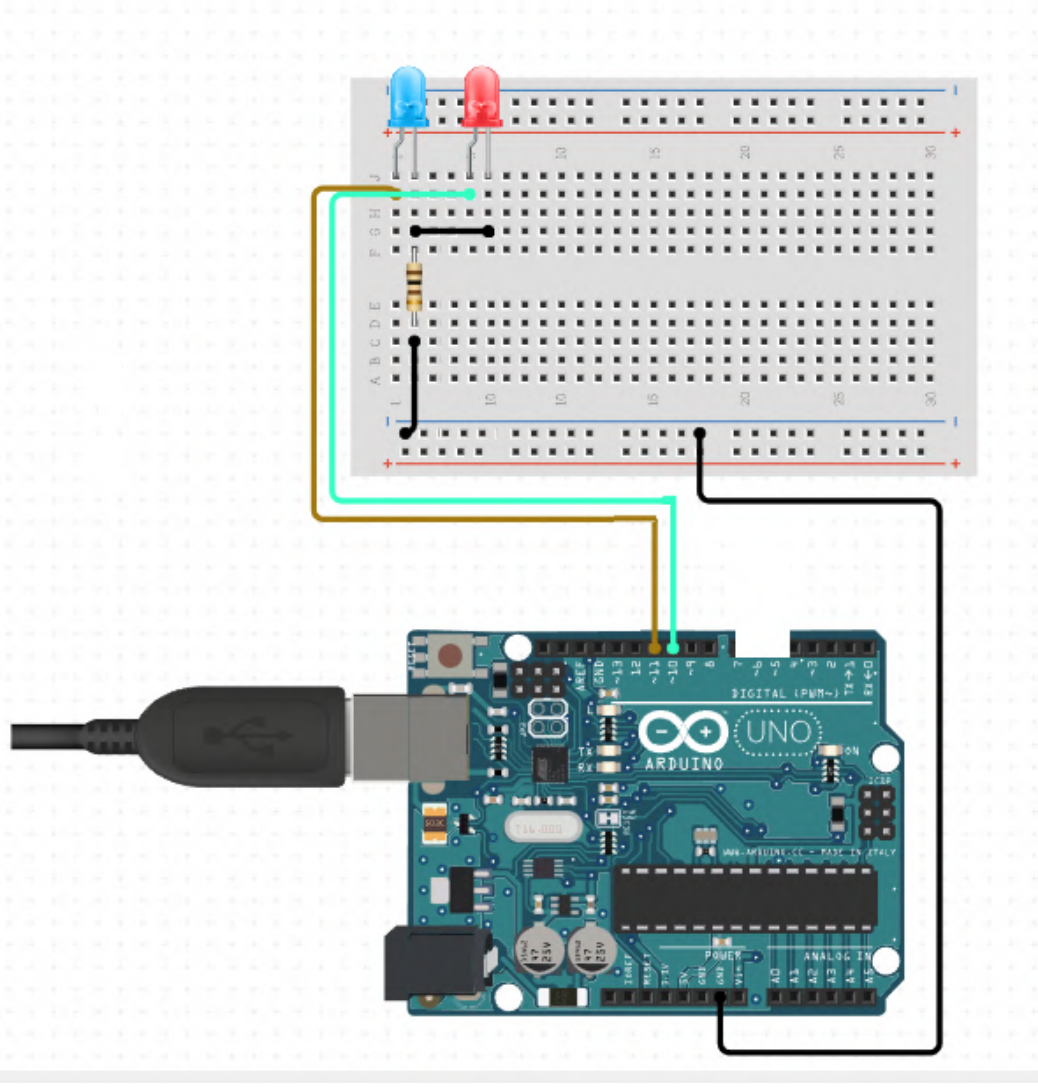


Figura 3.26: Diagrama del circuito. [68]

Capítulo 4

Pruebas y resultados

Se realizaron diferentes pruebas a lo largo del desarrollo del proyecto, en las que se requiere un análisis profundo para que el proyecto tenga un bajo porcentaje de error y sea lo más eficiente posible. Se hablará del análisis del parámetro de factor escala y cuadros vecinos llamado "Scalefactor y MinNeighbors" dónde se ajusta la parte de sensibilidad para la correcta detección facial. En el segundo análisis se explicará sobre la decisión de entrenar el modelo con una cantidad adecuada de fotogramas. Como tercer análisis, se evaluará el umbral de decisión para una correcta identificación facial.

Dentro de todo el análisis se presentan ilustraciones dónde se explica el desarrollo de este capítulo siendo todas de creación propia.

4.1. Primer análisis "Scalefactor y MinNeighbors"

Dentro del primer módulo (Registro facial) en el bloque de código de la figura 4.1, se ajustaron los parámetros ***Scalefactor*** y ***MinNeighbors*** para la correcta detección facial. Este ajuste es muy importante porque sin él no se puede detectar ningún rostro y es primordial para continuar con el análisis de imágenes, puede ocurrir que al momento de tomar los fotogramas para almacenarlos en las carpetas tome algunas imágenes basura con falsos positivos, lo cual genera un problema en el momento de llevar esa información al modelo de entrenamiento.

En la figura 4.1 se muestran 4 características de configuración que deben de tener

```
caras = modeloClasificador.detectMultiScale(gray,
                        scaleFactor = 1.1,
                        minNeighbors = 15,
                        minsie=(30,30),maxsize=(200,200))
```

Figura 4.1: Características del detector a modificar.

un análisis para su correcto funcionamiento, donde: *ScaleFactor* es un porcentaje de escala de la imagen, quiere decir que se buscarán imágenes positivas dentro del *frame* en una escala establecida, lo cual ayuda a encontrar el rostro a las diferentes distancias que el usuario se coloque, ya que eso muestra un objetivo de diferente tamaño debido a las diferentes profundidades. Con *MinNeighbors* se indica el número de cuadros vecinos a encontrar, si se coloca un número muy alto será muy sensible en su búsqueda de rostros encontrando falsos positivos, de lo contrario, si es muy bajo el número, el modelo no identificará rostros en el *frame*. Por ultimo *minsize* y *maxsize* son parámetros que modifican el tamaño de los pixeles a buscar si es que se tienen imágenes cargadas y el proceso no se realiza en *streaming*.

Para disminuir lo más posible el porcentaje de error y hacer más eficiente la detección de rostros se realizaron 2 rondas de 4 modificaciones cada una, ajustando las dos características mencionados. El registro de estas pruebas se pueden observar de la figura 4.2 a la 4.5. Dentro de cada ronda, las 4 modificaciones de parámetros se registraron con las siguientes condiciones:

- Tomando 100 fotogramas por modificación, ya que el sistema esta diseñado para trabajar con un número reducido de fotogramas y es innecesario sobrecargar el entrenamiento.

- Con un único usuario.

- Un tono de luz similar al entorno de aplicación del sistema.

Evaluando los siguientes tres aspectos:

- Tiempo de operación que tarda en almacenar los 100 fotogramas.
- Porcentaje de falsos positivos, es decir, que el fotograma carezca de un rostro.
- Porcentaje de fotogramas defectuosos, ya que el identificador no siempre puede tomar un fotograma de buena calidad, a pesar de haber identificado el rostro existe una posibilidad de almacenar un rostro borroso, con exceso de luz, muy obscuro o que presente una detección facial insuficiente que no cumpla con el 70 % del tamaño del rostro original.

4.1.1. Ronda 1

Dentro de la figura 4.2, en la primera modificación, se encontraron 15 falsos positivos de los 100 fotogramas tomados (figura 4.2e), con detección de rostros donde no existen (figura 4.2c); problemas con la cantidad de rostros a encontrar (figura 4.2a y 4.2b); dentro de los rostros detectados hay uno que siempre detecta correctamente a pesar de estar cerca o alejado de la cámara; con 5 fotogramas defectuosos que no cumplen con la condición del tamaño del rostro recomendado, capturando solamente la boca del usuario. **[9 seg de operación, 15 % falsos positivos, 5 % fotogramas defectuosos.]**

En la segunda modificación en la figura 4.3, dentro del tiempo de ejecución, la detección de rostros tuvo 100 % de asertividad (figura 4.3e), en ningún momento tuvo falsos positivos (figuras 4.3a, 4.3b, 4.3c y 4.3d) pero obtiene 10 fotogramas defectuosos en los cuales el proceso de captura fue demasiado rapido y almacena el rostro borroso. **[6 seg de operación, 0 % de falsos positivos, 10 % fotogramas defectuosos.]**

En la modificación número tres de parámetros, figura 4.4, hubo una correcta detección a dimensiones medianas y pequeñas de rostros (4.4b y 4.4d), pero problemas con la detección de rostros muy grandes o muy pegados a la cámara como en las

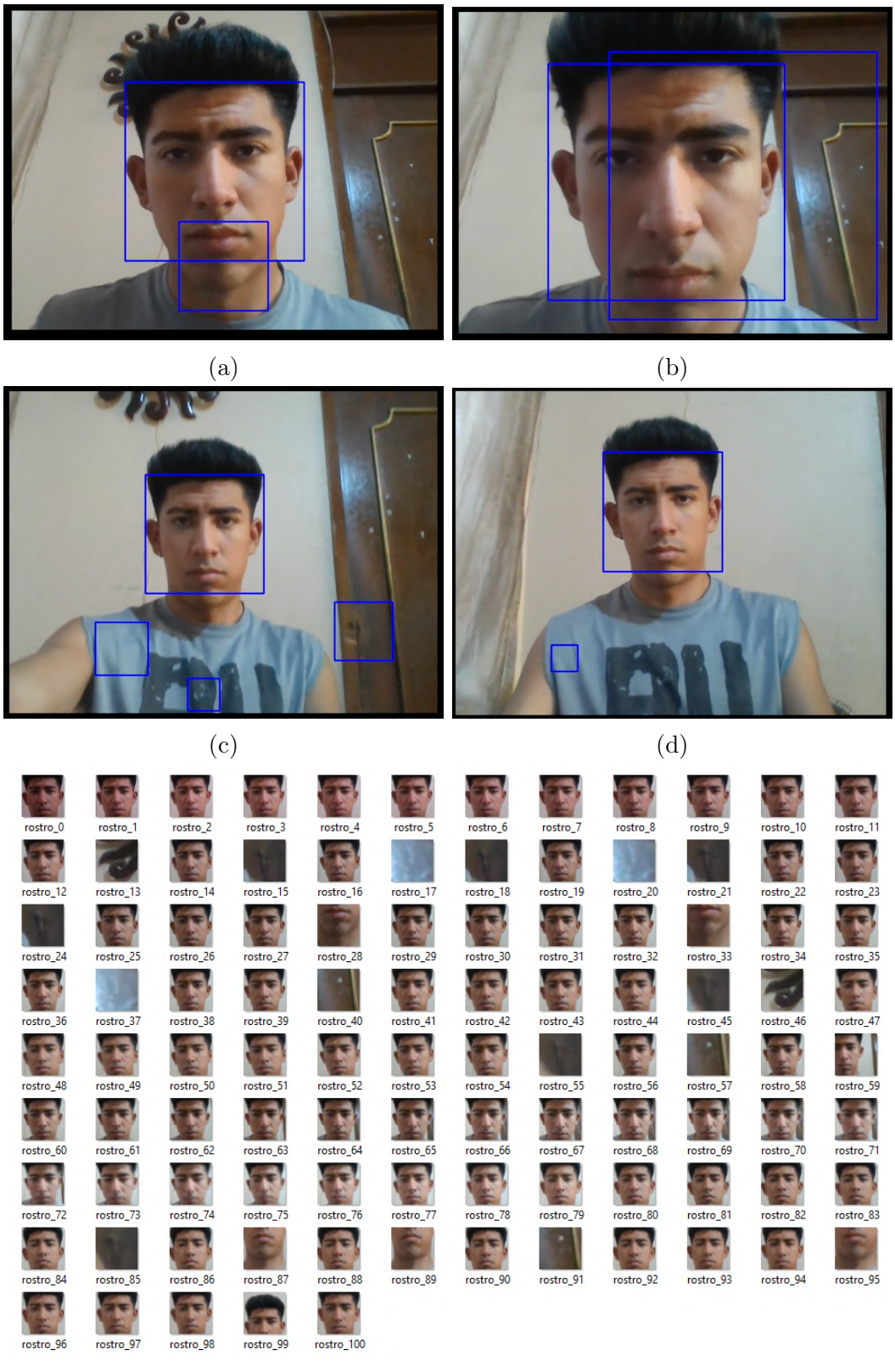


Figura 4.2: Primera modificación de parámetros
ScaleFactor = 1.1 / MinNeighbors= 1.

