# SISTEMA DE DETECCIÓN, EXTRACCIÓN Y RECONOCIMIENTO DE ROSTROS EN ESCENAS DE MÁXIMO 4 PERSONAS, PARA APLICACIONES DE VIDEOVIGILANCIA RESIDENCIAL UTILIZANDO HERRAMIENTAS DE SOFTWARE LIBRE, EN LUGARES CERRADOS

**UNIVERSIDAD DISTRITAL**
FRANCISCO JOSÉ DE CALDAS


**David Fernando Vásquez Bohórquez**

**COD: 20182383009**

**Cristian Steven Cortes Martínez**

**COD: 20182383003**

**UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS**
**Facultad Tecnológica**
**Ingeniería en Control y Automatización**


Bogotá D.C. Agosto, 2021

# SISTEMA DE DETECCIÓN, EXTRACCIÓN Y RECONOCIMIENTO DE ROSTROS EN ESCENAS DE MÁXIMO 4 PERSONAS, PARA APLICACIONES DE VIDEOVIGILANCIA RESIDENCIAL UTILIZANDO HERRAMIENTAS DE SOFTWARE LIBRE, EN LUGARES CERRADOS

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

**David Fernando Vásquez Bohórquez**
**COD: 20182383009**
**Email: dfvasquezb@correo.udistrital.edu.co**

**Cristian Steven Cortes Martínez**
**COD: 20182383003**
**Email: cscortesm@correo.udistrital.edu.co**

**Monografía**

**Presentado para optar al título de: Ingeniero(a) en Control y Automatización**

**Director**
**Ing. Miguel Pérez Pereira**

**UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS**
**Facultad Tecnológica**
**Ingeniería en Control y Automatización**
Bogotá D.C. Agosto, 2021

# Índice

Índice ..... 1
Índice de Figuras ..... 3
Índice de Tablas ..... 5
Índice de Anexos ..... 6
Resumen ..... 7
1. Introducción ..... 1
2. Planteamiento del problema ..... 2
3. Justificación ..... 3
4. Objetivos ..... 4
4.1. Objetivo General ..... 4
4.2. Objetivos específicos ..... 4
5. Marco de referencia ..... 5
5.1. Antecedentes ..... 5
5.2. Marco teórico ..... 10
5.2.1. Visión artificial. ..... 10
5.2.2. Cámaras. ..... 12
5.2.3. Procesamiento digital de imágenes. ..... 12
5.2.4. Reconocimiento Facial. ..... 13
5.2.5. Python ..... 15
5.2.6. Librería OpenCV. ..... 16
5.2.7. Machine Learning. ..... 16
5.3. Marco Legal ..... 17
6. Metodología ..... 18
6.1. Identificación de métodos de detección, extracción y reconocimiento de Rostro ..... 18
6.1.1. Métodos de detección de rostros. ..... 18
6.1.2. Métodos de extracción de rostros. ..... 24
6.1.3. Métodos de reconocimiento de rostros. ..... 26
6.2. Desarrollo de sistema de detección, extracción y reconocimiento de rostros en escena de múltiples personas. ..... 32
6.2.1. Reconocimiento facial mediante la librería Face-Recognition ..... 33

6.2.2. Reconocimiento facial mediante OpenCV ..... 35
6.3. Diseño interfaz gráfica de usuario GUI. ..... 39
7. Resultados ..... 46
7.1. Diseño definitivo. ..... 46
7.2. Restricciones a la solución. ..... 46
7.2.1. Restricciones algoritmo Face-Recognition ..... 46
7.2.2. Restricciones algoritmo OpenCV. ..... 46
7.3. Validación de los resultados. ..... 47
7.3.1. Prueba de distancias. ..... 47
7.3.2. Prueba de tiempo. ..... 49
7.3.3. Prueba de ángulo. ..... 52
7.4. Validar el funcionamiento del sistema de videovigilancia residencial. ..... 54
8. Conclusiones y Recomendaciones ..... 67
9. Trabajo futuro ..... 68
10. Referencias ..... 69
11. Anexos. ..... 72

