UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA DE COMPUTACIÓN

DESARROLLO DE UN SISTEMA PORTABLE DE PROCESAMIENTO DE DATOS OBTENIDOS POR CÁMARA PARA GENERAR DATASETS DE FIGURA HUMANA PARA LOS PROFESORES DE LA UNIVERSIDAD POLITÉCNICA SALESIANA

Trabajo de titulación previo a la obtención del

Título de Ingeniero en Ciencias de la Computación

AUTOR: Carlos Gabriel Revelo Landeta

TUTOR: Holger Raúl Ortega Martínez

Quito-Ecuador

2024

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Carlos Gabriel Revelo Landeta con documento de identificación N° 1725454613 manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 28 de febrero del 2024

Atentamente,

Carlos Gabriel Revelo Landeta

1725454613

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Carlos Gabriel Revelo Landeta con documento de identificación No. 1725454613, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del Proyecto Técnico: "Desarrollo de un sistema portable de procesamiento de datos obtenidos por cámara para generar datasets de figura humana para los profesores de la universidad politécnica salesiana" , el cual ha sido desarrollado para optar por el título de: Ingeniero en Ciencias de la Computación , en la Universidad Politécnica Salesiana", quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 28 de febrero del 2024

Atentamente,

Carlos Gabriel Revelo Landeta

1725454613

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Holger Raúl Ortega Martínez con documento de identificación N° 1708182728, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DESARROLLO DE UN SISTEMA PORTABLE DE PROCESAMIENTO DE DATOS OBTENIDOS POR CÁMARA PARA GENERAR DATASETS DE FIGURA HUMANA PARA LOS PROFESORES DE LA UNIVERSIDAD POLITÉCNICA SALESIANA, realizado por Carlos Gabriel Revelo Landeta con documento de identificación N° 1725454613, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 28 de febrero del 2024

Atentamente,

Fis. Holger Raúl Ortega Martínez, MSc.

1708182728

DEDICATORIA Y AGRADECIMIENTO

Quisiera expresar mi profundo agradecimiento a quienes han contribuido de manera significativa en la culminación de este arduo pero gratificante viaje académico.

A mi querido padre, fuente inagotable de inspiración y apoyo incondicional, tus enseñanzas y valores han sido el faro que iluminó mi camino hacia el conocimiento. A pesar de la falta de compresión y los problemas eres una guía en cada logro alcanzado.

A mi amada madre, aunque ya no esté entre nosotros, su legado de amor, fortaleza y perseverancia continúa siendo mi mayor motivación y este logro es también en honor a ti, que siempre creíste en mis capacidades y tuviste fe en mí.

A mis apreciados hermanos, compañeros silenciosos de este trayecto, su apoyo constante y aliento han sido mi fortaleza. Aunque no estuvieron tan presentes de manera física, siempre sentí su respaldo y comprensión en los momentos más desafiantes.

A mi apreciado docente tutor, cuya sabiduría, orientación y paciencia han sido fundamentales para mi desarrollo académico. Sus consejos y dedicación han dejado una huella imborrable en mi formación.

A la Universidad Politécnica Salesiana, mi segunda casa durante estos años de estudio. Agradezco a esta institución por brindarme las herramientas y conocimientos necesarios para enfrentar los desafíos académicos y personales.

A todos aquellos que, de una forma u otra, contribuyeron a mi crecimiento y éxito académico, les estoy profundamente agradecido. Este logro no solo es mío, sino también de quienes han sido parte de mi trayectoria.

Con gratitud y dedicación, Carlos Revelo

ÍNDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN

El sistema es capaz de satisfacer la necesidad de conjuntos de datos para el entrenamiento de sistemas de reconocimiento de personas, ya sea a través de reconocimiento facial o mediante la información adicional obtenida de la figura humana. La información recopilada se obtiene de un circuito integrado tipo FPGA con un sensor de cámara, programado mediante un software desarrollado en Python, que captura imágenes en tiempo real del entorno donde se ubica. Posteriormente, procesa estas imágenes para identificar la figura o figuras presentes en el foco de la cámara, determinando si están de frente o no, y finalmente, separando sus rostros.

Toda esta información se guarda en carpetas organizadas por cada vez que una persona entra en el foco de la cámara. Luego, pasa por un proceso para identificar qué personas estuvieron en la cámara en más de una ocasión, agrupando sus carpetas mediante una comparación de rostros. Esto resulta en un conjunto de datos preciso y prácticamente listo para el entrenamiento de otros sistemas en el ámbito relacionado con la visión por computadora.

PALABRAS CLAVE:

Circuito integrado FPGA, Conjuntos de datos, Segmentación de imágenes, Visión por computadora y Yolo.

ABSTRACT

The system is capable of meeting the need for datasets for training person recognition systems, whether through facial recognition or additional information obtained from the human figure. The collected information comes from an FPGA-type integrated circuit with a camera sensor, programmed with software developed in Python, capturing real-time images from the environment where it is placed. Subsequently, it processes these images to identify the figure or figures within the camera's focus, determining whether they are facing it or not, and finally separating their faces.

All this information is stored in folders organized for each time a person enters the camera's focus. Then, it undergoes a process to identify which individuals were in front of the camera more than once, grouping their folders through a facial comparison. This results in a precise dataset, practically ready for training other systems in the field related to computer vision.

KEYWORDS:

FPGA Integrated Circuit, Datasets, Image Segmentation, Computer Vision and Yolo.

INTRODUCCIÓN

Antecedentes

Los sistemas de reconocimiento facial se enfrentan a un desafío considerable en la actualidad si hablamos sobre la detección y reconocimiento de características de una persona, donde se han evidencia con errores notables durante el proceso pruebas a estos y el problema se intensifica debido a la necesidad constante de realizar ajustes a la cámara para obtener datos precisos, lo cual representa un desafío persistente por la falta de variación en los datos para el entrenamiento de los modelos, volviéndose un factor crítico que afecta adversamente la capacidad del sistema para operar eficazmente en condiciones desafiantes las cuales se pueden llegar a encontrar dificultades en el mundo real. (S. Dwijayanti, 2022)

En la actualidad, la constante necesidad de sistemas biométricos fundamentados en el reconocimiento facial adquiere una relevancia crucial, especialmente en el ámbito de la seguridad, con estos sistemas no solo proporcionan una autenticación robusta y segura, sino que también verifican la identidad a través de las características únicas del rostro, convirtiéndose en una herramienta necesaria en entornos donde la seguridad es prioritaria, como por ejemplo en acceso a edificios, dispositivos móviles y aplicaciones bancarias. Su capacidad para detectar impostores y prevenir suplantaciones de identidad refuerza significativamente los protocolos de seguridad. (B. Meneses-Claudio, 2022)

La mejora de los sistemas de seguridad se evidencia especialmente en la automatización basada en el rostro, logrando resultados notables en eficiencia y garantizando una experiencia fluida para el usuario, con estas tecnologías emergentes la inteligencia artificial y el aprendizaje automático, han elevado la precisión y la

adaptabilidad de estos sistemas, consolidándolos como herramientas fundamentales para garantizar la seguridad en una amplia variedad de contextos. (B. Meneses-Claudio, 2022)

La demanda de sistemas de reconocimiento facial se extiende tanto a instituciones públicas como privadas, especialmente en sectores que requieren un riguroso control de acceso, estos sistemas se vuelven esenciales al gestionar grandes volúmenes de datos de personas en entornos donde es crucial regular y supervisar quién tiene acceso, al implementar el reconocimiento facial se optimiza el registro de asistencia, se agiliza la notificación de incumplimientos y se refuerza la seguridad al detectar vulnerabilidades potenciales. La esencia de estos sistemas radica en su capacidad para reducir significativamente la carga de trabajo de los encargados de administrar estos aspectos, mejorando la eficiencia y precisión en la supervisión de áreas, más allá de la simplificación de tareas administrativas, estos sistemas contribuyen a fomentar la productividad y la responsabilidad. Al ser gestionados de manera automática, las amonestaciones se vuelven instantáneas y objetivas, lo que no solo fortalece la seguridad, sino que también establece un ambiente donde la rendición de cuentas es inherente al sistema. La implementación de sistemas de reconocimiento facial no solo optimiza las operaciones cotidianas, sino que también promueve un entorno más seguro y eficiente en diversos contextos institucionales. (N. Narkhede, 2023)

Problemática

La recopilación de datos es una tarea laboriosa, la cual toma mucho tiempo en obtenerse por personal humano, además de tener que ser procesada para poder ser usada posteriormente para el entrenamiento de sistemas derivados de la visión por computadora, esto dificulta el desarrollo de nuevas tecnologías en estos ámbitos que actualmente es una necesidad para reducir costos en personal. El sector en donde se encarga este tipo de

sistemas requiere horarios extensos en el área de seguridad y gestión. (Tena-Parera, D, 2019)

Los sistemas basados en visión por computadora siempre necesitarán conjuntos de datos para llevar a cabo tareas como detección, seguimiento, clasificación y reconocimiento. Estos sistemas complejos permiten analizar imágenes del mundo real mediante el procesamiento con redes neuronales convolucionales entrenadas. En diversas áreas donde se requiere la intervención humana para la recopilación de imágenes, es necesario un postprocesamiento para que puedan utilizarse en sistemas de seguridad, registros de personas o cualquier sistema que necesite información específica sobre individuos en una institución o área determinada.

Con el aumento gradual en el número de cámaras tanto en sectores públicos como privados, donde es factible implementar sistemas de visión por computadora, los dispositivos de IoT (Internet de las cosas) se vuelven una parte fundamental en este proceso de desarrollo de nuevas tecnologías. Estas tecnologías son aplicables en diversos sectores, tanto comerciales como privados. (Kumar, D., Shen, K., Case, B., Garg, D., Alperovich, G., Kuznetsov, D., ... & Durumeric, Z, 2019).

