UNIVERSIDAD ESTATAL PENÍNSULA
DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

TÍTULO

PROTOTIPO DE SISTEMA INTELIGENTE PARA EL REGISTRO DE ASISTENCIA LABORAL POR MEDIO DE RECONOCIMIENTO FACIAL MEDIANTE REDES NEURONALES

AUTOR

Uquillas Daquilema, Alexis Manolo

TRABAJO DE TITULACIÓN

Previo a la obtención del grado académico en
MAGÍSTER EN ELECTRÓNICA Y AUTOMATIZACIÓN

TUTOR

Morales Escobar, Luis Alberto

Santa Elena, Ecuador

Año 2024

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

TRIBUNAL DE SUSTENTACIÓN

Firmado electrónicamente por:
ALICIA GERMANIA ANDRADE VERA

Ing. Alicia Andrade Vera, Mgtr.
COORDINADORA DEL PROGRAMA

Firmado electrónicamente por:
LUIS ALBERTO MORALES ESCOBAR

Ing. Luis Morales Escobar, Ph.D.
TUTOR

Firmado electrónicamente por:
BYRON XAVIER LIMA CEDILLO

Ing. Byron Lima Cedillo, Mgtr.
DOCENTE ESPECIALISTA

Firmado electrónicamente por:
JUNIOR RAFAEL FIGUEROA OLMEDO

Ing. Junior Figueroa Olmedo, Mgtr.
DOCENTE ESPECIALISTA

Firmado electrónicamente por:
MARIA MARGARITA RIVERA GONZALEZ

Abg. María Rivera González, Mgtr.
SECRETARIA GENERAL
UPSE

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

CERTIFICACIÓN

Certifico que luego de haber dirigido científica y técnicamente el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos, razón por el cual apruebo en todas sus partes el presente trabajo de titulación que fue realizado en su totalidad por Uquillas Daquilema Alexis Manolo, como requerimiento para la obtención del título de Magíster en Electrónica y Automatización.

TUTOR

Firmado electrónicamente por:
LUIS ALBERTO
MORALES ESCOBAR

Ing. Luis Alberto Morales Escobar, Ph. D.

Santa Elena, 15 de octubre de 2024

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

DECLARACIÓN DE RESPONSABILIDAD

Yo, **Uquillas Daquilema Alexis Manolo**

DECLARO QUE:

El trabajo de Titulación, Prototipo de Sistema Inteligente Para el Registro de Asistencia Laboral por Medio de Reconocimiento Facial Mediante Redes Neuronales previo a la obtención del título en Magíster en Electrónica y Automatización, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Santa Elena, 15 de octubre de 2024

EL AUTOR

Firmado electrónicamente por:
ALEXIS MANOLO
UQUILLAS DAQUILEMA

Alexis Manolo Uquillas Daquilema

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

CERTIFICACIÓN DE ANTIPLAGIO

Certifico que después de revisar el documento final del trabajo de titulación denominado, Prototipo de Sistema inteligente para el registro de asistencia laboral por medio de reconocimiento facial mediante Redes Neuronales, presentado por el estudiante, Uquillas Daquilema Alexis Manolo fue enviado al Sistema Antiplagio COMPILATIO, presentando un porcentaje de similitud correspondiente al 10%, por lo que se aprueba el trabajo para que continúe con el proceso de titulación.


CERTIFICADO DE ANÁLISIS
magister

ALEXIS_UQUILLAS_TT

10%
Textos sospechosos

4% Similitudes
0% similitudes entre comillas
0% entre las fuentes mencionadas
6% Idiomas no reconocidos

Nombre del documento: ALEXIS_UQUILLAS_TT.pdf
ID del documento: e322b41695f86b6d7b97277eb4cf91ba6c1e4b38
Tamaño del documento original: 2,92 MB
Autores: []

Depositante: LUIS ALBERTO MORALES ESCOBAR
Fecha de depósito: 15/10/2024
Tipo de carga: interface
fecha de fin de análisis: 15/10/2024

Número de palabras: 16.370
Número de caracteres: 122.040

TUTOR

Firmado electrónicamente por:
LUIS ALBERTO
MORALES ESCOBAR

Ing. Luis Alberto Morales Escobar, Ph. D.

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

AUTORIZACIÓN

Yo, **Uquillas Daquilema Alexis Manolo**

Autorizo a la Universidad Estatal Península de Santa Elena, para que haga de este trabajo de titulación o parte de él, un documento disponible para su lectura consulta y procesos de investigación, según las normas de la Institución.

Cedo los derechos en línea patrimoniales de mi proyecto de titulación con componentes de investigación aplicada y/o de desarrollo con fines de difusión pública, además apruebo la reproducción de este proyecto de titulación con componentes de investigación aplicada y/o de desarrollo dentro de las regulaciones de la Universidad, siempre y cuando esta reproducción no suponga una ganancia económica y se realice respetando mis derechos de autor.

Santa Elena, 15 de octubre de 2024

EL AUTOR

Firmado electrónicamente por:
ALEXIS MANOLO
UQUILLAS DAQUILEMA

Alexis Manolo Uquillas Daquilema

AGRADECIMIENTO

Agradezco a Dios porque durante todo este tiempo me ha dado salud y vida para culminar esta nueva etapa de profesionalismo.

A mis padres Manuel y Martha por confiar en mí y apoyarme en cada decisión de mi vida y siempre guiarme a ser una persona de bien.

A mi hermano Danny y su familia quien me aconseja y me guía por el buen camino y nunca rendirme hasta lograr mis objetivos.

A mi enamorada Blanca por su aliento y compañía que me han dado la fuerza necesaria para superar los obstáculos y seguir adelante.

A mi grupo de amigos por su ayuda y consejos en el camino del profesionalismo.

A mi tutor Ph. D. Luis Morales por sus enseñanzas, tiempo y dedicación a lo largo del proyecto de titulación.

Alexis Manolo, Uquillas Daquilema

DEDICATORIA

En mi primer lugar dedico este trabajo de titulación a Dios, quien fortalece mi espíritu y guía mi camino día tras día, por darme salud y su bendición para alcanzar todas las metas que me proponga a lo largo de mi vida. Segundo a mi familia quienes me apoyan incondicionalmente y forman un pilar muy importante en mi vida.

Por último, a mi hermana quien no se encuentra ya en este mundo, sin embargo, me brinda su apoyo desde donde quiera que esté.

Alexis Manolo, Uquillas Daquilema

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En esta propuesta se presenta el diseño e implementación de un de un prototipo que automatiza el control de asistencia laboral mediante tecnologías de reconocimiento facial a través de inteligencia artificial. El objetivo principal es llevar un control de ingreso del personal a la empresa y mejorar la eficiencia del registro de asistencia a un nivel digital, de esta manera superando las limitaciones de métodos tradicionales como el uso de formularios físicos. Para el desarrollo del sistema se ha usado el algoritmo de Redes Neuronales Convolucionales, el cual es un modelo de reconocimiento. Así mismo se ha utilizado el lenguaje de programación Python y su librería Face Recognition para la implementación. En conclusión, la implementación y puesta en marcha del sistema inteligente con reconocimiento facial y redes neuronales ha demostrado ser una solución efectiva para el control de registro de asistencia laboral reconociendo y verificando los rostros de los empleados que se encuentran almacenados en una base de datos.

Palabras claves: Reconocimiento Facial, Redes Neuronales, Python

ABSTRACT

This thesis presents the design and implementation of a prototype that automates work attendance control using facial recognition technologies through artificial intelligence. The main objective is to control the entry of personnel into the company and improve the efficiency of the attendance record at a digital level, thus overcoming the limitations of traditional methods such as the use of physical forms. For the development of the system, the Convolutional Neural Networks algorithm has been used, which is a recognition model. Likewise, the Python programming language and its Face Recognition library have been used for the implementation. In conclusion, the implementation and launch of the intelligent system with facial recognition and neural networks has proven to be an effective solution for controlling work attendance records by recognizing and verifying the faces of employees that are stored in a database.

Keywords: Facial Recognition, Neural Networks, Python

INTRODUCCIÓN

La incorporación de nuevas tecnologías en la automatización de procesos, como el control de asistencia mediante reconocimiento facial, ofrece a empresas, industrias y al sector educativo una mayor seguridad y control sobre el acceso del personal a sus instalaciones. Este enfoque permite agilizar el registro de asistencia, optimizando el tiempo disponible para llevar a cabo diversas actividades de manera más eficiente. (Bastidas, 2018)

El registro de acceso de personal mediante video puede implementarse utilizando reconocimiento facial, lo que puede mejorar la precisión y confiabilidad de estos sistemas. A diferencia de los métodos tradicionales, como los formularios físicos, que pueden ser alterados fácilmente y son susceptibles a la suplantación de identidad o inconsistencias, las tecnologías basadas en biometría aseguran la autenticidad de la información. Tecnologías basadas en la inteligencia artificial, emplean los rasgos físicos únicos de cada persona, garantizando un reconocimiento más seguro y eficiente. . (Caro & López, 2018)

Actualmente, la empresa SUMELTRONIC S.A.S. no posee un sistema de control inteligente para el registro de asistencia del personal laboral, el cual permita verificar el ingreso de cada trabajador. En el país, un gran porcentaje de las empresas emplean la forma tradicional de registro de asistencia por medio de reconocimiento de huella dactilar o por escrito A través del sistema propuesto el cual utiliza redes neuronales, se busca evitar fraudes de identificación de personal o tener la certeza que el personal está presente en el lugar de trabajo.

La verificación de la asistencia laboral tradicionalmente se realiza en hojas físicas que son archivadas, lo cual presenta limitaciones en cuanto a eficiencia y seguridad. Este proyecto propone mejorar el proceso de registro de asistencia mediante una solución digital basada en reconocimiento facial y algoritmos basados en redes neuronales. Este sistema no solo registra la asistencia de los trabajadores de manera automatizada, sino que también permite un control y verificación diario del personal. La implementación de estas nuevas tecnologías en la automatización del control de asistencia inteligente proporciona a la empresa una mayor seguridad y un control más efectivo del personal. Además, optimizará el tiempo destinado al registro de asistencia, permitiendo a los trabajadores cumplir con sus actividades de manera puntual y aprovechando de mejor forma el tiempo del personal encargado del sistema de control laboral. Para cumplir los

objetivos planteados, este trabajo se ha dividido en capítulos que presentan la información a detalle del sistema propuesto.

El Capítulo I presenta el marco teórico referencial, partiendo de la inquietud sobre la viabilidad de realizar un registro de asistencia utilizando redes neuronales. A partir de esta cuestión, se formulan una serie de preguntas que este trabajo de investigación busca responder. El desarrollo de la investigación ha permitido alcanzar los objetivos propuestos, logrando así ofrecer una solución al problema planteado.

El Capítulo II hace referencia a la metodología a ser empleada en el trabajo desarrollado considerando los componentes de hardware y software necesario para obtener un prototipo funcional que cumpla con los requerimientos del sistema de reconocimiento automático para registro de asistencia laboral.

En el Capítulo III se presenta la recopilación, discusión y análisis de los resultados obtenidos, con el fin de extraer conclusiones sobre el proyecto realizado. De este modo, se espera que este trabajo ofrezca un aporte significativo al campo de la investigación y sirva como base para la resolución de problemas tanto en el presente como en el futuro.

Formulación del problema de investigación

¿Cuál es el beneficio que se obtiene al desarrollar un registro automático de asistencia laboral por medio de reconocimiento facial utilizando inteligencia artificial?

Objetivo General:

Desarrollar un sistema inteligente que realice el registro de asistencia laboral a un grupo de personas utilizando reconocimiento de rostros mediante métodos de inteligencia artificial.

Objetivos Específicos:

•Implementar un prototipo para el control de registro laboral, del personal de la Empresa "Sumeltronic S.A.S." con una base de datos que me permita la elaboración de reportes del sistema implementado.

•Determinar las características del proceso de reconocimiento facial en sistemas que utilicen Inteligencia Artificial.

•Implementar un algoritmo de reconocimiento facial con inteligencia artificial para poder identificar diferentes rostros de individuos para el registro de asistencia laboral.

Planteamiento hipotético

¿Es factible desarrollar un sistema que permita hacer el registro de asistencia laboral por medio de reconocimiento facial utilizando inteligencia artificial?

¿Cuáles son las técnicas más relevantes utilizadas para el reconocimiento facial?

CAPÍTULO 1. MARCO TEÓRICO REFERENCIAL

En los últimos años, se ha explorado ampliamente la implementación del reconocimiento facial mediante procesos computacionales, cuyo funcionamiento se basa en algoritmos de inteligencia artificial. Este desarrollo involucra diversos métodos, tanto para el procesamiento de las imágenes como para el entrenamiento del modelo, lo que permite mejorar la precisión y eficiencia de la tecnología.

1.1. Revisión de literatura

Rodeados por una sociedad en continuo cambio y con un afán de desarrollo tecnológico sin límites, la inteligencia artificial se posiciona como una de las áreas que mayor repercusión está teniendo actualmente (Costa, 2020). Dentro de este contexto, se encuentra el reconocimiento facial, una herramienta cada vez más usada en cualquier ámbito o en cualquier situación; desde el desbloqueo de dispositivos móviles, pasando por cientos de aplicaciones, hasta su uso como medida de seguridad tomada por algunos gobiernos. Este proyecto se centra en el análisis, tanto teórico como experimental, de un sistema de reconocimiento facial utilizando una base de datos. La implementación del código se llevó a cabo en Python 3.6, empleando la librería OpenCV para machine learning y visión por computador. El proceso de identificación de un rostro se divide en dos subprocesos consecutivos. Por lo que respecta al primer subproceso, es decir, a la detección facial, el método empleado se centra en el algoritmo de Viola-Jones basado en Haar Cascades. En cuanto al segundo subproceso, el reconocimiento facial, los métodos utilizados en este proyecto son Fisherfaces, Eigenfaces y LBPH (Local Binary Pattern Histogram), de los cuales se ha implementado su código y se ha realizado un estudio detallado.

El proyecto de (Vizuete, 2020), se centra en el desarrollo de un sistema de reconocimiento facial utilizando técnicas de Deep Learning y la biblioteca OpenCV. A pesar de que el reconocimiento facial ha existido durante décadas, su rápida evolución en los últimos años ha permitido su integración en diversas aplicaciones innovadoras, como autenticación en dispositivos móviles y aplicaciones de fotos personales. El sistema desarrollado permite gestionar usuarios registrados, ofreciendo funcionalidades para añadir o eliminar información, así como la detección de personas con vida frente a objetos inanimados. Además, se contempla tres modos de uso: usuario, administrador y

desarrollador. Finalmente, se realizan pruebas de rendimiento para evaluar los tiempos de ejecución y la eficacia en la detección e identificación de caras, analizando los resultados obtenidos.

El trabajo realizado por (Chen, 2020), en su investigación resaltan las redes neuronales artificiales que se forman por unidades, también llamados nodos o neuronas artificiales, que combinan múltiples entradas para producir una sola salida. En cada nodo se realiza una suma de sus entradas, donde cada una tienen un parámetro de peso que las multiplica. El resultado de la suma, antes de pasarse al siguiente nodo, se pasa por una función de activación, cuyo objetivo es traer no linealidad a la predicción, ya que muchos problemas del mundo real no se pueden modelar sólo con funciones lineales. En los últimos años, la autenticación de personas ha cobrado gran relevancia debido a los avances tecnológicos y las investigaciones que se han llevado a cabo en este campo. En este proceso, se emplean técnicas de visión por computadora que permiten analizar imágenes o videos con el objetivo de verificar la identidad de un individuo. Estos métodos han demostrado ser efectivos en diversas aplicaciones, gracias a su capacidad para procesar información visual de manera rápida y precisa.

En (Arroyo, Pineda, & López, 2021), se analizan trabajos relacionados con el proceso de autenticación de personas, haciendo un análisis profundo en los trabajos basados en Máquina de Vectores de Soporte (Support Vector Machines). De igual manera, se explican a grandes rasgos las diferentes etapas que conforman el proceso de autenticación de personas. Finalmente, se presenta un conjunto de experimentos realizados, utilizando una combinación de características basadas en color, textura y simetría, mientras que, para la etapa de clasificación se utiliza SVM. Esta combinación de características aunada con el clasificador, muestra ser una alternativa para la autenticación de personas.

En (Gómez J., 2021), presenta los estudios psicométricos permiten realizar mediciones basadas en correlaciones de vectores que producen predicciones de rasgos del comportamiento humano. Con la creciente digitalización, la psicometría es usada para diseñar perfiles de los individuos a través de diversos mecanismos: big data, machine mearning, entre otros. Lejos de ser neutrales, estos perfiles están basados en metodologías que producen sesgos que afectan a los individuos perfilados. Así, este artículo muestra cómo la psicometría elabora un perfil-dividuo que genera una reducción de los individuos

y produce formas en las que estos tienen que comportarse. Para ello, se ejemplifica con un caso de medición psicométrica basada en el reconocimiento facial.

El trabajo realizado por (Aznarte, 2022), Demuestra como las tecnologías de identificación biométrica han experimentado un gran desarrollo en los últimos años, siendo aplicadas a decenas de ámbitos diferentes, entre los que se encuentra el ámbito educativo, en particular el universitario. Sin embargo, en este artículo argumentamos que dicha tendencia puede impactar de formas inesperadas a los procesos de enseñanza/aprendizaje. Así, se exponen algunas consideraciones principales acerca del uso de tecnologías de identificación biométrica en general y más particularmente de técnicas de reconocimiento facial en el marco de los exámenes universitarios, prestando especial atención al problema de realizar los exámenes presenciales por medios remotos durante la pandemia de la COVID-19.

En (Navarro A., 2022), se presenta una panorámica de las propuestas de reconocimiento facial utilizando un sensor Kinect para adquirir las imágenes, El reconocimiento facial en una imagen o video, es un tema que ha sido investigado y desarrollado en los últimos años. Sin embargo, continúa siendo una tarea abierta con grandes complejidades, desde el punto de vista científico y económico Se plantean tres enfoques, los que se basan exclusivamente en imágenes RGB, de profundidad, o la fusión de ambas modalidades.

En (F. Trujillo, 2022), se presenta la implementación de un sistema de reconocimiento de objetos, el cual utiliza solo cuatro vistas de cada objeto para entrenar a la red neuronal de Mapas Auto organizados. En específico, lo que se propone es la extracción y generación de clústeres a partir de la distribución generada por el descriptor de los objetos utilizado. Una vez generados, los clústeres son etiquetados como pertenecientes a una misma clase o tipo de objeto u objetos si fuera el caso. Se propone implementar un sistema de reconocimiento de objetos utilizando para ello una red neuronal auto asociativa como lo es la red de Kohonen. La entrada para esta red neuronal será el vector de características morfológicas basado en el histograma de gradientes orientados. Este será el descriptor de los objetos que se encuentran en la base de imágenes que se va a utilizar para entrenar, probar y validar el algoritmo a implementar.

El trabajo realizado por (P. Canazas, 2022), tiene como principal objetivo el desarrollo de un sistema que permita identificar las emociones de una persona mediante el

reconocimiento de rostros utilizando inteligencia artificial. Para el desarrollo del sistema se tuvo como base el algoritmo básico de Eigenfaces o Análisis de Componente Principal, el cual es uno de los modelos de reconocimiento de rostros más utilizado. Así mismo fue utilizado el lenguaje Python y algunas de sus librerías disponibles como Numpy, OpenCV y Sklearn para la implementación.

En (Centurión & Almeida, 2022), se propone el uso de un sistema biométrico de reconocimiento facial. La ventaja de este tipo de sistema de reconocimiento sobre otros sistemas biométricos como el reconocimiento de huella dactilar, se encuentra en la reducción del tiempo de identificación de las personas activas en el entorno. Para ello se aplican técnicas de inteligencia artificial y visión por computadora en una placa Raspberry Pi 3. El prototipo consta de una cámara para la captura del rostro a identificar y la placa Raspberry pi, la cual es controlada de forma remota por otra computadora. Para la detección de rostros se utiliza un clasificador tipo cascada. Una vez detectado el rostro, se procede con la captura y el pre-procesamiento de la imagen facial del individuo para que posteriormente sirva de entrada para el entrenamiento de la red neuronal artificial. El tipo de red neuronal artificial que se utiliza es la red Neuronal Convolucional (RNC), una red especialmente diseñada para trabajar en la clasificación de imágenes aplicando la técnica del aprendizaje profundo. Los resultados obtenidos de las pruebas demuestran que el sistema biométrico de reconocimiento facial puede reconocer exitosamente a los sujetos con una tasa de precisión bastante significativa y confiable.

(H. Vega, 2023), Desarrollaron un control de acceso a áreas restringidas enfrenta problemas con sus sistemas de verificación de identidad para los directores ejecutivos (CEOs) basados en tarjetas de acceso, vulnerables a la clonación, suplantación de identidad y ataques cibernéticos como el Whaling. Para darles solución, se desarrolló un sistema de reconocimiento facial basado en el aprendizaje por transferencia con el uso de la librería face-recognition. La metodología para la implementación del sistema de reconocimiento facial fue basada en Scrum y Kanban y se usó como lenguaje de programación Python. Como resultado, el sistema logra extraer las características principales del rostro y las compara con las almacenadas en una base de datos, así como notificar los intentos fallidos de personal que no está autorizado.

(R. Campos, 2023) desarrollaron un sistema de reconocimiento facial para el control de accesos mediante Inteligencia Artificial. El sistema se basa en el uso de Redes Neuronales Convolucionales (CNN), un modelo efectivo para el reconocimiento facial. Para su implementación, se empleó el lenguaje de programación Python junto con las librerías Numpy, Os, OpenCV e Imutils. Los resultados obtenidos, utilizando un dataset de 450 imágenes por individuo, alcanzaron una precisión de aproximadamente el 88% en la predicción por persona. Esto sugiere que el sistema de reconocimiento es eficaz y su eficiencia aumenta al incrementar el tamaño de los datasets utilizados.

Finalmente, (S. Altamirano, 2024), presenta el desarrollo de un sistema de control de acceso automático para el personal de la Dirección de Tecnologías de la Información y Comunicación (DTIC). Su enfoque ha sido usar inteligencia artificial para crear un sistema biométrico que registre la entrada y salida del personal a la bodega del departamento de redes de la DTIC, proporcionando así una capa de seguridad a los equipos de alto valor económico que se guardan en sus instalaciones.