## Índice de Figuras

Figura 1. Sistema basado en visión artificial(Gonzáles & Zamara, 2014) ..... 11
Figura 2. Técnicas de Reconocimiento Facial (Lu et al., 2003) ..... 13
Figura 3 Imagen Integral desde el pixel (0,0) hasta pixel (x,y)(Barroso, 2014) ..... 18
Figura 4. Subregiones en una imagen integral(Barroso, 2014) ..... 19
Figura 5. Ejemplo de característica para evaluar la existencia del conjunto de ojos y nariz. a) Evaluación sobre una ventana que no es una cara; b) Evaluación sobre una ventana con una cara(Barroso, 2014) ..... 19
Figura 6. Características usadas por el clasificador(Barroso, 2014). ..... 20
Figura 7. Representación de características sobre una imagen. Fuente: Autor ..... 20
Figura 8. Características entrenadas con AdaBoost (Barroso, 2014) ..... 21
Figura 9. Diagrama de funcionamiento HOG(Alcántara Montiel, 2019)..... 22
Figura 10. Filtro o kernel para extracción de características. (Cowley, 2018)..... 23
Figura 11. Cambio de una imagen al aplicar un kernel.a) Imagen de entrada sin aplicar filtros.b)Imagen resultante al aplicar un filtro o kernel (Cowley, 2018) ..... 23
Figura 12. Usos de filtro en una matriz 4 x4 (Bagnato, 2020) ..... 24
Figura 13. Muestreo mediante Max Pooling (Bagnato, 2020) ..... 24
Figura 14. Detección y extracción de las coordenadas del rostro, Fuente: Autor ..... 25
Figura 15. Línea de código de extracción de rostro. Fuente. Autor ..... 25
Figura 16. Extracción de rostro. Fuente: Autor ..... 25
Figura 17. Recta que minimiza distancias de los puntos(Costa Mari, 2020). ..... 26
Figura 18. Conversión de la región de una imagen a su resultante mediante LBPH. (Costa Mari, 2020)..... 29
Figura 19. Zonas definidas por operador BLP. (Costa Mari, 2020)..... 29
Figura 20. Extracción de rasgos faciales mediante la variación de la iluminación de la imagen(Costa Mari, 2020). ..... 30
Figura 21. Puntos de referencia del rostro humano. (Geitgey, 2016)..... 30
Figura 22. Transformación de la imagen para realizar un centrado de imagen. (Geitgey, 2016) . 31
Figura 23. Procedimiento para el entrenamiento óptimo de la red neuronal.(Geitgey, 2016) ..... 31
Figura 24. Diagrama general del algoritmo desarrollado. ..... 32
Figura 25. Etapas sistema de reconocimiento facial..... 33
Figura 26. Diagrama de flujo técnicas implementadas del algoritmo de Face-Recognition. ..... 34
Figura 27. Diagrama de flujo técnicas implementadas del algoritmo de OpenCV. ..... 35
Figura 28. Diagrama de flujo sistema de videovigilancia residencial Fuente: Autor..... 38
Figura 29. Ventana principal Fuente: Autor ..... 39
Figura 30. Selección de cámara Fuente: Autor ..... 39
Figura 31. Selección de la resolución Fuente: Autor..... 40
Figura 32. Ventana administrar usuarios de Face-Recognition Fuente: Autor ..... 40
Figura 33. zona para eliminar un usuario Fuente: Autor ..... 41
Figura 34. Ventana nombre y foto a archivar Fuente: Autor..... 41
Figura 35. Dialogo confirmando la decisión de añadir al usuario Fuente: Autor ..... 42
Figura 36. Reconocimiento Facial con librería Face-Recognition Fuente: Autor..... 42
Figura 37. Ventana administrar usuarios de OpenCV Fuente: Autor..... 43
Figura 38. Dialogo de advertencia Fuente: Autor ..... 43
Figura 39. Frame para captura de fotos de OpenCV Fuente: Autor..... 44
Figura 40. Dialogo para la confirmación de guardar usuario Fuente: Autor..... 44