Justificación

La visión por computadora es una de la tecnologías y herramientas que en la actualidad ha aumentado su desarrollo y ha alcanzado un pico muy alto de innovación, el cual va de la mano con todos los tipos de inteligencia artificial y se espera el desarrollo de muchas aplicaciones más para esta tecnología. El procesamiento en tiempo real de imágenes de sistemas pre entrenados se ha perfeccionado al punto de poder llegar a solventar los requerimientos de varias áreas donde se necesitaba la presencia del trabajo humano de jornadas muy largas como en el caso de la vigilancia, ya que en la actualidad

la visión por computadora es capaz de discernir e identificar cuando ocurre un evento o detectar un objeto el cual aparezca en los sensores de sus sistemas. (G. Casanova, D. Yandún and G. Guerrero, 2020).

Existe una gran necesidad de nuevas tecnologías en área de seguridad, gestión y administración, se ha expandido en la actualidad debido a su bajo costo de instalación y desarrollo en comparación a cualquier otro método que sea realizado por personal humano, esa necesidad se ve reflejada en la cantidad de dispositivos FPGA con sensores de cámara que se encuentran usándose en la actualidad, los cuales son considerados la mejor opción para cualquiera de los ámbitos antes mencionados, ya sea a nivel de costos y personal. (Álvarez, L, 2019).

El mundo ha revolucionado con el uso del Internet de las cosas (IoT) o sistemas implementados con sistemas FPGA utilizado en plataformas empresariales, sectores industriales, organizacional y gubernamental. Estos sistemas tienen el potencial de brindar seguridad al sector público y con esta confiabilidad ha permitido que la inteligencia artificial basada en sistemas de visión por computadora sean una opción más viable y económica, (M. H. Rohit, 2020).

Estos sistemas tienen la capacidad de procesar gran cantidad de datos en tiempo real, lo que permite generar una rápida respuesta a cualquier evento observado en cámara con la posibilidad de convertirse en un sistema IoT que permite seguir el desarrollo de una ciudad inteligente. (Y. -Y. Chen, Y. -H. Lin, Y. -C. Hu, C. -H. Hsia, Y. -A. Lian and S. -Y. Jhong, 2022)

Objetivos

Objetivo general:

Desarrollar un sistema portable de procesamiento de datos obtenidos por cámara para generar datasets de figura humana de los profesores de la universidad politécnica salesiana, para entrenamiento de otros sistemas basados en visión por computadora.

Objetivos específicos:

- Investigar e identificar opción más adecuada en hardware para generación de datasets para sistemas basados en visión por computadora
- Desarrollar un sistema de obtención y detección de figura humana por cámara para generación de dataset con las librerías de OpenCV y Yolov8.
- Evaluar la calidad de los datasets resultantes del sistema, verificando si logra una adecuada precisión en la detección de la figura humana.

Metodología

El sistema de primera instancia ha sido diseñado con la capacidad de capturar imágenes en tiempo real de individuos que aparecen en el encuadre, llevando a cabo un seguimiento y detección de estos. Posteriormente, estas imágenes se archivan en carpetas organizadas correspondientes a cada persona identificada, permitiendo una comparativa de similitud entre las imágenes recopiladas para corregir posibles errores en caso de superposición de personas. Se utiliza un índice de similitud del 45% para determinar si una imagen debe ser eliminada de una carpeta debido a un error de ubicación. Además, se procede a segmentar los cuerpos y rostros, especialmente cuando la persona está de frente,

mediante un algoritmo de detección facial. Estos elementos recortados se almacenan en una carpeta designada como "persona_frente" para su reconocimiento posterior.

En la última fase de depuración de datos, se examinan las imágenes de cada carpeta facial obtenida para realizar una comparación. Si al menos el 50% de las imágenes comparadas con otras carpetas coinciden como el mismo rostro, se fusionan en una única carpeta que representa a una persona. Es importante destacar que la cantidad de carpetas resultantes se corresponde con la cantidad de observaciones que se han registrado para esa persona a lo largo de todo el proceso.

Al concluir el proceso, se generan varios conjuntos de datos que representan a las personas observadas, con errores reducidos gracias al procesamiento exhaustivo que la información ha experimentado antes de llegar al resultado final. Este software desarrollado se ha implementado de manera integral en una microcomputadora con una notable capacidad de cómputo gráfico, como el Jetson Nano, que dispone de múltiples puertos para facilitar su configuración, incluyendo un socket destinado a una cámara serial. Todo este sistema se encuentra contenido en una carcasa de plástico PLA con refrigeración líquida, proporcionando protección y enfriamiento eficientes al procesador de la microcomputadora.

Con el fin de permitir la manipulación del dispositivo sin necesidad de conectarlo siempre a una computadora, se ha creado un sistema sobre un servidor Flask. Este servidor permite realizar solicitudes para visualizar el video en vivo desde un teléfono o computadora, a través de una interfaz sencilla que facilita detener y ejecutar los scripts responsables de llevar a cabo el proceso de generación de los conjuntos de datos.

MARCO TEÓRICO

Herramientas de desarrollo

El desarrollo del sistema demanda el uso de diversas herramientas de software, cada una esencial para una función específica. Estas herramientas incluyen lenguajes de programación, librerías, frameworks y motores de plantillas web, todos los cuales desempeñan un papel crucial en su funcionamiento integral.

- **Python:** Python es un software versátil con la capacidad de integrarse en diversos tipos de sistemas. Es notablemente fácil de comprender y aprender, permitiendo su implementación en una amplia variedad de contextos, como aplicaciones web, desarrollo de software, machine learning y ciencia de datos. Además, Python cuenta con la capacidad de establecer entornos estables, donde se pueden instalar las librerías deseadas. Python ofrece una extensa variedad de librerías específicamente diseñadas para el desarrollo de sistemas de inteligencia artificial. Es compatible con todos los sistemas operativos más comunes, lo que facilita el desarrollo de una amplia gama de proyectos en cualquier ámbito. (¿Qué es Python? - Explicación del lenguaje Python - AWS, s. f.-b)

    - **OpenCV:** Principalmente empleada por librerías dedicadas a la visión por computadora, Python ofrece una amplia gama de funciones que se adaptan a diversas aplicaciones. Estas funciones son especialmente útiles para analizar las características de imágenes capturadas por cámaras, siendo comúnmente utilizadas en plataformas móviles. Una de las principales ventajas de esta librería es su naturaleza de código abierto, lo que posibilita su uso libre

en proyectos académicos o en aplicaciones prácticas de visión por computadora en la vida real. (Suarez, O, 2014)

- **Yolov8:** La versión 8 de YOLO es reconocida como un modelo de vanguardia con la capacidad de llevar a cabo una detección de alta calidad, gracias a su amplio conjunto de tareas y las adiciones que se han incorporado a esta potente herramienta. YOLO se destaca por su versatilidad, ya que puede ser empleada en diversas aplicaciones del mundo real, incluyendo la visión por computadora. Sus modelos están preentrenados y son adaptables a distintos tamaños y capacidades de cómputo, lo que hace que sea accesible para una variedad extensa de hardware. Es importante señalar que, aunque se trata de un modelo de código abierto preciso, su utilización puede requerir recursos significativos. (Ultralytics, s. f.)

- **Flask:** Flask se define como un microframework que proporciona las herramientas esenciales para el desarrollo de aplicaciones web bajo el patrón de modelo vista controlador. Actúa como un servidor que alberga todas las funcionalidades de Python y, adicionalmente, permite la integración de lenguajes para páginas web como JavaScript o PHP. Esto posibilita el desarrollo de proyectos web complejos y flexibles, ofreciendo una extensa gama de posibilidades. Además, Flask pone a disposición las propias librerías de Python, simplificando así el proceso de desarrollo. (Muñoz, 2023)

    - **Jinja2:** Son plantillas que nos permiten utilizar html con python pasando las variables funciones o lo que se requiera para trabajar con código web, conocida como una de las librerías más populares y útiles en el ámbito web. (Plantillas Jinja2, s. f.)

- **JavaScript:** Es conocido como un lenguaje de secuencia de comandos o lenguaje de programación especificado para funciones complejas en páginas web, nacido por la necesidad que generaba las tecnologías estándar de HTML y CSS que no solventan ninguna necesidad más de la visual. (¿Qué es JavaScript? - Aprende Desarrollo web | MDN, s. f.-b)

  JavaScript permite hacer los que muchos otros lenguajes hacen con la limitación de que se puede implementar solo en sistemas web, también contando con varias librerías para el uso de machine learning útiles para clasificación cálculos y graficación. (GraphEverywhere, 2021)

Placa de microordenador

Se busco un circuito integrado de tipo FPGA que nos permita un desarrollo simple e instalación sencilla para nuestro proyecto, teniendo dos opciones importantes a discutir según sus características.

- **Nvidia Jetson:** Un circuito integrado diseñado con enfoque específico en proyectos de inteligencia artificial de alto rendimiento, con la capacidad de interactuar de manera efectiva en el mundo real. Este dispositivo, similar a una computadora, es operable con un sistema operativo análogo a Ubuntu, diseñado especialmente para plataformas Nvidia. Destacando por su eficiencia energética, estos dispositivos ofrecen una notable capacidad de cómputo gráfico gracias a sus tarjetas gráficas integradas. (NVIDIA , s. f.)

Figura 1

Jetson Nano

Nota. Placa de Nvidia orientada para IA (Del autor)

- **IMX219:** Camara Serial compatible con Nvidia

Figura 2

Cámara IMX219

Nota. Camara serial compatible con placas de Nvidia marca Sony (IMX219-77

Camera - Waveshare Wiki, s. f.)