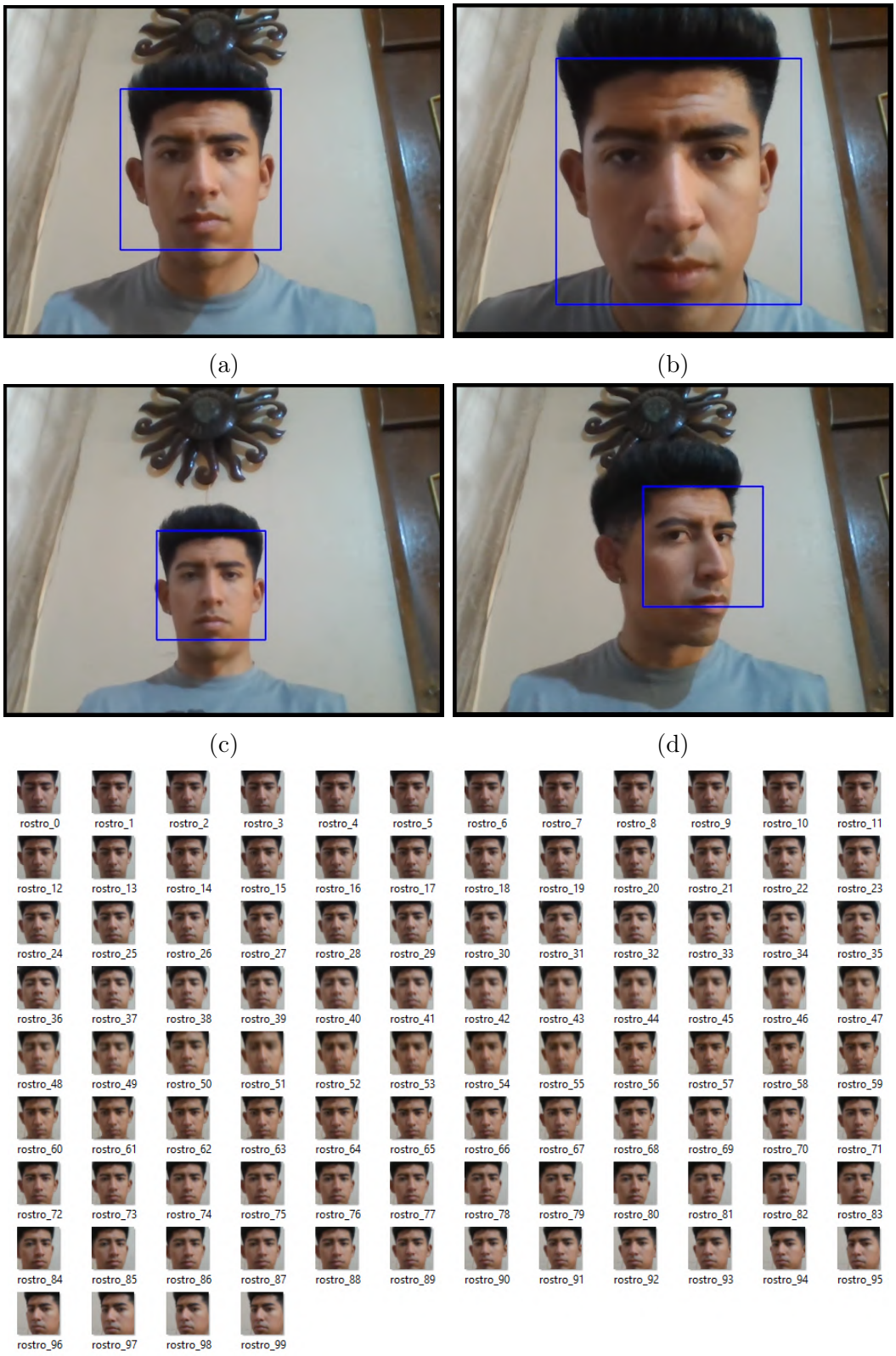


Figura 4.3: Segunda modificación de parámetros
ScaleFactor = 1.3 / MinNeighbors= 5.

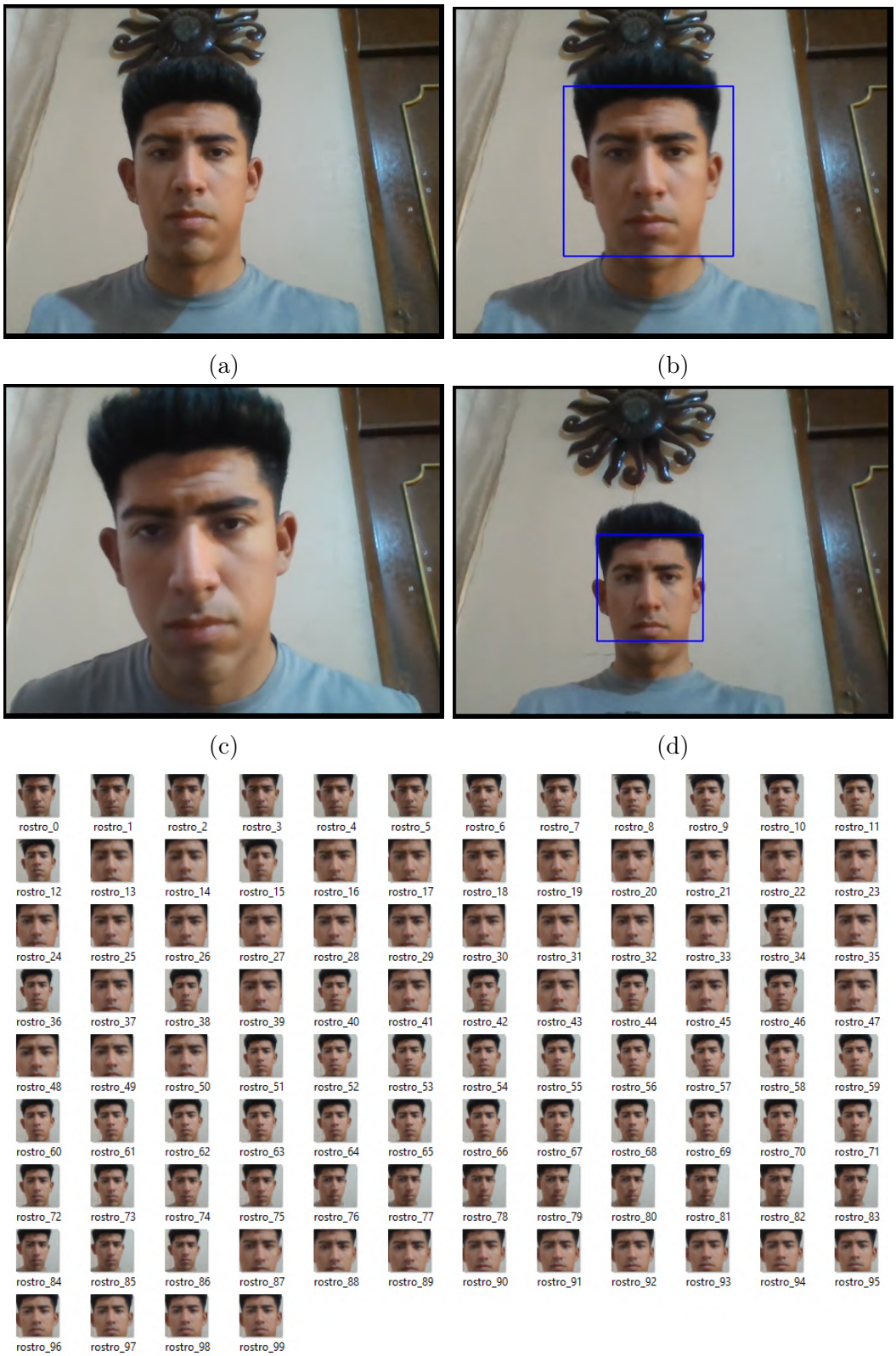


Figura 4.4: Tercera modificación de parámetros
ScaleFactor = 1.6 / MinNeighbors= 10.

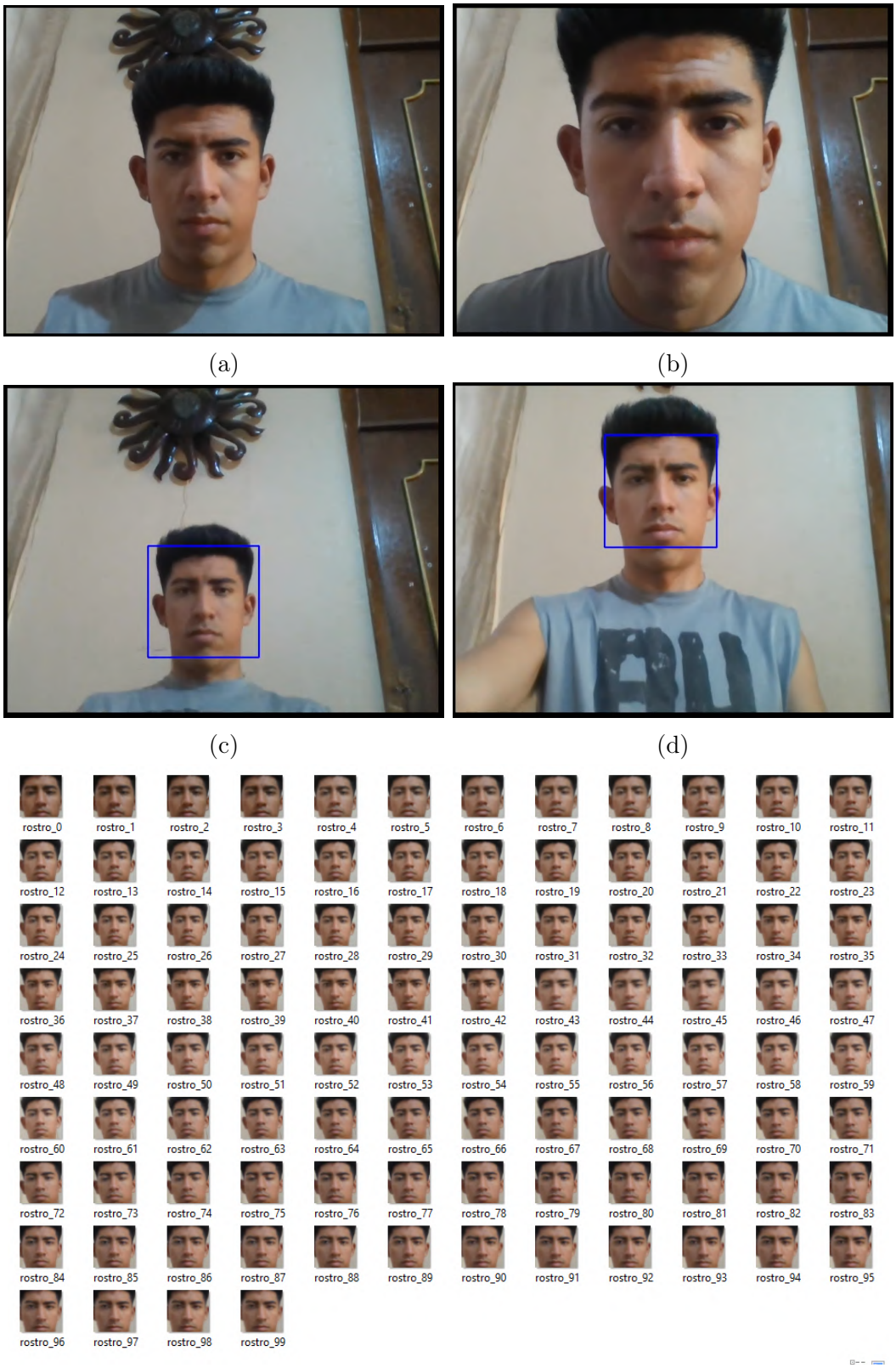


Figura 4.5: Cuarta modificación de parámetros
ScaleFactor = 1.9 / MinNeighbors= 15.

figuras 4.4a y 4.4a. **7 seg de operación, 0 % de falsos positivos, 0 % fotogramas defectuosos.]**

Con la cuarta modificación (figura 4.5), se obtiene una nula detección con rostros grandes pegados a la cámara (4.5a y 4.5b, con rostros alejados o pequeños muestra un comportamiento mejor ya que logra detectarlos, pero se observa que el proceso tarda mucho tiempo (4.5c y 4.5d). **[17 seg de operación, 0 % de falsos positivos, 0 % fotogramas defectuosos.]**

Los resultados se registraron en la tabla 4.1.

Registro de resultados ronda 1 (Captura de fotogramas)			
Parámetros	Tiempo total de ejecución. (seg)	Porcentaje falsos positivos	Porcentaje fotogramas defectuosos
ScaleFactor = 1.1 MinNeighbors= 1	9 seg	15 %	5 %
ScaleFactor = 1.3 MinNeighbors= 5	6 seg	0 %	10 %
ScaleFactor = 1.6 MinNeighbors= 10	7 seg	0 %	0 %
ScaleFactor = 1.9 MinNeighbors= 15	17 seg	0 %	0 %

Tabla 4.1: Tabla comparativa de resultados con las 4 primeras modificaciones de parámetros. Ronda 1.

Se registraron los datos obtenidos en la tabla 4.1 y con una lectura rápida de las primeras 4 modificaciones de parámetros se identifica dentro de qué rango se comporta mejor el sistema de detección de rostros, con el análisis explicado se puede notar que se necesita un valor de *Scalefactor* bajo para poder detectar los rostros a diferentes distancias, con las primeras pruebas se obtiene un mejor comportamiento a escalas bajas. Se observa que el tiempo de operación es muy importante, ya que se busca una ejecución rápida para que el usuario no espere un tiempo excesivo, pero surgieron algunos problemas con la velocidad, ya que la cámara no logra enfocar tan rápido y captura fotogramas borrosos. El otro error a considerar es el porcentaje de falsos positivos, se tuvo buen comportamiento manteniendo un parámetro más estricto del *MinNeighbors*, se detecta que a una escala baja arroja muchos falsos

positivos dando un porcentaje de error más alto, a partir del índice `minNeighbors = 5` se pudo observar el 0 % de falsos positivos lo cual aumenta la asertividad en la detección por un enrenamiento con datos de calidad. Con un 10 % de fotogramas defectuosos en el proceso mas rápido de 6 seg y un 5 % en la doble captura de rostros.

4.1.2. Ronda 2

Considerando las observaciones anteriores se realizará una nueva serie de 4 pruebas con diferentes combinaciones de parámetros. Manteniendo un *ScaleFactor* por debajo de 1.5 y un *MinNeighbor* por encima de 5, hasta 15.