Figura 41. Dialogo confirmar si desea añadir otro usuario Fuente: Autor ..... 44
Figura 42. Dialogo informativo acerca del modelo de entrenamiento Fuente: Autor ..... 45
Figura 43. Reconocimiento Facial con librería Open CV Fuente: Autor ..... 45
Figura 44. Comportamiento resolución cámara vs distancia vs resolución detección Face-Recognition..... 48
Figura 45. Comportamiento resolución cámara vs distancia vs resolución detección OpenCV ... 49
Figura 46. Grafica de comportamiento Resolución vs personas vs tiempo Face-Recognition ..... 50
Figura 47. Grafica de comportamiento Resolución vs personas vs tiempo OpenCV ..... 51
Figura 48. Prueba ángulo de reconocimiento Fuente: Autor ..... 52
Figura 49. Prueba de ángulo Face-Recognition Fuente: Autor ..... 53
Figura 50. Prueba de ángulo OpenCV Fuente: Autor ..... 53
Figura 51. Residencia en la cual se validó el funcionamiento Fuente: Autor ..... 54
Figura 52. Validación algoritmo de Face-Recognition reconociendo la identidad de una persona en un lugar residencial recorrido izquierda a derecha Fuente: Autor..... 55
Figura 53. Validación algoritmo de Face-Recognition reconociendo la identidad de una persona en un lugar residencial recorrido derecha a izquierda Fuente: Autor..... 55
Figura 54. Verificación del reconocimiento de 2 personas mediante la librería Face-Recognition en residencia Fuente: Autor..... 56
Figura 55. Verificación del reconocimiento de 3 personas mediante la librería Face-Recognition en residencia. Fuente: Autor..... 56
Figura 56. Verificación del reconocimiento de 4 personas mediante la librería Face-Recognition en residencia Fuente: Autor..... 57
Figura 57. Prueba de ángulo librería OpenCV en zona residencial Fuente: Autor ..... 58
Figura 58. Prueba de reconocimiento sin ángulo de giro. Fuente: Autor ..... 58
Figura 59. Prueba de reconocimiento facial de dos personas en una residencia. Fuente: Autor... 59
Figura 60. Prueba de reconocimiento facial de tres personas en una residencia. Fuente: Autor .. 59
Figura 61. Prueba de reconocimiento facial de tres personas en una residencia y un desconocido. Fuente: Autor ..... 60
Figura 62. Validación del algoritmo de Face-Recognition en un centro comercial reconociendo 1 persona Fuente: Autor ..... 61
Figura 63. Validación del algoritmo de Face-Recognition en un centro comercial reconociendo 2 personas Fuente: Autor ..... 61
Figura 64. Validación del algoritmo de Face-Recognition en un centro comercial reconociendo 3 personas. Fuente: Autor ..... 62
Figura 65. Validación del algoritmo de Face-Recognition en un centro comercial reconociendo 4 personas. Fuente: Autor ..... 63
Figura 66. Validación del algoritmo de OpenCV en un centro comercial reconociendo 1 persona. Fuente: Autor ..... 63
Figura 67. Validación del algoritmo de OpenCV en un centro comercial reconociendo 2 personas Fuente: Autor ..... 64
Figura 68. Validación del algoritmo de OpenCV en un centro comercial reconociendo 3 personas. Fuente: Autor ..... 65
Figura 69. Validación del algoritmo de OpenCV en un centro comercial reconociendo 4 personas. Fuente: Autor ..... 66

## Índice de Tablas

Tabla 1. Resultado prueba de distancias Face-Recognition ..... 47
Tabla 2. Resultado prueba de distancias OpenCV ..... 48
Tabla 3. Resultados prueba de tiempo Face-Recognition ..... 50
Tabla 4. Resultados prueba de tiempo OpenCV..... 51
Tabla 5. Resultados prueba de ángulo Face Recognition..... 52
Tabla 6. Resultados prueba de ángulo OpenCV..... 53

## Índice de Anexos

Anexo A. Código implementado para el sistema de reconocimiento..... 72

## Resumen

La protección de las personas y sus bienes es un punto que se busca mejorar día tras día, el avance de la tecnología permite que el campo de la seguridad electrónica sea cada vez más efectiva para brindar un confort a los usuarios, quitando esa carga mental de estar pensando que ellos o sus bienes serán afectados por personas inescrupulosas, el desarrollo de sistemas de seguridad ha avanzado mucho, y uno de los campos más nuevos y que ha manifestado este avance es el reconocimiento facial, dicho campo, proporciona la capacidad de detectar y reconocer rostros humanos en video automáticamente, generando una señal de alarma o control, ya sea para alertar a la persona o permitirle el ingreso.