- **Adaptador Wifi:** Con capacidad de crear una red Costo

Figura 3

USB Wifi adaptador

Nota: Adaptador compatible con hotspot TP-LINK (Del autor)

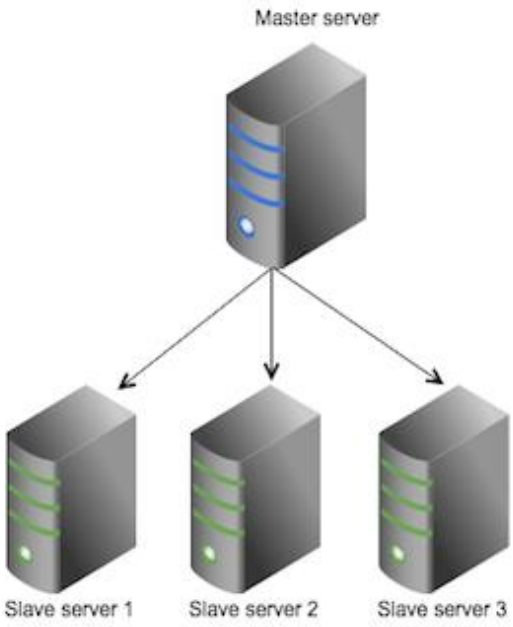
Arquitecturas

La arquitectura es importante si debemos conseguir el mejor patrón de arquitectura para nuestro proyecto lo que nos permitirá un mejor rendimiento y resultado a la hora de su ejecución.

- **Patrón cliente-servidor:** Es el tipo de arquitectura más especializado para sistemas de peticiones distribuidas debido a su centralización a un punto de administración, simple de realizar mantenimientos y con una alta posibilidad de escalabilidad, que por lo general usa TCP/IP otorgando una comunicación bidireccional que se mantiene conectada. (Arquitectura Cliente-Servidor, s. f.)

Figura 4

Patrón cliente-servidor

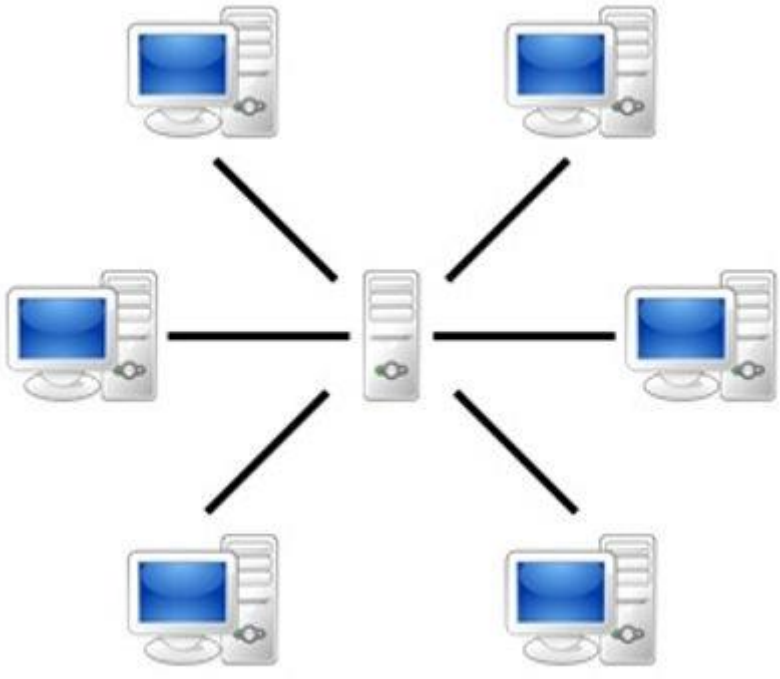


Nota: Como se estructura si se vincula o un servidor ((Alfresco Docs - Solr Replication, s. f.)

- **Patrón master-slave:** Se lo llama así por sus tipos de grupos el "master" y los "slaves", el master envia a los slaves a realizar una tarea propuesta y los resultados se envían de nuevo a este y realiza cualquier acción que requiera el sistema con estos datos ya procesados, muy ventajoso en proyectos que son multitarea ya que mantiene una funcionalidad de bucle basada en módulos. (Patrones de diseño de aplicaciones: Maestro / Esclavo - NI, 2020)

Figura 5

Patrón Master-Slave



Nota: Arquitectura que nos permite la escalabilidad del sistema (ARQUITECTURA CLIENTE - SERVIDOR, 2015)

Metodología de desarrollo

Design Sprint es una metodología concebida inicialmente en las oficinas de Google, la cual ha evolucionado con el tiempo y ha llegado a ser reconocida como una metodología ágil. Esta metodología resulta apropiada para nuestro proyecto, ya que implica la realización de revisiones semanales con entregables del sistema. Durante estas revisiones, se incorporarán y corregirán las novedades identificadas, lo que permite un enfoque dinámico y adaptativo a lo largo del desarrollo del proyecto. (Del Prado, J. A, 2022).

La metodología Design Sprint se estructura en cinco etapas fundamentales, las cuales generalmente se asocian con cada día de una semana laboral estándar de 40 horas. Se inicia con un enfoque o idea, formando un equipo multidisciplinario capaz de llevar a cabo el proyecto. A lo largo de esta semana, se da forma de manera rápida a la idea inicial, permitiendo una exploración ágil y eficiente del concepto. (Becas Santander, 2023).

Esta metodología comienza centrándose en la adecuada definición de los objetivos a alcanzar, procurando que sean lo más realistas posible. Luego, se inicia la generación de bocetos e ideas para abordar de manera innovadora el problema que debe resolverse. Se seleccionan las ideas que serán utilizadas en el prototipo final de la semana, y con este avance, se llevan a cabo pruebas para evaluar su idoneidad en la resolución del problema. Este ciclo se repite a lo largo del proyecto.

El principal objetivo del Design Sprint es generar ideas de manera rápida, permitiendo a las empresas tomar decisiones cruciales en cuestiones como el público objetivo o la estrategia de precios en tan solo unos días, mediante la implementación de sprints de diseño. (Ribas, E, 2022)

Esta metodología aporta valiosos beneficios a nuestro proyecto al poner la resolución del problema como prioridad. Facilita la generación de nuevas ideas y acelera el desarrollo, proporcionando resultados concretos que pueden ser medidos tanto por el equipo de trabajo como por el cliente final (Becas Santander, 2023).

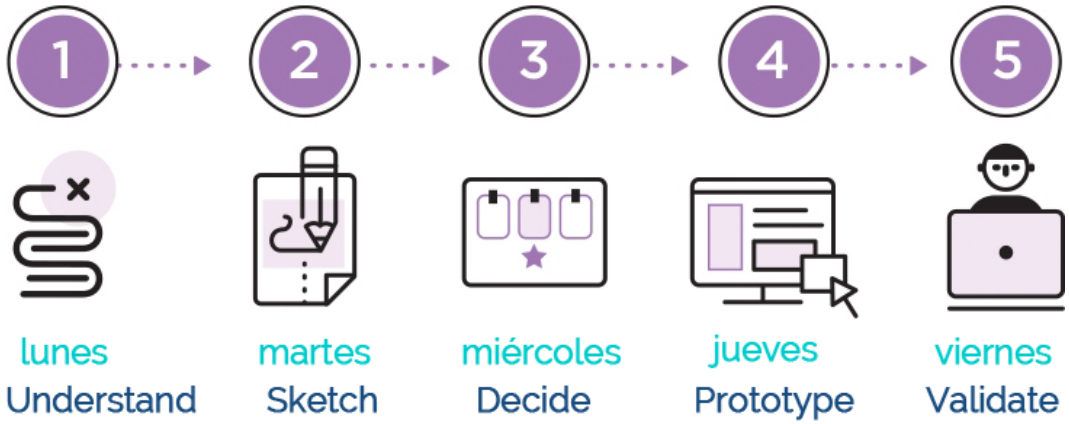
- Reducción de tiempos de entrega exponenciales debido a la organización de ideas claras y fallos corregidos en cada iteración
- Nos permite reducir los riesgos se hace siempre una comprobación antes de la implementación de una idea en el sistema, logrando que existan menos retrocesos en el desarrollo.

- Se puede obtener una retroalimentación continua del usuario final, además de lograr demostrar avances funcionales e innovadores en cada iteración.
- El equipo podrá trabajar de una manera más eficiente sobre los avances previos logrando un ambiente de trabajo optimo reduciendo la cantidad de esfuerzo innecesario aportado a cada etapa.

(Becas Santander, 2023)

Figura 6

Metodología Sprint



Nota. Proceso de pasos de la metodología de desarrollo sprint (Becas Santander, 2023)

METODOLOGÍA

Configuración

Sistema Operativo

La instalación del sistema operativo se lleva a cabo mediante el flasheo de una microSD, la cual será insertada en el Jetson Nano. Los pasos detallados para este proceso se encuentran descritos en la página principal de Nvidia, utilizando la imagen oficial del sistema operativo que incluye el módulo CUDA, esencial para trabajar con la gráfica de la placa. Además, se instalan los componentes necesarios para la arquitectura, los cuales son prácticos para el uso de inteligencia artificial.

Una vez que el sistema operativo está listo, se procede con la actualización de paquetes, una práctica común en nuevas máquinas con Ubuntu, para asegurar que todos los componentes estén actualizados y funcionando correctamente. (Get started with Jetson Nano Developer Kit, s. f.)

Módulos

El Jetson Nano necesita varios módulos para su adecuado funcionamiento del sistema, además de algunos que deben ser descargados de su repositorio de manera manual y uno que debe ser construido con CMAKE para que pueda ser compatible con todos los requerimientos que se necesita para el sistema, esto se hace de manera automática gracias a un archivo de secuencia de comandos Bash Shell colocado en la raíz de proyecto y los más relevantes son los siguientes:

- **Cuda**

La instalación u actualización de cuda se puede llegar a realizar con cierta precaución hasta la versión 11 de manera no oficial, esto podría llegar a ocasionar problemas y errores en el sistema, por lo que no se recomienda. Esto podría evitar el funcionamiento total del sistema por algún error lo que impediría el acceso al computador ya que no encendería, dejando como única opción el formatear la microSD flasheada y volver a empezar el proceso.

- **Python**

El sistema requiere Python 3.9, y para su instalación, es necesario primero descargar la versión desde el repositorio oficial. Una vez descargado, la instalación se realiza de manera similar a otros módulos, pero con la diferencia de que se especifica el nombre del archivo descargado o la ruta completa. Es crucial llevar a cabo este proceso con precaución, ya que existen paquetes que pueden no ser compatibles con el sistema, dado que el soporte para esta placa ya no está disponible en versiones más recientes.