1.2. Desarrollo teórico y conceptual

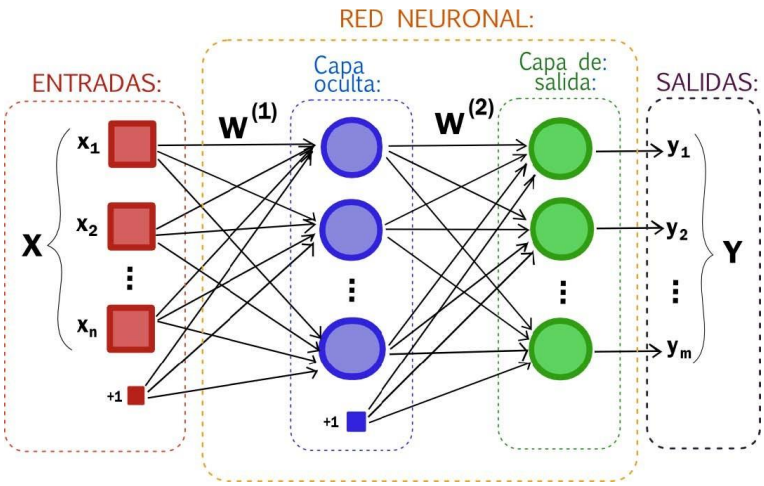
1.2.1 Inteligencia Artificial

La inteligencia artificial es un desarrollo científico por el cual una máquina es diseñada para llevar a cabo acciones similares a las de la inteligencia humana. El principal objetivo de esta ciencia es hacer que los ordenadores hagan aquello que los humanos son capaces de hacer. Según su aplicación, se puede dividir la tecnología en dos tipos. Por un lado, aquellas máquinas diseñadas para especializarse en una tarea específica se les denomina IA aplicada o débil, y es la más común. Un ejemplo de ello sería un módulo de ajedrez o un coche autónomo. Por otro lado, si la inteligencia tiene la capacidad de realizar cualquier acción que se le asigne se denomina IA generalizada o robusta. Hoy en día, existen algoritmos de IA que cumplen estas condiciones. (Bedoya, 2022)

1.2.2 Redes Neuronales

Figura 1

Red Neuronal



Nota. Entrenamiento de la red neuronal, por Marcelo Cajas,2018, researchgate (https://n9.cl/umni8)

Una red neuronal artificial es un modelo matemático computacional que trata de emular el comportamiento de las neuronas del cerebro humano desarrollando un aprendizaje mediante experiencias. Está compuesta de unidades procesadoras interconectas unas con otras, capaz de reconocer patrones, clasificar datos y pronosticar eventos futuros con precisión y exactitud. En la Figura 1 se presenta la arquitectura de una red neuronal con una capa de entrada, dos capas ocultas y una capa de salida. La estructura de una red neuronal, consta de cuatro elementos fundamentales refieren que la estructura interna de la red está determinada por los pesos o sinapsis, estas conexiones pueden ser excitadoras o inhibidoras; un sumador, encargado de ponderar la suma de todas las entradas multiplicadas por las sinapsis; una función de activación no lineal y un umbral de exterior. (Fernando Alain Incio Flores, 2023)

Red Neuronal Convolucional

Una red neuronal convolucional, también conocida como CNN (convolutional neural network) o ConvNet, es una clase de red neuronal diseñada para procesar datos utilizando un sistema que imita al ojo humano.

Son las más empleadas para el reconocimiento de imágenes (debido a sus características), pero también pueden usarse para clasificar datos procedentes de audio, señales. Estas redes contienen una serie de píxeles dispuestos en forma de cuadrícula, cada uno con unos valores que indican el brillo y el color de cada punto de la imagen (píxel).

Las redes neuronales convolucionales están detrás de múltiples aplicaciones con las que interactuamos diariamente y forman parte de la inteligencia artificial (IA), como es el caso del reconocimiento de personas en redes sociales para etiquetarlas, en cámaras de seguridad o de la matrícula de los vehículos para acceder a un parking. (Unir, 2024)

Convolución

La mayoría de los cálculos se realizan en esta capa, que es el componente central de una CNN. Una segunda capa convolucional puede seguir a la capa convolucional inicial. El proceso de convolución implica que un núcleo o filtro (kernel) dentro de esta capa se mueve a través de los campos receptivos de la imagen, verificando si una característica está presente en la imagen.

Durante múltiples iteraciones, el kernel recorre toda la imagen. Después de cada iteración, se calcula un producto escalar entre los píxeles de entrada y el filtro. El resultado final de la serie de puntos se conoce como mapa de características o característica convolucionada. En última instancia, la imagen se convierte en valores numéricos en esta capa, lo que permite a la CNN interpretar la imagen y extraer patrones relevantes de ella.

Reducción o muestreo

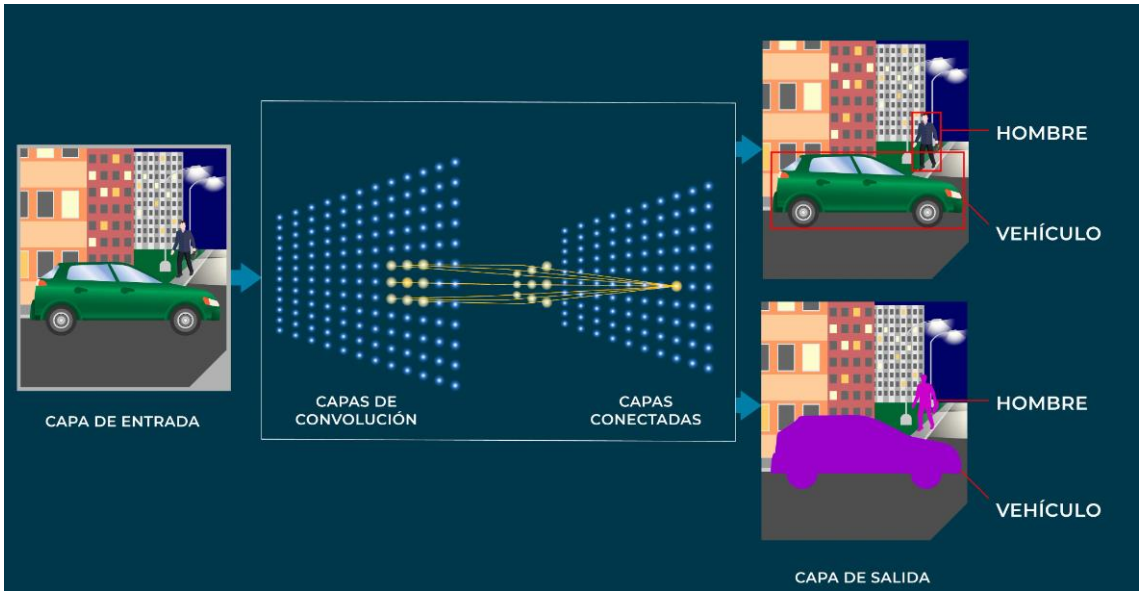
Al igual que la capa convolucional, está también barre un núcleo o filtro a través de la imagen de entrada, pero se diferencia en que reduce la cantidad de parámetros en la entrada y también cierta información. Como dato positivo señalar que esta capa reduce la complejidad y mejora la eficiencia de la CNN.

Completamente conectada

En esta capa es donde ocurre la clasificación de imágenes en función de las características extraídas en las anteriores. Aquí, completamente conectado significa que todas las entradas o nodos de una capa están conectados a cada unidad de activación o nodo de la siguiente capa, finalmente mostrando la salida reconocimientos los objetos previamente entrenados, como se observa en la figura 2. (Unir, 2024)

Figura 2

Proceso clasificatorio de una Red neuronal Convolucional



Nota. Proceso clasificatorio de una Red Neuronal Convolucional, Algotive Partner ,2022, IA (http://surl.li/vhdcat)

1.2.3 Reconocimiento Facial

El reconocimiento facial es una técnica que se enmarca en el campo de la biometría. La biometría se define como el "estudio mensurativo o estadístico de los fenómenos o procesos biológicos". Para que la biometría sea efectiva, es necesario que se sustente en tres elementos fundamentales:

a) Universalidad. - todos los individuos somos portadores de determinadas características aptas para ser medidas y cuantificadas.

b) Singularidad. - esas características que todo sujeto posee son muchas veces dispares y sirven para resaltar la individualización de cada uno frente al resto.

c) Permanencia. - pues se trata de elementos inalterables que perduran en el tiempo. Las técnicas de reconocimiento más conocidas son seguramente el análisis de las huellas dactilares o de los rasgos faciales de un individuo; no obstante, en realidad la biometría comprende técnicas muy variadas. Se distingue tres clases de datos biométricos: En primer lugar, pueden mencionarse los datos llamados fisiológicos, que se refieren a características físicas y fisiológicas de la persona; los datos fisiológicos más frecuentemente utilizados son la huella dactilar, el iris, la geometría de la mano, la retina, los vasos sanguíneos en determinadas partes del cuerpo, la voz, el sudor, las orejas y el

ADN. En segundo lugar, con una categoría conformada por aquellos datos biométricos relacionados con el comportamiento de la persona: con sus actuaciones o con la forma en que realizan ciertas conductas; entre éstos destacan la escritura de un sujeto, su ritmo cardíaco, ritmo respiratorio, la firma, la manera en que utiliza un teclado, la forma de conducir, la forma de andar o de moverse, y la marcha. identifica un tercer tipo de controles biométricos, al que este organismo califica como emergente, y que serían los de corte psicológico: la forma de reaccionar de la persona ante ciertas situaciones o pruebas, que pueden dar lugar a un perfil psicológico de ésta. (Montero, 2022)

Face Recognition

Face Recognition es una biblioteca diseñada para el reconocimiento facial, destacada por ser una de las más sencillas de utilizar para el reconocimiento y manipulación de rostros, tanto desde Python como desde la línea de comandos. Esta herramienta se basa en la tecnología de reconocimiento facial de última generación proporcionada por Dlib, la cual ha sido desarrollada utilizando técnicas de aprendizaje profundo. Según las pruebas realizadas por los desarrolladores de la biblioteca, el modelo de Face Recognition alcanza una precisión del 99,38 %, lo que la posiciona como una opción altamente confiable para aplicaciones de reconocimiento facial.

Detección facial

Face Recognition permite la detección de rostros mediante su función «face_locations» que devuelve una lista de cuatro tuplas (x, y) por cada cara detectada con las coordenadas de las esquinas de los marcos que enmarcan las caras detectadas. El primer parámetro de esta función es la imagen como matriz numpy; el segundo, un número para indicar el número de veces que se va a aumentar la imagen para tratar de encontrar caras (valor uno por defecto) y el tercero es el modelo que se va a usar para detectar caras: «hog» (por defecto) o «cnn». El modelo CNN es un modelo de aprendizaje profundo más preciso, acelerado por GPU (Graphics Processing Unit). Este modelo es importado de la biblioteca Dlib.

Posicionamiento y proyección de caras

Esta biblioteca también tiene una función llamada «face_landmarks» que devuelve una lista de diccionarios Python con las localizaciones (puntos de referencia faciales) de las características faciales (nariz, ojos, etc.) para cada cara de la imagen. El tercer parámetro de esta función («model») puede ser «large» (por defecto) o «small». La opción «large» devuelve las coordenadas de los 68 puntos de referencia faciales (landmarks) calculados por un predictor de Dlib mientras que la opción «small» solamente devuelve cinco, pero es más rápido. Esta función puede usarse para cambiar de posición y proyectar caras haciendo transformaciones como, por ejemplo, rotar o escalar la cara, para así poder centrar (alinear, normalizar) e identificar un rostro.

Codificación facial

Antes de identificar un rostro, es fundamental obtener una representación cuantitativa o conjunto de características de este, así como de otros rostros, para realizar comparaciones o extraer rasgos distintivos. Face Recognition cuenta con una función denominada "face_encodings", la cual utiliza un modelo previamente entrenado. Este modelo, basado en una red neuronal que ha aprendido a generar de manera precisa las representaciones faciales, emplea puntos de referencia faciales (face landmarks) para devolver una lista de codificaciones de 128 dimensiones. Estas codificaciones representan las características únicas de cada rostro en la imagen, proporcionando una lista por cada cara detectada.

Identificación facial

Para la identificación y verificación facial en Face Recognition se usa la función «compare_faces», que compara una lista de caras codificadas con una cara codificada para ver si hay coincidencias (devuelve una lista de valores True o False según haya coincidencias o no, respectivamente) y también se usa la función «face_distance» dada una lista de codificaciones de caras, las compara con una codificación de una cara conocida y obtiene la distancia euclidiana para cada cara de comparación. La distancia indica cuan similares son las caras.

(Vizuete, 2020)

1.2.4 Raspberry Pi

Figura 3

Tarjeta Electrónica Raspberry Pi 4 Tipo B

La Raspberry Pi 4 modelo b mostrada en la figura 3, es un ordenador de bajo costo y de tamaño compacto destinado al desarrollado para hacer accesible la informática a todos los usuarios también se caracteriza por ser muy utilizada para desarrollar pequeños prototipos y para la formación sobre informática y electrónica. Puede ser conectada a un monitor de computador o un TV, y usarse con un mouse y teclado estándar. La Raspberry pi cuenta con un sistema en un chip o SoC (System On a Chip) de Broadcom que incorpora una unidad de procesamiento central o CPU compatible con ARM y una unidad de procesamiento gráfico o GPU. (Bautista, 2022)

Tabla 1

Características del Raspberry PI 4 Tipo B

Características Raspberry PI 4 Tipo B	
Sistema:	Broadcom BCM2711
CPU:	Procesador de cuatro núcleos a 1,5 GHz
GPU:	Video Core VI
Memoria:	4GB LPDDR4 RAM
Conectividad:	802.11ac Wi-Fi / Bluetooth 5.0, Gigabit Ethernet
Puertos:	2 x USB 3.0, 2 x USB 2.0

Alimentación:	5V/3A vía USB-C, 5V vía cabezal GPIO
Expansión:	Cabezal GPIO de 40 pines

1.2.5 Cámara Web

Figura 4

Cámara Web

Los sistemas de visión son ampliamente utilizados en la actualidad en el control de la calidad de procesos industriales, en sistemas de vigilancia e identificación de objetos móviles, en la robótica, etc. Uno de los tipos de cámaras que pueden usarse para obtener información visual son las cámaras web USB (webcam), como se observa en la figura 4. Sin embargo, su uso dentro del mundo del diseño electrónico se ha visto frenado por la diversidad de controladores existentes entre fabricantes y por solo encontrarse estos desarrollados para los principales sistemas operativos. (Delgado, 2020)

1.2.6 Lenguaje de Programación

Python cuenta con facilidades para la programación orientada a objetos, imperativa y funcional, por lo que se considera un lenguaje multi-paradigmas. Es un lenguaje de alto nivel ya que contiene implícitas algunas estructuras de datos , que permiten realizar tareas complejas en pocas líneas de código y de manera legible. Python, a diferencia de otros lenguajes interpretados, ha implementado toda su librería estándar en el lenguaje C, lo que hace que sus funciones primitivas sean bastante eficientes, lo que optimiza aún más el proceso de interpretación. Una de las fortalezas de Python, y quizás la mayor, es la librería estándar que cuenta con decenas de módulos que cubre la mayoría de las necesidades básicas de un programador. (C. Pérez, 2014)

CAPÍTULO 2. METODOLOGÍA

2.1. Contexto de la investigación

El proyecto se ha desarrollado en la Empresa Sumeltronic S.A.S. ubicado en el Local Nro 1, Urdenor 1 - Mz 112 - Solar 16, que se encuentra ubicada en la ciudad de Guayaquil, donde se realiza proyectos de ingeniería para las industrias y venta de suministros eléctricos y electrónicos.

2.2. Diseño y alcance de la investigación

Es fundamental reforzar y ampliar los conocimientos en inteligencia artificial mediante un enfoque teórico-práctico. Esta investigación experimental, representada a través del prototipo presentado, permite complementar el conocimiento en redes neuronales y reconocimiento facial, brindando así una solución moderna y práctica para el registro de asistencia de personal en la empresa Sumeltronic S.A.S. Adicionalmente, se espera que esta investigación exploratoria brinde una visión general sobre el problema. El propósito principal es explorar y generar ideas que permitan identificar patrones preliminares y formular hipótesis para futuras investigaciones más detalladas y profundas en el campo de aplicaciones de visión por computador e inteligencia artificial.

2.3. Tipo y métodos de investigación

El objetivo principal de esta investigación es el estudio de las redes neuronales y su proceso de entrenamiento. Se puede deducir que la naturaleza de este trabajo es cualitativa, ya que se enfoca en la relación entre variables y patrones dentro de un conjunto de datos. El análisis se centrará en interpretar el significado y las percepciones subyacentes en lugar de medir cuantitativamente dichas variables. Este enfoque ofrece una perspectiva diferente para estudiar la inteligencia artificial de manera más profunda y los resultados obtenidos de los modelos entrenados

2.4. Población y muestra

La población de estudio se enfoca en el personal de la Empresa Sumeltronic S.A.S y se enfoca en el reconocimiento facial para el registro de asistencia laboral, donde este proyecto es de carácter Muestreo No probabilístico debido que el enfoque o de la selección de las unidades de la muestra no depende del azar, sino de criterios subjetivos o conveniencia del investigador.

Se fundamentará principalmente en las características faciales mediante la muestra de un grupo de trabajadores de la empresa con la ayuda de una cámara web y luego podrá ser comparada con imágenes en una base de datos. Finalmente, con la ayuda del algoritmo de redes neuronales convolucionales se verificará si es parte de la empresa y se procederá a registrar su asistencia.

2.5. Técnicas e instrumentos de recolección de datos

A través de una conversación con el gerente general de la empresa Sumeltronic S.A.S., se propuso la implementación de este nuevo proyecto de innovación tecnológica de registro laboral para su empresa, que se enfoca en el reconocimiento facial y registro de asistencia mediante inteligencia artificial.

Este proyecto ha sido desarrollado en respuesta a la necesidad del gerente de mejorar el sistema de registro laboral de los empleados y optimizar el proceso de recolección de datos, que tradicionalmente se realizaba de forma manual mediante hojas de papel. Además, el proyecto busca reducir el tiempo dedicado a esta tarea y minimizar las inconsistencias que pueden surgir al revisar las nóminas y los registros diarios. De este modo, se pretende lograr una mayor eficiencia y precisión en la gestión de los datos laborales.

La base de datos que se emplea para el desarrollo del sistema consiste en la recolección de los rostros de los empleados que se encuentran registrados en un repositorio virtual por parte del investigador en el dispositivo Raspberry, donde el sistema captura la imagen mediante la cámara web que mediante el algoritmo programado abordará todo el procesamiento de imágenes y extraerá cada característica de la persona registrada y de

esta manera obtener una clara respuesta de quien es cada una de las personas que están ingresando a la empresa.

Tabla 2

Nómina de Trabajadores de la empresa Sumeltronic S.A.S.

	Trabajadores Sumeltronic S.A.S.
1	Ing. Cristian Naranjo (Gerente)
2	Ing. Blanca Pezo (Proyectos)
3	Ing. Stalin Coloma (Logística)
4	Abg. Gregorio Anastacio (Importaciones)
5	Ing. Miguel Vega (Técnico)
6	Lcda. Allison Chávez (Vendedora)
7	Sr. Teófilo Palma (Transporte)

2.6. Metodología de desarrollo

La metodología que se lleva a cabo en esta propuesta consta de la fase de diseño del prototipo, la programación del algoritmo y la interfaz del sistema, las cuales se detallan a continuación:

Diseño del Prototipo

El diseño de la caja del prototipo fue creado en el programa Fusion 360 de Autodesk como se lo puede observar en las figuras 5 y 6, fue impreso 3D en el dispositivo Ender 3 V1, finalizado se observa como en la figura 7.

El prototipo está diseñado de una manera compacta para que contenga la Raspberry y la cámara web de tal manera que sea un modelo cómodo de utilizar, como se puede observar en la figura 8.

Figura 5

Diseño de Tapa de la Caja del Prototipo

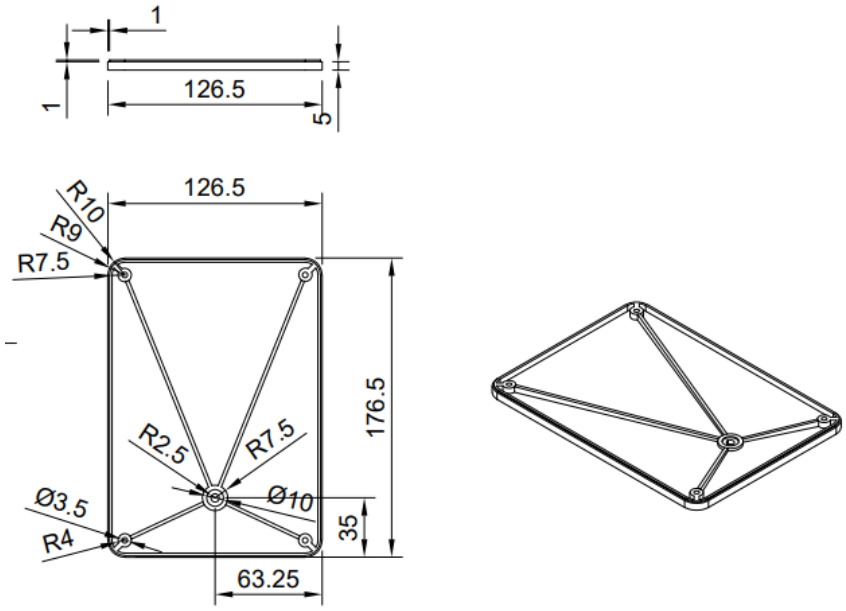


Figura 6

Diseño de la base de la caja del prototipo

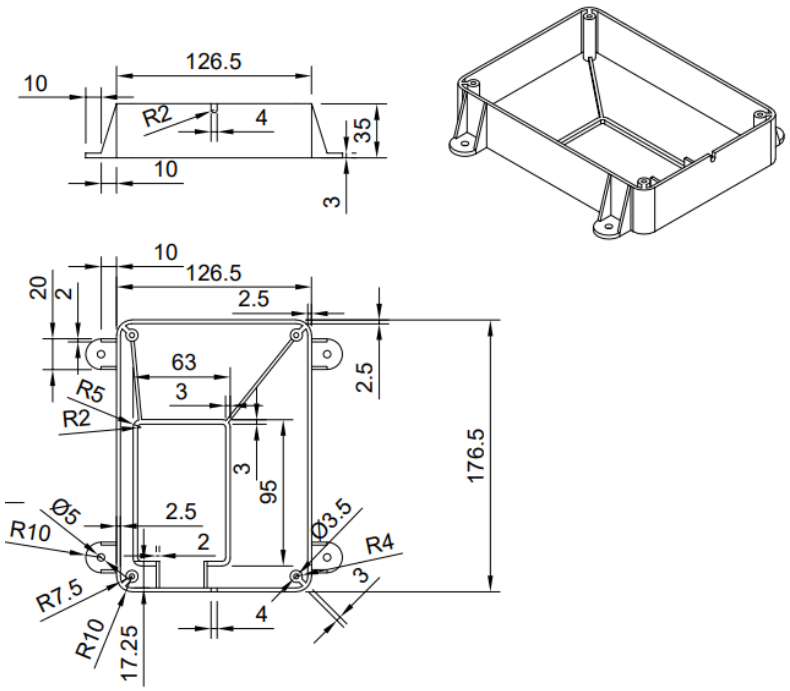


Figura 7

Diseño de Caja del prototipo

Figura 8

Vista Interna del prototipo con Raspberry Pi

Interfaz del Prototipo

La interfaz del prototipo fue codificada utilizando las librerías de Python: tkinter, esta librería permite la creación de la ventana que brinda el control al prototipo, como se observa en la figura 9, se le asigna una variable y se la ubica en algún lugar de la ventana según el fotograma de la pantalla asignado. Este método requiere que se indique la posición del control en la ventana especificando la posición en las coordenadas X e Y. Con width y height se puede asignar al control un ancho y un alto; de no ser el caso, la librería provee un tamaño por defecto, como se observa en la figura 10 la configuración de esta librería para la creación del interfaz.