Los desarrollos actuales que cuentan con la característica de reconocer múltiples rostros en video, suelen tener elevados costos, debido a que son implementados en plataformas que no están basadas en software libre, lo que aumenta el costo debido a licencias de pago por uso, Además de esto, se presenta una escasez de estudios que demuestren cuán eficiente es un sistema que esté desarrollado en software libre, y que cuente con la detección facial al interactuar con múltiples rostros, que sea confiable para la protección de bienes, y que cuente con la capacidad de identificar personas que deseen acceder a lugares de residencia sin contar con la respectiva autorización.

En esta investigación se encontrará en primera instancia una contextualización teórica de los tipos de algoritmos y librerías para detección y reconocimiento facial que existen en Python, dichas librerías son las denominadas OpenCV y Face-Recognition, en vista de que no se tiene un estudio que muestre distancia máxima de detección, tiempos de ejecución y ángulos de operación, entre otros, en esta investigación se decidió hacer pruebas a ambos algoritmos para determinar cuál es más efectivo o cuál se ajusta mejor al sistemas de videovigilancia propuesto en esta investigación.

A continuación, se mostrará el algoritmo propuesto por los investigadores el cual fusiona diferentes programas y librerías que permiten la detección, extracción y reconocimiento de múltiples rostros en tiempo real, además, se mostrará la interfaz gráfica de usuario (GUI) que permite la total manipulación de este sistema, por último, se mostrará las pruebas que realizaron al sistema en dos ambientes diferentes, uno residencial y uno comercial. Es de aclarar, que todo el sistema fue desarrollado en el lenguaje de programación Python el cual es libre.

*Palabras claves: Reconocimiento facial, OpenCV, FaceRecognition, Video Vigilancia, Biometría, Procesamiento de Imagen.*

# 1. Introducción

Los sistemas de videovigilancia residencial son dispositivos de seguridad que hoy en día se implementan principalmente en zonas residenciales con alta afluencia de personas, para la adquisición, mantenimiento y uso de estos sistemas, los residentes determinado conjunto, pagan una cuota moderadora que les permite hacer uso de este servicio, el cual se encarga de prevenir hurtos o identificar los responsables que cometan actos delictivos, teniendo en cuenta que la mayoría de estos dispositivos cuentan con funciones como grabación de video, grabación de audio y detección de movimiento.

Hay que robustecer los sistemas de seguridad con las herramientas que han surgido en los últimos años y como menciona Juan Yanes en un artículo en la revista innovación *"no basta con tener una imagen de alta calidad y resolución, sino que también son necesarias analíticas inteligentes de video que les permitan a los operadores beneficiarse de las herramientas necesarias para efectuar una mejor labor de seguridad"*(Yanez, 2018), algunas de estas herramientas son conteo de personas, mapas de calor, tracking, detección facial de emociones, detección de objetos, reconocimiento facial, herramientas que se pueden implementar en áreas que van desde lo comercial, hasta el área de seguridad, para potenciar mediante el empleo de herramientas tecnológicas los servicios anteriormente mencionados .

Actualmente los avances de video vigilancia se han unificado con herramientas inteligentes para brindar soluciones más efectivas ante cualquier acción que altere la tranquilidad y seguridad en una zona residencial, sin embargo, obtener un servicio de videovigilancia residencial inteligente que brinden funciones como reconocimiento facial, trae consigo que el costo del sistema aumente exponencialmente, cifra que puede ser desproporcionada a la hora de emplear estos dispositivos en casas familiares o aun en zonas residenciales estrato 1,2,3 o 4.

Hoy en día, gracias al acceso a la información e investigaciones, es posible implementar o diseñar sistemas de videovigilancia que reduzca notablemente los costos de los sistemas comerciales, sistemas que son diseñados y desarrollados sobre software libre, empleando herramientas de uso cotidiano como una laptop, o un computador de escritorio en conjunto con una cámara web, por lo cual en este documento se desarrolla un sistema de detección, extracción y reconocimiento de rostros en escenas de múltiples personas para aplicaciones de videovigilancia residencial de bajo costo. Para ello se realizó un proceso de investigación de algoritmos de detección, extracción y reconocimiento de rostros que permitiera detectar e identificar varios rostros en video para desarrollar un sistema de video vigilancia residencial, en el cual se evaluaron aspectos como distancia máxima de reconocimiento, resolución de la detección facial en relación con la distancia, tiempos de respuesta del algoritmo variando la cantidad de personas que el sistema debe detectar e identificar, y ángulos de giro del rostro para un óptimo reconocimiento facial, y a su vez permitiera registrar a los residentes de la vivienda, detectar, extraer el rostro e identificar las personas que transitan por determinada área de la residencia.