Cabe destacar que la última versión de Python que recibió soporte oficial para esta placa fue hasta Python 3.6, junto con las librerías disponibles hasta ese momento. Así fue como se configuró el toolkit con Jetpack 4.6, que es compatible con la placa.. (JetPack SDK 4.6 Release Page, s. f.)

- **Jetson-stats**

Es una interfaz gráfica que se muestra en la terminal que nos permite observar el uso de la computadora en todos sus componentes y más información adicional de las librerías instaladas, el tema de interés en esta interfaz en revisar el uso de recursos del programa y si se encuentra

habilitado CUDA en nuestra computadora. Esta librería tiene un inconveniente si se instala de la manera común por lo que se necesita usar este comando para que tenga una funcionalidad adecuada “sudo -H pip3 install --no-cache-dir -U jetson-stats”.

- **OpenCV**

OpenCV siendo una de las librerías más esenciales, cuando se instala de manera normal no tiene acceso a la cámara serial del Jetson Nano debido a que no está disponible el Gstreamer, esto lo podemos comprobar ejecutando un script de Python con el comando “cv. getBuildInformation()”. Debe tener dos opciones habilitadas el Cuda y Gstreamer en caso de que no, se debe construir la librería obteniendo el repositorio de github de opencv y opencv-contrib, con las etiquetas adecuadas se debe ejecutar “cmake”, esto puede llegar a tardar hasta 8 horas en realizarse y se debe estar atento de que no haya ningún error, con este objetivo existe un script que automatiza este proceso en la raíz del proyecto.

- **Torch**

Es esta librería debido a la naturaleza del proyecto y la falta de soporte como se mencionó antes, solo se puede instalar ciertas versiones de torch funcionales para este sistema, cuando se instale se debe obtener la versión que se requiere con “pip” con la opción “-f” y se buscara el link del repositorio de torch para nuestra versión de cuda que en este caso es “https://download.pytorch.org/whl/cu102/torch_se.html”.

Las versiones especificas necesarias se encuentran en el archivo de requerimientos del proyecto.

Adicionales

El proyecto opera de dos maneras, a través de wifi y ethernet. Para ambas opciones, se requiere realizar la configuración adecuada en el servidor de Nginx para su funcionamiento, en caso de que se esté utilizando. Se otorga siempre prioridad a la IP de la red ethernet; por lo tanto, si está conectada, se asignará como la principal, permitiendo el acceso al servidor a través de esta conexión.

Este proceso se lleva a cabo de manera automática mediante un archivo de secuencia de comandos Bash Shell y un archivo cron. Estos archivos permiten cambiar esta información cada vez que se inicia la computadora, por lo que es esencial configurar esta tarea automática para garantizar un funcionamiento óptimo.

Hardware

El sistema siendo basado en visión por computadora requiere hardware adecuado capaz de manejar un alto rendimiento en computo gráfico, por este motivo se optó por un Jetson Nano de Nvidia el cual tiene una arquitectura específica para estas tareas, en un tamaño muy compacto de tan solo 70x45 mm ofreciendo hasta 472 GFLOP con la capacidad de ejecutar múltiples redes neuronales en paralelo, volviéndolo el dispositivo perfecto para el uso de IA con un consumo bajo de 10 vatios. (Jetson Nano de NVIDIA, s. f.)

El código fue desarrollado con las herramientas antes mencionadas se instalará en el circuito integrado programable Jetson Nano, trabajando con sistema operativo Linux más específicamente Ubuntu 18.04 con un sensor de cámara compatible con este dispositivo y refrigeración líquida para mejor funcionamiento.

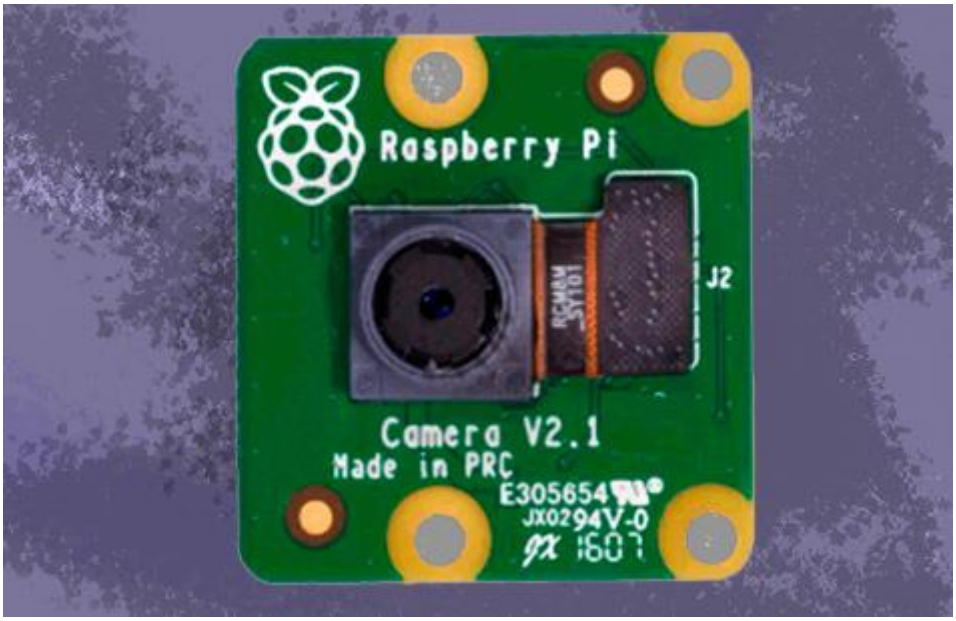
Cámara

La placa utilizada es un poco especifica en que cámaras es compatible, ya que a la fecha de este documento la cámara IMX219 para esta placa no se puede conseguir, a menos que se compre en el exterior se probó con varias cámaras debido a un desperfecto de la cámara compatible que se tenía, las cuales ninguna de las siguientes fue compatible:

1. Raspberry Pi camera v2: No compatible.

Figura 7

Cámara Raspberry Pi v2



Nota; Cámara que resulto no ser compatible, pero tiene características similares (Del autor)

2. Jevois A33: No tenia soporte a fecha del proyecto para configurarla, pero si daba imagen.

Figura 8

Cámara Jevois A33

Nota; Cámara con procesador incluido para reconocimiento integrado (Del autor)

3. Cámara USB: Detenía el funcionamiento de la placa cuando estaba conectada.

Al final se debe optar por comprar una cámara compatible con la placa de Jetson Nano como la antes mencionada o las que se recomienda en la página oficial, en el caso de que una cámara compatible genere un error de “buffer HD” sin haberla usada antes se recomienda comprar una nueva.

Adaptador WIFI

Si se va a usar un adaptador WIFI, este debe tener la posibilidad de poder crear una red Hotspot y esto se debe leer en las especificaciones del adaptador, si no, no será compatible para todas las opciones de uso del sistema.

Case

La placa fue colocada en una caja hecha por impresión 3D en plástico PLA a medida con espacio para la cámara y la refrigeración líquida, está siendo desmontable por tornillos permitiendo sacar a la placa cuando sea necesario.

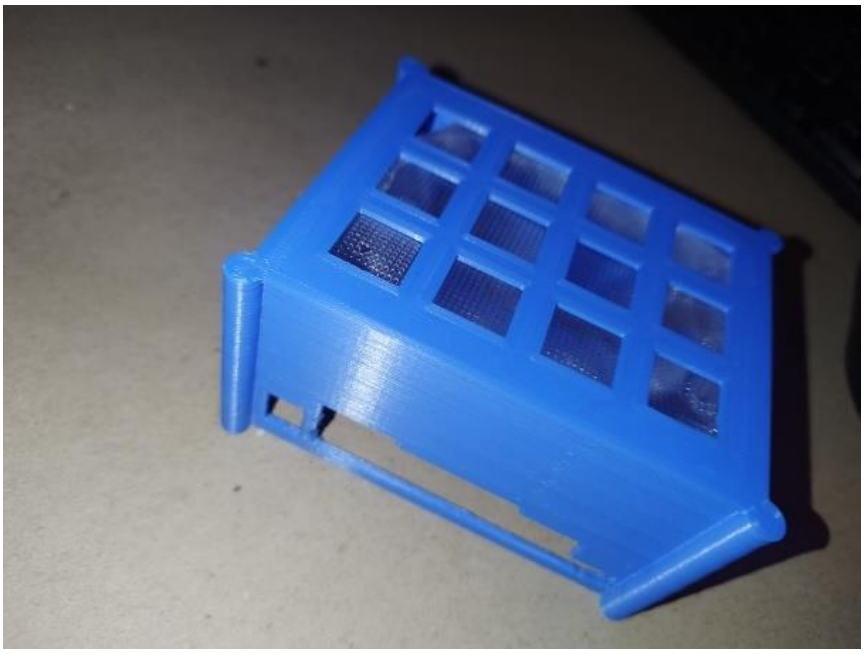
Figura 8

Base case para Jetson Nano

Nota: Base azul celeste que sostiene a la placa (Del autor)

Figura 9

Tapa case Jetson Nano



Nota: Tapa del case color celeste de la placa (Del autor)

Refrigeración liquida

La refrigeración líquida se implementó mediante una funda para empacado al vacío, la cual se llenó con agua y se colocó dentro de la caja en la parte superior, asegurándola con silicona selladora para que hiciese contacto directo con el disipador del procesador. Este método permite un enfriamiento constante al esparcir el calor al exterior. La alta resistencia al calor de la funda, combinada con el contacto con el agua, crea una eficaz barrera térmica, proporcionando varias horas de trabajo sin preocupaciones por el recalentamiento del procesador.

Figura 10

Refrigeración líquida case

Nota: Case colocado la refrigeración por agua (Del autor)

Detección y seguimiento de personas

OpenCV para el tratamiento de las imágenes debido a que es el más utilizado y con gran variedad de opciones para trabajar los vectores de imágenes. (S. Sambolek and M. Ivasic-Kos, 2021).