Figura 9

Creación de la interfaz con librería tkinter de Python.

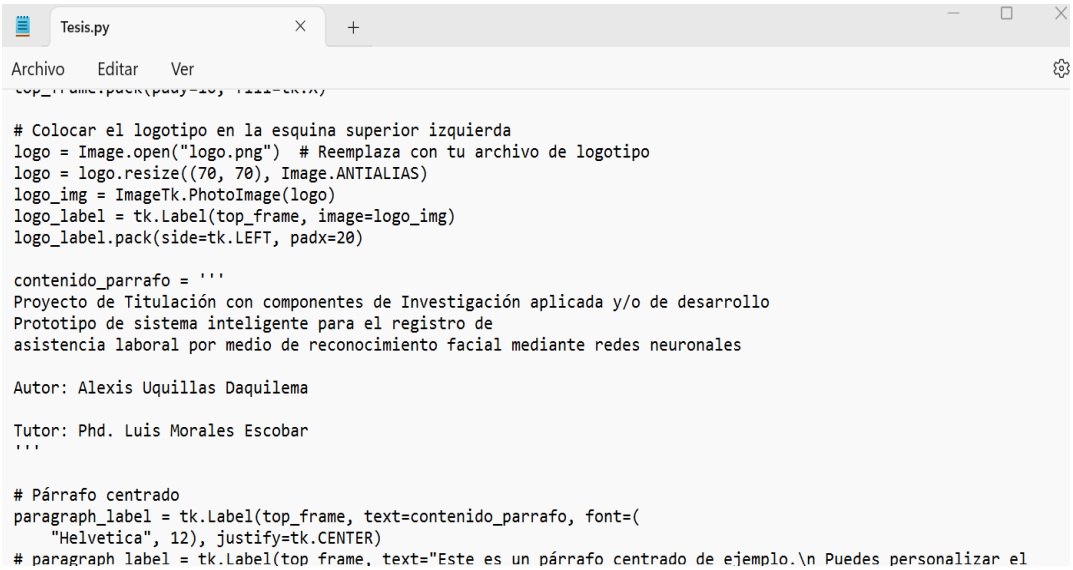
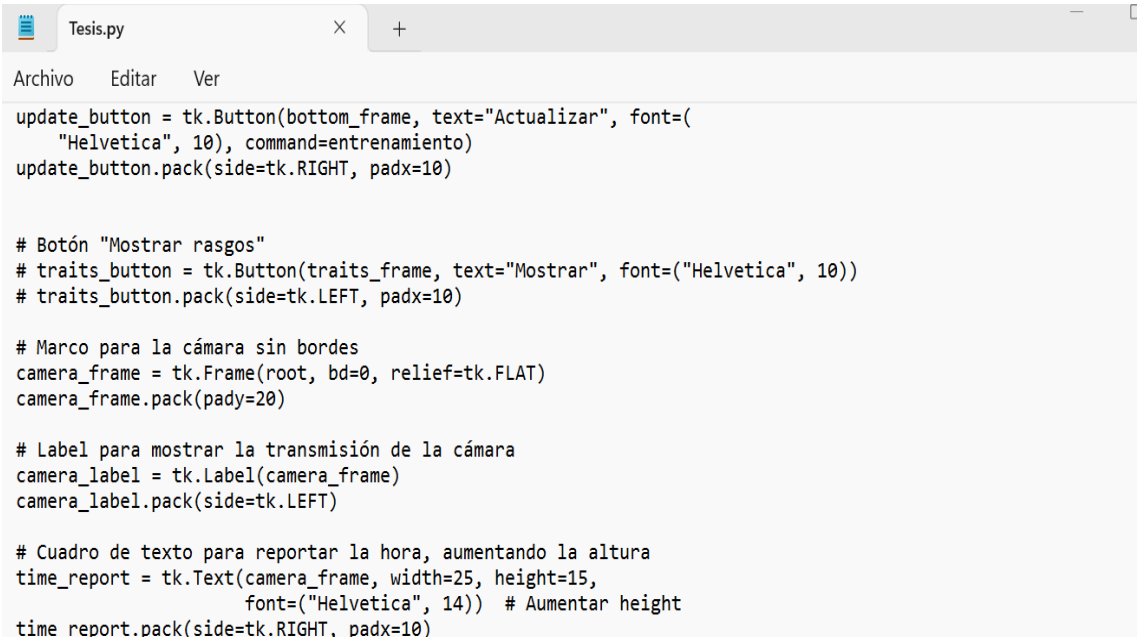


Figura 10

Configuraciones de la librería tkinter

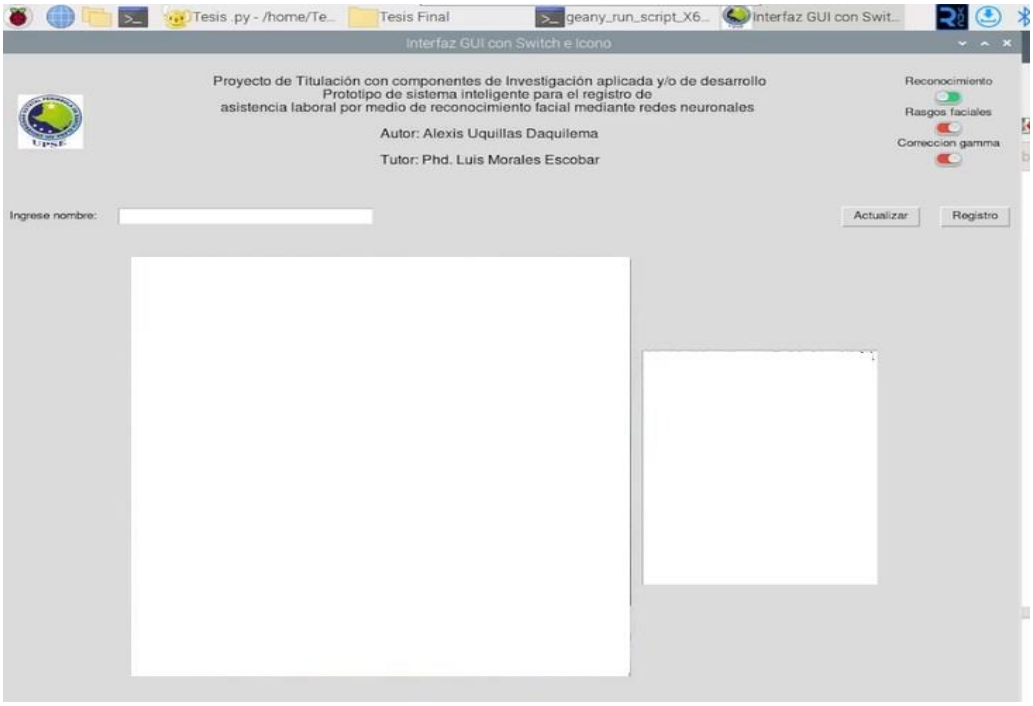


Una vez diseñado la interfaz, ejecutamos el programa de Python para observar su contenido, como se muestra en la figura 11.

La interfaz de menú del prototipo incluye varias opciones, como el encendido y apagado del sistema de reconocimiento, así como la activación o desactivación de la visualización del proceso de extracción de descriptores, que corresponde a los rasgos faciales. También cuenta con un botón para corregir la saturación de la imagen, en caso de que las condiciones de iluminación no sean óptimas, y un botón adicional para realizar el registro instantáneo de nuevos rostros, optimizando así el sistema. Cuando se agrega un nuevo rostro, es necesario entrenar nuevamente la red neuronal con la nueva información. Para ello, se utiliza la opción "Actualizar". Adicionalmente, la interfaz presenta una breve descripción del proyecto de titulación.

Figura 11

Visualización del interfaz del prototipo



Programación del algoritmo

Esta fase se enfoca en la creación del algoritmo que será fundamental para el funcionamiento del prototipo. En la tabla 3 se muestran las librerías más relevantes para la creación del algoritmo con su respectiva descripción.

Tabla 3

Principales librerías del sistema inteligente

Librerías	
import tkinter as tk:	Realiza Interfaz Gráfica
from PIL import Image, ImageTk, ImageDraw:	Permite la edición de imágenes
import cv2:	Manejo de fotogramas
import time:	Para registrar hora actual
import os:	Manejo de archivos y carpetas
from imutils import paths:	Realiza funciones matemáticas con imágenes
import face recognition:	realiza el reconocimiento facial
import pickle:	almacena la base de datos

import csv:	Formato de archivo
import numpy as np:	Realiza el cálculo numérico y el análisis de datos

Captura de Imagen

En este proyecto se ha utilizado una cámara web para la captura de los rostros. En la figura 12 se muestra su codificación para realizar esta función y en la figura 13 su diagrama de flujo correspondiente a la función del código, demostrando como la imagen BGR la transforma a una escala de grises, al transformarlo de esta manera al localizador de rostros le brindara una ayuda a su identificación de los descriptores.

Figura 12

Código de Captura de imagen

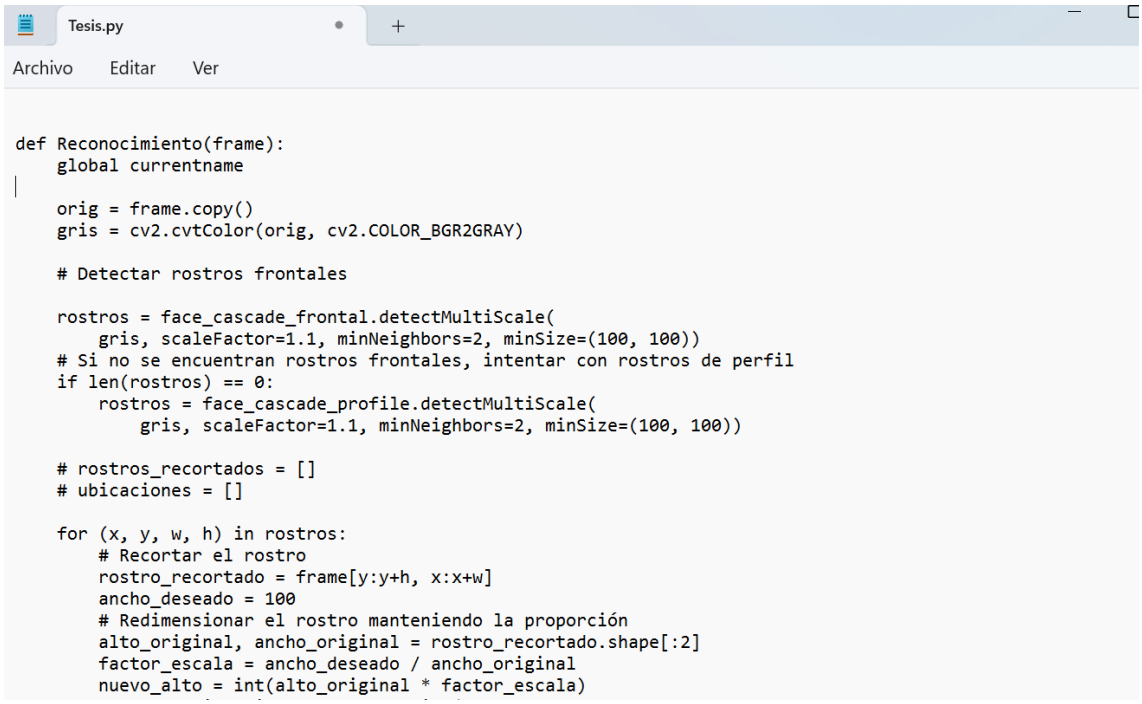
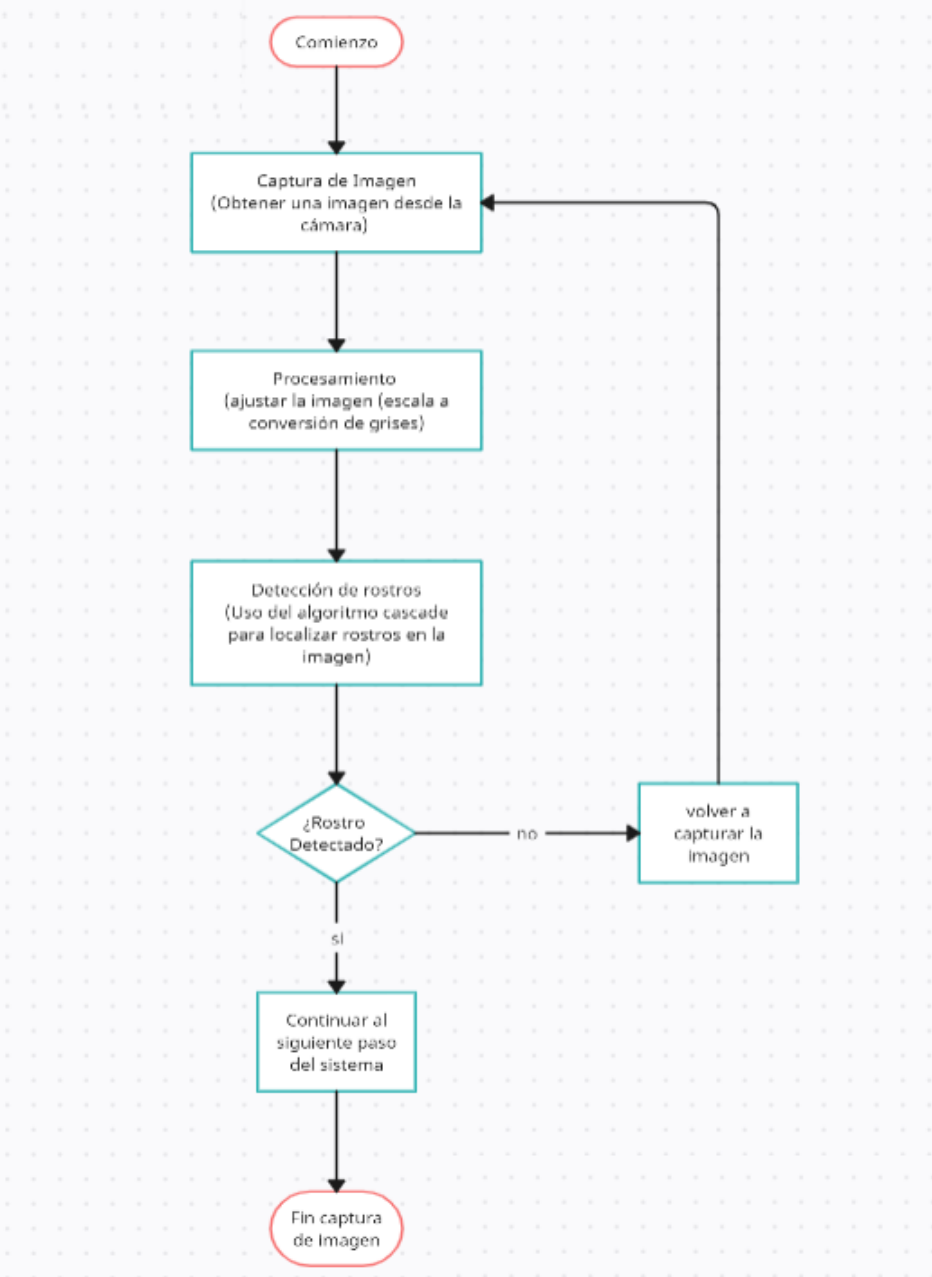


Figura 13

Diagrama de flujo del proceso de la captura de imagen del sistema



Procesamiento de Imagen

Después de adquirir la imagen, el sistema debe detectar la presencia del rostro. Esto se hace mediante algoritmos de inteligencia artificial, como:

Redes Neuronales Convolucionales: Algoritmos de aprendizaje profundo que permiten la detección más precisa y robusta de rostros en diferentes condiciones, como se muestra en la figura 14.

Figura 14

Procesamiento de imagen con modelo CNN

```
# Detectar rostros frontales

rostros = face_cascade_frontal.detectMultiScale(
    gris, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
# Si no se encuentran rostros frontales, intentar con rostros de perfil
if len(rostros) == 0:
    rostros = face_cascade_profile.detectMultiScale(
        gris, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

# rostros_recortados = []
# ubicaciones = []

for (x, y, w, h) in rostros:
    # Recortar el rostro

    rostro_recortado = image[y:y+h, x:x+w]

    # Redimensionar el rostro manteniendo la proporción
    alto_original, ancho_original = rostro_recortado.shape[:2]
    factor_escala = 150 / ancho_original
    nuevo_alto = int(alto_original * factor_escala)
    rostro_redimensionado = cv2.resize(
        rostro_recortado, (150, nuevo_alto))

    boxes = face_recognition.face_locations(
        rostro_redimensionado, model="cnn")
```

En la figura 15 se observa el diagrama de flujo correspondiente a la función del código del procesamiento de la imagen.

Una vez detectado el rostro, es necesario normalizar la imagen para mejorar la precisión del reconocimiento. Esto incluye:

Alineación facial: Alinear el rostro respecto a ciertos puntos clave (ojos, nariz, boca) para evitar variaciones debido a la pose.

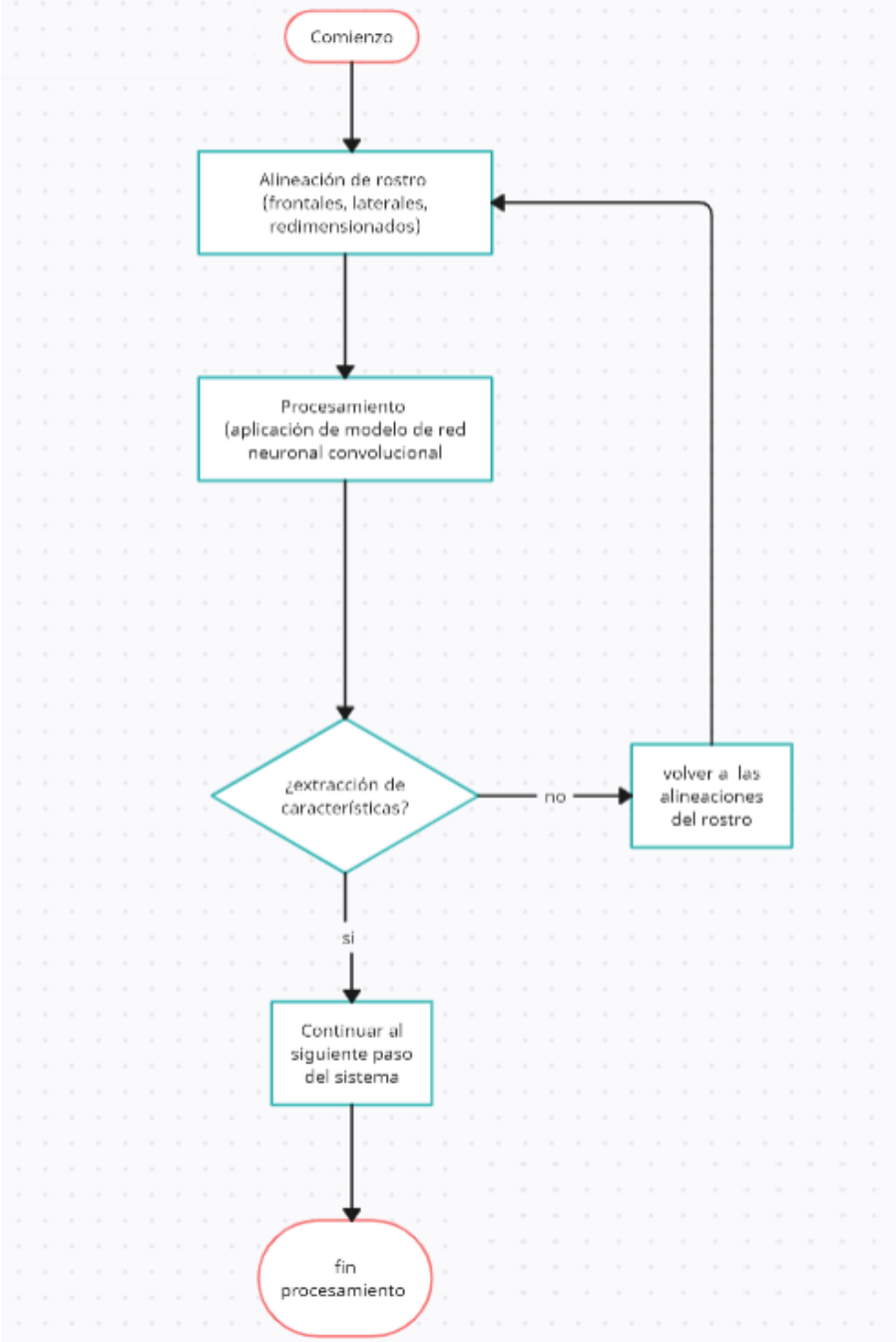
Redimensionamiento: Escalar la imagen del rostro a un tamaño estándar que el sistema pueda procesar eficientemente.