## 2. Planteamiento del problema

Según los datos entregados por la policía Nacional de Colombia en el año 2020 se presentaron 8284 hurtos a residencia en Bogotá entre el periodo de 1 de Enero al 31 de Diciembre del mismo año, y a nivel nacional se presentaron 32324 hurtos residenciales(Angel, 2021). Los medios utilizados para cometer este delito fueron, llaves maestras, palancas, escopolamina, armas blancas y armas de fuego.

El experto en seguridad Jofre Montoto en un debate realizado por la vanguardia sobre la eficacia de las cámaras de videovigilancia en la prevención de delitos afirmó *"Más que reducir la violencia lo que hacen es prevenirla". "Está demostrado que su colocación y sobre todo su publicitación tienen un efecto disuasorio y en caso que se cometa un delito ejercen como prueba inexcusable ante un juez"*(Meritxell, 2011) lo cual demuestra que ante el punto de vista de Montoto el tener cámaras en una zona residencial ayuda a prevenir hurtos y en caso de hurto sirve como material probatorio para encontrar, capturar y judicializar, en caso de ser necesario, a los responsables.

Los sistemas de videovigilancia que tienen como función principal almacenar horas de video para posteriormente obtener los registros de las personas que ingresan a determinado lugar, pueden ser potenciados al implementar tecnologías de reconocimiento facial que se encarguen de comparar los rostros captados en pantalla, con una base preestablecida que permita saber quién es la persona que está en la residencia o recinto.

En la actualidad la necesidad de proteger bienes o personas hace que se busquen medidas de protección cada vez más avanzadas y eficientes que permitan fortalecer e incrementar la seguridad (Sánchez Calle, 2005). El desarrollo en hardware y software que disponen los sistemas de seguridad han tenido gran evolución en cuanto a reconocimiento facial (Pereyra, 2018). Múltiples investigaciones y desarrollos presentan detección e identificación de un rostro en video y generan una señal de alarma o control para dar acceso a la persona (Porras Aldana, 2018) (Castaño & Sierra, 2019), el problema surge cuando se desea detectar, extraer e identificar varios rostros en escenas de múltiples personas, pues la mayoría de plataformas desarrolladas sobre software libre solo reconocen un rostro a la vez , además lo sistemas que se encargan de detectar e identificar varios rostros presentan elevados costos de adquisición  y mantenimiento, esto representa un gran riesgo a la seguridad, pues en caso de requerir un sistema que videovigilancia que interactúe con múltiples rostros, los excesivos costos no permiten que cualquier persona pueda acceder a un sistema de video vigilancia residencial. (Encalada, 2015)(Arellano & Cortijo, 2020) (Morcote, 2020)

¿Es posible desarrollar un sistema de reconocimiento facial de bajo costo, fácil implementación y óptimo desempeño que reconozca más de un rostro en un mismo video?

## 3. Justificación

El proyecto va orientado al sector de videovigilancia residencial, aplicado específicamente a los sistemas de seguridad, sector que en los últimos años ha implementado nuevas tecnologías mediante las cuales el usuario final ha adquirido una mayor sensación de tranquilidad y seguridad, de ahí que, estos sistemas de seguridad son de gran aplicación en acceso a hogares, empresas, cuidado de instalaciones, entre otros, pero hay poco desarrollo en cuanto a algoritmos de detección de un rostro e identificación, que permitan la protección de bienes materiales o la integridad del usuario, y que identifique personas desconocidas en lugares concurridos.
Este proyecto proporcionará la información necesaria para el diseño, desarrollo e implementación de sistemas de videovigilancia de bajo costo debido al empleo de software libre, los cuales podrán ser usados en el territorio nacional con la finalidad de aumentar la seguridad en entornos residenciales. En términos de investigación existe muy poca información referente a detección e identificación de un rostro en escenas de múltiples personas en un video y que a su vez este sistema sea de bajo costo, por lo cual, este desarrollo sentará bases para nuevos proyectos los cuales se le puedan incorporar nuevas tecnologías aumentando su eficiencia o su área de aplicación.

# 4. Objetivos

## 4.1. Objetivo General

Desarrollar un sistema de detección, extracción y reconocimiento de rostros en escenas de máximo 4 personas, para aplicaciones de videovigilancia residencial utilizando herramientas de software libre, en lugares cerrados.