En la quinta modificación, el sistema no mostró errores ni complicaciones en la detección de rostros (figura 4.6) tanto cercanos y grandes (figuras 4.6b y 4.6d) como pequeños y alejados (figuras 4.6a y 4.6d). **[12 seg de operación, 0 % de falsos positivos, 0 % fotogramas defectuosos.]**

Con la sexta modificación mostrada en la figura 4.7, el sistema no pierde el rostro a pesar del movimiento del usuario, detecta rostros cercanos y grandes, cómo también los alejados y pequeños. Figuras 4.7a, 4.7b 4.7c y 4.7d. Con 4 fotogramas defectuosos capturando el rostro borroso. **[8 seg de operación, 0 % de falsos positivos, 4 % fotogramas defectuosos.]**

En la modificación número 7, que se presenta en la figura 4.8, no pierde el rostro a pesar de los movimientos de usuario, no toma falsos positivos y termina el proceso muy rápido. Pero debido a la alta velocidad los primeros 6 rostros aparecen muy borrosos, se puede observar en los primeros fotogramas de la figura 4.8e **[6 seg de operación, 0 % de falsos positivos, 6 % de fotogramas defectuosos.]**

Por último, en la octava modificación que aparece en la figura 4.9, se observan dos cuadros delimitadores identificando 2 rostros al mismo tiempo (figura 4.9a y 4.9d). Lo que provoca una base de datos con rostros de diferente tamaño (figura 4.9e) y eso es un problema al momento de entrenar el sistema, ya que se deben obtener datos lo más limpios posibles para que resulte eficaz. **[4 seg de operación, 0 % de falsos positivos, 37 % de fotogramas defectuosos.]**

Registrando los valores obtenidos en la tabla 4.2 se analiza que la cuarta

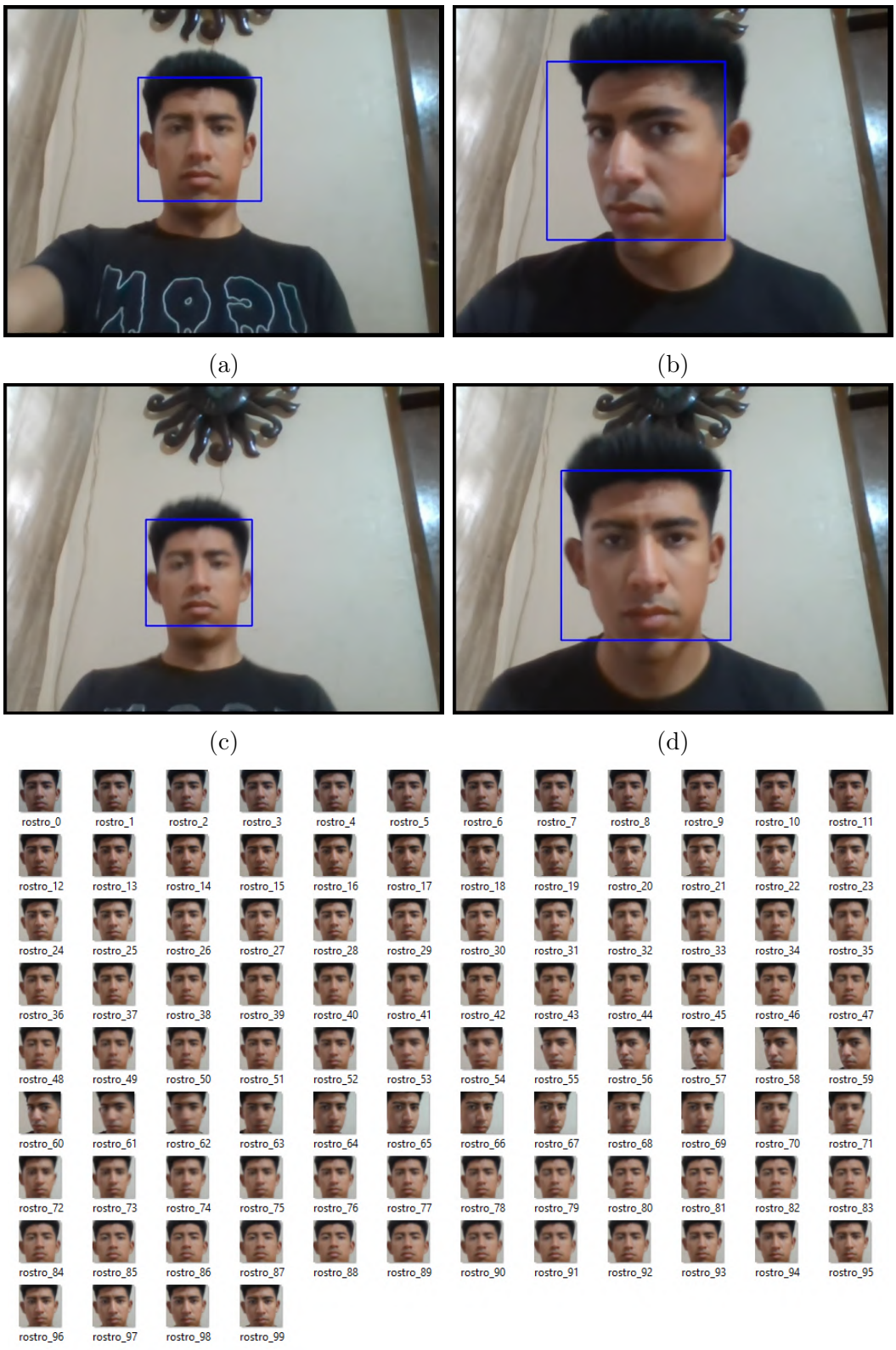


Figura 4.6: Quinta modificación de parámetros
ScaleFactor = 1.1 / MinNeighbors= 15.

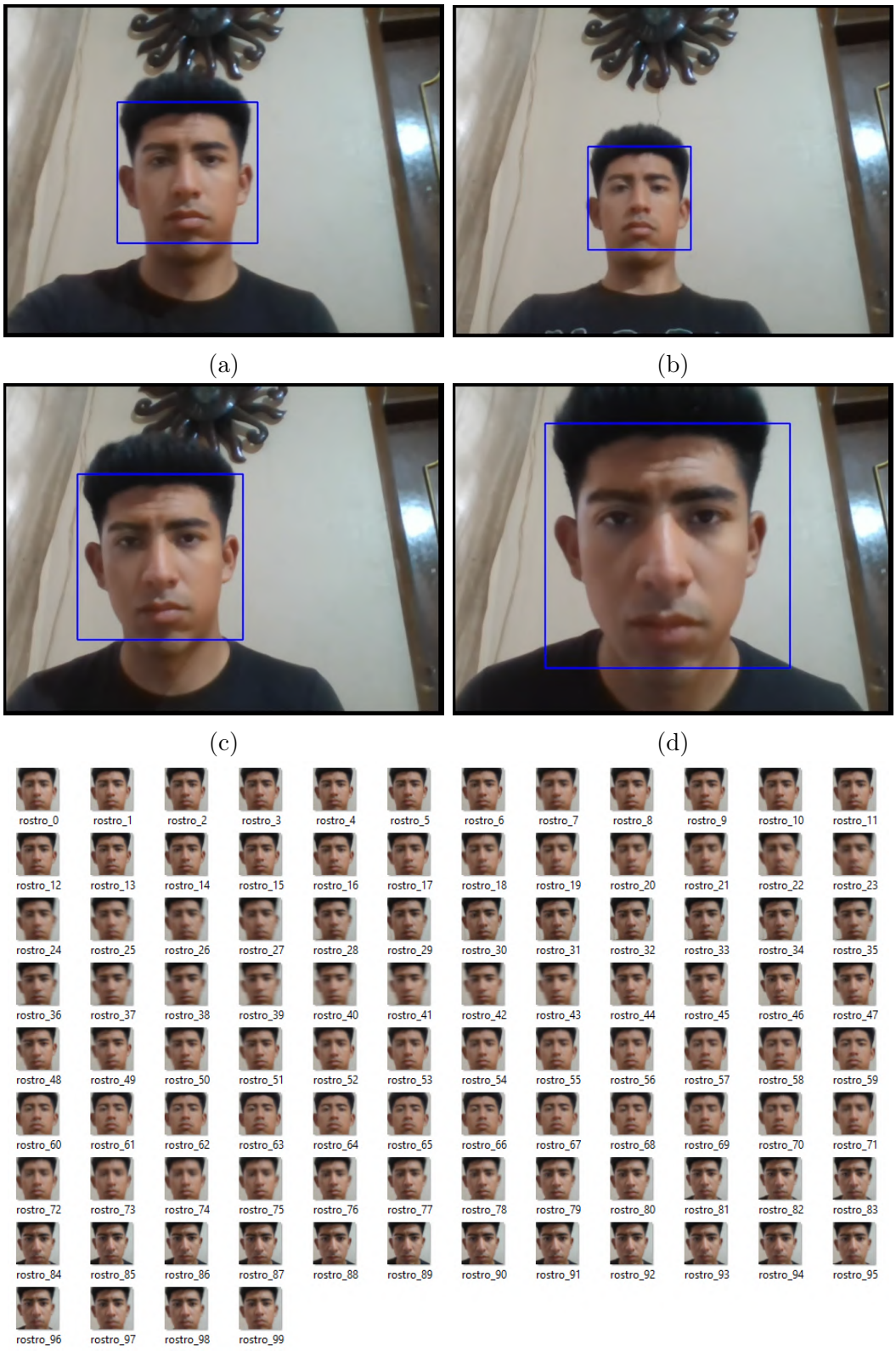


Figura 4.7: Sexta modificación de parámetros
ScaleFactor = 1.2 / MinNeighbors= 10.

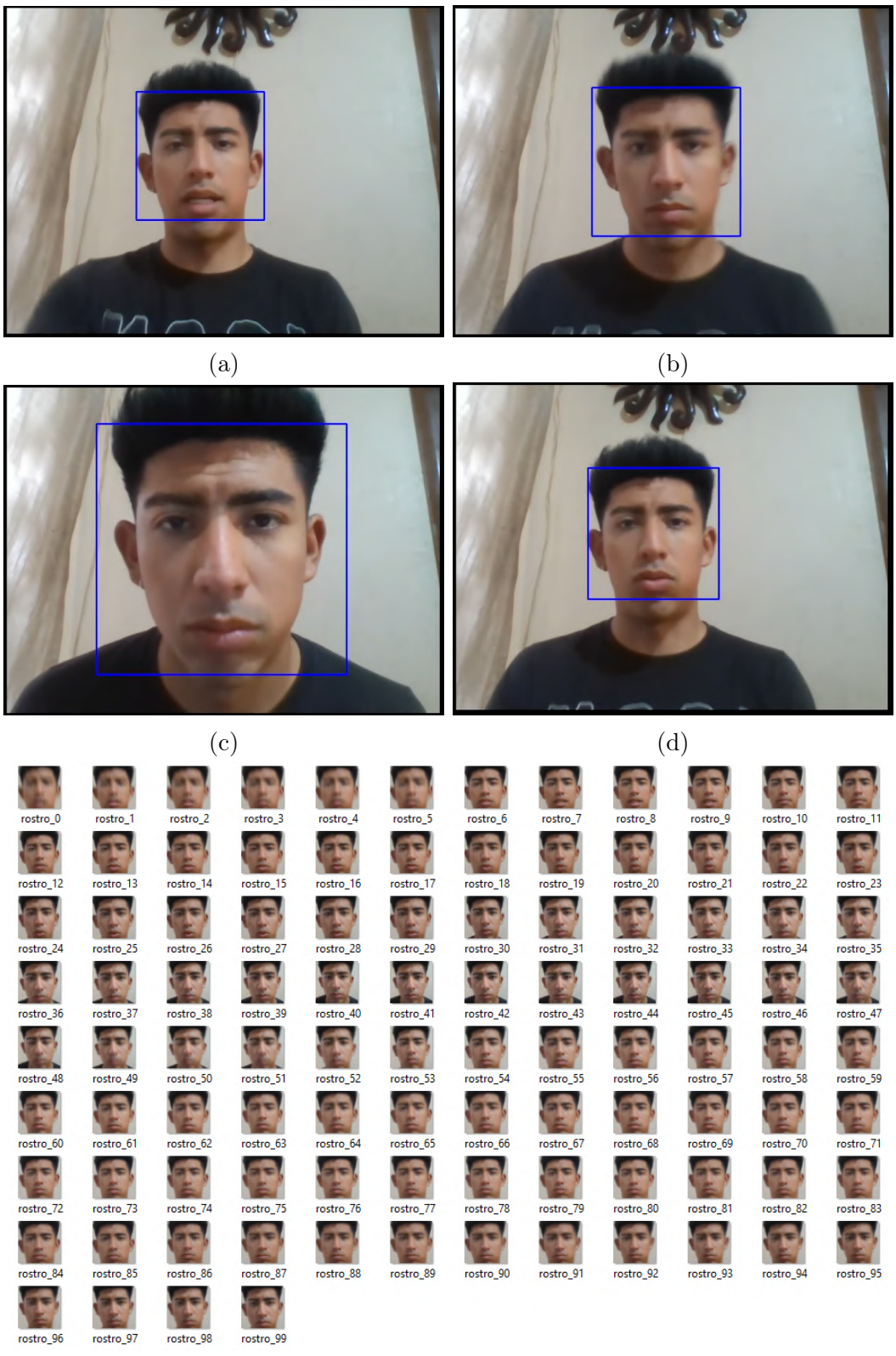


Figura 4.8: Séptima modificación de parámetros
ScaleFactor = 1.3 / MinNeighbors= 8.