Corrección de iluminación: Ajustar el brillo o el contraste para mejorar la calidad de la imagen capturada.

Solo en casos que sea necesario se utiliza este tipo de correcciones, debido que la red neuronal convolucional lo logra procesar sin librerías externas.

Figura 15

Diagrama de flujo del procesamiento de la imagen del sistema



Extracción de Características (Descriptores)

Este paso se refiere a la transformación de la imagen en una representación numérica que capture las características esenciales del rostro. Se utiliza el siguiente método:

Verificación e Identificación de los Descriptores

El sistema de reconocimiento facial compara el vector de características extraído con los vectores almacenados en una base de datos, se observa el código de este paso en la figura 16 y de la misma manera la figura 17 se observa el diagrama de flujo del proceso de la extracción de descriptores del sistema.

Los sistemas de IA utilizan modelos de aprendizaje profundo para convertir la imagen facial en un vector numérico o embedding que captura características únicas del rostro. Estos vectores suelen tener alta dimensionalidad y representan las características distintivas del rostro.

Un solo vector por rostro: Para cada rostro que se detecta en una imagen, la librería genera un único vector de características (embedding). Cada vector representa las características faciales más importantes del rostro, capturadas en un espacio matemático.

128 dimensiones: Cada vector generado por Face Recognition tiene 128 componentes. Estos son valores numéricos (decimales) que describen las particularidades del rostro. Cuantas más similitudes hay entre dos vectores, más probable es que representen el mismo rostro.

En conclusión, La librería Face Recognition se basa en el modelo de reconocimiento facial de Dlib, que utiliza una red neuronal entrenada en un conjunto amplio de datos de rostros humanos. El modelo extrae estos 128 descriptores para cada rostro detectado en una imagen. Estos descriptores están representados como un vector numérico de 128 dimensiones, también conocido como embedding.

Los descriptores son valores numéricos que encapsulan las características más importantes y únicas del rostro. Estos 128 números no representan rasgos faciales concretos (como ojos, nariz o boca), sino una combinación abstracta de características que el modelo ha aprendido a reconocer tras ser entrenado en un gran conjunto de datos de rostros. Esos descriptores permiten identificar y comparar rostros de manera eficiente.

Distancia Euclidiana: Para medir la similitud entre el vector del rostro capturado y los vectores almacenados es decir se utiliza para cuantificar la similitud entre puntos de datos, lo cual es esencial para diversas tareas como agrupación, clasificación y detección de

rostros. Para comprender la distancia euclidiana, observamos que está dada por dos puntos en un espacio bidimensional, P1(x1, y1) y P2(x2, y2).

La distancia euclidiana entre estos dos puntos viene dada por la fórmula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Esta fórmula calcula la raíz cuadrada de la suma de las diferencias al cuadrado entre las coordenadas de los dos puntos. Representa la longitud de la línea recta que conecta los dos puntos, estos son cálculos internos que la red neuronal realiza y la detecta en manera comparativa con las imágenes captadas en tiempo real y almacenadas en el sistema.

La distancia euclidiana se usa a menudo como una métrica de similitud para comparar vectores de características. Un vector de características representa un punto de datos en un espacio de alta dimensión, donde cada dimensión corresponde a una característica o atributo en específico. Al calcular la distancia euclidiana entre los vectores de características, se determina qué tan similares o diferentes son entre sí. (Europea, 2023)

Figura 16

Extracción de descriptores de las imágenes

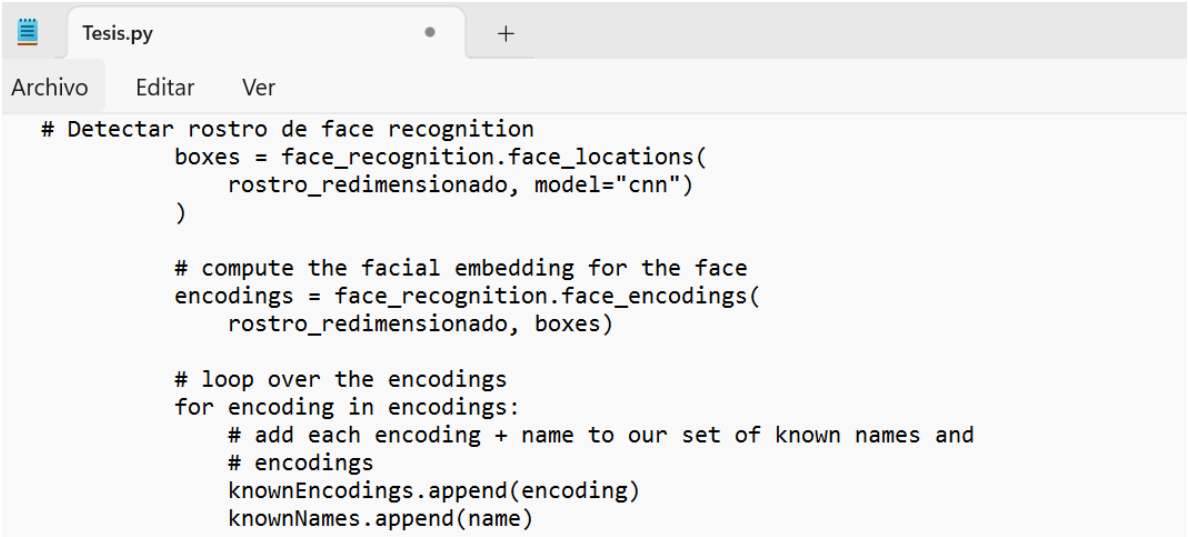
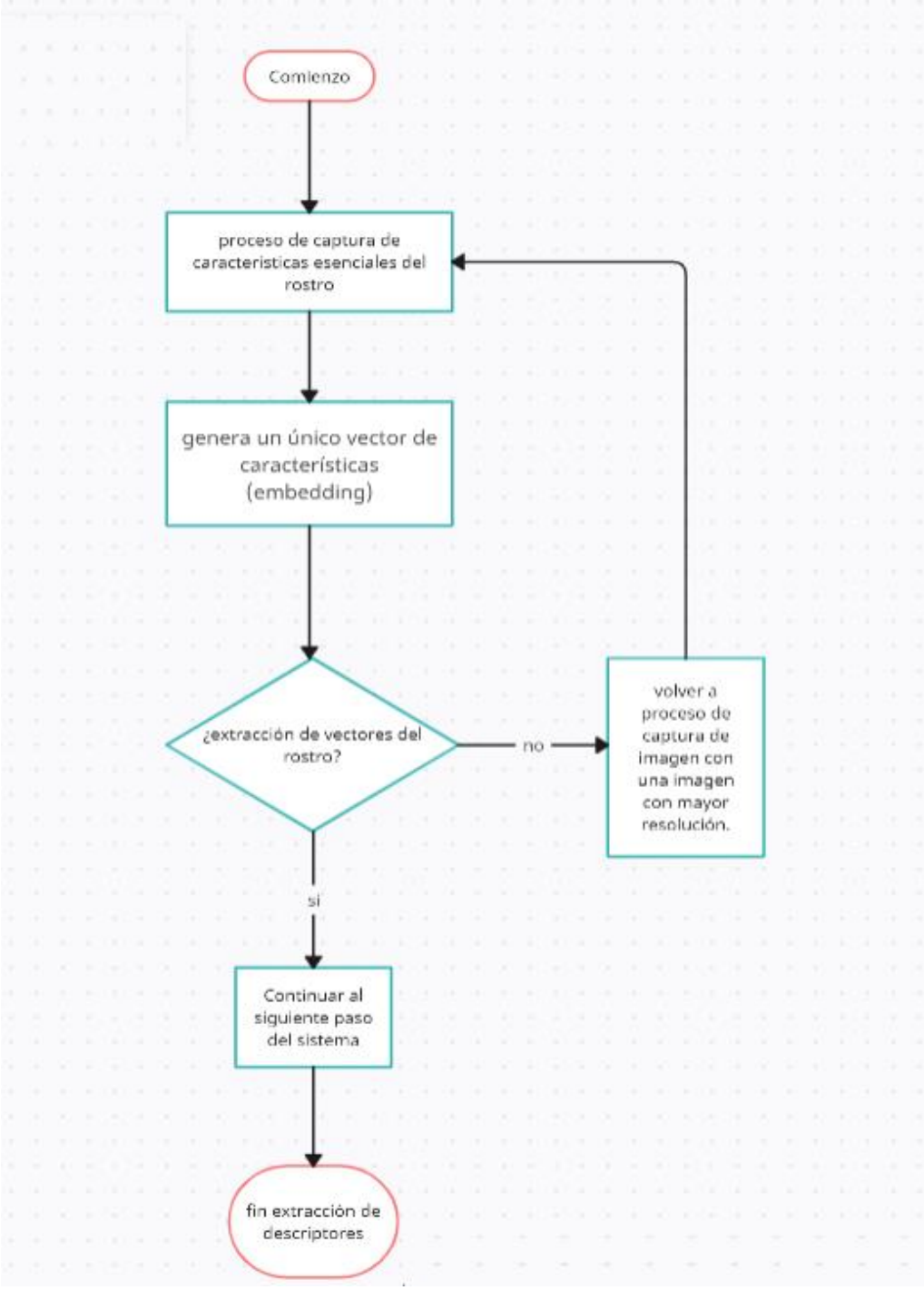


Figura 17

Diagrama de flujo de la extracción de descriptores de la imagen



Verificación o identificación:

Verificación: Se compara la imagen capturada con una imagen de referencia para verificar si son la misma persona.

Identificación: Se compara la imagen con una base de datos completa para identificar la persona. Se puede observar en la figura 18 como el código realiza la comparación con la

base de datos de los empleados de la empresa Sumeltronic, en la figura 19 se observa el diagrama de flujo del proceso comparativo de datos de los descriptores del sistema.

Figura 18

Código de la comparación de embedding de las características únicas del rostro.

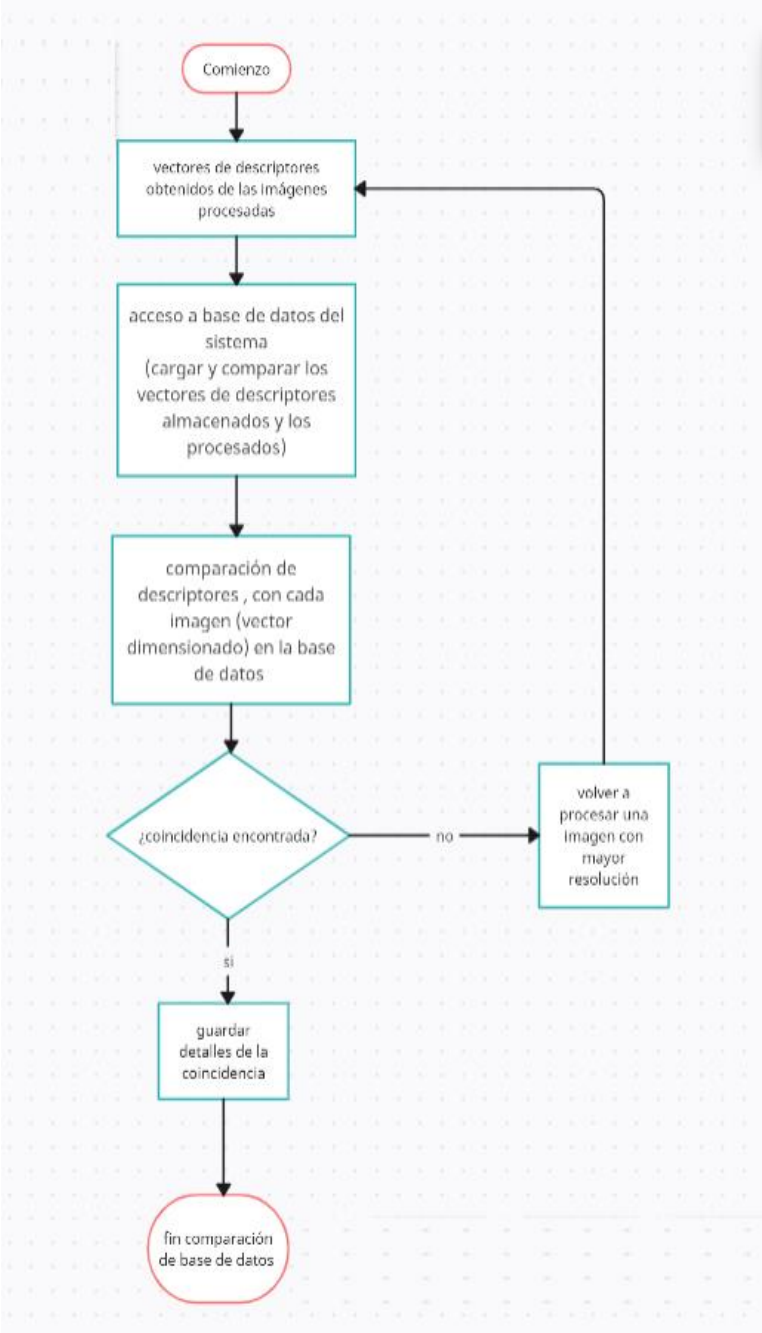
Tesis.py

Archivo Editar Ver

```
# loop over the facial embeddings
for encoding in encodings:
    # attempt to match each face in the input image to our known
    # encodings
    matches = face_recognition.compare_faces(data["encodings"],
                                             encoding)
    name = "Desconocido"  # if face is not recognized, then print Unknown

    # check to see if we have found a match
    if True in matches:
        # find the indexes of all matched faces then initialize a
        # dictionary to count the total number of times each face
        # was matched
        matchedIdxs = [i for (i, b) in enumerate(matches) if b]
        counts = {}

        # loop over the matched indexes and maintain a count for
        # each recognized face face
        for i in matchedIdxs:
            name = data["names"][i]
            counts[name] = counts.get(name, 0) + 1

        # determine the recognized face with the largest number
        # of votes (note: in the event of an unlikely tie Python
        # will select first entry in the dictionary)
        name = max(counts, key=counts.get)
```

Figura 19

Diagrama de flujo de la comparación de base de datos

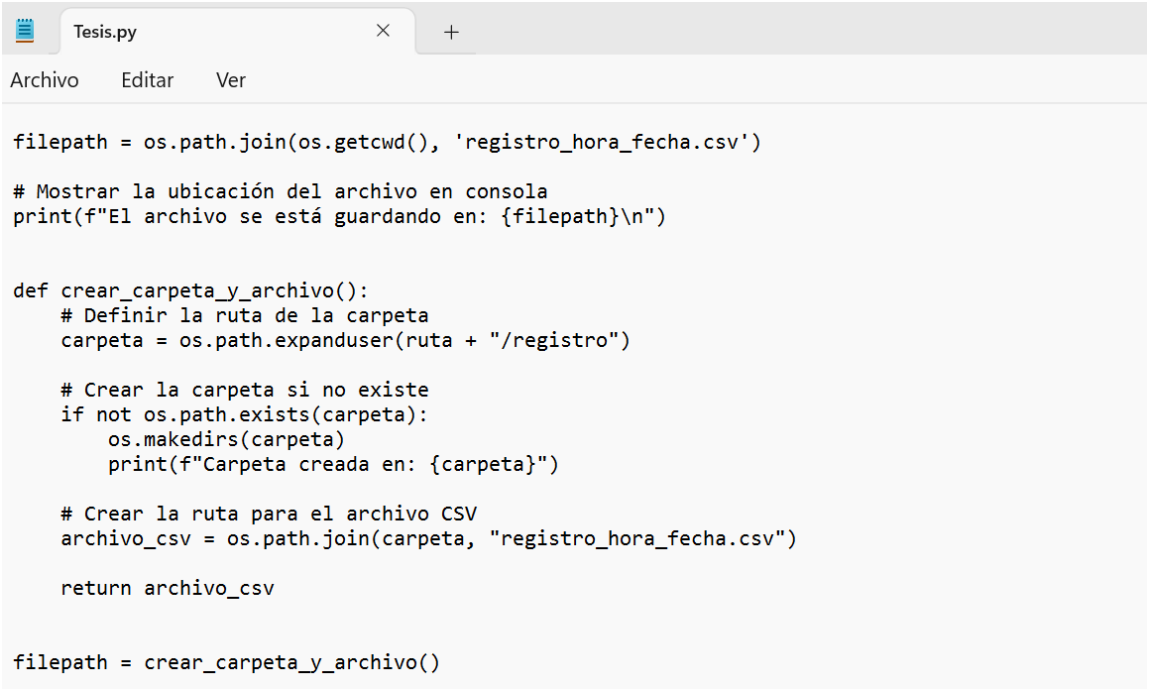


Creación del Registro Laboral

Una vez identificada la persona, en un archivo (.csv) que el algoritmo crea, se detalla toda la información (nombre, fecha y hora) y una captura de la persona del momento del ingreso a la empresa mientras este dentro de la base de datos del sistema, en la codificación de la figura 20 podemos observar las rutas que son creadas para el almacenamiento de esta información.

Figura 20

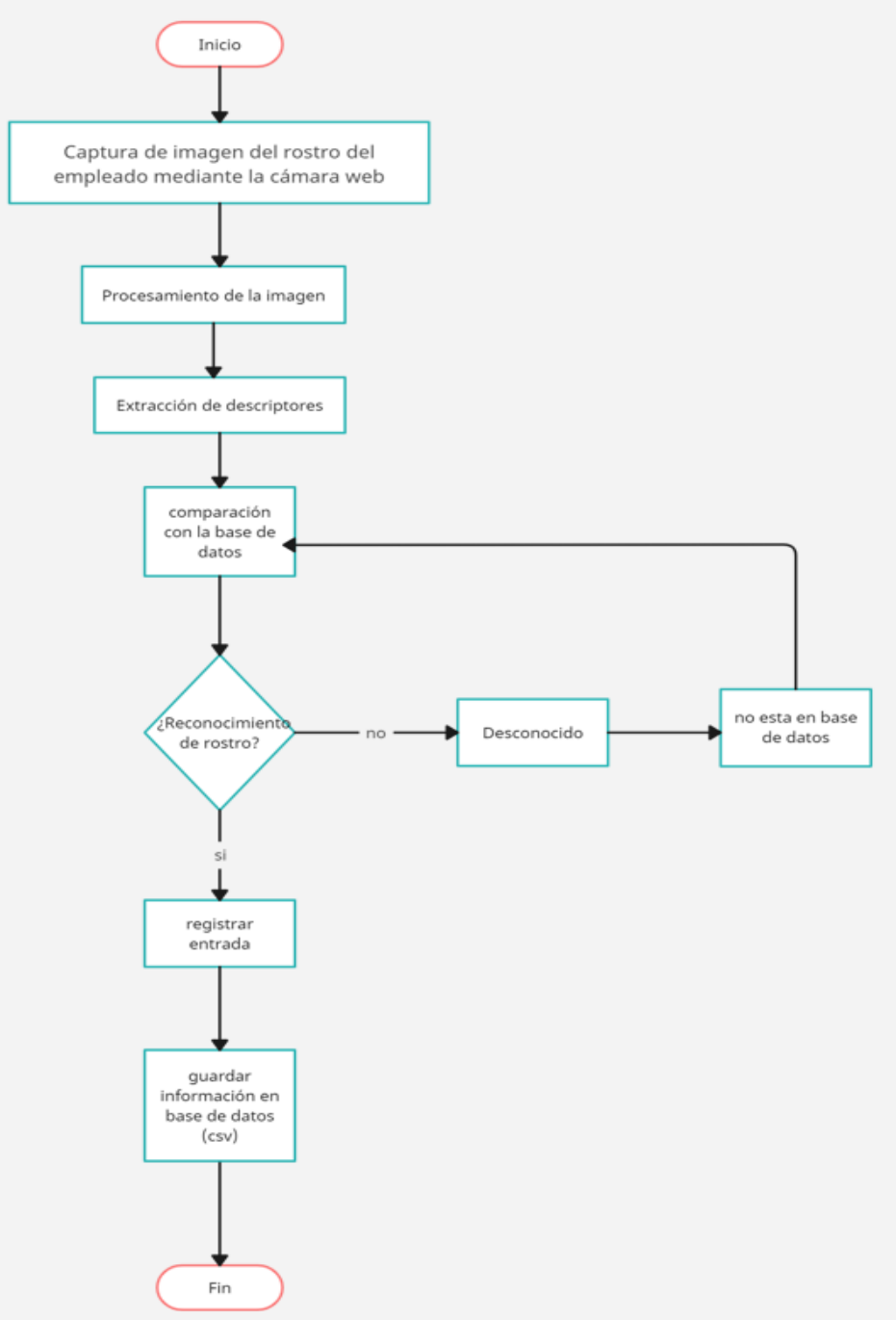
Ruta elaborada para la creación del registro laboral



Una vez que se ha demostrado el paso a paso del funcionamiento del sistema, lo representamos en un diagrama de flujo de manera general como se observa en la figura 21.

Figura 21

Diagrama de flujo del sistema


Inicio
Captura de imagen del rostro del empleado mediante la cámara web
Procesamiento de la imagen
Extracción de descriptores
comparación con la base de datos
¿Reconocimiento de rostro?
no
Desconocido
no esta en base de datos
si
registrar entrada
guardar información en base de datos (csv)
Fin

CAPÍTULO 3. RESULTADOS Y DISCUSIÓN

El proyecto plantea el diseño de un sistema de registro de asistencia laboral mediante redes neuronales y reconocimiento facial. Como se ha mencionado en el capítulo 2, el prototipo final mostrado en la figura 22 está compuesto por una Raspberry Pi conectada a una cámara web. El prototipo ha sido diseñado para que el usuario pueda interactuar con él y adaptarlo según sus necesidades.

Figura 22

Prototipo del Proyecto de Titulación

3.1 Pruebas del Interfaz

En la figura 23 se observa, el accionamiento de la botonera de reconocimiento facial permitiendo desde el momento de su activación identificar a la persona que visualizamos, siempre que este almacenada en nuestra base de datos.

Figura 23

Encendido del reconocimiento facial

En la figura 24, se observa el apagado del reconocimiento facial mediante el accionamiento de la misma botonera, de esta manera desactivando la función para ser observada como una cámara web en el entorno.