Para la detección de personas se usó **redes neuronales convucionales** (CNN) debido a la afinidad con el hardware actualmente usado y además de ser bastante conocido por su entrenamiento casi sin requerir un procesamiento de las imágenes a diferencia de otros tipos de arquitecturas para Deep Learning, El sistema, desarrollado en Python utilizando YOLO versión 8, constituye un modelo de detección y segmentación de objetos con la capacidad de trabajar con imágenes en tiempo real gracias a su alta velocidad y precisión, derivada de su

modelo preentrenado. Esta librería también ofrece la posibilidad de generar modelos personalizados y etiquetas según las necesidades del usuario.

Para la generación del modelo, se utiliza la librería YOLO, que recibe argumentos como los pesos utilizados para la predicción. En este caso, se empleará el modelo **YOLOv8n**, también conocido como "nano", debido a sus métricas de rendimiento eficientes en términos de recursos. El dispositivo de cómputo utilizado será **CUDA**, aprovechando la arquitectura de hardware que lo permite, y se realizarán configuraciones adicionales según las necesidades del proyecto. Posteriormente, se ejecuta el método de **non_max_suppression** para obtener la mejor delimitación del objeto identificado.

Los bordes resultantes se cargan en el software de seguimiento, en este caso, **deepsort**. Este último proporciona las detecciones en coordenadas, ancho y largo, lo que permite dibujar las cajas delimitadoras en la imagen con sus respectivas etiquetas de persona y/o personas para cada fotograma.

Segmentación de persona

Después de la detección, las imágenes resultantes se segmentan exclusivamente para aislar a la persona detectada. Este proceso se lleva a cabo según el registro del software de seguimiento, marcando el primer paso para asegurar que cada foto sea debidamente asociada a la misma persona antes de almacenarse en una carpeta externa.

Segmentación de postura

La postura nos referimos hacía que parte está apuntando el rostro y el cuerpo de las personas detectadas previamente este, cuando reconoce un rostro clasifica esta imagen como una persona parada de frente, caso contrario la tomara como una persona que está parada de espaldas con la capacidad de recocer desde personas que están

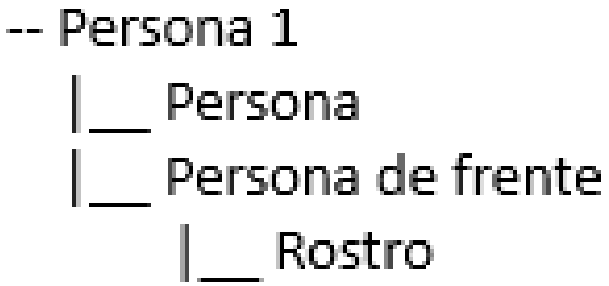
alejadas de la cámara como también las más cercanas y recopiladas en la carpeta “persona” y “persona de frente”

Segmentación de rostro

La detección de rostros se lleva a cabo utilizando la clasificación de la librería face_recognition, la cual nos brinda varias opciones para la detección de rostros. En este proyecto, se utiliza la clasificación Harr, conocida por su capacidad precisa de detectar rostros. Las imágenes detectadas se recopilan en la carpeta "persona de frente" y, dentro de esta, se crea una subcarpeta denominada "rostro".

Figura 11

Organización script de rostro



Nota: Estructura de la organización jerárquica Fase 2 (Del autor)

Organización de archivos y directorios

Como fase final, se realiza una comparación de histogramas con un umbral de similitud del 45%, que ha demostrado ser el más eficaz en el sistema. Las imágenes que caen por debajo de este umbral se eliminan para corregir posibles errores del software de seguimiento. Luego, se analiza cada carpeta correspondiente a cada persona para comparar sus rostros. Este paso es esencial, ya que cuando una persona sale completamente del rango

de visión de la cámara y regresa, se le considera como una persona nueva con una carpeta nueva.

Al concluir la comparación de rostros, si al menos el 60% de las imágenes de una carpeta se reconocen como la misma persona de otra carpeta, estas carpetas se fusionan. Este proceso resulta en una carpeta que representa a una persona con subcarpetas que indican la cantidad de observaciones detectadas en todo el proceso. Así, el dataset está listo para ser utilizado en el entrenamiento de sistemas. Se puede realizar correcciones adicionales según sea necesario. Es importante señalar que estos errores pueden ocurrir cuando una persona no aparece el tiempo suficiente en la cámara y no se cuenta con la información necesaria para la comparación.

Control del sistema vía remota

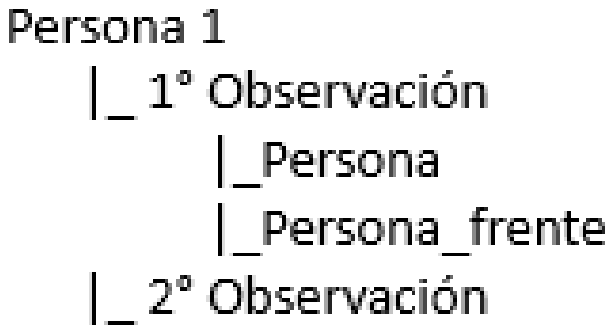
El sistema requería un adaptador wifi que permitiera el acceso a una red con internet para la descarga e instalación de todos los requisitos necesarios. Además, debía tener la capacidad de crear una red propia a la que los dispositivos pudieran conectarse para controlar el Jetson Nano mediante un servidor de peticiones, ejecutando los scripts que no se inician automáticamente al encender la placa. Por esta razón, se eligió el patrón de diseño cliente-servidor, que beneficia la estructura del proyecto.

Se realizaron peticiones a un servidor construido en Flask, permitiendo la obtención y almacenamiento de los datos organizados en carpetas por persona detectada. Finalmente, se ejecuta un script adicional que verifica la presencia de un rostro para clasificar entre personas de espaldas y de frente. En caso de detectarlo, separa el rostro de la persona y lo guarda de manera independiente en la misma carpeta de la persona, realizando comparaciones entre varias carpetas para fusionarlas en caso de coincidencia de rostros.

Como resultado se tendría una organización de carpetas por persona así:

Figura 12

Organización carpeta script final



Nota: Estructura de la organización jerárquica Fase 3 (Del autor)

RESULTADOS Y DISCUSIÓN

Se llevaron a cabo diversas pruebas con el equipo, destacando subcategorías que analizaremos para evaluar el desempeño en las áreas establecidas previamente. Estas pruebas proporcionaron resultados tanto positivos como negativos, ofreciendo información valiosa sobre cómo se podría mejorar posteriormente el software y obtener mejores resultados.

Pruebas por modelo

Se inicia evaluando el sistema de detección y seguimiento, probándolo en la Universidad Politécnica Salesiana en el aula de profesores del bloque G, a una altura de 2 metros del piso. Se verificó que el sistema es capaz de detectar de manera precisa a las personas dentro del cuadro de la cámara. Sin embargo, se observaron diferencias en los resultados entre los diferentes modelos utilizados:

- **Yolov8s.pt:** Este modelo, siendo uno de los más pequeños de YOLO y caracterizado por la sigla "S" que indica "small", logra una detección precisa de las personas en las imágenes. No se evidenció ningún caso en el que no reconociera a alguien en la zona donde se ubicó. Sin embargo, se presentó un inconveniente importante: el tiempo de captura por imagen osciló entre 6 a 9 segundos, lo cual es significativamente alto y afecta el rendimiento del sistema de seguimiento. Esta demora en la captura compromete la continuidad del seguimiento de la persona.

- **Yolov8n.pt:** Este modelo, siendo el más pequeño de YOLO y caracterizado por la sigla "N" que indica "nano", logra una detección precisa de las imágenes de las personas. No se evidenció que no reconociera a alguien en la zona donde se colocó. Sin embargo, se observaron falsos positivos en otros casos, donde el modelo identificó erróneamente objetos como personas cuando no lo eran. Una

ventaja en comparación con un modelo más grande es que este maneja un tiempo de captura de 3 a 5 segundos por imagen, lo cual es considerablemente más bajo que el modelo anterior. Esta reducción en el tiempo de captura afecta menos al rendimiento del sistema de seguimiento, lo que significa que la continuidad del seguimiento de la persona se ve menos afectada.

En esta evaluación, se llegó a la conclusión de que el modelo más pequeño es el más adecuado para esta placa, dada la cantidad limitada de recursos disponibles, así como la incapacidad de utilizar completamente la gráfica debido a la falta de soporte proporcionado por Nvidia. Los falsos positivos generados por este modelo más pequeño son resultado de realizar menos cálculos para lograr un tiempo de respuesta más rápido. Este aspecto no se considera problemático, ya que generalmente se eliminan estos falsos positivos en las fases posteriores del procesamiento.

Pruebas por conexión

Las pruebas de conexión mostraron una gran diferencia en el momento de sacar resultados.

Tabla 1

Comparativa de conexión

N°	**Tipo de conexión**	**N° Imágenes obtenidas**	**Tiempo de grabación**
1	Adaptador Wifi Hostpot	31 imagenes	3 horas y media
2	Ethernet	830 imagenes	2 horas y media

Nota; Tabla de resultados de cantidad de imágenes en relación al tiempo y conexión (Del autor)

En la prueba número 1, se observa una mejora notable al utilizar el cable de red. Esto se debe a que el adaptador wifi presenta fluctuaciones considerables en la conexión, y las imágenes, antes de ser guardadas, pasan por el servidor. Por lo tanto, la estabilidad de la conexión es un factor crítico al desarrollar sistemas de este tipo. Las imágenes tomadas durante esta prueba no resultaron útiles, ya que el sistema de seguimiento carecía de continuidad, lo que hacía imposible crear un dataset de calidad adecuada para ser utilizado en otro sistema.

En la prueba número 2, se obtuvieron resultados mucho mejores y más cercanos a lo esperado. Aunque persisten ciertos errores debido a la falta de continuidad en el seguimiento, estos son considerablemente menores en comparación con la prueba 1. En estas condiciones, se logra generar un dataset de 830 imágenes que se observó de una persona en cámara ya sea entrando o saliendo del rango de visión, la cual es la información necesaria para la implementación de una variedad de sistemas de reconocimiento.