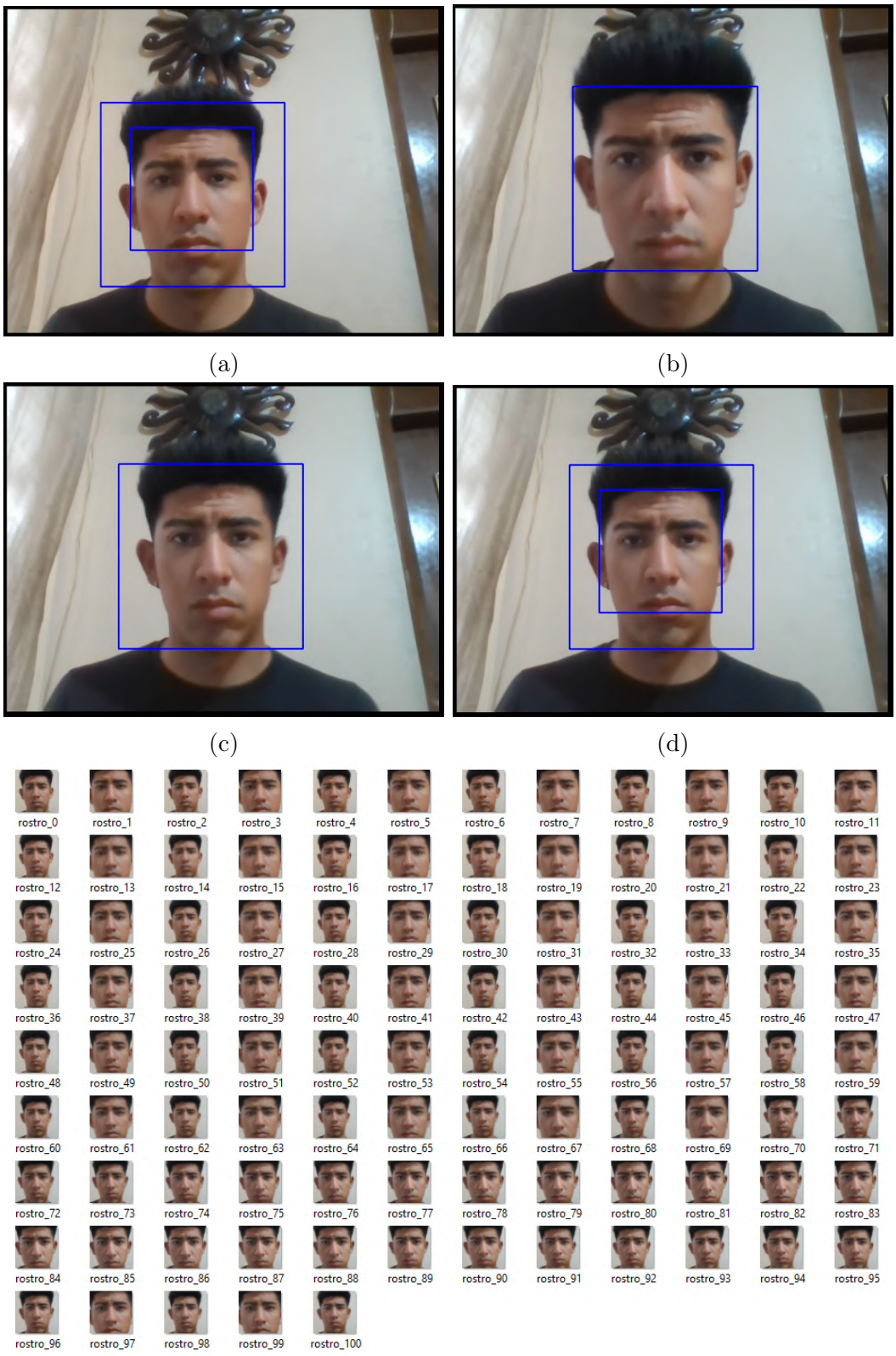


Figura 4.9: Octava modificación de parámetros
ScaleFactor = 1.5 / MinNeighbors= 5.

Registro de resultados ronda 2 (Captura de fotogramas)			
Parámetros	Tiempo total de ejecución. (seg)	Porcentaje falsos positivos	Porcentaje fotogramas defectuosos
ScaleFactor = 1.1 MinNeighbors= 15	12 seg	0 %	0 %
ScaleFactor = 1.2 MinNeighbors= 10	8 seg	0 %	4 %
ScaleFactor = 1.3 MinNeighbors= 8	6 seg	0 %	6 %
ScaleFactor = 1.5 MinNeighbors= 5	4 seg	0 %	37 %

Tabla 4.2: Tabla comparativa de resultados con las 4 segundas modificaciones de parámetros. Ronda 2.

combinación [1.5/5] será eliminada ya que, a pesar de ser la más rápida con tan solo 4 segundos, genera una base de datos duplicada con dos rostros, esto afectará en un modelo de entrenamiento con rostros defectuosos. La tercera combinación [1.3/8] generó en las primeras tomas una imagen borrosa del usuario, por lo cual tiene rostros defectuosos y es un problema para la generación del modelo de entrenamiento. En la primera combinación [1.1/15] se tiene un tiempo de operación muy lento, no tiene fallas, pero se tiene que optimizar el proceso lo más posible así que por eso se ocupará la segunda combinación [1.2/10], ya que a pesar de tener un 4 % en fotogramas defectuosos es la que menor porcentaje presenta, no tuvo errores de detección con falsos positivos y fue la más rápida.

4.2. Segundo análisis ”Fotogramas”

El segundo punto a analizar es ¿Cuántos fotogramas es adecuado capturar para un correcto entrenamiento? Debido a que el proceso de entrenamiento necesita información para poder catalogar los diferentes rostros y así realizar una correcta predicción es necesario conocer bajo qué cantidad de información se entrena mejor, no se puede saturar el entrenamiento con demasiados fotogramas, pero también es necesario una variedad de rostros para que sea un entrenamiento efectivo. En esta

aplicación, no se sabe exactamente cuántos usuarios estarán registrados, lo ideal sería dejar un parámetro estable para una cantidad promedio de usuarios, así que se tomará un parámetro para un registro dentro de un edificio pequeño con un total de 100 personas, de las cuales se registraran en el sistema un 30 % del total. Sabiendo hacia donde está dirigido el sistema, se buscara que sea eficiente para un rango desde 3 hasta 40 personas. Por último, es necesario explicar que entre más usuarios se quieran registrar al sistema son más los fotogramas que se deben adquirir, pero no se puede sobrecargar el entrenamiento con tantos fotogramas si solo se van a identificar 3 usuarios, por ello se necesita un parámetro amplio que funcione lo mejor posible para ambos casos.

Para ello se realizaron varias pruebas con diferentes cantidades comenzando desde 10 hasta 500 fotogramas. Este rango de prueba se decidió con base a la documentación del modelo LBPH, ya que este fue diseñado para que la identificación facial funcione con una base de datos reducida en datos de entrenamiento.

De dichas pruebas, se analizó el comportamiento del modelo viendo cual era el mejor rango para la identificación. Este análisis fue directamente por prueba y error, ya que se iban determinando los parámetros con los cuales se tenía un mejor comportamiento, se iniciaba con el rango más bajo, se realizaba la prueba y se analizaba el comportamiento, se volvía a modificar, se probaba de nuevo y se analizaba, así hasta que se pudo llegar a la conclusión que la mejor cantidad para un funcionamiento más eficiente era de 200 fotogramas.

4.3. Tercer análisis ”Result”

El último punto que se debe analizar es un índice con el cual se tomará la última decisión de si es una persona registrada o es un desconocido. Aunque parezca muy básico, decidir con un *if-else* es necesario porque todo este proceso del análisis de imágenes se maneja a través de datos numéricos, entonces, al final, se necesitan analizar los datos que arroja la identificación para que se pueda dar esa decisión. El proceso funciona a través del comando *predict* junto con el modelo de entrenamiento

que es el que realiza esta parte de los cálculos matemáticos y comparaciones; dentro del código se almacenó esta operación en una variable llamada *result* y con esa variable es con la que se trabajará para la decisión. Para darle un valor numérico adecuado a ese *if-else* es necesario conocer bajo que parámetro ronda el valor del cálculo mostrado en la variable. Para ello se realizó una identificación de una persona registrada y se almacenan los valores de vuelta que arroja la variable *result*, después se realizó el mismo proceso pero ahora con una persona no registrada y se almacenan los valores de igual manera. A partir de ahí se tomó la media de esos datos para llegar a un primer rango de valores y realizar otras pruebas y modificaciones con dos usuarios más.

El proceso se realizó modificando el código en el *script* de identificación, agregando una lista llamada *tupla*, para registrar los valores de la variable *result*. Esto se desarrolló con el comando *append*, agregando el valor de comparación a la lista de la variable cada que el modelo de identificación facial realiza los cálculos, estos cálculos son repetidos hasta que el ciclo se interrumpe debido a que es un análisis en tiempo real y el resultado cambia dependiendo las condiciones de luz, entrenamiento y enfoque de la cámara. Una vez creada la lista con los resultados de la comparación, el siguiente paso es almacenar dichos datos en dos archivos diferentes con extensión `.csv`. Para ello, con las instrucciones del código de la figura 4.10, se almacena la lista en el archivo correspondiente, uno para almacenar los datos con entrenamiento y otro sin entrenamiento para después poder hacer las comparaciones en un gráfico.

En este punto, es necesario recurrir a la minería de datos porque los valores que se analizarán vienen "sucios", con información que no es requerida, lo cual aumenta la dificultad de manipularlos ya que son tipo tupla y cuentan con dos datos por valor, como en el siguiente ejemplo `(0 59.3701)`, dónde el primer índice es un valor que indica la posición que ocupa el segundo índice dentro de un ciclo repetitivo y el segundo índice es el resultado de la operación matemática de comparación. En este caso el índice que se ocupará es el segundo, pero no se podrá almacenar por separado, entonces, después de crear la lista y juntar los datos tipo tupla se convertirán a tipo cadena y se almacenarán en el archivo `csv` del usuario como se ve en la figura 4.11.

Figura 4.10: Instrucciones para almacenar la lista en el archivo .csv y graficar el dataframe.

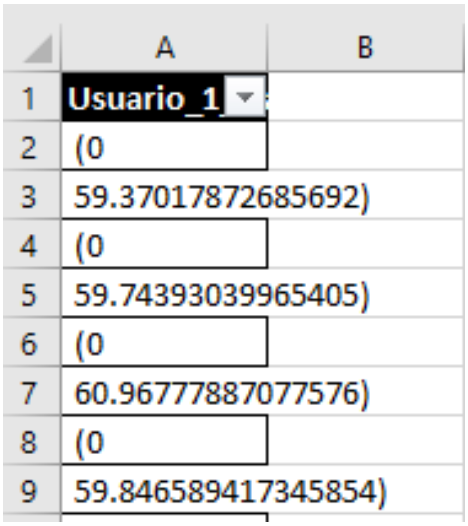
El siguiente paso será eliminar el primer índice de cada valor, es decir, los ceros junto con los paréntesis, para ello se ordenan los datos en una sola columna y se les dará un formato de tabla, como en la figura 4.12, después se filtrarán los ceros y los paréntesis, y se eliminarán sus filas para tener como resultado una base de datos "limpia". (figura 4.13).

	A	B	C	D	E	F	G
1	['Indices'	(0	59.37017872	(0	59.74393039	(0	60.96777887
2							

Figura 4.11: Muestra de datos sin limpieza en csv.

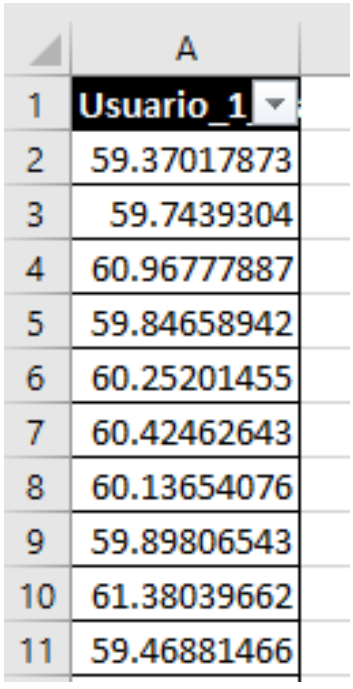
Con una estructura limpia de datos, se podrá manipular y complementar la tabla con la información requerida. Se toman 50 muestras para un análisis rápido pero certero. En la figura 4.14 se muestra la tabla final. Aumentar una columna con el promedio ayudará a ver gráficamente la tendencia de comportamiento, agregar columnas con el umbral inferior y superior da la oportunidad de hacer el análisis del parámetro adecuado para dar los límites de la decisión.

En la parte final de la minería de datos se transformará esta información en un *Dataframe* para poder realizar una gráfica dentro de *Python*. Para este análisis se ocupó *Jupyter Notebook* ya que su estructura facilita el proceso por su herramienta de compilar solo el bloque de código seleccionado, dando oportunidad a ejecutar



	A	B
1	Usuario_1	
2	(0	
3	59.37017872685692)	
4	(0	
5	59.74393039965405)	
6	(0	
7	60.96777887077576)	
8	(0	
9	59.846589417345854)	

Figura 4.12: Muestra de la estructura de la tabla.



	A
1	Usuario_1
2	59.37017873
3	59.7439304
4	60.96777887
5	59.84658942
6	60.25201455
7	60.42462643
8	60.13654076
9	59.89806543
10	61.38039662
11	59.46881466

Figura 4.13: Tabla con datos limpios.

	A	B	C	D
1	usuario_1	Promedio	Superior	Inferior
2	54.32042082	53.38119932	55.1565026	52.0376092
3	53.93157739	53.38119932	55.1565026	52.0376092
4	53.21991016	53.38119932	55.1565026	52.0376092
5	53.72683879	53.38119932	55.1565026	52.0376092
6	53.25400298	53.38119932	55.1565026	52.0376092
7	52.64102477	53.38119932	55.1565026	52.0376092
8	52.88511502	53.38119932	55.1565026	52.0376092
9	53.65896458	53.38119932	55.1565026	52.0376092
10	53.42256851	53.38119932	55.1565026	52.0376092

Figura 4.14: Tabla con promedio, valor máximo y valor mínimo.

parte por parte el código, también las bibliotecas enfocadas al análisis de datos como *Pandas y Matplotlib* hacen mas cómoda esta tarea. Dentro del código de la figura 4.15 se observa el *head* del *dataframe* con las 5 primeras muestras, esta función es necesaria para saber cómo queda el *dataframe* y cómo esta compuesto, así se podrán manipular desde la consola los datos, cambiar los encabezados y agregar o eliminar filas o columnas.

```
In [51]: archivo_nuevo = pd.read_csv('C:/Users/Lenovo/Documents/Proyectos/Proyectos Python/Reconocimiento facial
         archivo_nuevo.head()
         #archivo_nuevo.plot()
```

	usuario_1_ivan	Promedio	Superior	Inferior
0	54.320421	53.381199	55.156503	52.037609
1	53.931577	53.381199	55.156503	52.037609
2	53.219910	53.381199	55.156503	52.037609
3	53.726839	53.381199	55.156503	52.037609
4	53.254003	53.381199	55.156503	52.037609

Figura 4.15: Instrucciones para visualizar el dataframe.