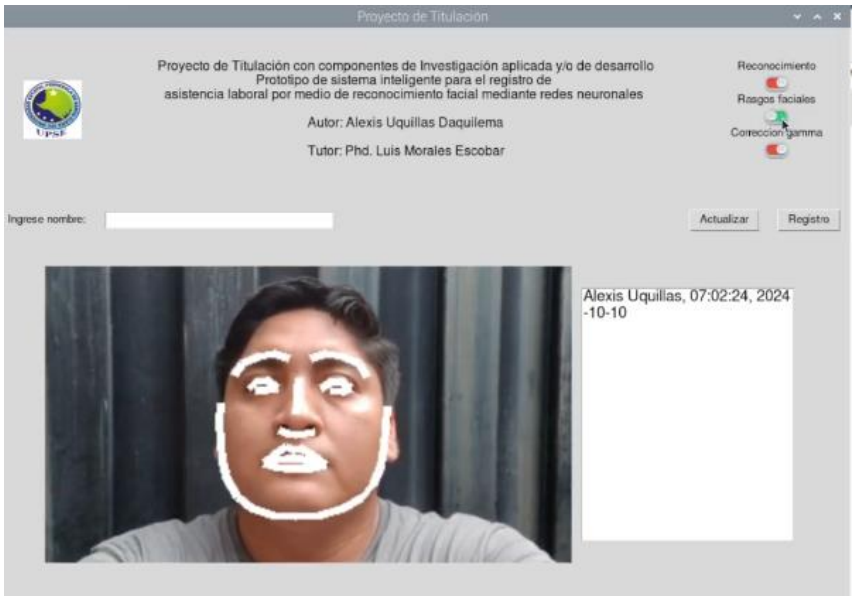
Figura 24

Apagado del reconocimiento facial del sistema

En la figura 25 se observa el encendido de la función de los rasgos faciales, la cual nos permite identificar los rasgos característicos que están siendo consideradas como descriptores para la identificación de la persona.

Figura 25

Visualización de los rasgos característicos para ser identificado en la base de datos



En la figura 26 se observa la corrección de saturación de la imagen, misma que esta programada en caso de un algún requerimiento en específico que se pueda dar por alguna condición lumínica que no sea favorable para el registro de la persona.

Figura 26

Corrección de saturación de la imagen desde el interfaz del sistema

3.2 Implementación del Prototipo

Una vez diseñado el prototipo de la caja en el software Fusion 360, se han colocado los dispositivos necesarios para este proyecto, entre los cuales se incluyen la Raspberry Pi 4 Modelo B y una cámara web, la cual ha sido instalada de manera compacta dentro del dispositivo. A continuación, se procede a cerrar el prototipo y a implementar una canalización adecuada para asegurar que el cable de alimentación no se desconecte accidentalmente. Como se muestra en la figura, finalmente, se energiza el prototipo para verificar su correcto funcionamiento. En la figura 27 se puede observar el montaje del prototipo en la pared.

Figura 27

Montaje del prototipo del proyecto

3.3 Evaluación del prototipo del sistema de reconocimiento facial

Para validar el funcionamiento del prototipo se ha evaluado su funcionamiento realizando la identificación del gerente de la empresa y sus empleados, demostrando alta precisión en la detección de las características faciales.

A continuación, en las figuras 28,29,30,31,32,33,34,35 se observa la interfaz del prototipo con su respectivo reconocimiento, mediante el cual se comprueba la efectividad del sistema:

Figura 28

Identificación Ing. Cristian Naranjo Gerente de la Empresa

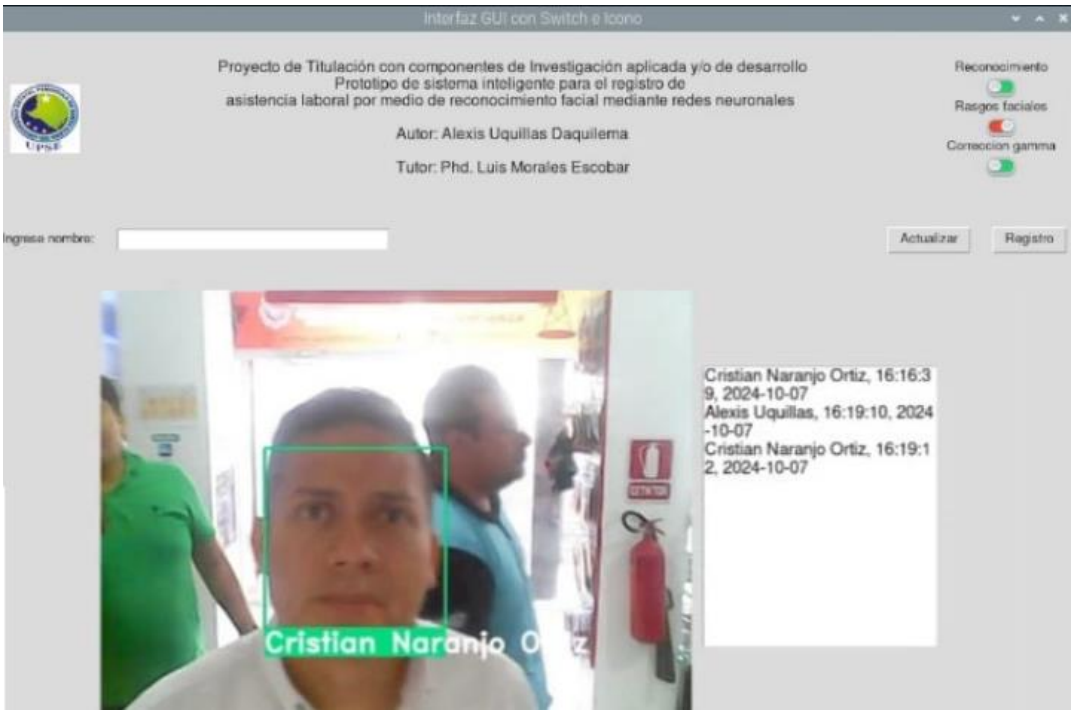


Figura 29

Identificación Ing. Blanca Pezo del Departamento de Proyectos

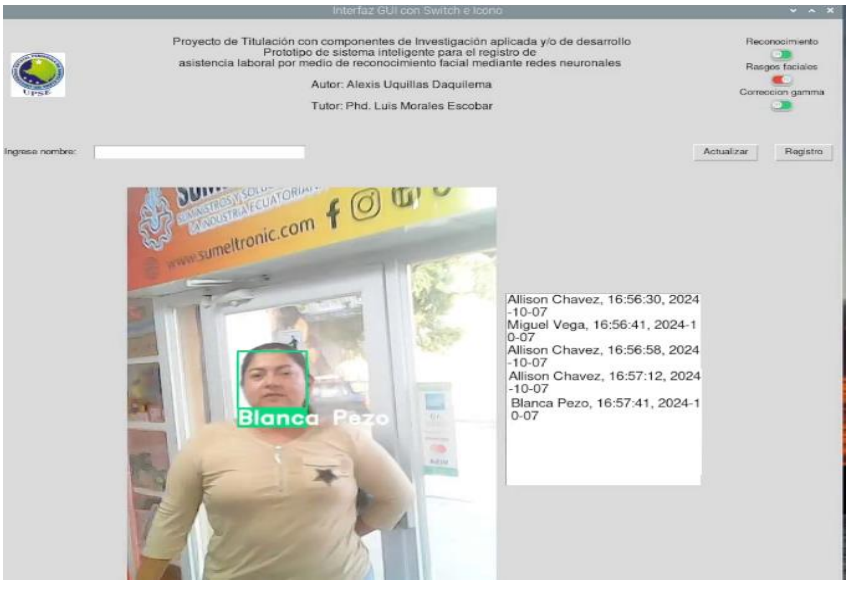


Figura 30

Identificación Lcda. Allison Chávez de Ventas

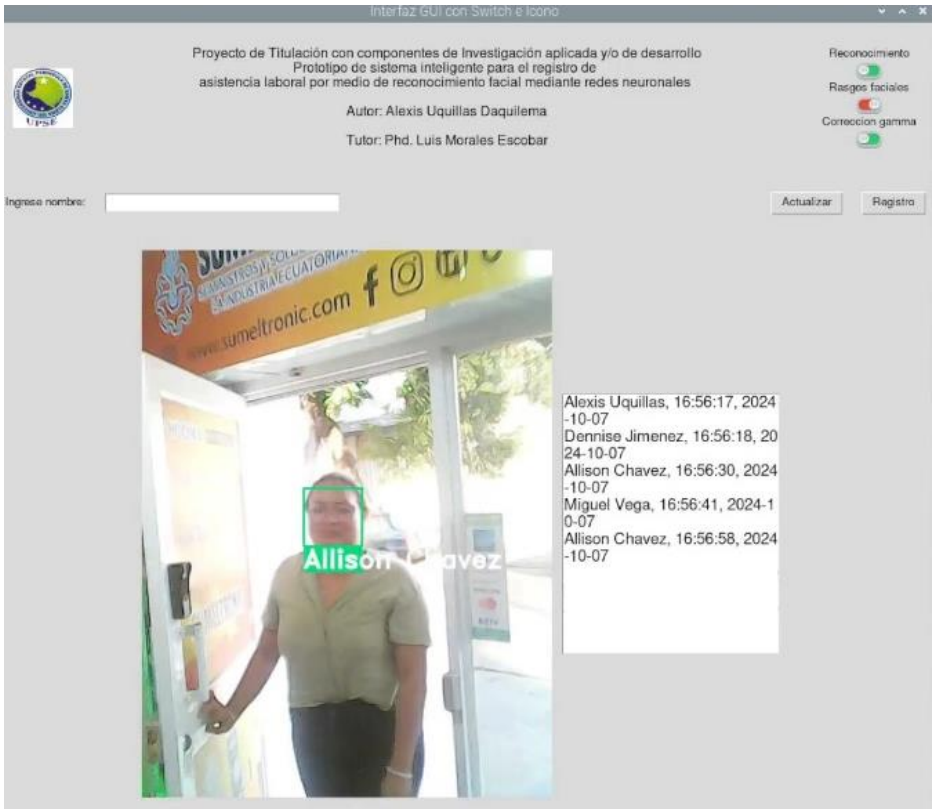


Figura 31

Identificación Ing. Stalin Coloma de Logística

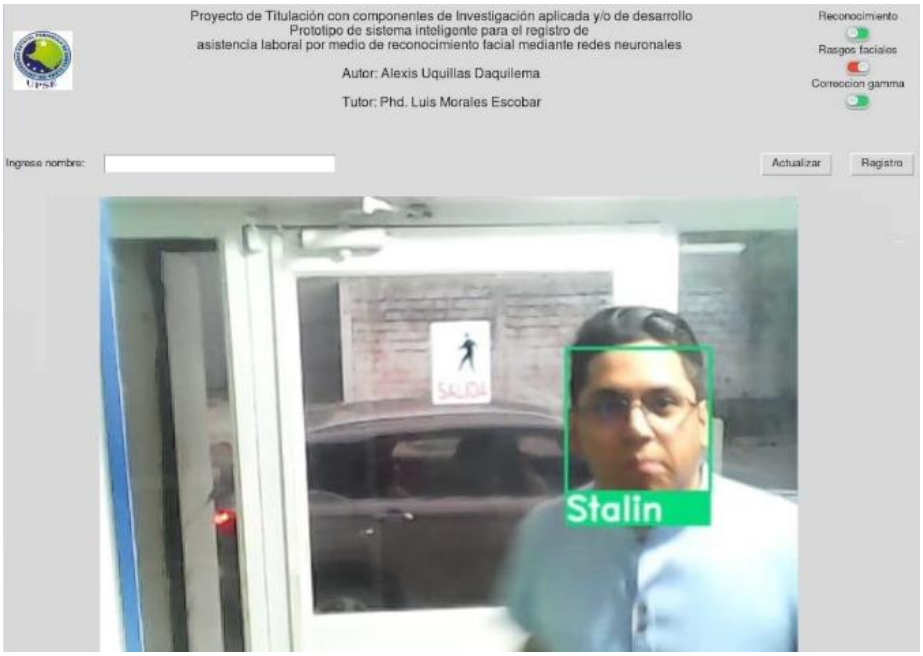


Figura 32

Identificación Ing. Miguel Vega jefe Técnico

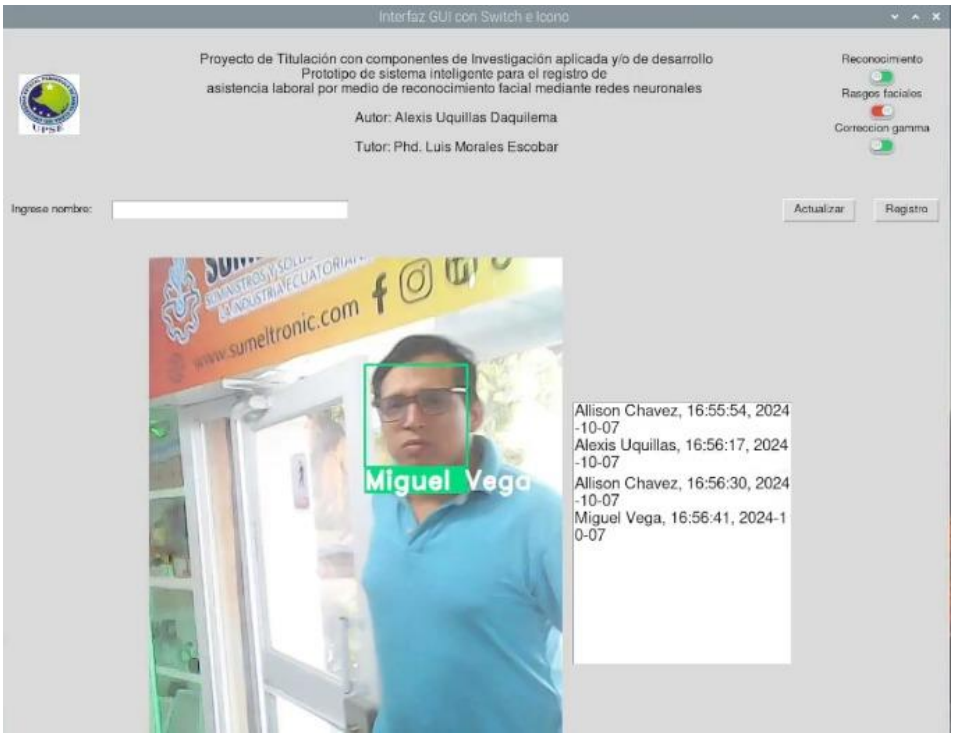


Figura 33

Identificación Abg. Gregorio Anastacio de Importaciones

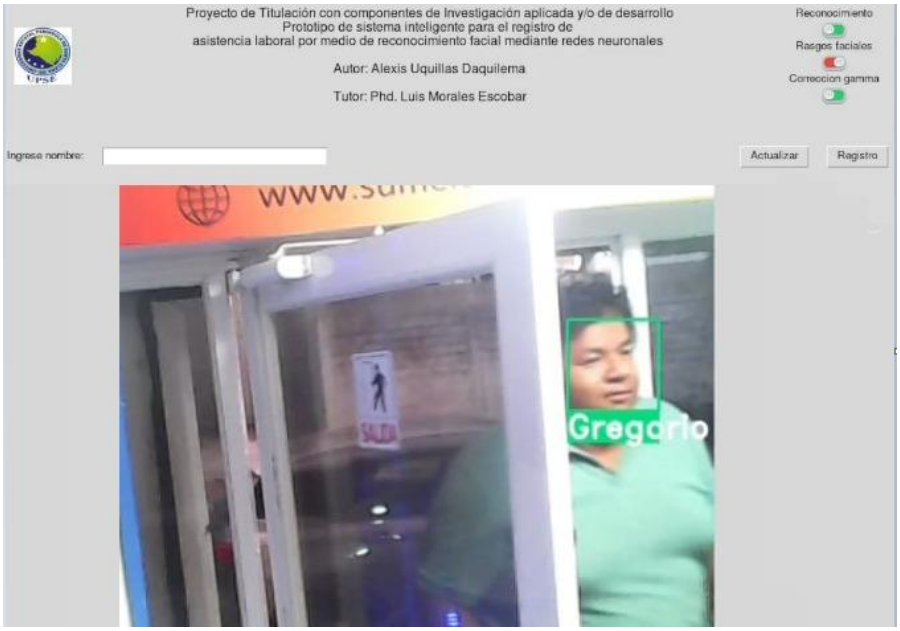


Figura 34

Identificación Ing. Teófilo Palma de Transporte de la empresa

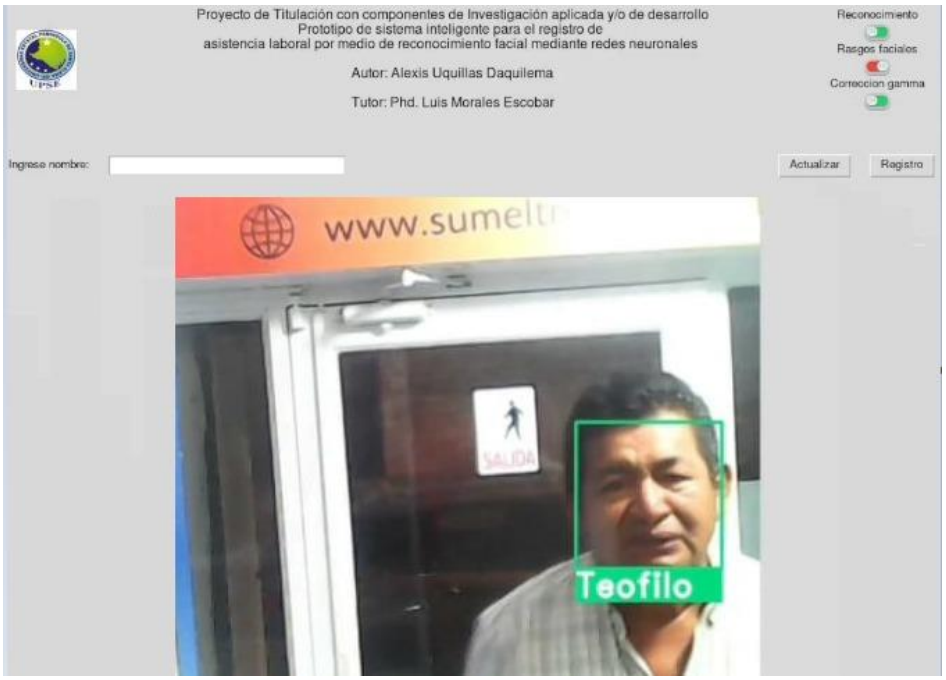
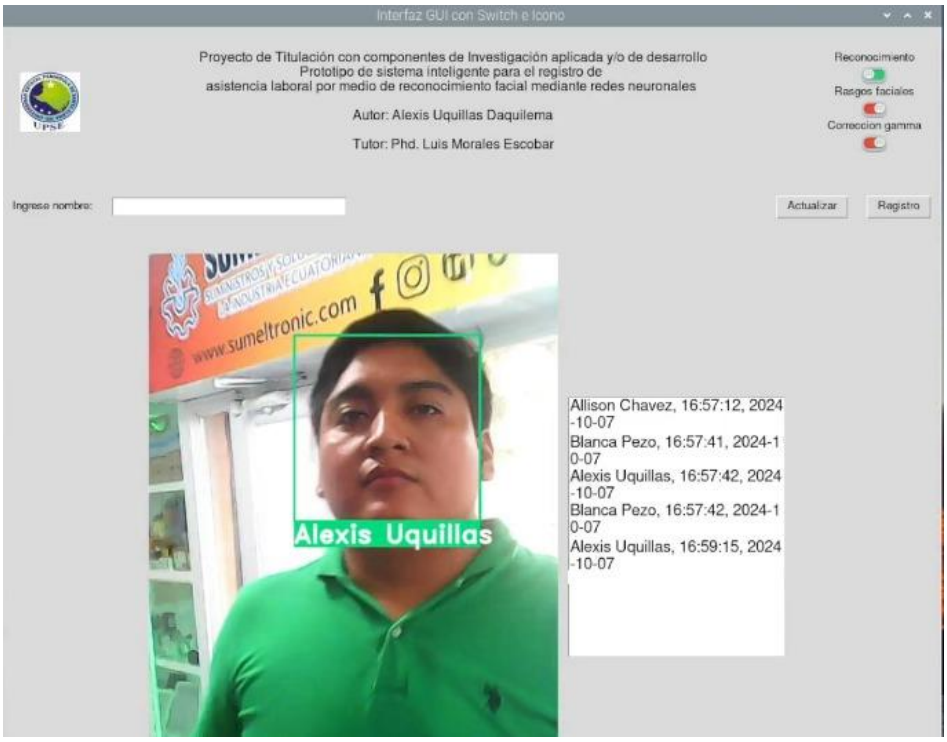


Figura 35

Identificación Alexis Uquillas (Autor)



3.4 Verificación del registro de asistencia laboral

Se implementó un sistema automatizado para el registro de la asistencia laboral a través del prototipo. La información requerida se almacena en una base de datos en Excel, la cual se genera un archivo a diario, y además se captura una imagen del rostro en el momento en que se realiza el registro, la capacidad máxima de base de datos permitida en el sistema dependerá de la unidad de almacenamiento instalada al raspberry , al ser imágenes y base de datos de Excel archivos de un tamaño muy ligero (aproximadamente 1MG CSV y la foto capturada con fecha y hora aproximadamente 95 KB) la capacidad máxima del prototipo que podría almacenar estaría más allá 5000 base de datos generados por el sistema . En la figura 36 se puede observar el registro de asistencia en Excel, mientras que en la figura 37 se muestra el proceso mediante el cual se generan los archivos de captura de imagen desde el prototipo.