## 4.2. Objetivos específicos

- Identificar algoritmos que se emplean en detección y reconocimiento facial, a fin de seleccionar los adecuados para el desarrollo del sistema.
- Implementar algoritmos de detección y extracción de rostros en escenas en donde se encuentre máximo 4 personas para aplicaciones residenciales.
- Implementar el algoritmo de reconocimiento de rostros aplicando software libre.
- Diseñar una interfaz gráfica de usuario que permita la correcta operación del sistema de detección, extracción e identificación de rostros en escenas de múltiples personas.
- Validar el funcionamiento de la aplicación mediante pruebas inicialmente en lugares cerrados(controlados).

# 5. Marco de referencia

## 5.1. Antecedentes

A continuación, se listan algunos desarrollos de reconocimiento facial a partir del procesamiento digital de imágenes.

En (Encalada, 2015) Juan Carlos Jiménez Encalada de la universidad Politécnica salesiana en el año 2015 presenta en su trabajo de grado diferentes metodologías de reconocimiento de rostro para posteriormente aplicarlas en videovigilancia, Jiménez inicialmente realiza una revisión de los métodos de identificación y reconocimiento principales dentro de las cuales se encuentran los métodos Holísticos y los métodos basados en características, Jiménez hace uso de PCA y LDA por parte de los métodos holísticos, y por parte de los métodos basados en características hace uso de SURF y SIFT. El autor en el desarrollo del trabajo hace uso de un software libre y aprovecha la librería OpenCV para todo el tratamiento de las imágenes, finalmente concluye que los métodos holísticos son una buena alternativa referente al reconocimiento facial siempre y cuando se puedan tener varias muestras de una misma persona ya que la persona a través del tiempo varía en cuanto a su barba, el método también puede fallar si se usa gafas, poca o mucha iluminación, si el rostro tiene cierta inclinación, además que para mejorar el desempeño es necesario ir alimentando una base de datos con mayor información, lo cual requiere una cantidad mayor de almacenamiento, por otro lado, los métodos por descriptores son muchos más precisos incluso invariantes ante ciertas condiciones como la barba, el uso de gafas o inclinación del rostro, pero es necesaria una iluminación constante. Afirma el autor que es indispensable que las cámaras que van a ser usadas para el reconocimiento estén a una altura promedio de 2 metros para evitar problemas con la inclinación del rostro en la captación del video y a su vez tengan una fuente constante de iluminación.

Damián Gutiérrez de la universidad politécnica Salesiana en 2021 hace uso del reconocimiento facial específicamente para sistemas de videovigilancia doméstica. La tesis tiene como objetivo el diseño e implementación de reconocimiento facial aplicado a seguridad en el hogar, el sistema monitorea el ingreso de las personas validando el rostro en un base de datos ya cargada previamente, otorgando o no el acceso a la persona, en caso de que la persona no esté autorizada enviará una alarma a los dispositivos móviles registrados indicando el acceso no deseado. Gutiérrez implementa 4 módulos para el funcionamiento total del sistema, módulo de ingreso de imágenes que se utiliza para entrenar el sistema, módulo de ingreso del video encargado del procesamiento y análisis, módulo del proceso de reconocimiento donde se etiquetara a la persona y el módulo de entrega de resultados. El autor desarrolla una aplicación Web que permite la interacción del sistema de reconocimiento facial con los usuarios. En la aplicación, el usuario puede desempeñar el papel de un usuario común o administrador, la diferencia entre estos dos usuarios es que el primer usuario puede registrarse y ver en el sistema informes y notificaciones que se han generado. El segundo se encarga de registrar nuevos datos y rostros. Los módulos de detección e identificación están desarrollados en Python haciendo uso de la biblioteca OpenCV, la aplicación Cascadas de Haar para la detección, OpenCV y Face-Recognition se utilizan para el reconocimiento facial. La cámara IP Hikvision se utiliza para capturar y adquirir imágenes y para el procesamiento de imágenes Dispositivo embebido Raspberry PI 4 Modelo B. El autor llega a la conclusión que el procesamiento que realiza el algoritmo es demasiado alto y por tanto no es recomendable el uso de Raspberry Pi 4 para la implementación, por otro lado, para que el algoritmo sea más efectivo se