Resultado de imágenes

Calidad

La calidad de las imágenes obtenidas está directamente relacionada con la distancia entre la persona y el dispositivo, esta influye en la capacidad de reconocer el rostro de manera adecuada, siendo más difícil lograr un reconocimiento preciso cuando la persona está muy alejada. En contraste, cuando la persona se encuentra en cercanía, se pueden captar una cantidad adecuada de píxeles, lo que facilita el reconocimiento facial.

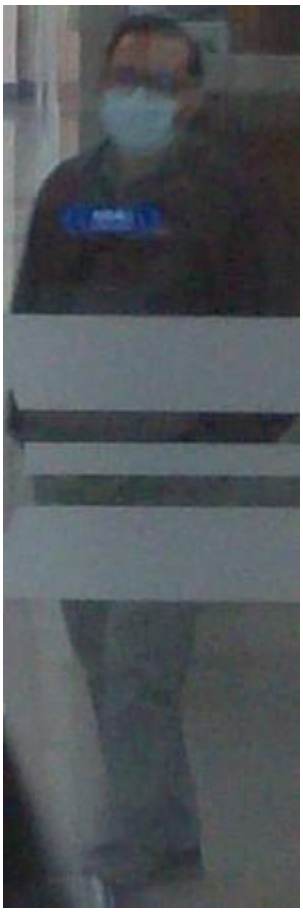
Figura 13

Comparativa de imágenes por distancia 3 metros

Nota: Se observa los detalles de la persona a distancia cercana (Del autor)

Figura 14

Comparativa de imágenes por distancia 5-6 metros



Nota: Se observa los detalles de la persona a distancia lejana (Del autor)

Este ejemplo ilustra claramente la diferencia en la calidad de las imágenes. En la primera imagen, se observa una captura tomada con la cámara IMX219-77 a una distancia de 3 metros, mientras que, en la segunda imagen la distancia es de 5 a 6 metros. La disparidad en la calidad es notable, ya que en la primera imagen se puede obtener el rostro de manera clara y separarlo, mientras que en la segunda es probable que sea eliminado por el sistema debido a la menor cantidad de detalles y píxeles capturados a mayor distancia.

Proceso de generación de datasets

Durante la prueba realizada en las instalaciones de la Universidad Politécnica Salesiana, el sistema fue implementado desde las 12:00 pm hasta las 2:30 pm, en este período

el Jetson Nano registró un total de 108 avistamientos de personas en el sistema principal de seguimiento y detección. Cada avistamiento generó automáticamente una carpeta correspondiente, resultando en un número igual de carpetas y avistamientos, cada carpeta contenía imágenes que capturaron exhaustivamente el trayecto de cada persona en el entorno monitorizado.

Para optimizar el almacenamiento y evitar una carga innecesaria de información, se estableció una limitación de 20 imágenes por carpeta, esta medida precautoria fue diseñada especialmente para situaciones en las que una persona podría permanecer frente a la cámara durante un período prolongado. así, se garantiza la eficiencia del sistema al almacenar únicamente la información esencial, contribuyendo a un uso más eficaz de la memoria sin comprometer la integridad de los datos recopilados.

Figura 15

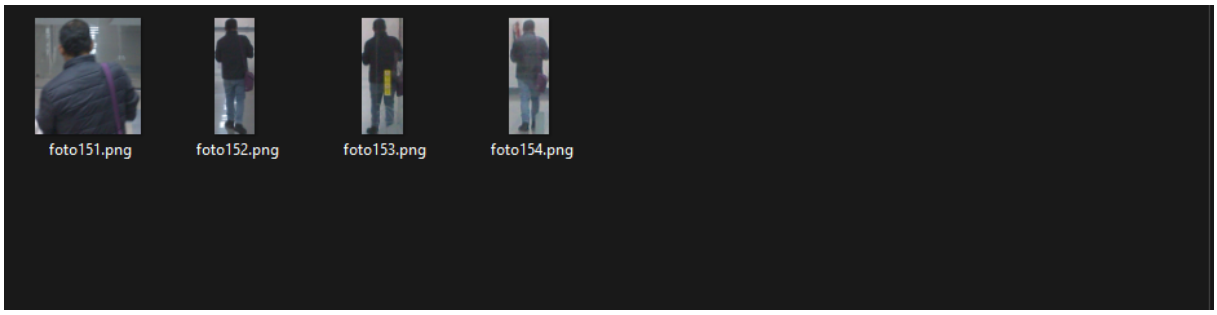
Ejemplo A con rostro Fase 1

Nota: Carpeta de una persona observada en la fase 1 (Del autor)

Se toma el ejemplo A en el cual se observa que solo existe una imagen en esta carpeta pero que nos proporciona una clara información del rostro, por lo que el sistema es capaz de obtener el código único del rostro de esta persona.

Figura 16

Ejemplo A sin rostro Fase 1



Nota: Otras posturas de la misma persona registrada en la fase 1 (Del autor)

En este caso observamos a la misma persona, pero con vestimenta ligeramente diferente, debido a que no aparece un rostro en cualquiera de las imagenes se puede predecir que esta carpeta será descartada, ya que al no existir un rostro no hay una validación segura de que sea la misma persona, por lo que se la etiquetará como persona desconocida.

Figura 17

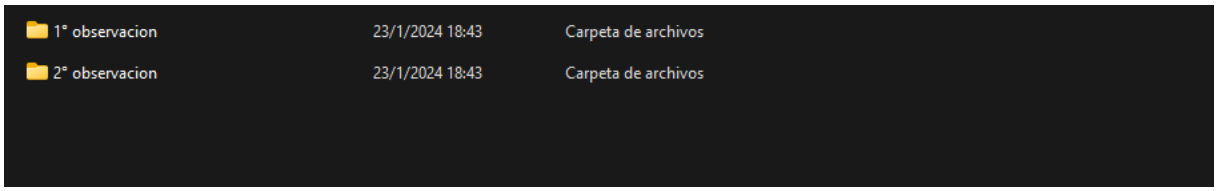
Ejemplo A Fase 2

Nota: Carpeta de persona de frente en la fase 2 (Del autor)

Como se observa en el ejemplo A de esta fase se segmenta de manera correcta el rostro de la persona, de la única foto que existía en esa carpeta, para posteriormente ser comparado con las demás carpetas en la siguiente fase.

Figura 18

Ejemplo A Fase 3 Carpetas



Nota: Numero de observaciones de una persona en la prueba (Del autor)

El resultado obtenido en la fase 3 es de dos observaciones eso quiere decir que la persona paso por cámara mostrando su rostro dos veces en esta prueba.

Figura 19

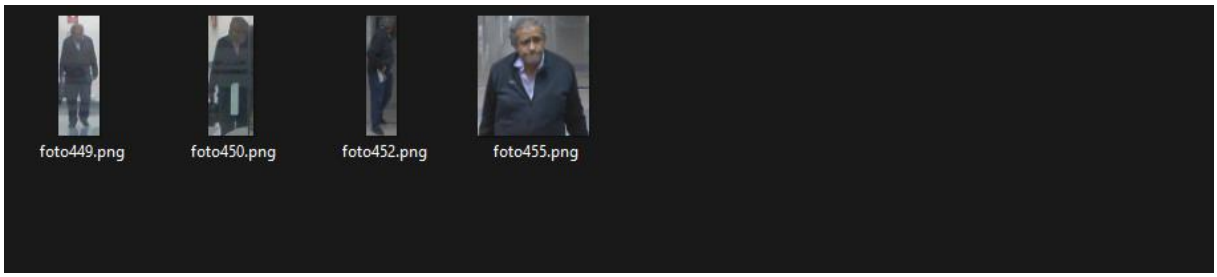
Ejemplo A Fase 3 Observación 2

Nota: Comprobación de la fase 3 que es la misma persona en las observaciones (Del autor)

Siendo esta la segunda observación se recalca que esta misma persona que no tenían rostro fue eliminada en la fase 3 dejándonos con una información precisa del rostro, esto se confirma debido a que el sistema junto a las dos carpetas como una sola persona, lo que nos indica que las imágenes son adecuadas para poder entrenar a un reconocimiento de rostros.

Figura 20

Ejemplo B Fase 1



Nota: Ejemplo de otra persona detectada en la primera fase (Del autor)

En el ejemplo B, se presenta una situación idéntica a la del caso anterior, con la única variación de que no se identificó ninguna observación adicional de la persona en cuestión, esto se debe a la ausencia de imágenes posteriores que permitieran la visualización de su rostro, resultando en la generación de una única carpeta en la fase

Figura 21

Ejemplo B Fase 2 Persona de frente

Nota: Resultados obtenidos en la fase 2 (Del autor)

Figura 22

Ejemplo B Fase 2 Otras posturas

Nota: Resultados obtenidos en la fase 2 con las posturas (Del autor)

Se han registrado las demás posturas dentro de la misma carpeta debido a la presencia de un rostro en ella, lo que permite agruparlas bajo la misma entidad.

En las imágenes subsecuentes correspondientes a los ejemplos C y D, se aplicará un difuminado a los rostros por razones de privacidad, ya que estas personas adicionales han contribuido con datos a este proyecto.