Ya que finaliza el proceso de limpieza de la información requerida se procederá a realizar el análisis. Para ello se realizarán una serie de gráficas para identificar de esta manera el umbral de decisión que se necesita. Registrando los resultados en una tabla comparativa 4.3 con los datos principales para identificar ese umbral de decisión.

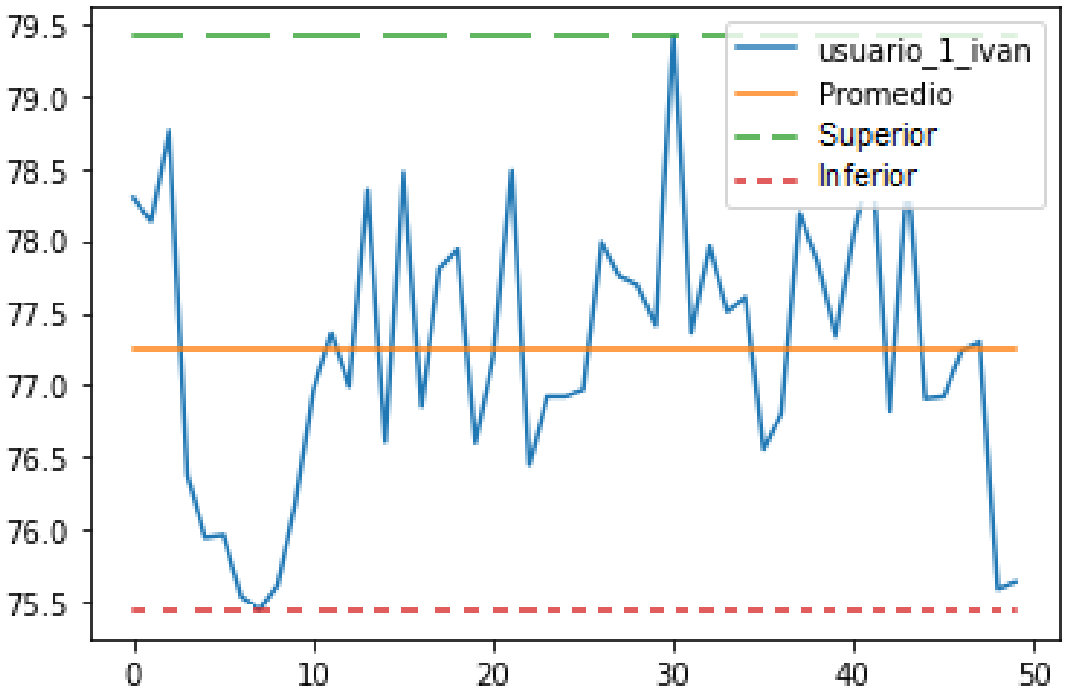


Figura 4.16: Resultados numéricos de la identificación con el usuario 1 (IVAN) sin entrenamiento.

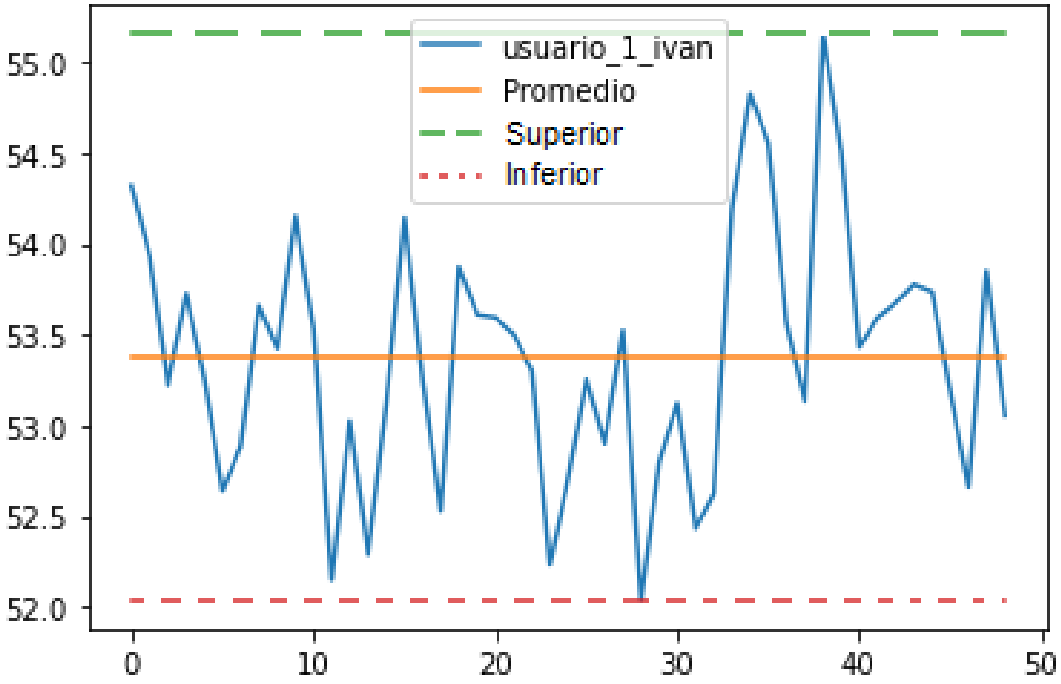


Figura 4.17: Resultados numéricos de la identificación con el usuario 1 (IVAN) con entrenamiento.

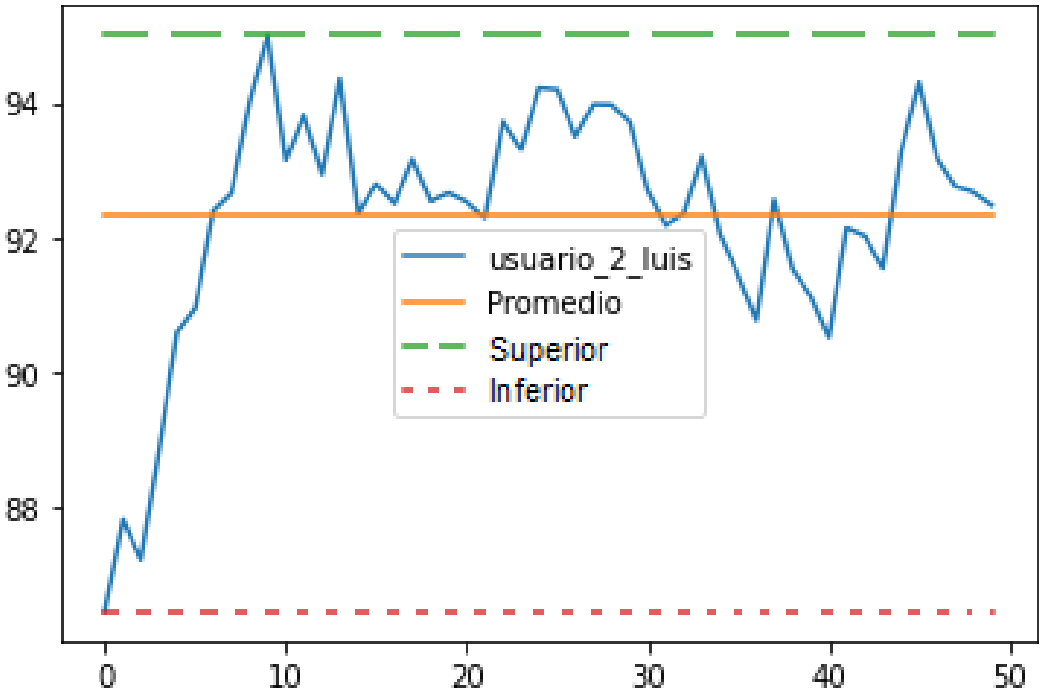


Figura 4.18: Resultados numéricos de la identificación con el usuario 2 (LUIS) sin entrenamiento.

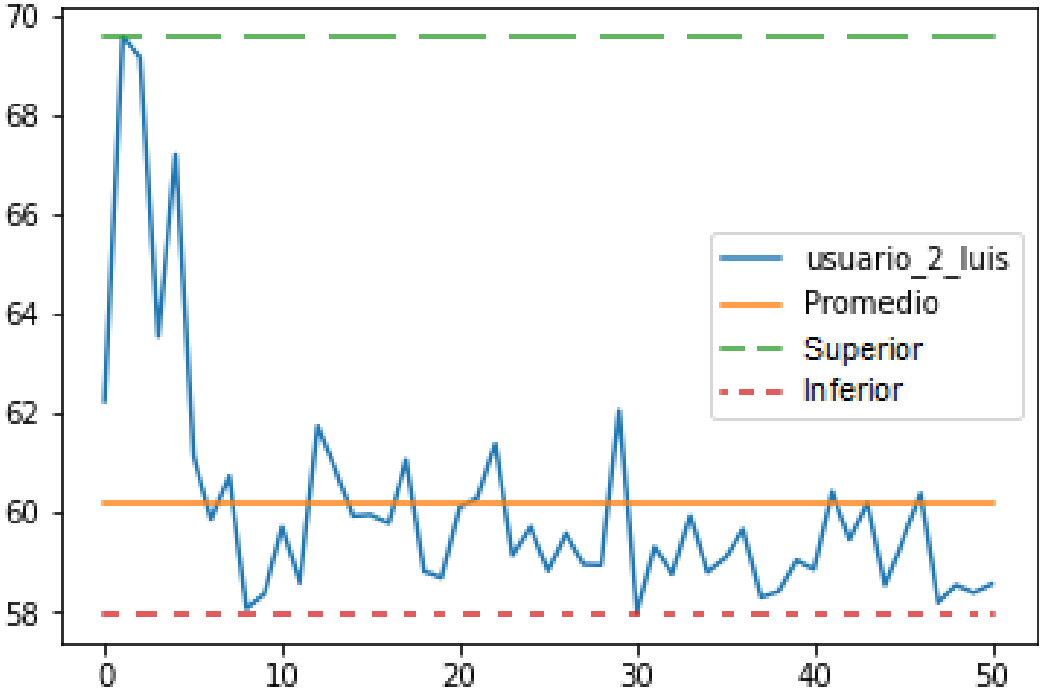


Figura 4.19: Resultados numéricos de la identificación con el usuario 2 (LUIS) con entrenamiento.

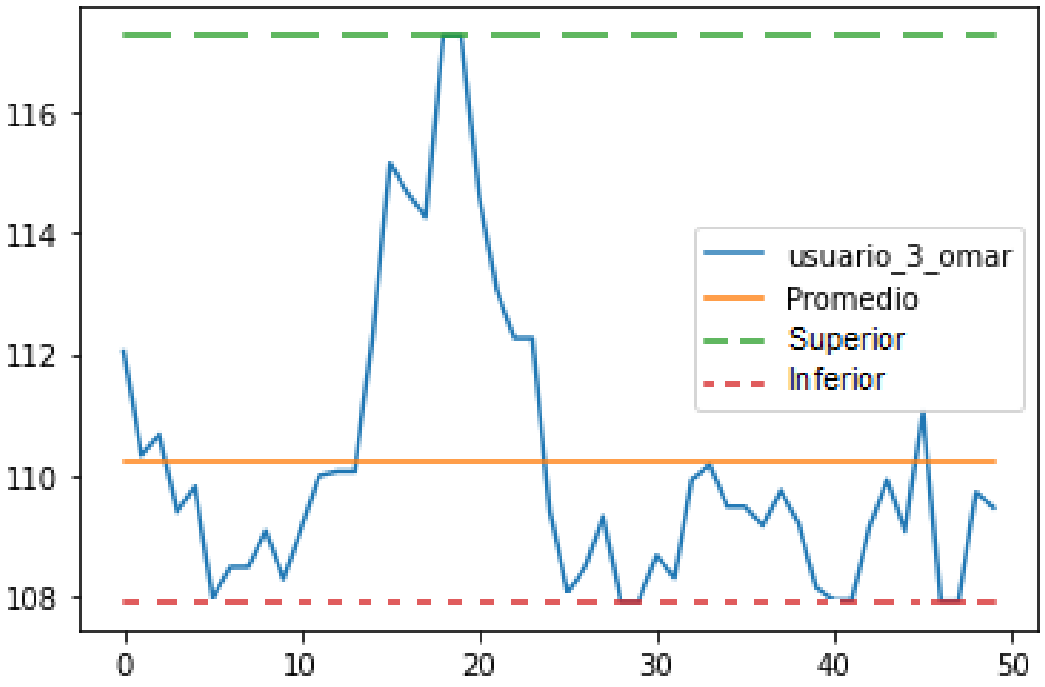


Figura 4.20: Resultados numéricos de la identificación con el usuario 3 (OMAR) sin entrenamiento.

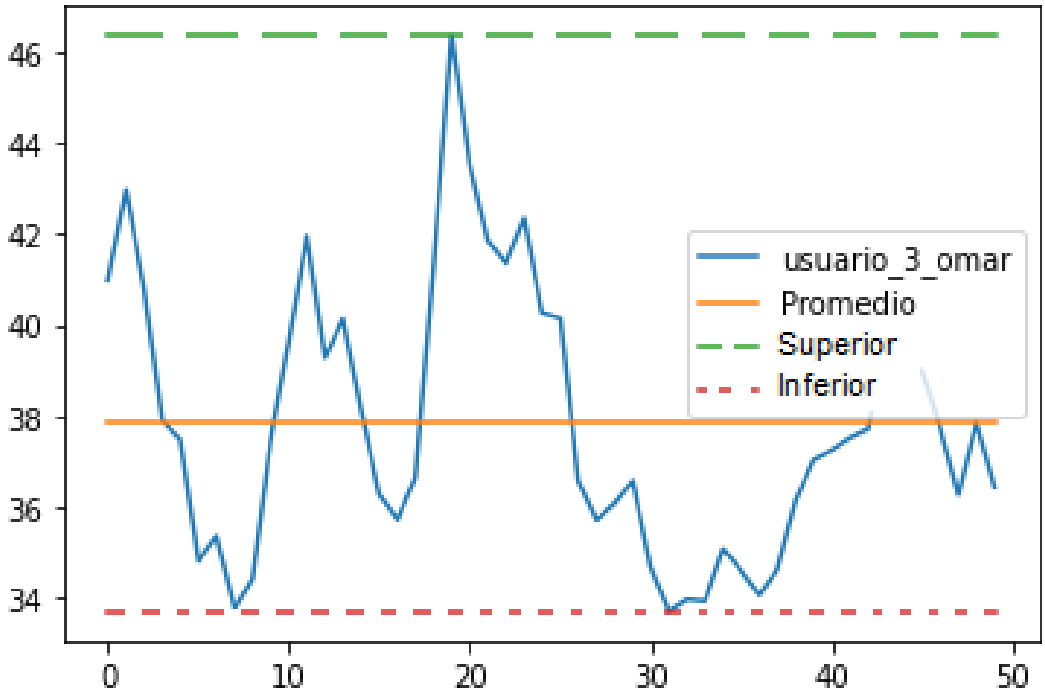


Figura 4.21: Resultados numéricos de la identificación con el usuario 3 (OMAR) con entrenamiento.