Figura 36

Registro laboral en Excel

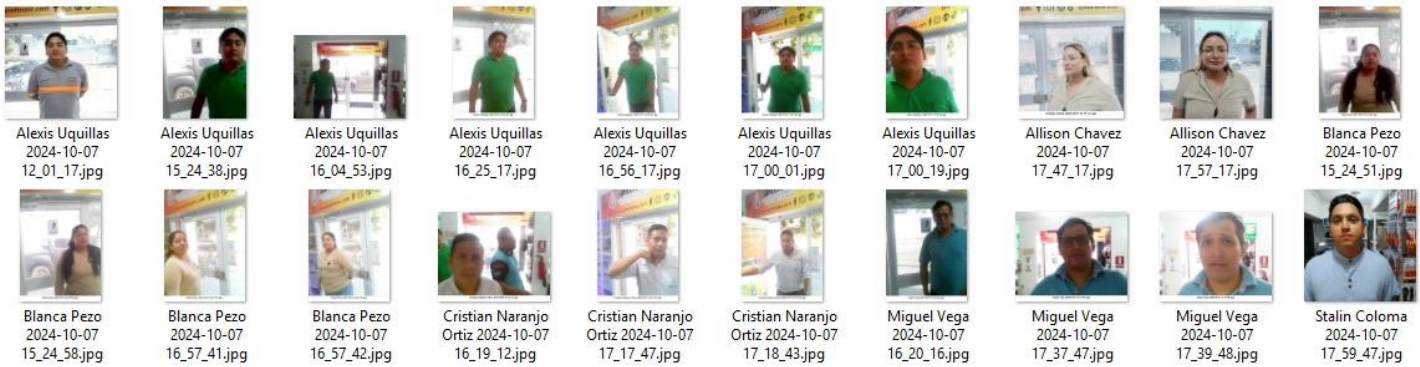
F8

	A	B	C
1	Nombre	Hora	Fecha
2	Miguel Vega	15:24:26	7/10/2024
3	Alexis Uquillas	15:44:18	7/10/2024
4	Blanca Pezo	16:21:24	7/10/2024
5	Alexis Uquillas	16:33:03	7/10/2024
6	Blanca Pezo	16:34:16	7/10/2024
7	Stalin Coloma	16:37:46	7/10/2024
8	Blanca Pezo	16:38:45	7/10/2024
9	Gregorio Anastacio	16:39:11	7/10/2024
10	Allison Chavez	16:39:56	7/10/2024
11	Teofilo Palma	16:41:17	7/10/2024
12	Cristian Naranjo Ortiz	16:44:56	7/10/2024
13	Alexis Uquillas	16:46:14	7/10/2024
14	Blanca Pezo	16:48:25	7/10/2024
15	Cristian Naranjo Ortiz	16:51:11	7/10/2024
16	Allison Chavez	16:54:50	7/10/2024
17	Cristian Naranjo Ortiz	16:54:58	7/10/2024
18	Alexis Uquillas	16:56:56	7/10/2024
19	Cristian Naranjo Ortiz	16:59:17	7/10/2024
20	Alexis Uquillas	17:04:10	7/10/2024

registro_hora_fecha

Figura 37

Visualización de la captura del registro reconocimiento facial del trabajador con nombre, fecha y hora.



3.5 Precisión del sistema

A continuación, se analizan aciertos y errores del reconocimiento facial según cada empleado al momento de ingresar a la empresa:

1. **Pruebas de Distancia**

Tabla 4

Datos de pruebas aciertos y errores de reconocimiento facial Ing. Cristian Naranjo

Ing. Cristian Naranjo
Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 5

Datos de pruebas aciertos y errores de reconocimiento facial Ing. Blanca Pezo

Ing. Blanca Pezo

Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 6

Datos de pruebas aciertos y errores de reconocimiento facial Ing. Miguel Vega

Ing. Miguel Vega
Pruebas: 15
Aciertos: 14
Errores: 1
Efectividad: E= $\frac{14}{15}$ x 100% = 93.33%

Tabla 7

Datos de pruebas aciertos y errores de reconocimiento facial Ing. Stalin Coloma

Ing. Stalin Coloma
Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 8

Datos de pruebas aciertos y errores de reconocimiento facial Abg. Gregorio Anastacio

Abg. Gregorio Anastacio
Pruebas: 15
Aciertos: 14
Errores: 1
Efectividad: $E= \frac{14}{15}$ x 100% = 93.33%

Tabla 9

Datos de pruebas aciertos y errores de reconocimiento facial Lcda. Allison Chávez

Lcda. Allison Chávez
Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 10

Datos de pruebas aciertos y errores de reconocimiento facial Sr. Teófilo Palma

Sr. Teófilo Palma
Pruebas: 15
Aciertos: 14
Errores: 1
Efectividad: $E= \frac{14}{15}$ x 100% = 93.33%

Tabla 11

Datos de pruebas aciertos y errores de reconocimiento facial Ing. Alexis Uquillas (Autor)

Ing. Alexis Uquillas (Autor)
Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

2. Pruebas de condiciones de iluminación

- **Iluminación Baja**

  Tabla 12

  Datos de pruebas aciertos y errores de reconocimiento facial por iluminación baja del Ing. Cristian Naranjo

Ing. Cristian Naranjo
Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E= \frac{15}{15}$ x 100% = 100%

Tabla 13

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación baja de Ing. Blanca Pezo

Ing. Blanca Pezo
Pruebas: 15

Aciertos: 14
Errores: 0
Efectividad: $E = \frac{14}{15}\ x100\% = 93.33\%$

Tabla 14

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación baja del Ing. Miguel Vega

Ing. Miguel Vega
Pruebas: 15
Aciertos: 14
Errores: 0
Efectividad: $E = \frac{14}{15}\ x100\% = 93.33\%$

Tabla 15

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación baja del Ing. Stalin Coloma

Ing. Stalin Coloma
Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 16

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación baja del Abg. Gregorio Anastacio

Abg. Gregorio Anastacio
Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 17

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación baja de Lcda. Allison Chávez

Lcda. Allison Chávez
Pruebas: 15
Aciertos: 14
Errores: 1
Efectividad: $E = \frac{14}{15}\ x100\% = 93.33\%$

Tabla 18

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación baja del Sr. Teófilo Palma

Sr. Teófilo Palma
Pruebas: 15
Aciertos: 14

Errores: 1
Efectividad: $E = \frac{14}{15}\ x100\% = 93.33\%$

Tabla 19

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación baja del Ing. Alexis Uquillas (Autor)

Ing. Alexis Uquillas (Autor)
Pruebas: 15 Aciertos: 14 Errores: 0
Efectividad: $E = \frac{14}{15}\ x100\% = 93.33\%$

- **Iluminación Natural**

Tabla 20

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación natural del Ing. Cristian Naranjo

Ing. Cristian Naranjo
Pruebas: 15 Aciertos: 15 Errores: 0
Efectividad: E= $\frac{15}{15}$ x 100% = 100%

Tabla 21

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación natural de Ing. Blanca Pezo

Ing. Blanca Pezo
Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 22

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación natural del Ing. Miguel Vega

Ing. Miguel Vega
Pruebas: 15
Aciertos: 15
Errores: 1
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 23

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación natural del Ing. Stalin Coloma

Ing. Stalin Coloma
Pruebas: 15
Aciertos: 15

Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 24

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación natural del Abg. Gregorio Anastacio

Abg. Gregorio Anastacio
Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 25

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación natural de Lcda. Allison Chávez

Lcda. Allison Chávez
Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 26

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación natural del Sr. Teófilo Palma

Sr. Teófilo Palma
Pruebas: 15 Aciertos: 15 Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 27

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación natural del Ing. Alexis Uquillas (Autor)

Ing. Alexis Uquillas (Autor)
Pruebas: 15 Aciertos: 15 Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Iluminación Intensa

Tabla 28

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación intensa del Ing. Cristian Naranjo

Ing. Cristian Naranjo
Pruebas: 15 Aciertos: 14

Errores: 1

Efectividad: $E = \frac{14}{15}$ x 100% = 93.33%

Tabla 29

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación intensa de Ing. Blanca Pezo

Ing. Blanca Pezo
Pruebas: 15
Aciertos: 14
Errores: 1
Efectividad: $E = \frac{14}{15}\ x100\% = 93.33\%$

Tabla 30

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación intensa del Ing. Miguel Vega

Ing. Miguel Vega
Pruebas: 15
Aciertos: 14
Errores: 1
Efectividad: $E = \frac{14}{15}\ x100\% = 93.33\%$

Tabla 31

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación intensa del Ing. Stalin Coloma

Ing. Stalin Coloma
Pruebas: 15
Aciertos: 14
Errores: 1
Efectividad: $E = \frac{14}{15}\ x100\% = 93.33\%$

Tabla 32

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación intensa del Abg. Gregorio Anastacio

Abg. Gregorio Anastacio
Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 33

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación intensa de Lcda. Allison Chávez

Lcda. Allison Chávez
Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 34

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación intensa del Sr. Teófilo Palma

Sr. Teófilo Palma
Pruebas: 15 Aciertos: 15 Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 35

Datos de pruebas aciertos y errores de reconocimiento facial por iluminación intensa del Ing. Alexis Uquillas (Autor)

Ing. Alexis Uquillas (Autor)
Pruebas: 15 Aciertos: 14 Errores: 0
Efectividad: $E = \frac{14}{15}\ x100\% = 93.33\%$

3. Pruebas de ángulos del rostro

Tabla 36

Datos de pruebas aciertos y errores de reconocimiento facial por ángulos del Ing. Cristian Naranjo

Ing. Cristian Naranjo

Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E= \frac{15}{15}$ x 100% = 100%

Tabla 37

Datos de pruebas aciertos y errores de reconocimiento facial por ángulos de Ing. Blanca Pezo

Ing. Blanca Pezo
Pruebas: 15
Aciertos: 14
Errores: 1
Efectividad: $E = \frac{14}{15}\ x100\% = 93.33\%$

Tabla 38

Datos de pruebas aciertos y errores de reconocimiento facial por ángulos del Ing. Miguel Vega

Ing. Miguel Vega
Pruebas: 15
Aciertos: 14
Errores: 1
Efectividad: $E = \frac{14}{15}\ x100\% = 93.33\%$

Tabla 39

Datos de pruebas aciertos y errores de reconocimiento facial por ángulos del Ing. Stalin Coloma

Ing. Stalin Coloma
Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 40

Datos de pruebas aciertos y errores de reconocimiento facial por ángulos del Abg. Gregorio Anastacio

Abg. Gregorio Anastacio
Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 41

Datos de pruebas aciertos y errores de reconocimiento facial por ángulos de Lcda. Allison Chávez

Lcda. Allison Chávez
Pruebas: 15
Aciertos: 15

Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 42

Datos de pruebas aciertos y errores de reconocimiento facial por ángulos del Sr. Teófilo Palma

Sr. Teófilo Palma
Pruebas: 15
Aciertos: 15
Errores: 0
Efectividad: $E = \frac{15}{15}\ x100\% = 100\%$

Tabla 43

Datos de pruebas aciertos y errores de reconocimiento facial por ángulos del Ing. Alexis Uquillas (Autor)

Ing. Alexis Uquillas (Autor)
Pruebas: 15
Aciertos: 14
Errores: 1
Efectividad: $E = \frac{14}{15}\ x100\% = 93.33\%$

Efectividad del sistema en general

Distancia

Se obtuvo que 5 personas tuvieron sus pruebas experimentales con satisfacción mientras que a los otros 3 se tuvo un error en durante las pruebas de distancia (1m), se calculó la media de los datos obtenidos:

$$M= \frac{100+100+100+100+100+93.33+93.33+93.33}{8} \% = 97.49\%$$

Iluminación

Se obtuvo que 3 personas tuvieron sus pruebas experimentales con satisfacción mientras que a los otros 5 se tuvo un error en durante las pruebas de iluminación baja, se calculó la media de los datos obtenidos:

$$M= \frac{100+100+100+93.33+93.33+93.33+93.33+93.33}{8} \% = 95.83\%$$

Se obtuvo que 8 personas tuvieron sus pruebas experimentales con satisfacción durante las pruebas de iluminación natural, se calculó la media de los datos obtenidos:

$$M= \frac{100+100+100+100+100+100+100+100}{8} \% = 100\%$$

Se obtuvo que 3 personas tuvieron sus pruebas experimentales con satisfacción mientras que a los otros 5 se tuvo un error en durante las pruebas de iluminación intensa, se calculó la media de los datos obtenidos:

$$M= \frac{100+100+100+93.33+93.33+93.33+93.33+93.33}{8} \% = 95.83\%$$

Ángulo del rostro

Se obtuvo que 5 personas tuvieron sus pruebas experimentales con satisfacción mientras que a los otros 3 se tuvo un error en durante las pruebas de ángulos del rostro, se calculó la media de los datos obtenidos:

$$M= \frac{100+100+100+100+100+93.33+93.33+93.33}{8} \% = 97.49\%$$

A continuación, se realiza el estudio correspondiente a la hipótesis de la investigación:

Con los datos anteriormente mencionados se calculó la media y se demuestra que el porcentaje de aciertos del reconocimiento facial a distancia de 1m en el grupo de personas de la empresa Sumeltronic S.A.S. mediante el prototipo implementado alcanza un 97.49% de efectividad, con una variable de ángulo de rostros se ha observado que de igual manera alcanza un 97.49% de efectividad y con condiciones lumínicas variables se ha concluido que con una luz baja se obtiene un 95.83% de efectividad, mientras que con la luz natural se ha obtenido un 100% de efectividad, siendo la luz intensa la de menor porcentaje de efectividad se tiene con un 95.83% , los fallos del sistema en ciertas ocasiones se da en que el dispositivo raspberry tiene un retraso en el video capturado en tiempo real, debido al esfuerzo computacional requerido para las operaciones de procesamiento de imágenes además de que las condiciones de captura de rostros influyen de tal manera que el procesamiento de imágenes tiende a fallar por la errónea captura de imagen. En este caso, según la investigación realizada, se le atribuye que es un sistema con muy buena precisión, debido a sus altos porcentajes de asertividad durante las pruebas del sistema.

CONCLUSIONES

Al culminar el presente proyecto de titulación se cumplieron con los objetivos específicos trazados al inicio del mismo, generando los siguientes puntos:

Se realizó el diseño y la implementación del prototipo en la empresa Sumeltronic S.A.S.

Se determinó el proceso de extracción de características de los rostros mediante redes neuronales convolucionales y la función face encoding que sirve para generar los vectores (descriptores) de los rostros detectados de una imagen, ha sido pieza clave para el desarrollo de este sistema.

Se diseñó e implementó un algoritmo a través del lenguaje de programación en Python utilizando librerías especializadas para el reconocimiento facial como lo es Face Recognition para la extracción e identificación de descriptores de los rostros.

Se implementó este sistema reduciendo los costos asociados con el control de asistencia, al eliminar la necesidad de dispositivos físicos como detectores de proximidad o lectores de huellas dactilares. Además, se disminuyen los errores humanos y se evita la intervención de personal dedicado exclusivamente al monitoreo de estos sistemas, teniendo en consideración que este proyecto de titulación se basa exclusivamente en el registro de control de asistencia (ingreso) del personal de la empresa Sumeltronic S.A.S.

Se automatizó el proceso de registro de asistencia laboral de manera eficiente y precisa. Al reemplazar los métodos manuales o basados en tarjetas de acceso, se eliminan los riesgos de fraude o inconsistencias y se optimizan los tiempos de entrada personal.

El lenguaje de programación Python brinda muchas facilidades para la programación de estos algoritmos, refiriéndonos a reconocimiento facial como a Inteligencia artificial, con la gran cantidad de librerías que ofrecen métodos eficientes para la programación de este proyecto.

Mediante los análisis estadísticos sobre los resultados en la experimentación con diferentes variables como lo es distancia, iluminación y ángulo se obtuvo que el porcentaje de aciertos varia entre 95.83% y el 97.49% dependiendo de las configuraciones de las variables utilizadas. Estos resultados son alentadores por lo que se puede confiar en el prototipo que brinda el sistema de registro día a día en su función de registro laboral.

RECOMENDACIONES

Para el correcto funcionamiento del sistema se deben de seguir las siguientes recomendaciones:

Verificar la lista de usuarios entrenados antes de cada registro al sistema, para su correcto funcionamiento, la red debe volver a entrenarse cuando hay nuevos usuarios.

Mantener buena iluminación dentro de la oficina para garantizar condiciones favorables de captura en los rostros para poder mantener la calidad de las imágenes y, por lo tanto, minimizar posibles errores relacionados con la variabilidad en las condiciones de iluminación.

Procurar colocarse a una distancia prudente para la detección del sistema, lo recomendado para este tipo de prototipos es de 2 metros como mínimo.

Recomendar al usuario de este sistema que para ejecutar esta aplicación se requiere un computador con hardware moderno, enfocándose en una GPU con una cantidad de memoria ram considerable. Esto ya que durante la experimentación se observó un retraso en el video capturado en tiempo real, debido al esfuerzo computacional requerido para las operaciones de procesamiento que constantemente se realiza. Al seleccionar un hardware más avanzado, se espera mitigar este problema, mejorando la fluidez de la aplicación.

Crear una base de datos local para cada usuario en el sistema con un área despejada en las imágenes, para así evitar tener identificaciones erróneas del reconocimiento del rostro.

Utilizar una raspberry pi de últimas generaciones para no tener problemas de procesamiento del sistema.

Impartir charlas informativas al personal que lleva el control del registro de asistencia laboral sobre estas nuevas innovaciones y recursos tecnológicos disponibles para ser empleadas en su labor diaria y que faciliten el entendimiento del sistema.

REFERENCIAS

Bibliografía

Arroyo, A., Pineda, & López, J. A. (2021). Autenticación de personas utilizando un clasificador SVM. *Revista Colombiana de Computación*, 48-57. Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=8171295

Aznarte, J. L. (2022). Sobre el uso de tecnologías de reconocimiento facial en la universidad: el caso de la UNED. *Revista Iberoamericana de Educación a Distancia*, 261-277. Obtenido de https://doi.org/10.5944/ried.25.1.31533

Bastidas, G. J. (2018). *REGISTRO DE ASISTENCIA DE ALUMNOS POR MEDIO DE RECONOCIMIENTO FACIAL UTILIZANDO VISIÓN ARTIFICIAL.*

Bautista, O. V. (2022). *Raspberry Pi.*

Bedoya, A. (6 de 2 de 2022). AVANCES Y DESAFÍOS DE LA INTELIGENCIA ARTIFICIAL. Obtenido de https://www.nationalgeographicla.com/ciencia/2023/02/que-es-la-inteligencia-artificial

C. Pérez, I. D.-R.-G. (2014). El lenguaje de programación Python. *Ciencias Holguin*, 1-13.

Caro & López, D. C. (2018). *Sistema Inteligente para el Registro de Asistencia Basado en Procesamiento Digital de Imágenes y Redes Neuronales Convolucionales.*

Centurión, D., & Almeida, C. (2022). Reconocimiento facial utilizando redes neuronales artificiales en Raspberry Pi. *FPUNE Scientific*. Obtenido de http://servicios.fpune.edu.py:83/fpunescientific/index.php/fpunescientific/article/view/234

Chen, L. S. (2020). *revisión de los modelos de redes neuronales artificiales.* Obtenido de https://www.mdpi.com/2076-3417/10/17/5776

Costa, D. (2020). *ANÁLISIS DE UN SISTEMA DE RECONOCIMIENTO FACIAL A PARTIR DE UNA BASE DE DATOS REALIZADO MEDIANTE PYTHON.*

Delgado, L. D. (2020). *Diseño de un sistema de adquisición de imágenes basado en cámaras web USB y hardware reconfigurable.*

Europea, A. T. (2023). *¿Qué es la distancia euclidiana y por qué es importante en el aprendizaje automático?* Obtenido de ACADEMIA EUROPEA DE CERTIFICACIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN: https://es.eitca.org/artificial-intelligence/eitc-ai-mlp-machine-learning-with-python/programming-machine-learning/euclidean-distance/examination-review-euclidean-distance/what-is-euclidean-distance-and-why-is-it-important-in-machine-learning/

F. Trujillo. (2022). Reconocimiento de objetos usando conjuntos pequeños de entrenamiento neuronal. *Reaxion Ciencia y Tecnologia Universitaria*. Obtenido de http://reaxion.utleon.edu.mx/Revista_ano_9_numero_3.pdf

Fernando Alain Incio Flores, D. L.-S.-U. (2023). *Modelo de red neuronal artificial para predecir resultados.*

Gómez J. (2021). Psicometría, perfiles y sesgos. El caso del reconocimiento facial. *InMediaciones de la Comunicación*, 63-81. Obtenido de https://doi.org/10.18861/ic.2021.16.2.3156

H. Vega, M. P. (2023). Reconocimiento facial mediante aprendizaje por transferencia para el control de acceso a áreas restringidas. *Revista Ibérica de Sistemas e Tecnologias de InformaçãoIberian Journal of Information Systems and Technologies*, 261-273. Obtenido de https://www.researchgate.net/publication/380631054_Reconocimiento_facial_mediante_aprendizaje_por_transferencia_para_el_control_de_acceso_a_areas_restringidas

Montero, A. F. (2022). *EL RECONOCIMIENTO FACIAL COMO INSTRUMENTO DE INVESTIGACIÓN Y PREVENCIÓN DEL DELITO.*

Navarro A., M. E. (2022). Revisión de los métodos de reconocimiento facial en imágenes RGB-D adquiridas mediante un sensor Kinect. *Revista Cubana de Ciencias Informáticas*. Obtenido de

http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992022000200157

P. Canazas, A. R. (2022). Sistema de identificación de emociones a través de reconocimiento facial utilizando inteligencia artificial. *Innovación y Software*, 140-150. Obtenido de https://www.redalyc.org/journal/6738/673870841013/

R. Campos, J. E. (2023). Sistema de reconocimiento facial para el control de accesos mediante Inteligencia Artificial. *Innovación y Software*, 24-36. Obtenido de https://www.redalyc.org/journal/6738/673874721016/

S. Altamirano, S. E. (2024). *Implementación de un sistema de control de acceso de personal utilizando inteligencia artificial para la Dirección de Tecnologías de Información y Comunicación de la UNACH.* Riobamba. Obtenido de http://dspace.unach.edu.ec/bitstream/51000/12399/1/Sampedro%20S%2C%20Stalyn%20E.%282024%29%20Implementaci%C3%B3n%20de%20un%20sistema%20de%20control%20de%20acceso%20de%20personal%20utilizando%20inteligencia%20artificial%20para%20la%20Dire.pdf

Unir. (2024). ¿Qué son las redes convolucionales y para qué se usan? *INGENIERÍA Y TECNOLOGÍA*, 1.

Vizuete, F. M. (2020). Desarrollo de un sistema de reconocimiento facial utilizando Deep Learning con OpenCV. En U. P. València.