Figura 23

Ejemplo C Fase 1

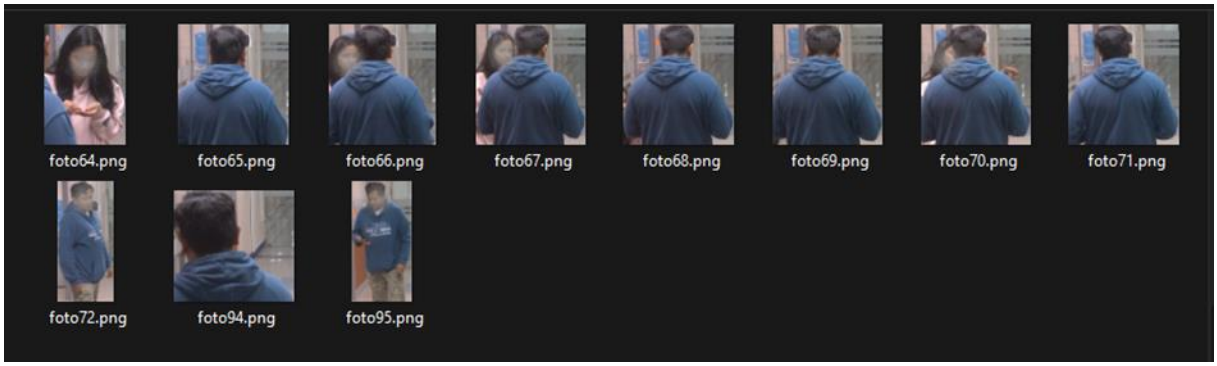
Nota: Se muestran los errores en la detección con software de seguimiento (Del autor)

En este caso se toma el ejemplo C en donde podemos observar un error muy común en el sistema de seguimiento, ya que existen dos imágenes que no correspondes a esta carpeta

esto se da debido a un cruce de una persona y otra en el mismo lugar en donde se pierde el seguimiento.

Figura 24

Ejemplo D Fase 1

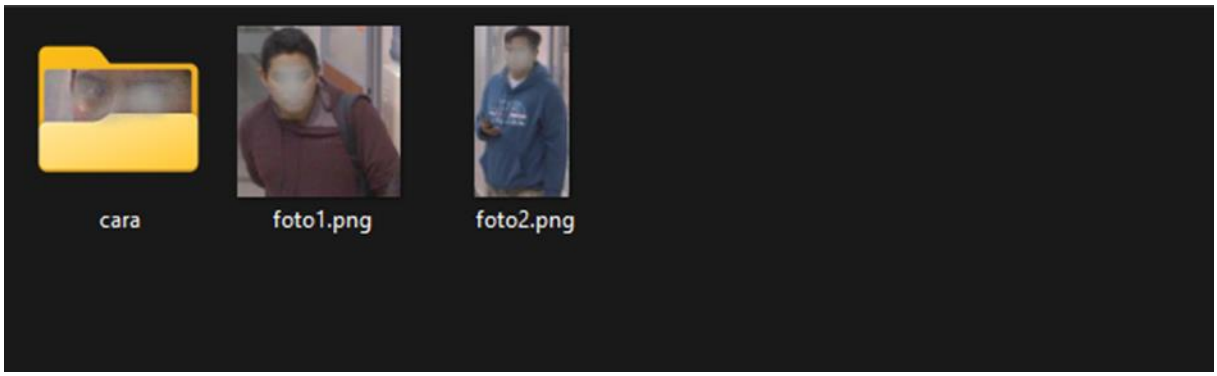


Nota: Otro ejemplo similar al error de seguimiento (Del autor)

Identificamos un escenario similar durante el ejemplo D, donde se inicia el seguimiento de una persona y, en el momento del cruce, el sistema procede a rastrear a otra. Estos desafíos evidenciados se abordan de manera efectiva en la fase siguiente de depuración de las imágenes, en esta etapa cada imagen es meticulosamente procesada para segmentar los rostros. En situaciones donde una carpeta no contiene al menos una imagen con la presencia de un rostro, se descarta, ya que carece de utilidad para la creación de un dataset completo de la persona. La eliminación de estas carpetas asegura la integridad y relevancia del dataset, ya que no es posible vincular de manera segura a una persona de un avistamiento a otro sin la presencia de imágenes faciales correspondientes. Este enfoque garantiza la calidad y coherencia del conjunto de datos generado durante el seguimiento y la detección.

Figura 25

Ejemplo C Fase 2



Nota: En la primera fase no se elimina aun estos falsos positivos (Del autor)

Tras llevar a cabo el segundo proceso de depuración, el ejemplo C presenta dos rostros, permitiendo la obtención adecuada de sus códigos únicos. Esto implica que la información disponible es suficiente para desarrollar un sistema básico de reconocimiento facial con garantías de precisión. En la tercera fase de depuración se realiza una comparación del histograma de las imágenes resultantes con el fin de observar el nivel de similitud, donde si el índice de similitud es menor al 45% tomando de referencia la primera imagen de rostro obtenida, es eliminada.

En la fase final, se extrae el código único de los rostros presentes en las imágenes de la carpeta bajo observación, se lleva a cabo una comparación exhaustiva entre cada uno de estos códigos con el propósito de descartar identificaciones erróneas, utilizando como referencia la primera observación realizada. Este proceso se repite tanto en la carpeta actual como en otras, y si más del 50% de las imágenes indican que se trata de la misma persona, las carpetas se fusionan, consolidando así la identidad de esa persona a lo largo de diferentes observaciones de cámaras. Este enfoque asegura una mayor precisión y coherencia en la identificación de individuos a través del sistema.

Figura 26

Ejemplo C Observación 1 Fase 3

Nota: Resultados obtenidos en la fase 3 sin errores (Del autor)

Figura 27

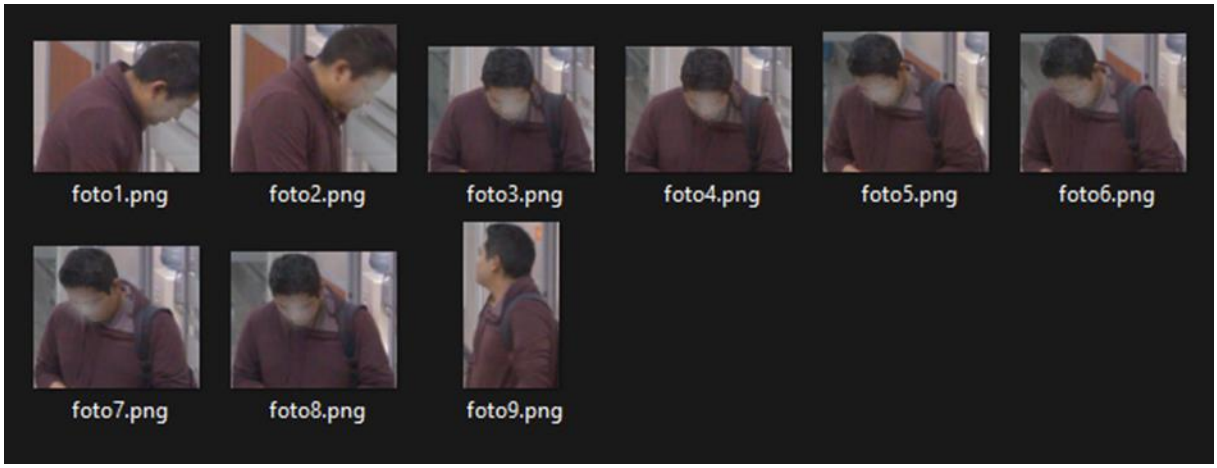
Ejemplo C Observación 2 Fase 3

Nota: Observación 2 de la persona en fase 3 sin errores (Del autor)

El resultado final nos proporciona un dataset completo que incluye imágenes de la persona cuando se encuentra frente a la cámara, revelando claramente su rostro, así como imágenes capturadas en otras posturas adicionales.

Figura 28

Ejemplo C Observación 1 Posturas Fase 3



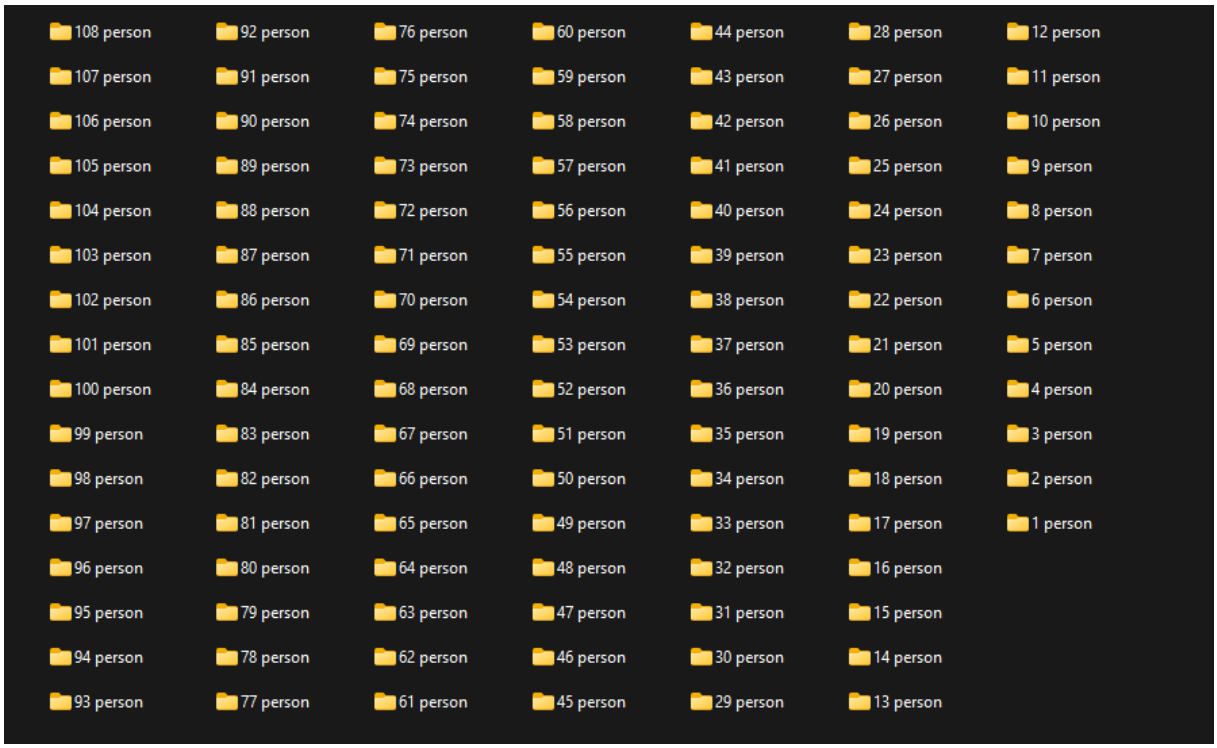
Nota: Comprobación de otras posturas sin errores (Del autor)

Este enfoque garantiza una representación integral de la persona en diversas orientaciones, proporcionando un conjunto de datos completo y diversificado que abarca todas las posiciones posibles en las que pudo ser capturada por la cámara, además de todas las observaciones que tuvo en la sala de profesores del bloque G.