Se debe prestar especial atención a los valores máximos y mínimos de cada gráfica, ya que con ellos se va a decidir qué número se le otorga a la condición *if-else* para discriminar entre usuario conocido o desconocido. Este rango de valores son el resultado de la operación matemática del análisis de comparación del usuario entrenado y no entrenado expresado en unidades del 0 en adelante. Se considerará que a medida que los valores se acerquen a cero, el sistema de identificación presentará una mayor confianza en la asignación del usuario correspondiente. Por otra parte, valores elevados cercanos al cien indicarán una menor capacidad del sistema para reconocer a un usuario con las características físicas asociadas.

Para el primer usuario (Iván) se observa que cuando no cuenta con un entrenamiento el umbral superior es de 79.5 unidades y el umbral inferior es de 75.5 unidades, cómo se muestra en la gráfica figura 4.16, con un índice promedio de 77.2 unidades. En la gráfica de la figura 4.17, con un entrenamiento previo el umbral superior es de 55.1 unidades y el umbral inferior es de 52.03 unidades, con un índice promedio de 53.4 unidades.

Para el segundo usuario (Luis), se observa en la gráfica de la figura 4.18 que el umbral superior es de 93.0 unidades y el inferior de 86.0 unidades con un índice promedio de 92.03 unidades sin entrenamiento, mientras que con entrenamiento el umbral superior fue de 69.8 unidades y el inferior de 58.0 unidades con índice promedio de 60.01 (figura 4.19).

Por último, en el usuario numero 3 (Omar) el umbral superior sin entrenamiento es de 115.0 unidades, inferior de 108.2 unidades e índice promedio de 110.01. Mientras que con entrenamiento el superior es de 46.1 unidades, inferior de 33.9 unidades con un índice promedio de 38.0 unidades. (Figura 4.21 y figura 4.20).

Con el análisis de las gráficas, concretamente, se busca encontrar el umbral de decisión adecuado para poder discriminar 2 opciones (Conocido y Desconocido). Para ello, es necesario identificar el umbral superior e inferior de cada usuario con entrenamiento y sin entrenamiento, para observar el rango de diferencia y escoger el valor más adecuado.

Se registraron y compararon los datos de cada gráfica dentro de la tabla 4.3.

Las medidas importantes para identificar el valor discriminante son: el umbral inferior sin entrenamiento y el umbral superior con entrenamiento ya que son los dos datos adyacentes, entre los cuales se encontrará el umbral de decisión. Para el primer caso con el usuario 1 se identifica que el umbral de decisión debe de estar entre 55.1 y 75.5 para escoger el valor en la condición *if-else*. En el caso del usuario 2 el umbral de decisión debe de estar entre 69.8 y 86.0, mientras que en el caso del usuario 3 el umbral de decisión va desde 46.1 hasta 108.2.

	Usuario 1 (Ivan)	**Usuario 2 (Luis)**	**Usuario 3 (Omar)**
Umbral superior sin entrenamiento	79.5	93.0	110.01
Umbral inferior sin entrenamiento	**75.5**	86	108.2
Umbral superior con entrenamiento	55.1	**69.8**	46.1
Umbral inferior con entrenamiento	52.03	58.0	33.9
Índice promedio con entrenamiento	53.4	60.01	38.0
Índice promedio sin entrenamiento	77.2	92.03	110.01

Tabla 4.3: Registro de resultados en unidades para el análisis del umbral de decisión.

Cada usuario tiene un rango propio en el que se puede escoger ese umbral de decisión, por lo cual es necesario comparar entre los tres usuarios y llegar al valor general adecuado.

Dentro de la tabla comparativa 4.3 se observa que el umbral superior más alto entre todos los usuarios con entrenamiento es de 69.8 unidades y el umbral inferior entre todos los usuarios más bajo de una identificación sin entrenamiento es de 75.5 unidades. Entre estos dos valores se ubica el umbral de decisión ideal, por lo cual se identifica que existe una diferencia solamente de 5.7 unidades. Para seleccionar el valor discriminante se calcula el punto medio de esta diferencia y se sumará al umbral superior con entrenamiento para así obtener el valor discriminante. Entonces,

5.7 unidades se dividirá entre dos obteniendo 2.85 unidades, se sumará al umbral superior con entrenamiento (69.8 unidades) dando un resultado de 72.65 unidades, el cual se redondeará a 72 unidades y ese será el valor discriminante a colocar dentro de la sentencia condicional *if-else*.

Con estas pruebas se llegó al resultado más eficiente, se analizaron los gráficos junto con las tablas y se tomó la mejor decisión para otorgar el rango de operación. Aun así, no queda exento de algún fallo en la toma de decisión, porque es un valor muy amplio. Cómo se sabe, el algoritmo es capaz de identificar y dar una respuesta a través de datos numéricos, la función aquí es dar un parámetro de funcionamiento, en dado caso que llegue un usuario nuevo en alguna condición diferente puede alterar ese valor. La función como programador es colocar un índice adecuado dependiendo de los datos que s estén observando, es decir, si se realizan cambios con respecto a lo entrenado puede que tenga que modificarse este parámetro, por ejemplo, factores como condiciones de iluminación, cantidad de usuarios registrados, enfoque, calidad y tipo de cámara, son algunos motivos para un diferente funcionamiento.

Como resumen de estas pruebas se llega a la conclusión de que algunos de los parámetros se tiene que modificar dependiendo de las condiciones en las que operará el sistema. Este proyecto planea ser instalado en un ambiente fijo dentro de un edificio, por lo cual la serie de pruebas se realizaron en un contexto lo mas parecido posible, aunque queda libre a cualquier tipo de ajuste y modificación.

En la tabla 4.4 se presenta un resumen de los parámetros con los que opera con máxima eficiencia el sistema de identificación.

Resumen de resultados	
ScaleFactor	Para el factor escala se identificó el parámetro de **1.2 unidades** con el cúal se obtuvo el mejor resultado al poder identificar rostros a diferentes tamaños y distancias.
MinNeighbors	Con el parámetro de cuadros vecinos, se llegó a la conclusión de **10 unidades**, debido a que es la sensibilidad adecuada para encontrar rostros dentro del frame.
Fotogramas (Número de fotogramas para entrenamiento)	Para un registro cómodo y eficiente en el entrenamiento, se decidió capturar **200 fotogramas** por usuario.
Result (Umbral de decisión)	Para el umbral de decisión, **72 unidades** fue la medida indicada para el rango de usuarios.

Tabla 4.4: Resumen de resultados, parámetros elegidos.

Conclusiones y trabajo futuro

Dentro de la creación de este proyecto, se observó el uso de diferentes tecnologías pertenecientes a la rama de sistemas digitales, se aplicaron varios fundamentos que nos brinda la carrera como el planteamiento correcto de un procedimiento experimental, la recopilación estructural de datos e información y el análisis adecuado de los resultados obtenidos. El trabajo basado en módulos fue un reto que se tuvo que afrontar para un desarrollo eficiente del proyecto, a pesar de ya tener un conocimiento previo de cómo funcionan y de las arquitecturas de los algoritmos, fue en algunos casos complicado, ya que hay muchas bibliotecas que nos ayudan a desarrollar el algoritmo avanzado y no desde cero. Un ejemplo de esto es la biblioteca de Open CV que nos brinda módulos pre entrenados con redes neuronales convolucionales o con formatos estadísticos para un análisis de imágenes y visión por computadora, gracias a esta biblioteca se ahorraron varias líneas de código programando un modelo desde cero, recopilando imágenes de entrenamiento positivas, negativas e imágenes para pruebas, lo cual en este caso fue mejor porque el objetivo del proyecto era darles una aplicación a la visión por computadora.

Este trabajo nos hizo desarrollar aún más la búsqueda de información, elegir correctamente las fuentes de consulta y poner a prueba de manera autodidacta estas nuevas herramientas para la inteligencia artificial, de programación y el uso de placass de desarrollo. La búsqueda dentro de los manuales y documentación fue clave para conocer y tener una introducción del funcionamiento de las herramientas. Estas diferentes herramientas se ocuparon a lo largo de los módulos establecidos, cada uno cumple una función específica, pero en la mayoría de los casos se complementaron con más de una herramienta. En el caso de los módulos 1, 2 y 3 tienen un tipo de

programación especializada en la inteligencia artificial con machine learning, pero también cuentan con una programación estándar para la creación de carpetas, tomas de decisión, contadores y demás; todo esto trabajado con el lenguaje Python con el editor de código de Visual Studio Code. El módulo 4 que fue el sistema físico se realizó con una programación estándar, lineal donde se configura para realizar una comunicación serial entre estos dos lenguajes que es C y Python, esto para la operación de la placa de desarrollo Arduino. El módulo 5 que fue la interfaz gráfica, se desarrolló dentro de Python con una programación lineal, con funciones y botones que le dan la estructura general al código. Por ultimo, en la parte del análisis de datos, se realizaron pruebas cumpliendo el objetivo que era utilizar el análisis de datos con las herramientas aprendidas de minería de datos. Este análisis se realizó en Jupyter Notebook que es un ambiente en el que se facilita mucho el trabajar en el análisis de datos, todo esto con el lenguaje Python.

Al termino de este proyecto nos damos cuenta que es un sistema que se puede aplicar en muchos ámbitos y no solo para el objetivo que se planteó. Como mencionó el asesor de proyecto, uno de los objetivos es tener un proyecto de código abierto en el que podamos apoyarnos entre alumnos y si en algún momento alguien necesita este proyecto para conocer esta tecnología o alimentar su propia idea pueda tomarlo y crear algo con un impacto mayor. Finalmente mi proyecto se presta a este fin, con algunas modificaciones dentro del código y una estructura especial podría ayudar a gran parte de la comunidad para la creación de sus propios proyectos.

Trabajo futuro

Como trabajo futuro de este proyecto se planea llevar a cabo la implementación en algún edificio escolar para salir de la fase del prototipo con esta simulación primitiva. Para ello, es importante realizar algunas modificaciones, por ejemplo, mejorar la base de datos con un registro de usuarios que tenga un diferenciador, es decir, agregar un *ID* único al nombre del usuario para evitar duplicar el registro. Este punto es importante, ya que dentro de la zona de implementación es posible que existan homónimos y corre

el riesgo de que se almacenen los fotogramas de un usuario dentro de la carpeta de otro con el mismo nombre.

En la parte de entrenamiento, se tiene como objetivo modificar la sección de registro de usuario para que cuando se realice el proceso, el algoritmo no borre los datos previos y parta desde cero, si no que actualice los datos nuevos adquiridos y los agregue al modelo de entrenado.

Implementar un mecanismo de motores y compuertas para llevarlo a la practica dentro de un edificio pequeño como primer paso. Con ese propósito, se necesita agregar un controlador extra para nivelar la velocidad de los motores y la fuerza con la que se abren y cierran las compuertas.

En el tema de redes, expandir el proyecto a diferentes puntos de acceso de vigilancia y tener el control de registro y supervisión en una misma computadora sería lo ideal, ya que el objetivo es cubrir todas las áreas restringidas del inmueble.

Estos son algunos puntos que son importantes cubrir si es que se quiere llevar a la expansión del proyecto. Son temas necesarios a desarrollar para el correcto funcionamiento de las demandas que exige el campo para poder sustituir las tarjetas magnéticas o teclados numéricos con *PIN* de acceso.