ANEXOS

Anexo 1: Diseño de Serigrafia del Prototipo

Figura 38

Diseño de serigrafia del prototipo

Anexo 2: Datasheet Cámara web

Figura 39

Medidas Cámara Web

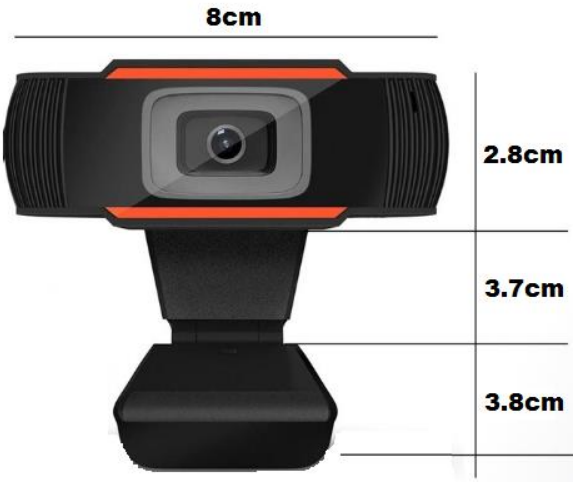


Tabla 44

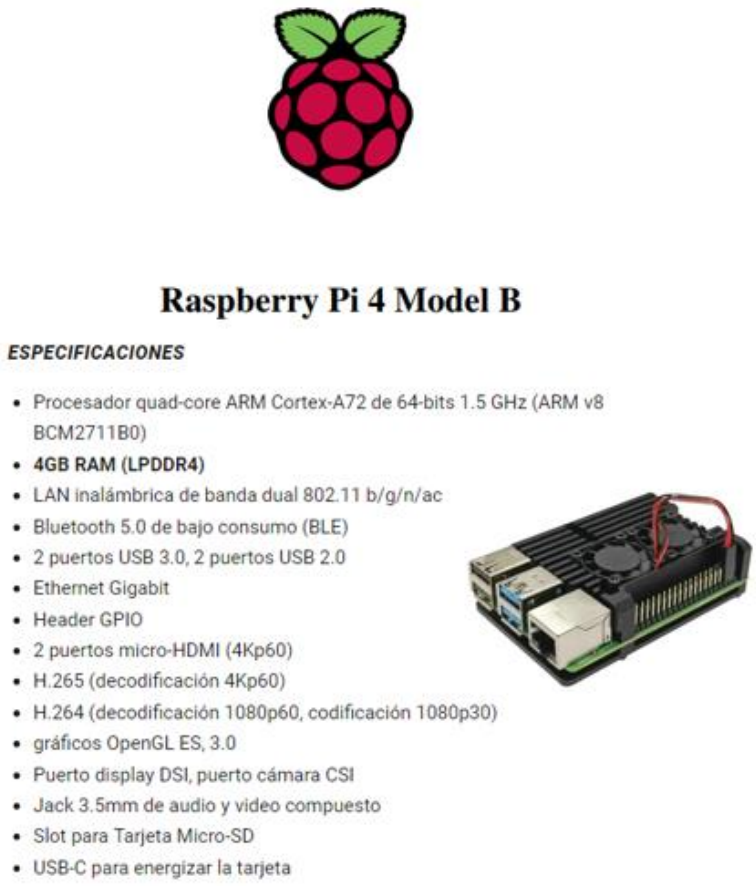
Especificaciones Cámara Web

Resolución:	720P 1280 x 720
Velocidad de fotogramas:	30fps
Tipo de sensor:	CMOS
Interfaz:	USB 2.0 + cable de audio 3.5mm
Compatible Windows 32/64 Bits	

Anexo 3: Datasheet Raspberry Pi 4 Tipo B

Figura 40

Datasheet Raspberry Pi 4 Tipo B



Raspberry Pi 4 Model B

ESPECIFICACIONES

- Procesador quad-core ARM Cortex-A72 de 64-bits 1.5 GHz (ARM v8 BCM2711B0)
- **4GB RAM (LPDDR4)**
- LAN inalámbrica de banda dual 802.11 b/g/n/ac
- Bluetooth 5.0 de bajo consumo (BLE)
- 2 puertos USB 3.0, 2 puertos USB 2.0
- Ethernet Gigabit
- Header GPIO
- 2 puertos micro-HDMI (4Kp60)
- H.265 (decodificación 4Kp60)
- H.264 (decodificación 1080p60, codificación 1080p30)
- gráficos OpenGL ES, 3.0
- Puerto display DSI, puerto cámara CSI
- Jack 3.5mm de audio y video compuesto
- Slot para Tarjeta Micro-SD
- USB-C para energizar la tarjeta

Anexo 4: Manual de Usuario

- Lo primero que debemos hacer es acceder al dispositivo raspberry, la cual se lo realiza de la siguiente manera, se busca en las opciones de Windows: escritorio remoto como se observa en la figura 41, luego accedemos a la opción introducir la dirección IP local (o remota si nos conectamos desde fuera de nuestra red) de nuestro Raspberry Pi para establecer así la conexión como se observa en la figura 42,pulsamos sobre conectar y ya estaremos dentro del escritorio de raspberry , en la figura 43 se observa el escritorio de raspberry pi.

Figura 41

Icono de conexión a acceso remoto de Windows

Figura 42

Configuración para establecer conexión a escritorio remoto

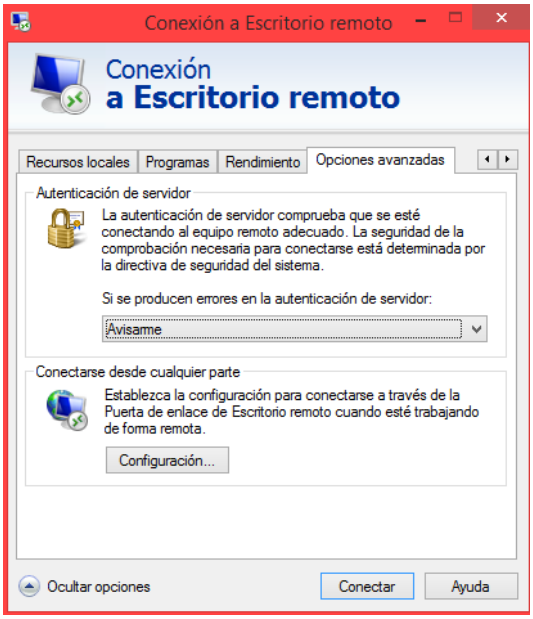
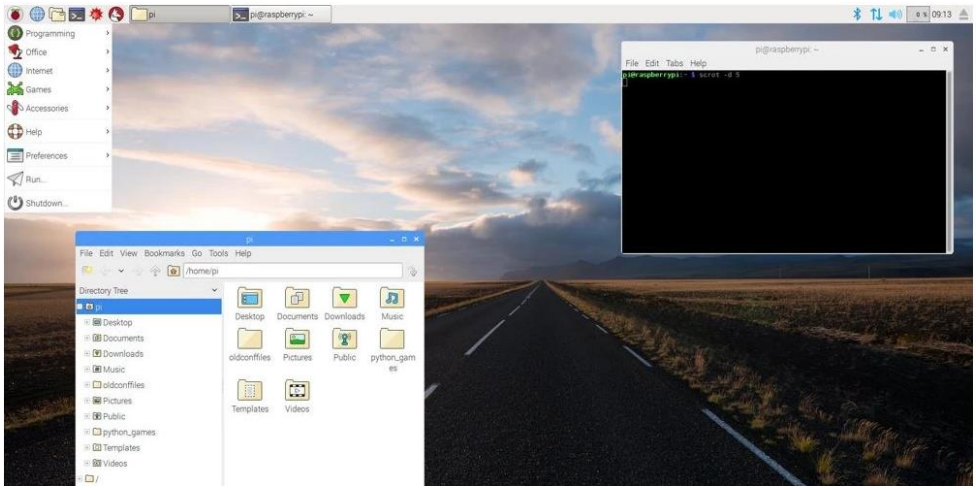


Figura 43

Escritorio de raspberry pi



- Abrimos nuestra consola de comando en el escritorio, véase figura 44, para poder acceder de forma inmediata al programa del sistema.

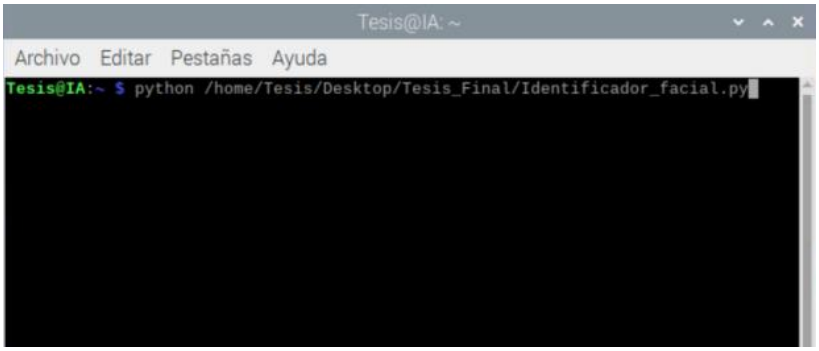
Figura 44

Consola de comandos de raspberry pi

- Como se observa en la figura 45, escribimos la palabra tesis y automáticamente aparecerá el archivo, luego lo ejecutamos el programa por medio la dirección de que se ha mostrado

Figura 45

Ruta de acceso a la programación del proyecto



- Una vez iniciado el sistema, las opciones siempre están apagadas lo que el usuario debe hacer es encenderlas según su necesidad.

Figura 46

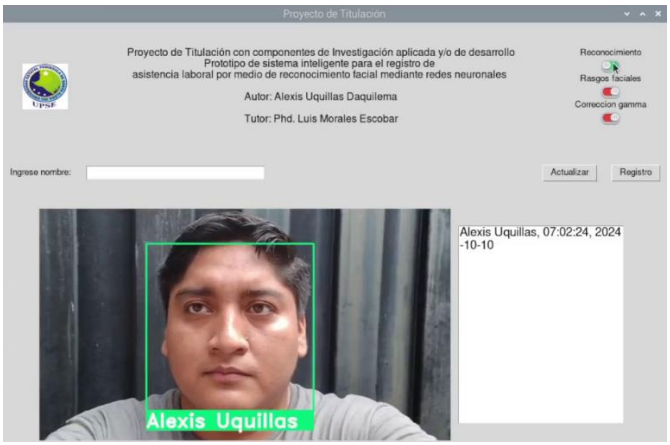
Interfaz del sistema

Tenemos 3 botones:

- El botón “reconocimiento” enciende y apaga la rutina de reconocimiento facial y reporta en el cuadro de a lado, en nombre de la persona identificada y el momento en que fue identificado, como se observa en la figura 47

Figura 47

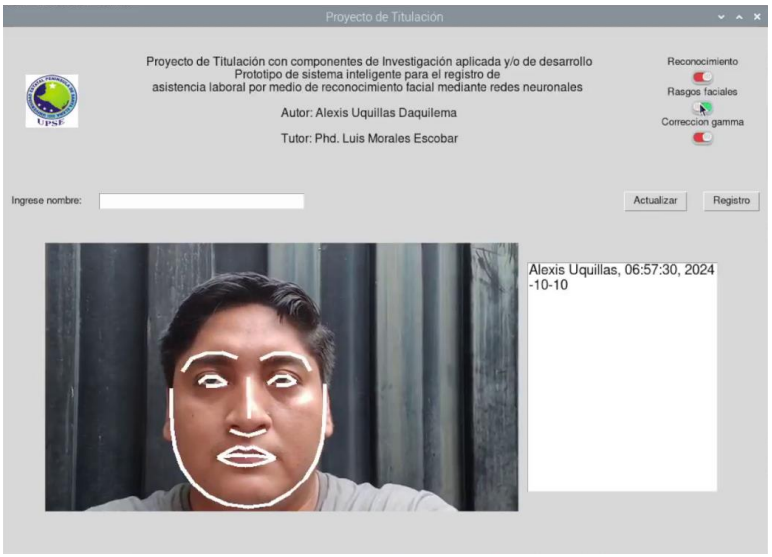
Encendido del Reconocimiento Facial



- El botón de “Rasgos faciales” realiza una demostración de cómo el programa identifica los rasgos faciales de la persona, como se observa en la figura 48.

Figura 48

Rasgos Faciales de la Persona capturada por el sistema.



- El botón de “Corrección Gama” hace realiza una corrección de valores de gamma en la imagen para que en casos que el rostro deseado se vea muy oscuro, pueda aclarar evitando en lo posible que se pierda detalles en los rasgos faciales, como se observa en la figura 49.

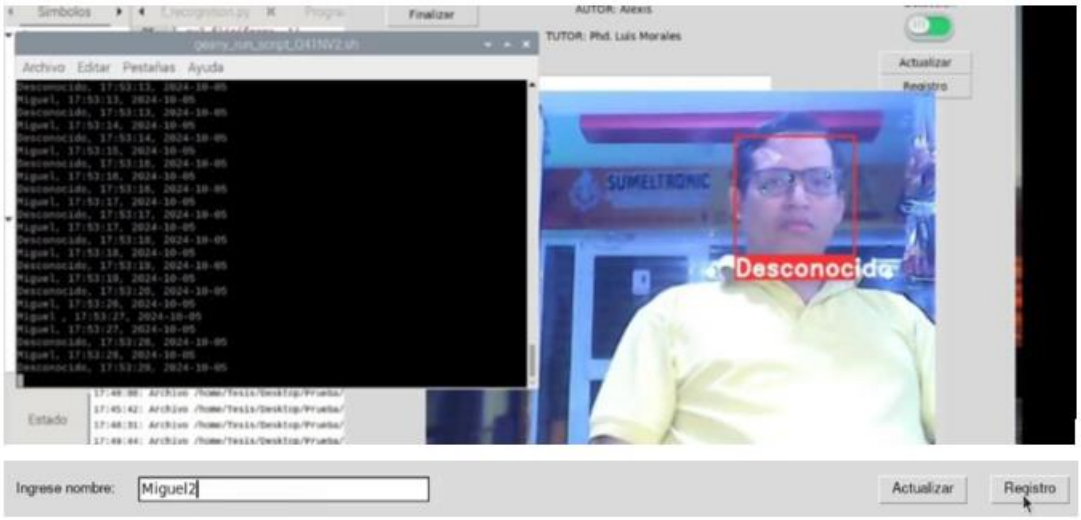
Figura 49

Corrección gamma del programa para mayor resolución.

- El botón “Registro” toma una foto en tiempo real de la persona en la cámara, para guardarla con el nombre ingresado en la barra de texto en el cual el texto a su izquierda indica “ingrese nombre”, y esta imagen llegue ser considerada en la actualización de la base de datos, como se observa en la figura 50.

Figura 50

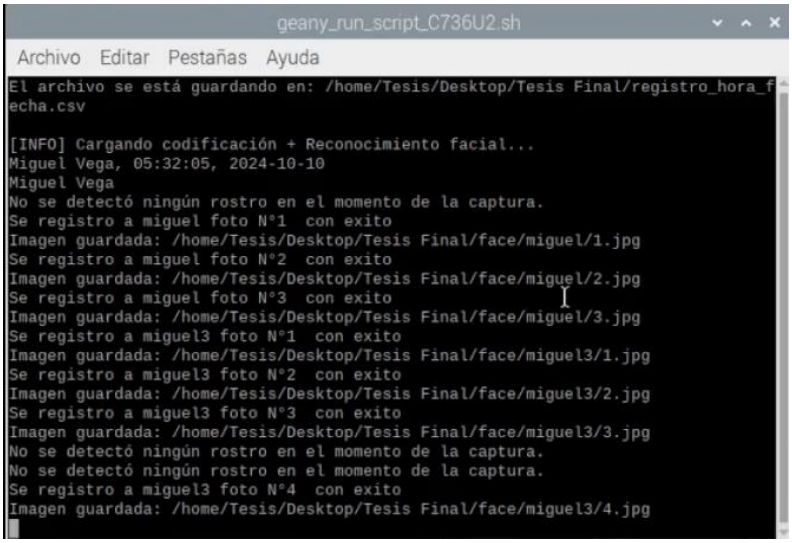
Registro De una nueva persona para el reconocimiento facial dentro del sistema



- A continuación, el reporte en la terminal de las fotos registradas del nuevo ingreso, como se observa en la figura 51

Figura 51

Registro de Nuevo Ingreso a base de datos

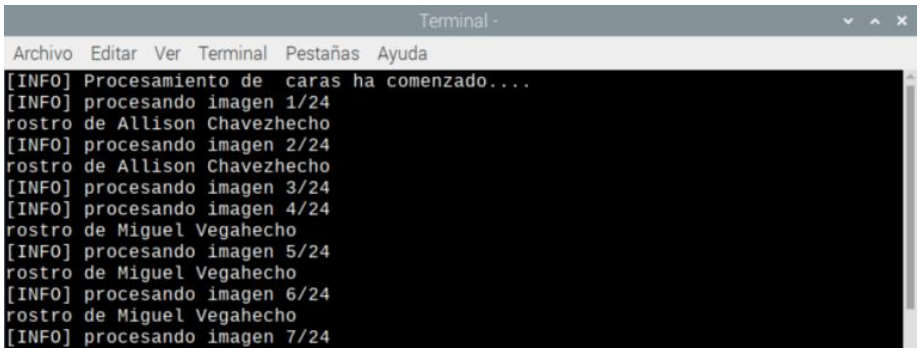


- Y el Botón “Actualizar” actualiza la base de datos con las nuevas fotos para que pueda reconocer a las nuevas personas que fueron registradas previamente, se

observa el entrenamiento con las nuevas adiciones a la base de datos en la figura 52, luego es necesario reiniciar el programa una vez realizado esto.

Figura 52

Entrenamiento de la red neuronal



- El sistema está listo para verificar los ingresos a la empresa, se observa en la figura 53.

  Figura 53

  Pruebas del prototipo en la Empresa Sumeltronic S.A.S.

- Finalmente, corroboramos el almacenamiento de datos del sistema de la captura de imagen al momento de registrarse y la creación de los archivos de Excel del día en el que recopilo los registros, como se observa en la figura 54.

Figura 54.

Visualización de registro de datos de reconocimiento facial del sistema

Anexo 5: Oficio del Gerente General de la empresa “Sumeltronic S.A.S.”

Figura 55

Carta emitida por el gerente de la empresa Sumeltronic S.A.S

Guayaquil, Lunes 07 de Octubre del 2024

Ing. Andrade Vera Alicia, Msc.

COORDINADORA MAESTRÍA EN ELECTRÓNICA Y AUTOMATIZACIÓN

Presente. –

SUMELTRONIC

De mis consideraciones:

Por medio de la presente notificamos a usted, que el estudiante Alexis Manolo Uquillas Daquilema con CEDULA DE IDENTIDAD N.0952506558, cumplió con éxito el trabajo de titulación " Prototipo de Sistema Inteligente Para el Registro de Asistencia Laboral por Medio de Reconocimiento Facial Mediante Redes Neuronales" para la empresa SUMELTRONIC S.A.S, el cual se encuentra totalmente operativo.

Suscribo el presente pronunciamiento, que será usado para trámites académicos.

Atentamente,

Ing. Cristian Fabian Naranjo Ortiz

Gerente General

SUMELTRONIC S.A.S.