Como resultado final se observará la comparativa de la cantidad de personas registradas antes y después de haber pasado por el sistema de reconocimiento de figura humana.

Figura 29

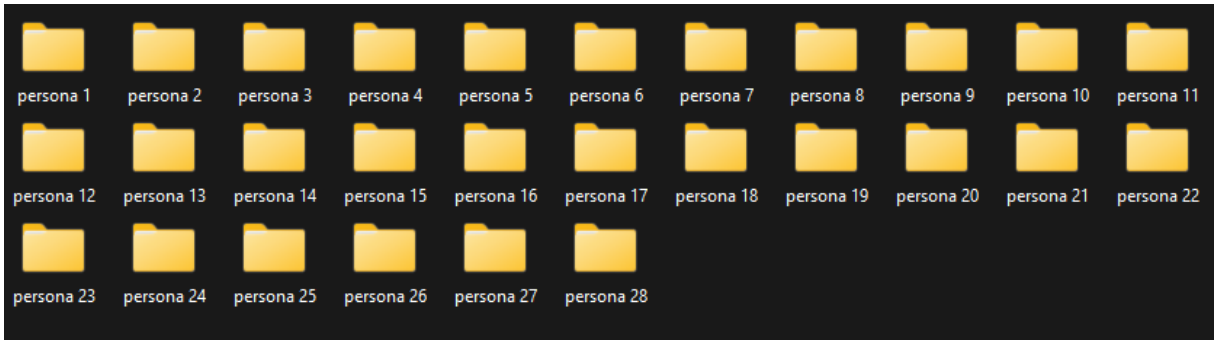
Carpetas Fase 1



Nota: Carpetas obtenidas sin el proceso de limpieza (Del autor)

Figura 30

Carpetas Fase 3



Nota: Carpetas obtenidas con el proceso de limpieza (Del autor)

La cantidad de personas identificadas se reduce después de completar todos los procesos de filtrado de datos, reteniendo únicamente la información que el sistema considera

relevante para los datasets. En consecuencia, se confirman con precisión 28 personas que han sido capturadas por la cámara.

CONCLUSIONES

El sistema trabaja de manera óptima con herramientas de vanguardia o similares como Python 3.9, este lenguaje de programación proporciona una amplia gama de herramientas que satisfacen las necesidades del proyecto, complementado con el uso del dispositivo Nvidia Jetson Nano, diseñado específicamente para tareas relacionadas con la inteligencia artificial y el computo gráfico.

El software desarrollado demuestra una alta precisión en las tareas de detección, reconocimiento y seguimiento de la figura humana en las imágenes capturadas. Se observan resultados significativamente mejores cuando el dispositivo que ejecuta el software dispone de más recursos. Los procesos posteriores a la toma de imágenes pueden requerir un tiempo considerable para la organización y clasificación, y este tiempo variará en función de la cantidad de imágenes procesadas.

Los datasets generados se consideran de calidad y confiables con respecto a los 3 procesos de depuración de la información que tienen que pasar, especialmente cuando se cuenta con una cantidad sustancial de datos de las personas observadas, que como mínimo los rostros deben aparecer una vez de manera reconocible en cámara y los errores residuales que puedan surgir no representan un inconveniente a su posterior uso.

RECOMENDACIONES

Se recomienda realizar pruebas con diferentes parámetros en el código para lograr un mejor resultado, ya que estos pueden variar según las condiciones del entorno donde se implemente el sistema o los tiempos deseados para obtener el dataset final. La experimentación con distintos ajustes permitirá optimizar el rendimiento y la eficacia del sistema de reconocimiento y seguimiento de la figura humana.

Dado que el sistema está alojado en un servidor Flask, sería fácilmente factible integrarlo con un servidor externo para enviar las imágenes generadas y procesarlas en proyectos más complejos. La capacidad de conectarse a un servidor externo proporciona flexibilidad y posibilidades adicionales para la implementación del sistema en contextos más amplios o en colaboración con otros proyectos.

Se recomienda instalar el sistema en computadoras con mayor capacidad de cómputo. En entornos donde no hay retrasos significativos en la obtención de imágenes, el sistema tiene la capacidad de generar datasets sin fallos, convirtiéndolo en un sistema preciso y confiable. La mejora en la capacidad de cómputo contribuirá a un rendimiento más eficiente y a la obtención de resultados más precisos y consistentes.

LISTA DE REFERENCIAS

S. Dwijayanti, M. Iqbal and B. Y. Suprapto, "Real-Time Implementation of Face Recognition and Emotion Recognition in a Humanoid Robot Using a Convolutional Neural Network," in IEEE Access, vol. 10, pp. 89876-89886, 2022, doi: 10.1109/ACCESS.2022.3200762.

B. Meneses-Claudio, L. Tarmeño-Bernuy, M. Yauri-Machaca and E. L. Huamaní, "Biometric Facial Security System with Telemetry Module," 2022 Second International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Bhilai, India, 2022, pp. 1-5, doi: 10.1109/ICAECT54875.2022.9807916.

N. Narkhede, A. Menon, I. Mathane, S. Nikam and S. Dange, "Facial Recognition and Machine Learning-based Student Attendance Monitoring System," 2023 3rd International Conference on Intelligent Technologies (CONIT), Hubli, India, 2023, pp. 1-7, doi: 10.1109/CONIT59222.2023.10205631.

Tena-Parera, D. (2019). El valor de los datos. Questiones publicitarias, 2(24), 71-76.

Kumar, D., Shen, K., Case, B., Garg, D., Alperovich, G., Kuznetsov, D., ... & Durumeric, Z. (2019). All Things Considered: An Analysis of {IoT} Devices on Home Networks. In *28th USENIX security symposium (USENIX Security 19)* (pp. 1169-1185).

G. Casanova, D. Yandún and G. Guerrero, "Analysis of video surveillance images using computer vision in a controlled security environment," 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), Seville, Spain, 2020, pp. 1-6, doi: 10.23919/CISTI49556.2020.9141068.

Álvarez, L., & Astudillo, C. (2019). Diseño y desarrollo de entrenador de fpga libre de bajo costo basado en fpga alhambra ii.

M. H. Rohit, "An IoT based System for Public Transport Surveillance using real-time Data Analysis and Computer Vision," 2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC), Bengaluru, India, 2020, pp. 1-6, doi: 10.1109/ICAECC50550.2020.9339485.

Y. -Y. Chen, Y. -H. Lin, Y. -C. Hu, C. -H. Hsia, Y. -A. Lian and S. -Y. Jhong, "Distributed Real-Time Object Detection Based on Edge-Cloud Collaboration for Smart Video Surveillance Applications," in IEEE Access, vol. 10, pp. 93745-93759, 2022, doi: 10.1109/ACCESS.2022.3203053.

¿Qué es Python? - Explicación del lenguaje Python - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/python/

Suarez, O. D., Carrobles, M. D. M. F., Enano, N. V., Garcia, G. B., & Gracia, I. S. (2014). Opencv essentials. Packt Publishing, Limited.

Ultralytics. (s. f.). Inicio. Documentación Ultralytics YOLOv8. https://docs.ultralytics.com/es/#yolo-una-breve-historia

Muñoz, J. D. (2023, 14 abril). Qué es Flask. OpenWebinars.net. https://openwebinars.net/blog/que-es-flask/

Plantillas Jinja2. (s. f.). CódigoFacilito. https://codigofacilito.com/articulos/plantillas-jinja2

¿Qué es JavaScript? - Aprende Desarrollo web | MDN. (s. f.). MDN Web Docs. https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript

GraphEverywhere, E. (2021, 15 febrero). Machine learning en Python. GraphEverywhere. https://www.grapheverywhere.com/machine-learning-en-python/

Jetson Nano de NVIDIA. (s. f.). NVIDIA. https://www.nvidia.com/es-la/autonomous-machines/embedded-systems/jetson-nano/product-development/

Arquitectura Cliente-Servidor. (s. f.). https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/cliente-servidor

Patrones de diseño de aplicaciones: Maestro / Esclavo - NI. (2020, 2 septiembre). https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q000000x1r9CAA&l=es-ES

Del Prado, J. A. (2022). Qué es la metodología Design Sprint y sus fases. UXABLES | Blog. http://www.uxables.com/diseno-ux-ui/que-es-la-metodologiadesign-sprint-y-sus-fases/

Design Sprint: qué es, fases, ventajas y limitaciones. (2023, June 9). Becas Santander. https://www.becas-santander.com/es/blog/design-sprint.html

Ribas, E. (2022, October 26). Design Sprint, medir la viabilidad de un proyecto en 5 días. Thinking for Innovation. https://iebschool.com/blog/google-sprint-medirviabilidad negocio-analitica-usabilidad/

Get started with Jetson Nano Developer Kit. (s. f.). NVIDIA Developer. https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#intro

JetPack SDK 4.6 Release Page. (s. f.). NVIDIA Developer.https://developer.nvidia.com/embedded/jetpack-sdk-46

S. Sambolek and M. Ivasic-Kos, "Automatic Person Detection in Search and Rescue Operations Using Deep CNN Detectors," in IEEE Access, vol. 9, pp. 37905-37922, 2021, doi: 10.1109/ACCESS.2021.3063681.

Alfresco Docs - Solr replication. (s. f.). https://docs.alfresco.com/search-services/latest/config/replication/

ARQUITECTURA CLIENTE - SERVIDOR. (2015, 21 marzo). Inv101. https://tannyk.wixsite.com/inv101/single-post/2015/03/21/arquitectura-cliente-servidor