Anexos

Código general del sistema

```
from socket import timeout
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from sympy import li
from PIL import Image,ImageTk
import cv2
import numpy as np
import os
import imutils
from matplotlib.pyplot import gray
from itertools import count
import serial
import time

#----------Configurando la comunicacion serial con el arduino----------

ser=serial.Serial('COM3',9600,timeout=1)
time.sleep(2)


#----------Ventana principal----------
ventana = Tk()
ventana.title('MENU_PRINCIPAL')
ventana.geometry("1200x630")
frame=Frame(ventana)
frame.pack()

#----------Ventana de Registro----------
imagen2=Image.open("C:/Users/Pavilion/Documents/Ivan/Proyecto_SS/Imagenes_interfaz/
```

```
                                                    fondoRegistro.gif")
#es necesario colocar la ruta donde estan almacenadas las imagenes de la interfaz
imagenRegistro=ImageTk.PhotoImage(imagen2)

def registro():
    #----------Mensaje de alerta----------------
    def enviar():
        alertaEnviar=messagebox.askokcancel("ATENCION","Para registrar un usuario es
                                    necesario acceder a la camara. \n Estas de acuerdo?")
        if alertaEnviar==True:

            #----------Script de registro facial----------
            #--------Se ejecuta el script que crea una carpeta con tu nombre
            #--------y captura los rostros necesarios
            nombre=entryNombre.get()

            rutaBase = 'C:/Users/Pavilion/Documents/Ivan/Proyecto SS/
                                                    Base_de_datos_LBPH'
        #colocamos la direccion de la carpeta donde almacenaremos los fotogramas
            carpetaPersonal = rutaBase + '/' + nombre

            if not os.path.exists(carpetaPersonal):
                print('Carpeta creada: ',carpetaPersonal)
                os.makedirs(carpetaPersonal)

            camara= cv2.VideoCapture(0)
            modeloClasificador = cv2.CascadeClassifier(cv2.data.haarcascades+
                                        'haarcascade_frontalface_default.xml')
            count=0

            while True:
                (ret,ventana) = camara.read()

                if ret==False: break
                        #Si no obtenemos ninguna lectura de alguna camara por defaul,
                        #con esto no tendremos problemas o errores
                ventana = cv2.flip(ventana,1)
                ventana =  imutils.resize(ventana, width=640)
                        #redimencionamos la imagen de la ventana
                gray= cv2.cvtColor(ventana, cv2.COLOR_BGR2GRAY)
                        #la colocamos en escala de grises
                auximagen = ventana.copy()
                        #creamos una copia de la imagen del rostro

                caras = modeloClasificador.detectMultiScale(gray, 1.1, 15)
```

```
                    #aplicamos a gray el detector multicascada


                for(x,y,w,h) in caras:
                    cv2.rectangle(ventana, (x,y),(x+w, y+h), (255,0,0),2)
                        #marcamos los rostros con un cuadrado
                    rostro = auximagen[y:y+h,x:x+w]
                        #tomamos el puro rostro sin el cuadrado delimitador
                    rostro = cv2.resize(rostro,(150,150),interpolation=
                                                                cv2.INTER_CUBIC)
                        #redimensionamos el rostro a 150x150
                    cv2.imwrite(carpetaPersonal + '/rostro_{}.jpg'.format(count),
                                                                    rostro)
                        #nombramos ese fotograma con el nombre rostro y el numero
                        #del contador y lo agregamos en la carpeta del usuario
                    count = count + 1
                        #aumentamos el contador cada fotograma nuevo
                cv2.imshow('Capturando rostro', ventana)

                key= cv2.waitKey(10)
                if key == 27 or count >=100:
                    break
            camara.release()
            cv2.destroyAllWindows()

            #--------Menu de ventana de advertencia--------

            usuarioNuevo=messagebox.askyesnocancel("LISTO","Perfil almacenado. \n
----------------------------------------------------Registrar un usuario mas??")
            if usuarioNuevo==True:
                registro()
                ventanaRegistro.destroy()

            if usuarioNuevo==False:

                #----------Script de entrenamiento----------

                messagebox.showinfo("Proceso de entrenamiento",
                                                'Entrenando... dame un minuto...')

                rutaBase = 'C:/Users/Pavilion/Documents/Ivan/Proyecto SS/
----------------------------------------------------------------Base_de_datos_LBPH'
                        #Volvemos a colocar la direccion de la carpeta donde se
                        #almacenaron los fotogramas
                usRegistrados = os.listdir(rutaBase)
```

```
                etiquetas = []
                        #declaramos una variable donde se almacenaran las etiquetas
                        #dependiendo la persona
                datosRostros = []
                        #declaramos una variable donde se almacenaran las imagenes
                        #de los rostros
                contador = 0    #iniciamos un contador

                for nombreDir in usRegistrados:
                    carpetaPersonal = rutaBase + '/' + nombreDir
                    for fileName in os.listdir(carpetaPersonal):
                        etiquetas.append(contador)
                        datosRostros.append(cv2.imread(carpetaPersonal+'/'+
                                                                    fileName,0))
                        image = cv2.imread(carpetaPersonal+'/'+fileName,0)
                    contador = contador + 1

                ### comienzo del entrenamiento del modelo

                        #crear el algoritmo "LBPH"

                reconocedor_facial = cv2.face.LBPHFaceRecognizer_create()

                        #entrenando el reconocedor facial

                reconocedor_facial.train(datosRostros, np.array(etiquetas))

                        #almacenando el entrenamiento para no volver a entrenar cada
                        #vez que se ejecute.

                reconocedor_facial.write('C:/Users/Pavilion/Documents/Ivan/
____________________________________Proyecto_SS/entrenamiento_facial_LBPH2.xml')
                        #le colocamos la ruta donde queremos que se almacene modelo
                        #de entrenamiento
                messagebox.showinfo("Proceso_de_entrenamiento",'Modelo_almacenado
_____________________________________________________________exitosamente...')

            if usuarioNuevo==None:
                ventana.destroy()

        else:
            ventanaRegistro.destroy()
            ventana.destroy()
```

```
#----------Configuracion de la ventana de registro-------
    ventanaRegistro=Toplevel()
    ventanaRegistro.title("REGISTRO")
    ventanaRegistro.geometry("626x357")
    nombre=StringVar
    imagenFondoRegistro=Label(ventanaRegistro,image=imagenRegistro).grid
                                        (row=0,column=0,columnspan=2,rowspan=3)
    labelRegistro=Label(ventanaRegistro,text="Se registrar un nuevo usuario,
------------------------------------\n favor de llenar los campos solicitados",
                                        font=("System",17)).grid(row=0,column=0,
                                        padx=1,pady=1,columnspan=3)
    labelNombre=Label(ventanaRegistro,text="Nombre:",font=("System",15)).
                                                            grid(row=1,column=0)
    entryNombre=Entry(ventanaRegistro,textvariable=nombre)
    entryNombre.grid(row=1,column=1)
    botonEnviar=Button(ventanaRegistro,text="Enviar",font=("System",14),
                                                    width=10,height=2,command=enviar).
                                    grid(row=2,column=0,padx=1,pady=1,columnspan=2)
    ventanaRegistro.mainloop()

#----------Funcion de Script Identificar-----------

def identificar():

    messagebox.showinfo("Identificacion","Comenzar el proceso de identificacion \n
------------------------------------------------------Colocate frente a la camara")

        ## cargamos la direccion de la base de datos donde estan las imagenes para
        #tomar los nombres de los usuarios
    rutaBase = 'C:/Users/Pavilion/Documents/Ivan/Proyecto SS/Base_de_datos_LBPH'
    carpetas_us= os.listdir(rutaBase)

        #volvemos a crear el modelo
    reconocedor_facial= cv2.face.LBPHFaceRecognizer_create()
        #leemos el modelo de entrenamiento que creamos
    reconocedor_facial.read('entrenamiento_facial_LBPH.xml')
        #como el modelo ya esta dentro de la carpeta del script,
        #ya no es necesario colocar toda la ruta completa
        #sera un reconocimiento en streaming
    camara= cv2.VideoCapture(0)
        #cargamos el modelo para identificar rostros
    modeloClasificador = cv2.CascadeClassifier(cv2.data.haarcascades+
                                        'haarcascade_frontalface_default.xml')
```

```
    while True:
        (ret,ventana) = camara.read()
        if ret==False: break
                #Si no obtenemos ninguna lectura de alguna camara
                #por defaul, con esto no tendremos problemas o errores
        ventana = cv2.flip(ventana,1)
        gray= cv2.cvtColor(ventana, cv2.COLOR_BGR2GRAY)
                #la colocamos en escala de grises
        auximagen = gray.copy()
                #creamos una copia de la imagen del rostro
        caras = modeloClasificador.detectMultiScale(gray,1.1,15)
                #aplicamos a gray el detector multicascada

        for(x,y,w,h) in caras:
            rostro = auximagen[y:y+h,x:x+w]
                #tomamos el puro rostro sin el cuadrado delimitador
            rostro = cv2.resize(rostro,(150,150),interpolation=cv2.INTER_CUBIC)
                #redimensionamos el rostro a 150x150
            result = reconocedor_facial.predict(rostro)
            cv2.putText(ventana,'{}'.format(result),(x,y-3),1,1.3,(255,0,0),1,
                                                                    cv2.LINE_AA)
            if result[1] < 72:
                cv2.putText(ventana,'{}'.format(carpetas_us[result[0]]),(x,y-25),
                                                2,1.1,(0,255,0),1,cv2.LINE_AA)
                cv2.rectangle(ventana, (x,y),(x+w,y+h),(0,255,0),2)
                ser.write(b'A')
            else:
                cv2.putText(ventana,'Desconocido',(x,y-20),2,0.8,(0,0,255),1,
                                                                    cv2.LINE_AA)
                cv2.rectangle(ventana, (x,y),(x+w,y+h),(0,0,255),2)
                ser.write(b'C')


        cv2.imshow('Identificacion facial',ventana)
        key= cv2.waitKey(10)
        if key == 27:
            break
    camara.release()
    cv2.destroyAllWindows()
    ser.close()
    messagebox.showinfo("Adios","Fu un placer atenderte\n Hay nos vidrios!!!")
#----------Menu principal----------

imagen=PhotoImage(file="C:/Users/Pavilion/Documents/Ivan/Proyecto SS/
                                                    Imagenes_interfaz/fondo.gif")
```

```
        #Le cargamos la direccion de la imagen de fondo para la interfaz
imagenInicio=Label(frame,image=imagen).grid(row=0,column=0,columnspan=2,rowspan=4)
labelBienvenida=Label(frame,text="Bienvenido al sistema \n de seguridad facial",
                                                        font=("System",30)).
                                                        grid(row=0,column=0,
                                                        padx=1,pady=1,columnspan=2)

botonRegistro=Button(frame,text="Registrar Usuarios",font=("System",14),width=18,
                                                    height=3,command=registro).
                                                    grid(row=2,column=0,padx=1,pady=30)
botonIdentificar=Button(frame,text="Identificar Usuarios",font=("System",14),
                                                width=18,height=3,command=identificar).
                                                grid(row=2,column=1,padx=1,pady=30)



ventana.mainloop()
```

Script de comunicación serial en arduino para el control de LED´s

```
int led_red = 10;
int led_blue = 11;
int option;

void setup() {

  Serial.begin(9600);
  pinMode(led_red, OUTPUT);
  pinMode(led_blue, OUTPUT);
}

void loop() {
  if (Serial.available() > 0) {
    option = Serial.read();
    Serial.println(option);
    if (option == 'C') {
      digitalWrite(led_red, HIGH);
      digitalWrite(led_blue, LOW);
    }
    if (option == 'A') {
      digitalWrite(led_red, LOW);
```

```
        digitalWrite(led_blue, HIGH);
    }
  }
}
```

Glosario

AIoT Artificial Intelligence of Thing. XVIII

Algoritmo Procedimiento computacional bien definido que parte de un estado inicial y un valor o un conjunto de valores de entrada, a los cuales se les aplica una secuencia de pasos computacionales finitos, produciendo una salida o solución [69]. XIV

API (Interfaz de Programación de Aplicaciones) Conjunto de reglas y definiciones que permiten que diferentes software se comuniquen entre sí. XVI

Arduino Plataforma de desarrollo basada en una placa electrónica de hardware libre que incorpora un microcontrolador reprogramable y una serie de pines hembra. Estos permiten establecer conexiones entre el microcontrolador y los diferentes sensores y actuadores [70]. XII

Biblioteca Son conjuntos de archivos de código que se utilizan para desarrollar software. Su objetivo es facilitar la programación, al proporcionar funcionalidades comunes, que ya han sido resueltas previamente por otros programadores [71]. XII

Clustering Método donde se buscan patrones y similitudes en conjuntos de datos no etiquetados. XVIII

Dataframe Es una estructura de datos con dos dimensiones en la cual se puede guardar datos de distintos tipos (como caracteres, enteros, valores de punto flotante, factores y más) en columnas [72]. 65

Dataset Es un conjunto o colección de datos habitualmente tabulada. Corresponde a los contenidos de una única tabla de base de datos o una única matriz de datos de estadística, donde cada columna de la tabla representa una variable en particular, y cada fila representa a un miembro determinado del conjunto de datos tratado [73]. XVII

Deep Layers Son las capas profundas en una red neuronal, especialmente en modelos de aprendizaje profundo. Estas capas son responsables de aprender representaciones jerárquicas complejas de los datos, permitiendo una mayor capacidad de abstracción y comprensión de patrones.. 10

Feeds Secuencia de publicaciones o actualizaciones que aparecen en el perfil de un usuario en la plataforma de microblogging Twitter.. 11

HaarCascade Técnica de visión por computadora que contienen información sobre características específicas de objetos que son utilizadas para entrenar modelos de detección.. XV

JavaScript Lenguaje de programación utilizado para el desarrollo web. XVI

Jupyter Notebook Es una aplicación web de código abierto que permite crear y compartir código y documentos. Estos cuadernos (notebooks) permiten combinar texto y código, organizados en celdas [74]. 65

K-means Algoritmo de agrupamiento que clasifica un conjunto de datos en k grupos basándose en características similares. XVIII

LBPH (Local Binary Pattern Histograms) Algoritmo de descripción de texturas utilizado en visión por computadora y reconocimiento de patrones. Se emplea para analizar la textura de una imagen y es particularmente útil en tareas como reconocimiento facial. XV

MaixCube Dispositivo diseñado para ejecutar modelos de aprendizaje automático de manera eficiente. XIX

Matplotlib Biblioteca para la generación de visualizaciones estáticas, animadas e interactivas a partir de datos contenidos en listas o arrays en el lenguaje de programación Python [71]. 67

Metadatos Información adicional que describe y proporciona contexto sobre otros datos. 12

MobilNet Arquitectura de red neuronal diseñada específicamente para aplicaciones móviles y de bajo consumo. Se centra en la eficiencia computacional, lo que lo hace adecuado para implementaciones en dispositivos con recursos limitados. XVIII

MySQL Sistema de gestión de bases de datos relacional de código abierto. MySQL es ampliamente utilizado para almacenar y gestionar datos en aplicaciones web y es conocido por su rapidez y confiabilidad. XVI

Node.js Entorno de ejecución de JavaScript que permite la creación de aplicaciones web escalables y de alto rendimiento. Es ampliamente utilizado para desarrollar aplicaciones web en tiempo real. XVI

OpenCV Es una biblioteca de programación de código abierto dirigida principalmente a la visión por computador en tiempo real y es una herramienta multiplataforma [75]. Permite desarrollar proyectos como en *C, C++, Python* y otros lenguajes. Capaz de:

- Identificar objetos o caras (reconocimiento facial).
- Encontrar imágenes similares.
- Eliminar los ojos rojos de las fotografías.
- Reconocer escenarios.
- Seguir los movimientos de los ojos.
- Clasificar acciones humanas que estén en vídeos.
- Extraer modelos 3D.