SUMELTRONIC

Anexo 6: Código Fuente del Proyecto

```
# biblioteca para crear interfaces gráficas de usuario (GUI).
import tkinter as tk
# módulos de biblioteca Pillow (PIL).
from PIL import Image, ImageTk, ImageDraw
# biblioteca OpenCV ampliamente utilizada para procesamiento de imágenes y visión por computadora.
import cv2
# datetime permite trabajar con fechas y horas.
from datetime import datetime
# interactua con el sistema operativo, como trabajar con rutas de archivos y directorios.
import os
# contiene funciones auxiliares para trabajar con archivos de imágenes o videos.
from imutils import paths
# permite realizar tareas como detección y reconocimiento facial usando modelos preentrenados.
import face_recognition
# que permite la serialización y deserialización de objetos en Python.
import pickle
# que permite leer y escribir archivos CSV
import csv
# trabaja con arreglos multidimensionales y realizar cálculos matemáticos de alto rendimiento.
import numpy as np


# variable que guarda la ruta de el directorio de el proyecto.
```

```
ruta = "/home/Tesis/Desktop/Tesis Final"

# clasificador en cascada (Haar Cascade) que se especializa en detectar caras frontales.
face_cascade_frontal = cv2.CascadeClassifier(
    cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

# clasificador en cascada que detecta perfiles de caras (rostros de lado)
face_cascade_profile = cv2.CascadeClassifier(
    cv2.data.haarcascades + 'haarcascade_profileface.xml')

# Variable 'currentname' para modificarse solo cuando se identifique a una nueva persona.
currentname = "Desconocido"

# Determina rostros a partir del archivo encodings.pickle, modelo creado desde la función "entrenamiento".py
encodingsP = "150*150-encodings.pickle"


# crear una carpeta y un archivo CSV en una ubicación específica si aún no existen
def crear_carpeta_y_archivo():
    # Definir la ruta de la carpeta
    carpeta = os.path.expanduser(ruta + "/registro")

    # Crear la carpeta si no existe
    if not os.path.exists(carpeta):
```

```
        os.makedirs(carpeta)
        print(f"Carpeta creada en: {carpeta}")

    # Crear la ruta para el archivo CSV
    archivo_csv = os.path.join(carpeta, "registro_hora_fecha.csv")

    return archivo_csv


# llama a la función crear_carpeta_y_archivo() y almacena la ruta del archivo CSV resultante en la variable
filepath = crear_carpeta_y_archivo()

# Carga los  descriptores de los rostros conocidos para la detección de rostros
print("[INFO] Cargando codificación + Reconocimiento facial...")
data = pickle.loads(open(encodingsP, "rb").read())


# Función para crear una carpeta si no existe
def crear_carpeta_si_no_existe(ruta):
    if not os.path.exists(ruta):
        os.makedirs(ruta)

# Función para contar las imágenes existentes en la carpeta de la persona
```

```
def contar_imagenes_existentes(ruta_carpeta_persona):
    imagenes_existentes = [archivo for archivo in os.listdir(
        ruta_carpeta_persona) if archivo.endswith(('.png', '.jpg', '.jpeg'))]
    return len(imagenes_existentes)



# Función de registro
def registro():
    global frame
    # Obtener el texto de la barra de entrada
    nombre_persona = name_entry .get()
    # Definir la ruta de la carpeta principal y la subcarpeta con el nombre de la persona
    ruta_carpeta_face = os.path.join(os.path.expanduser(ruta), "face")
    ruta_carpeta_persona = os.path.join(ruta_carpeta_face, nombre_persona)

    # Crear las carpetas si no existen
    crear_carpeta_si_no_existe(ruta_carpeta_face)
    crear_carpeta_si_no_existe(ruta_carpeta_persona)

    # Contar cuántas imágenes ya existen en la carpeta de la persona
    contador_imagenes = contar_imagenes_existentes(ruta_carpeta_persona)

    # Mostrar la imagen en un recuadro
    contador_imagenes += 1
```

```
    # Guardar la imagen con un nombre progresivo
    nombre_imagen = f"{contador_imagenes}.jpg"

    # ruta completa donde se guardará la imagen, uniendo la ruta de la carpeta de la
persona
    ruta_imagen = os.path.join(ruta_carpeta_persona, nombre_imagen)
    # Convierte el fotograma (frame) de su formato original BGR  a GREY.
    gris = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # devuelve una lista de las coordenadas de los rostros detectados
    faces = face_cascade_frontal.detectMultiScale(gris, 1.2, 5)
    # Itera sobre cada rostro detectado en la imagen.
    for (x, y, w, h) in faces:
        # Guarda el fotograma en la ruta especificada.
        cv2.imwrite(ruta_imagen + ".jpg", frame)
        # Muestra un mensaje en la consola indicando el registro
        print(f"Se registro a {nombre_persona} foto N°{
            contador_imagenes} con exito en {ruta_imagen}")
        break  # Guardar solo el primer rostro detectado
    # se ejecutará solo si no se detectan rostros
    else:
        # Muestra un mensaje en la consola indicando el registro fallido
        print("No se detectó ningún rostro en el momento de la captura.")
```

```
# Función de entrenamiento
def entrenamiento():
    global data

    # Nuestras imágenes están ubicadas en la carpeta de datos
    print("[INFO] Procesamiento de  caras ha comenzado....")

    # Define la subcarpeta dentro de la ruta principal
    ruta2 = ruta + "/face"

    # función para listar todas las rutas de las imágenes en la carpeta especificada.
    # se almacenan en la lista imagePaths.
    imagePaths = list(paths.list_images(ruta2))

    # Inicializar la lista de codificaciones conocidas y nombres conocidos
    # Crea una lista vacía para almacenar las codificaciones de las caras.
    knownEncodings = []
    # Crea una lista vacía para almacenar los nombres de las personas.
    knownNames = []

    # Recorre las imágenes de la ruta
    # Bucle "for" que itera sobre las rutas de las imágenes.
    # enumerate() se utiliza para obtener tanto el índice i como la imagePath de cada imagen.
    for (i, imagePath) in enumerate(imagePaths):
```

```
    # Extrae el nombre de la persona de la ruta de la imagen
    print("[INFO] procesando imagen {}/{}".format(i + 1, len(imagePaths)))

    # Divide la ruta de la imagen en componentes con (os.path.sep).
    # Luego, obtiene el penúltimo componente de la ruta, que se asume que es el nombre de la persona.
    name = imagePath.split(os.path.sep)[-2]

    # Carga la imagen desde la ruta especificada
    image = cv2.imread(imagePath)

    # Convierte la imagen cargada de BGR a escala de grises
    gris = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Detectar rostros frontales

    rostros = face_cascade_frontal.detectMultiScale(
        gris, scaleFactor=1.1, minNeighbors=5, minSize=(150, 150))
    # Si no se encuentran rostros frontales, intentar con rostros de perfil
    if len(rostros) == 0:
        rostros = face_cascade_profile.detectMultiScale(
            gris, scaleFactor=1.1, minNeighbors=5, minSize=(150, 150))

    for (x, y, w, h) in rostros:
        # Recortar el rostro
```

```
rostro_recortado = image[y:y+h, x:x+w]
# Redimensionar el rostro manteniendo la proporción
alto_recorte, ancho_recorte = rostro_recortado.shape[:2]

# Definir las dimensiones máximas permitidas para el recorte

ancho_maximo = 150  # máximo ancho permitido
alto_maximo = 150   # máximo alto permitido

if ancho_recorte > ancho_maximo or alto_recorte > alto_maximo:
    # Calcular el factor de escala manteniendo la proporción
    factor_escala = min(
        ancho_maximo / ancho_recorte, alto_maximo / alto_recorte)
    nuevo_ancho = int(ancho_recorte * factor_escala)
    nuevo_alto = int(alto_recorte * factor_escala)
    rostro_recortado = cv2.resize(
        rostro_recortado, (nuevo_ancho, nuevo_alto))

# Ubica las caras en la imagen.
# Especifica el eso de el modelo basado en una red neuronal convolucional (CNN) .
boxes = face_recognition.face_locations(
    rostro_recortado, model="cnn")
```

```
        # Calcula los descriptores para los rostros detectados en la imagen.
        # Devuelve una lista de vectores de características de cada rostro detectado.
        encodings = face_recognition.face_encodings(
            rostro_recortado, boxes)

        # Inicia un bucle "for" que itera sobre cada descriptor en la lista encodings.
        for encoding in encodings:

            # Añade el descriptor actual a la lista knownEncodings
            knownEncodings.append(encoding)
            # Añade el nombre de la persona a la lista knownNames
            knownNames.append(name)

    # Deposita las codificaciones faciales + nombres en el disco
    print("[INFO] Serializando descriptores...")

    # Crea un diccionario llamado data que contiene dos claves: "encodings" y "names".
    # "encodings" almacena la lista de codificaciones faciales,
    # y "names" almacena la lista de nombres correspondientes.
    new_data = {"encodings": knownEncodings, "names": knownNames}

    # Abre (o crea) un archivo llamado 150*150-encodings.pickle en modo de escritura binaria ("wb").
    # Este archivo se utilizará para guardar los datos serializados.
    f = open("150*150-encodings.pickle", "wb")
```

```
    # Conveirte el diccionario data en un formato de bytes (serializarlo)
    # y luego escribe esos bytes en el archivo abierto (f).
    f.write(pickle.dumps(new_data))

    print("[INFO] Entrenamiento concluido...")

    # Cargar los  descriptores de los rostros conocidos para la detección de rostros
    data = pickle.loads(open(encodingsP, "rb").read())


def Rasgos_faciales(frame):
    # Crea una copia de la imagen original (frame) y la almacena en orig2
    orig2 = frame.copy()

    # Obtener dimensiones originales de la imagen
    alto_original, ancho_original = orig2.shape[:2]

    # Redimensionar la imagen a una tercera parte de su tamaño original
    ancho_tercio = ancho_original // 2
    alto_tercio = alto_original // 2
    orig2 = cv2.resize(orig2, (ancho_tercio, alto_tercio))

    # Función face_landmarks para detectar y extraer los puntos de referencia faciales.
    face_landmarks_list = face_recognition.face_landmarks(orig2)
```

```
    # Crea un objeto imagedraw de PIL para que podamos dibujar en la imagen
    pil_image = Image.fromarray(orig2)

    # Crea un objeto "d", que se utilizará para dibujar sobre la imagen pil_image.
    d = ImageDraw.Draw(pil_image)

    # Inicia un bucle for que itera sobre cada conjunto de puntos de referencia faciales detectados
    for face_landmarks in face_landmarks_list:

        # Imprime la ubicación de cada característica facial en esta imagen
        for facial_feature in face_landmarks.keys():
            print("The {} in this face has the following points: {}".format(
                facial_feature, face_landmarks[facial_feature]))

        # Trazamos cada característica facial en la imagen con una línea
        for facial_feature in face_landmarks.keys():
            d.line(face_landmarks[facial_feature], width=5)

    # Convierte el objeto de imagen de PIL de vuelta a un formato de matriz de NumPy
    image_with_drawing = np.array(pil_image)
    # Redimensiona la imagen dibujada
    imagen_restaurada = cv2.resize(
        image_with_drawing, (ancho_original, alto_original))
```

```
    # Devuelve la imagen al lugar donde se llamó a la función.
    return imagen_restaurada



def Reconocimiento(frame):
    global currentname

    orig = frame.copy()
    gris = cv2.cvtColor(orig, cv2.COLOR_BGR2GRAY)

    # Detectar rostros frontales ligero

    rostros = face_cascade_frontal.detectMultiScale(
        gris, scaleFactor=1.1, minNeighbors=2, minSize=(50, 50))
    # Si no se encuentran rostros frontales, intentar con rostros de perfil
    if len(rostros) == 0:

     # Detectar rostros laterales ligero
        rostros = face_cascade_profile.detectMultiScale(
            gris, scaleFactor=1.1, minNeighbors=2, minSize=(50, 50))

    for (x, y, w, h) in rostros:
        # Recortar el rostro
        rostro_recortado = frame[y:y+h, x:x+w]
        ancho_deseado = 100
```

```
# Redimensionar el rostro manteniendo la proporción
alto_original, ancho_original = rostro_recortado.shape[:2]

factor_escala = ancho_deseado / ancho_original
nuevo_alto = int(alto_original * factor_escala)

# Del recorte del rostro, a escalado menor
rostro_redimensionado = cv2.resize(
    rostro_recortado, (ancho_deseado, nuevo_alto))

# Detectar rostro de face recognition
boxes = face_recognition.face_locations(
    rostro_redimensionado, model="cnn")
encodings = face_recognition.face_encodings(
    rostro_redimensionado, boxes)

names = []

# Inicia un bucle "for" que itera sobre cada descriptor en la lista encodings.
for encoding in encodings:
    # Compara la codificación facial actual con todas las codificaciones conocidas.
    # matches será una lista de valores booleanos del resultaod de las coincidencias.
    matches = face_recognition.compare_faces(data["encodings"],
                                             encoding)
    # Si el rostro no es reconocido, imprimirá "Desconocido"
```

```
            name = "Desconocido"

            # Verifica si hemos encontrado una coincidencia.
            if True in matches:

                # Encuentra los índices de todas las caras coincidentes
                # y luego inicializa un diccionario para contar el número
                # ctotal de veces que se coincidió cada cara.
                matchedIdxs = [i for (i, b) in enumerate(matches) if b]
                counts = {}

                # Recorre los índices coincidentes y mantiene
                # un conteo para cada cara reconocida.
                for i in matchedIdxs:
                    name = data["names"][i]
                    counts[name] = counts.get(name, 0) + 1

                # Determina la cara reconocida con el mayor número de votos.
                name = max(counts, key=counts.get)

            # Si alguien en tu conjunto de datos es identificado, imprime su nombre en la
pantalla.
            if currentname != name:
                currentname = name
```

```
# Obtener la hora actual
hora_actual = datetime.now().strftime("%H:%M:%S")

# Obtener la fecha actual
fecha_actual = datetime.now().strftime("%Y-%m-%d")

print(f"{name}, {hora_actual}, {fecha_actual}")

# Guardar en el archivo CSV
# Comprobar si el archivo CSV ya existe
file_exists = os.path.isfile(filepath)

# Abrir el archivo CSV en modo 'a' (append)
with open(filepath, 'a', newline='') as archivo_csv:
    writer = csv.writer(archivo_csv)

    # Si el archivo no existe, escribir el encabezado
    if not file_exists:

        # Encabezados del CSV
        writer.writerow(["Nombre", "Fecha", "Hora"])

    # Escribir la hora y la fecha en el archivo CSV
    writer.writerow([name, fecha_actual, hora_actual])
```

```
                # Imprime el nombre, la fecha y la hora al cuadro de texto
                Reportaje(name, fecha_actual, hora_actual)

                # Establecimiento de un color (en este caso verde)
                color = (125, 220, 0)

                # Dibuja un rectángulo relleno en la imagen original para mostrar el
nombre.
                cv2.rectangle(orig, (x, y + h),
                              (x + w, y + h + 30), color, -1)

                # Dibuja un rectángulo alrededor del rostro en la imagen original
                cv2.rectangle(orig, (x, y), (x + w, y + h), color, 2)

                # Coloca el nombre en la imagen justo debajo del rectángulo que rodea la
cara
                # El color del texto es blanco.
                cv2.putText(orig, name, (x, y + h + 25), 2,
                            1, (255, 255, 255), 2, cv2.LINE_AA)

                # Se crea variable que contiene los datos de la imagencon la extensión
".jpg"
                nombre_imagen = f"{currentname} {
                   fecha_actual} {hora_actual}.jpg"

                # Se genera la ruta completa donde se almacenará la imagen.
```

```
                ruta_imagen = os.path.join(
                    ruta + "/registro", nombre_imagen)

                # Guarda la imagen en la ubicación especificada.
                cv2.imwrite(ruta_imagen + ".jpg", orig)

        # Se ejecuta cuando no se ha reconocido la cara.
        # Con nombre "Desconocido" y el color del recuadro a rojo.
        else:
            name = "Desconocido"
            color = (50, 50, 255)

        # Actualiza la lista de nombres
        names.append(name)

        # Misma funcion de dibujado de recuadro anteriorm pero de color rojo
        cv2.rectangle(frame, (x, y + h), (x + w, y + h + 30), color, -1)
        cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
        cv2.putText(frame, name, (x, y + h + 25), 2,
                    1, (255, 255, 255), 2, cv2.LINE_AA)

# Devuelve la imagen al lugar donde se llamó a la función.
return frame
```

```
def ajustar_gamma(image, gamma=1.0):
    # Calcula el valor inverso de gamma
    invGamma = 1.0 / gamma
    # Crea una tabla de transformación utilizando una lista por comprensión.
    table = np.array([(i / 255.0) ** invGamma *
                      255 for i in range(256)]).astype("uint8")

    # Aplica la tabla de transformación table a la imagen
    # El resultado es la imagen ajustada con la corrección gamma aplicada.
    # Retorna la imagen modificada.
    return cv2.LUT(image, table)


# Función que cambia la imagen del switch
def switch_toggle():

    # Declara la variable como global.
    # Accediendo a una variable que fue definida fuera de la función.
    # Esto permite modificar su valor dentro de la función.
    global switch_on

    # Evalúa si la variable es True
    if switch_on:
        # Cambia a la imagen de "off"
        switch_img.config(image=img_off)
```

```
        # Cambia el valor de switch_on a False
        switch_on = False
    else:
        # Cambia a la imagen de "on"
        switch_img.config(image=img_on)
        # Cambia el valor de switch_on a True
        switch_on = True

# Función que cambia la imagen del switch2


def switch2_toggle():
    global switch2_on
    if switch2_on:
        switch2_img.config(image=img_off)
        switch2_on = False
    else:
        switch2_img.config(image=img_on)
        switch2_on = True

# Función que cambia la imagen del switch3


def switch3_toggle():
    global switch3_on
```

```
    if switch3_on:
        switch3_img.config(image=img_off)
        switch3_on = False
    else:
        switch3_img.config(image=img_on)
        switch3_on = True

# Función para actualizar el cuadro de texto


def Reportaje(name, hora_actual, fecha_actual):

    time_report.insert(tk.END, f"{name}, {fecha_actual}, {hora_actual}\n")
    # Desplaza el texto hacia abajo para mostrar la última línea
    time_report.see(tk.END)


# Función para mostrar la transmisión de la cámara
def show_camera():
    global frame
    ret, frame = cap.read()  # Captura un frame de la cámara

    # Ajustar gamma (valores menores que 1 oscurecen, mayores que 1 aclaran)

    if ret:
```

```
    if switch3_on:
        # Ajustar gamma (valores menores que 1 oscurecen, mayores que 1 aclaran)
        frame = ajustar_gamma(frame, gamma=1.7)

    if switch_on:
        frame = Reconocimiento(frame)

    if switch2_on:
        frame = Rasgos_faciales(frame)
else:
    cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
    ret, frame = cap.read()

# Convierte el frame a RGB
frame2 = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
img = Image.fromarray(frame2)  # Convierte a objeto de imagen de PIL

# Mantener la relación de aspecto
max_width = 1080  # Aumentar el ancho máximo
max_height = 1080  # Aumentar la altura máxima
# Redimensiona manteniendo la relación de aspecto
# img.thumbnail((max_width, max_height), Image.ANTIALIAS)

img_tk = ImageTk.PhotoImage(img)  # Convierte a PhotoImage de Tkinter
```

```
    camera_label.imgtk = img_tk  # Asigna la imagen al label
    # Actualiza el label con la nueva imagen
    camera_label.configure(image=img_tk)
    # Vuelve a llamar a la función después de 10 ms
    camera_label.after(10, show_camera)



# Crear ventana principal
# representa la ventana principal de la aplicación.
root = tk.Tk()
# será el nombre que aparecerá en la barra de título de la ventana.
root.title("Proyecto de Titulación")
# Ajustar el tamaño de la ventana
root.geometry("1000x800")

# Cargar el ícono de la ventana
# Reemplazar con tu icono
root.iconphoto(False, tk.PhotoImage(file="icon.png"))

# Crear el marco superior sin bordes
# bd=0: borde del marco. relief=tk.FLAT:estilo del borde del marco.
top_frame = tk.Frame(root, bd=10, relief=tk.FLAT)
# organiza por método pack().
# espacio vertical de píxeles (padding) alrededor del marco.
# fill=tk.X: se expande horizontalmente para llenar la ventana
```

```
# fill=tk.Y: se expande verticalalmente para llenar la ventana
# fill=tk.BOTH: para llenar en ambas direcciones
top_frame.pack(pady=10, fill=tk.X)

# Colocar el logotipo en la esquina superior izquierda
logo = Image.open("logo.png")  # Reemplaza con tu archivo de logotipo
# logo.resize Cambia el tamaño de la imagen #Image.ANTIALIAS: suavizado de la imagen
logo = logo.resize((70, 70), Image.ANTIALIAS)
# es necesario porque Tkinter no maneja imágenes PIL sin conversión.
logo_img = ImageTk.PhotoImage(logo)
# top_frame es el contenedor de la etiqueta
logo_label = tk.Label(top_frame, image=logo_img)
# side=tk.LEFT: la etiqueta debe colocarse lado izquierdo del marco
logo_label.pack(side=tk.LEFT, padx=20)

contenido_parrafo = '''
Proyecto de Titulación con componentes de Investigación aplicada y/o de desarrollo
Prototipo de sistema inteligente para el registro de
asistencia laboral por medio de reconocimiento facial mediante redes neuronales

Autor: Alexis Uquillas Daquilema

Tutor: Phd. Luis Morales Escobar
'''
```

```
# Párrafo centrado
paragraph_label = tk.Label(top_frame, text=contenido_parrafo, font=(
    "Helvetica", 12), justify=tk.CENTER)
# paragraph_label = tk.Label(top_frame, text="Este es un párrafo centrado de ejemplo.\n Puedes personalizar el texto.", font=("Helvetica", 12), justify=tk.CENTER)
paragraph_label.pack(side=tk.LEFT, expand=True)



# Crear un marco para el switch y el texto de "Reconocimiento" sin bordes
switch_frame = tk.Frame(top_frame, bd=0, relief=tk.FLAT)
switch_frame.pack(side=tk.RIGHT, padx=20)

# Texto de "Reconocimiento" centrado encima del switch
switch_label = tk.Label(switch_frame, text="Reconocimiento",
                        font= ("Helvetica", 10), anchor="center")
switch_label.pack(side=tk.TOP)

# Cargar las imágenes del switch
img_on = ImageTk.PhotoImage(Image.open("switch_on.png").resize(
    (30, 20), Image.ANTIALIAS))  # Reemplaza con tu archivo de imagen "on"
img_off = ImageTk.PhotoImage(Image.open("switch_off.png").resize(
    (30, 20), Image.ANTIALIAS))  # Reemplaza con tu archivo de imagen "off"

# Configuración inicial del switch
```

```
switch_on = False
switch_img = tk.Label(switch_frame, image=img_off)
switch_img.pack(side=tk.TOP)
# "<Button-1>": evento que se dispara cuando se clickea con el botón izquierdo
switch_img.bind("<Button-1>", lambda event: switch_toggle())

# Texto de "Rasgos faciales" centrado encima del switch
switch2_label = tk.Label(switch_frame, text="Rasgos faciales", font=(
    "Helvetica", 10), anchor="center")
switch2_label.pack(side=tk.TOP)

# Configuración inicial del switch
switch2_on = False
switch2_img = tk.Label(switch_frame, image=img_off)
switch2_img.pack(side=tk.TOP)
switch2_img.bind("<Button-1>", lambda event: switch2_toggle())

# Texto de "Correccion gamma" centrado encima del switch
switch3_label = tk.Label(switch_frame, text="Correccion gamma", font=(
    "Helvetica", 10), anchor="center")
switch3_label.pack(side=tk.TOP)

# Configuración inicial del switch
switch3_on = False
switch3_img = tk.Label(switch_frame, image=img_off)
```

```
switch3_img.pack(side=tk.TOP)
switch3_img.bind("<Button-1>", lambda event: switch3_toggle())



# Crear un nuevo marco para los elementos de abajo (texto, barra de ingreso de texto y botones) sin bordes
bottom_frame = tk.Frame(root, bd=0, relief=tk.FLAT)
bottom_frame.pack(pady=20, fill=tk.X)

# Texto "Ingrese nombre"
name_label = tk.Label(bottom_frame, text="Ingrese nombre:",
                      font=("Helvetica", 10))
name_label.pack(side=tk.LEFT, padx=10)

# Barra de ingreso de texto
name_entry = tk.Entry(bottom_frame, width=30)
name_entry.pack(side=tk.LEFT, padx=10)

# Botón "Registro"
register_button = tk.Button(bottom_frame, text="Registro", font=(
    "Helvetica", 10), command=registro)
register_button.pack(side=tk.RIGHT, padx=10)

# Botón "Actualizar"
update_button = tk.Button(bottom_frame, text="Actualizar", font=(
```

```
    "Helvetica", 10), command=entrenamiento)
update_button.pack(side=tk.RIGHT, padx=10)

# Marco para la cámara sin bordes
camera_frame = tk.Frame(root, bd=0, relief=tk.FLAT)
camera_frame.pack(pady=20)

# Label para mostrar la transmisión de la cámara
camera_label = tk.Label(camera_frame)
camera_label.pack(side=tk.LEFT)

# Cuadro de texto para reportar la hora, aumentando la altura
time_report = tk.Text(camera_frame, width=25, height=15,
                      font=("Helvetica", 14))  # Aumentar height
time_report.pack(side=tk.RIGHT, padx=10)


# Iniciar la captura de video
cap = cv2.VideoCapture(0)  # Cambia el índice si tienes múltiples cámaras


if not cap.isOpened():
    print("Error al abrir función de video")
    exit()
```

```
# Llamar a las funciones para actualizar la hora y mostrar la cámara
show_camera()



# Iniciar el bucle principal
root.mainloop()

# Liberar la cámara al cerrar la aplicación
cap.release()
cv2.destroyAllWindows()
```