- Útil en campos como la robótica y la realidad aumentada.

. XIV

Pandas Biblioteca de software escrita como extensión de *NumPy* para manipulación y análisis de datos para el lenguaje de programación *Python*. En particular, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales [76]. 67

Pooling Procesamiento de imágenes en la que se reduce la resolución de una imagen, preservando las características más relevantes. Empleado en redes neuronales convolucionales para reducir la complejidad computacional y extraer características importantes. XVIII

Python Es un lenguaje de programación de código abierto con estructuras de datos de alto nivel eficientes y un simple pero efectivo sistema de programación orientado a objetos. La elegante sintaxis de Python y su tipado dinámico, junto a su naturaleza interpretada lo convierten en un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en muchas áreas, para la mayoría de plataformas [77]. XII

PyTorch Ambiente para el desarrollo de aprendizaje profundo de código abierto desarrollado por Facebook. XVII

Raspberry Placa de desarrollo de bajo costo y tamaño compacto que integra todos los componentes esenciales de una computadora en una tarjeta única. XIX

Script Fragmentos de código usados para dar forma a herramientas. Se trata de una parte fundamental del software, constituye el código de una aplicación en su totalidad o una de sus funciones [78]. 21

TensorFlow Es un marco de código abierto desarrollado por Google para implementar y desplegar modelos de aprendizaje automático y redes neuronales.

Es utilizado para tareas como reconocimiento de imágenes, procesamiento de lenguaje natural y otros problemas de aprendizaje automático. XVII

TinyFaces Modelo relacionado con la detección facial que se centra en la identificación y seguimiento de caras en imágenes con alta eficiencia y precisión. XVI

Tkinter Es una biblioteca del lenguaje de programación Python y funciona para la creación y el desarrollo de aplicaciones de escritorio. Esta biblioteca facilita el posicionamiento y desarrollo de una interfaz gráfica de escritorio con Python [79]. XVI

Trading Es la actividad de comprar y vender instrumentos financieros. 8

YOLO (You Only Look Once) Es un algoritmo de detección de objetos en imágenes que permite localizar y clasificar múltiples objetos en una sola pasada. XVII

Bibliografía

[1] C. D. Jaramillo, "Utilización del sistema de reconocimiento facial para preservar la seguridad ciudadana," *El Criminalista Digital. Papeles de Criminología*, n.º 9, págs. 20-37, 2021.

[2] E. R. Caballero Barriga et al., "Aplicación práctica de la visión artificial para el reconocimiento de rostros en una imagen, utilizando redes neuronales y algoritmos de reconocimiento de objetos de la biblioteca opencv," 2017.

[3] I. A. G. Torres, X. T. Borja, J. D. De La Torre y W. N. Espín, "API para control de asistencia con reconocimiento facial usando OpenCv. JS," *Revista Tecnológica Ciencia y Educación Edwards Deming*, vol. 5, n.º 1, 2021.

[4] V. J. Valbuena Bracho, "Sistema de bajo coste para detectar personas con mascarilla y su temperatura a través de redes neuronales," 2021.

[5] H. E. Alarcón Castro, J. D. Hincapié Alzate et al., "Control de acceso e integración CCTV en línea en el Edificio JC, municipio de Fusagasugá.," 2017.

[6] M. Leyva-Vázquez y F. Smarandache, "Inteligencia Artificial: retos, perspectivas y papel de la Neutrosofía," *Infinite Study*, 2018.

[7] S. Badaró, L. J. Ibañez y M. J. Agüero, "Sistemas expertos: fundamentos, metodologías y aplicaciones," *Ciencia y tecnología*, n.º 13, págs. 349-364, 2013.

[8] EDSrobotics. "Visión por computador: que es, objetivos y aplicaciones." (31 enero, 2022), dirección: `https://www.edsrobotics.com/blog/vision-computador-que-es/`.

[9] J. L. Alonso Berrocal, J. J. Álvarez Navarrete, A. P. Bogram et al., "Avances en Informática y Automática. Séptimo Workshop," 2013.

[10] MathWorks. "¿Qué es una red neuronal? Tres cosas que es necesario saber." (2023), dirección: https://es.mathworks.com/discovery/neural-network.html?s_tid=srchtitle_red+neuronal_1.

[11] E. W. H. Villa, L. A. A. Marica, L. Q. Flores y J. M. Huayhua, "Clasificador de estrellas de Neutrones con una red neuronal multicapa utilizando R," *Innovación y Software*, vol. 2, n.º 1, págs. 33-42, 2021.

[12] J. S. G. Prieto, *Redes neuronales convolucionales y redes neuronales recurrentes en la transcripción automática*, 2019.

[13] IBM. "¿Qué es el aprendizaje supervisado?" (2023), dirección: https://www.ibm.com/mx-es/topics/supervised-learning#:~:text=Las%20redes%20neuronales%20aprenden%20esta,para%20obtener%20la%20respuesta%20correcta..

[14] F. J. G. Quesada, M. A. F. Graciani, M. T. L. Bonal y M. A. Díaz-Mata, "Aprendizaje con redes neuronales artificiales," *Ensayos: Revista de la Facultad de Educación de Albacete*, n.º 9, págs. 169-180, 1994.

[15] C. propia, *Ejemplo de cómo operan los filtros en una RNC*, 2023.

[16] S. I. Inc. "Reconocimiento de imágenes: Qué es y por qué." (2023), dirección: https://www.sas.com/es_es/insights/analytics/computer-vision.html.

[17] C. propia, *Procedimiento para el reconocimiento de imagenes*, 2023.

[18] I. MathWorks. "¿Qué es el reconocimiento de imágenes?" (2023), dirección: https://es.mathworks.com/discovery/image-recognition-matlab.html.

[19] MathWorks. "¿Qué es Machine Learning? Cómo funciona, por qué es importante y primeros pasos." (2023), dirección: https://es.mathworks.com/discovery/machine-learning.html?s_tid=srchtitle_machine+learning_1.

[20] O. tecnologico de Hidalgo. “Machine Learning: ¿qué es y cuál es su relación con la IA?” (2023), dirección: https://otech.uaeh.edu.mx/noti/index.php/machine-learning/machine-learning-que-es-y-cual-es-su-relacion-con-la-ia/.

[21] I. México. “¿Qué es deep learning?” (2023), dirección: https://www.ibm.com/mx-es/cloud/deep-learning/.

[22] IBM. “An introduction to deep learning.” (2023), dirección: https://developer.ibm.com/articles/an-introduction-to-deep-learning/.

[23] U. C. S. Pablo. “Big data: definición, tipos, características y beneficios.” (2023), dirección: https://postgrado.ucsp.edu.pe/articulos/que-es-big-data/.

[24] O. México. “¿Qué es big data?” (2023), dirección: https://www.oracle.com/mx/big-data/what-is-big-data/#:~:text=Definici%C3%B3n%20de%20big%20data,-%C2%BFQu%C3%A9%20es%2C%20exactamente&text=El%20t%C3%A9rmino%20%E2%80%9Cbig%20data%E2%80%9D%20abarca,como%20%E2%80%9Clas%20tres%20V%E2%80%9D..

[25] E. Á. J. Ambrogio, “Reconocimiento de objetos a través de la metodología Haar Cascades,” *ARTÍCULOS PRESENTADOS A RADI| TECNOLOGÍA DE LA INFORMACIÓN Y COMUNICACIÓN*, 2020.

[26] O. López Rincón et al., “Reconocimiento de gestos en tiempo real basado en heurísticas y reconocimiento de patrones,” *REPOSITORIO NACIONAL CONACYT*, 2015.

[27] C. propia, *Filtros digitales en cascada para centros y contornos*, 2023.

[28] C. propia, *Ejemplo de obtención de parámetros del algoritmo LBPH*, 2023.

[29] OpenCV. “OpenCV: Reconocimiento facial con Open CV.” (2023), dirección: https://docs.opencv.org/4.x/da/d60/tutorial_face_main.html#tutorial_face_lbph.

[30] O. CV. "Face Recognition with OpenCV." (2023), dirección: https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms.

[31] A. M. Turing, "Computing machinery and intelligence (1950)," *The Essential Turing: the Ideas That Gave Birth to the Computer Age*, págs. 433-464, 2012.

[32] J. McCarthy, Stanford, USA, 1956.

[33] S. C. Shapiro, "Encyclopedia of artificial intelligence second edition," *New Jersey: A Wiley Interscience Publication*, 1992.

[34] C. propia, *Ramas de la inteligencia artificial*, 2023.

[35] OpenCV. "Open Computer Vision Library." (2023), dirección: https://opencv.org/.

[36] O. for the KIPR Link. "opencv/data/haarcascade at master." (2023), dirección: https://github.com/kipr/opencv/tree/master/data/haarcascades.

[37] C. propia, *Diagrama del funcionamiento general*, 2023.

[38] C. propia, *Funcionamiento del módulo individual Registro facial"*, 2024.

[39] C. propia, *Funcionamiento del módulo individual .Entrenamiento"*, 2024.

[40] C. propia, *Funcionamiento del módulo individual Ïdentificacion facial"*, 2024.

[41] C. propia, *Diagrama de flujo del funcionamiento general del sistema*, 2023.

[42] C. propia, *Funcionamiento del módulo individual "Sistema fisico"*, 2024.

[43] C. propia, *Instrucciones para la creación de carpetas automáticas*, 2023.

[44] OpenCV. "OpenCV modules." (2023), dirección: https://docs.opencv.org/3.4/index.html.

[45] OpenCV. "OpenCV: Cascade clasifier." (2023), dirección: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html.

[46] N. Garzón Rodríguez et al., "REMI sistema de recomendación de contenido por medio de reconocimiento facial," 2019.

[47] C. propia, *Captura de fotografías*, 2023.

[48] C. propia, *Carpetas creadas*, 2023.

[49] C. propia, *Líneas de código para la configuración de fotogramas*, 2023.

[50] C. propia, *Visualización de los fotogramas almacenados con una etiqueta de identificación especial*, 2023.

[51] OMES. "OMES, visión por computador." (2023), dirección: `https://omes-va.com/`.

[52] C. propia, *Líneas de código para la configuración de las etiquetas identificadoras*, 2023.

[53] C. propia, *Líneas de código para el entrenamiento*, 2023.

[54] O. CV. "Diagrama de clases de detección facial dentro del módulo de FaceRecognizer." (2023), dirección: `https://docs.opencv.org/3.4/dd/d65/classcv_1_1face_1_1FaceRecognizer.html`.

[55] O. CV. "LBPHFaceRecognizer Class Reference." (2023), dirección: `https://docs.opencv.org/4.x/df/d25/classcv_1_1face_1_1LBPHFaceRecognizer.html`.

[56] C. propia, *Preparación para la identificación facial*, 2023.

[57] C. propia, *Configuración del frame para la identificación*, 2023.

[58] C. propia, *Parametros de decision conocido/desconocido*, 2023.

[59] C. propia, *Identificación persona conocida*, 2023.

[60] C. propia, *Identificación persona desconocida*, 2023.

[61] Python. "Interfaces gráficas de usuario con Tk." (2023), dirección: `https://docs.python.org/es/3/library/tk.html`.

[62] C. propia, *Ventana principal*, 2023.

[63] C. propia, *Ventana de registro de usuarios*, 2023.

[64] C. propia, *Ventana emergente de aviso al proceso de identificación facial*, 2023.

[65] C. propia, *Líneas de código para configurar la comunicación serial Python-Arduino.* 2023.

[66] pySerial. "Welcome to pySerial's documentation." (2020), dirección: https://pyserial.readthedocs.io/en/latest/index.html.

[67] C. propia, *Script para control de LEDs*, 2023.

[68] C. propia, *Diagrama del circuito*, 2023.

[69] U. de la Empresa. "¿Qué entendemos por algoritmo?" (2023), dirección: https://ude.edu.uy/que-son-algoritmos/.

[70] Arduino. "¿Qué es Arduino?" (2023), dirección: https://arduino.cl/que-es-arduino/..

[71] G. de España. "11 librerías para crear visualizaciones de datos." (2023), dirección: https://datos.gob.es/es/blog/11-librerias-para-crear-visualizaciones-de-datos.

[72] R. H. Paula Andrea Martinez Heladia Salgado. "Análisis y visualización de datos usando Python: Comenzando con datos." (2023), dirección: https://datacarpentry.org/python-ecology-lesson-es/02-starting-with-data.html.

[73] D. Formación. "Datasets y Dataframes en Big Data: ¿qué son?" (2023), dirección: https://www.deustoformacion.com/blog/programacion-tic/que-son-datasets-dataframes-big-data.

[74] F. Quiroga. "Tutorial de Python con Jupyter Notebook." (2023), dirección: http://facundoq.github.io/courses/aa2018/res/02_python.html.

[75] OpenCV. "OpenCV: Introduction." (2023), dirección: https://docs.opencv.org/4.2.0/d1/dfb/intro.html.

[76] U. D. ALCALÁ. “PANDAS: HERRAMIENTA BÁSICA PARA EL DATA SCIENCE EN PYTHON.” (2023), dirección: https://www.master-data-scientist.com/pandas-herramienta-data-science/#:~:text=%C2%BFQu%C3%A9%20es%20pandas%3F,tablas%20num%C3%A9ricas%20y%20series%20temporales..

[77] P. S. Foundation. “El tutorial de Python.” (2023), dirección: https://docs.python.org/es/3/tutorial/.

[78] Arimetrics. “Qué es un Script.” (2023), dirección: https://www.arimetrics.com/glosario-digital/script#:~:text=El%20script%20es%20un%20t%C3%A9rmino,concreto%20para%20herramientas%20en%20internet)..

[79] keepcoding. “¿Qué es Tkinter?” (2023), dirección: https://keepcoding.io/blog/que-es-tkinter/#:~:text=Tkinter%20es%20una%20librer%C3%ADa%20del,gr%C3%A1fica%20de%20escritorio%20con%20Python..