requiere de una base de datos de fotografías más grande, pero al tener más fotografías el procesamiento aumenta y a su vez la lentitud en el procesamiento. En cuanto a la etapa de experimentación y pruebas concluyó que factores como: la distancia entre el rostro y la cámara, la resolución, el ángulo de captura de la cámara, el número de fotos de cada individuo y el nivel de luminosidad influyen significativamente en la identificación del sujeto y en la velocidad del procesamiento. Por último, Gutiérrez recomienda para futuros trabajos con visión artificial dispositivos con altas prestaciones en cuanto a GPU's y tarjetas gráficas; trabajar con una base de datos de al menos 50 fotografías por persona tomadas con distintos niveles de luminosidad, distancias diferentes y ángulos diversos con el fin de obtener resultados más precisos. (Meneses, 2021)

La seguridad en el Centro de Educación Inicial N. 1 se ha visto afectada debido a que no hay control sobre la puerta de ingreso, permitiendo que personas desconocidas logren ingresar, provocando pérdida de bienes y poniendo en peligro a los estudiantes de la institución, en consecuencia, Leonardo Rubén Arellano estudiante de la Universidad Tecnológica Israel en el año 2020 decide Implementar un sistema de video vigilancia mediante algoritmos de Deep Learning y machine Learning para el Centro de Educación No 1. El sistema consta de un sensor de proximidad, sensor magnético, placa Arduino, pantalla LCD de 5" y una chapa magnética. El sistema de reconocimiento facial se desarrolla en Python haciendo uso de librerías como: OpenCV, DLIB, Face-Recognition entre otras, y este a su vez consta de dos programas, el primero contiene el algoritmo de entrenamiento, el cual es utilizado cada que se requiera agregar o eliminar un usuario de la base de datos del sistema, para agregar un usuario, se requiere un mínimo de 5 fotografías digitales. El segundo algoritmo es el que se ejecuta permanentemente y es el encargado de reconocer el rostro y enviar la señal de control a la cerradura magnética en caso de tener una identificación acertada. Para la detección de rostros se hace uso de histograma de gradientes HOG y el reconocimiento es basado en Tensorflow con una base ya pre-entrenada. Como recomendaciones el autor sugiere tener una fuente de luz constante para cuando llegue la noche y tener un mínimo de 20 fotos por persona con el fin de mejorar la exactitud del sistema, también recomienda tener una computadora asignada solo para este fin ya que el procesamiento de las imágenes requiere un alto costo computacional.(Arellano & Cortijo, 2020)

El estudiante de la Escuela Politécnica Nacional Cristian Salazar, año 2017, realiza como trabajo de grado la evaluación teórica de dos algoritmos con diferentes técnicas de detección de rostros como lo es Viola&Jones y Distancia de Hausdorff, concluyendo que Viola&Jones presenta mejores características de acuerdo al análisis teórico, a partir de esto, el autor decide implementar el algoritmo de Viuola&Jones y compararlo con el código para detección y seguimiento de rostros de Mathworks de Matlab, sometiendo ambos códigos a diferentes escenarios con iluminación limitada, inclinación vertical y horizontal de la cámara web, rostros girados a 45 y 90 grados, rostros con facciones incompletas, rostros a blanco y negro y rostros confusos, de donde resulta que el código desarrollado tiene un porcentaje de detección favorable de 92% y el código de Mathworks tan solo 74%. Concluyendo que el método de Viola&Jones por su alta efectividad y velocidad de procesamiento es el adecuado para sistemas de videovigilancia en tiempo real, sin embargo, se aclara que para cualquiera de los métodos evaluados lo que genera mayor problema en la detección de un rostro son las condiciones de iluminación, objetos extraños en el fondo y posición de la persona. (Sanches, 2017)

En la universidad de Zaragoza en el año 2012 Martin Valero decide hacer uso del método de Viola&Jones para la detección de rostros en su trabajo de grado, concluyendo que este método tiene resultados satisfactorios con un bajo coste computacional, lo cual lo hace muy poderoso en aplicaciones de tiempo real. Para el reconocimiento de rostros hace uso de la técnica Eigenfaces, basada en el análisis de componentes principales (PCA) y recomienda realizar una correcta selección de las imágenes de entrenamiento para evitar falsas detecciones y no ingresar un gran número de caras de entrenamiento, ya que a mayor número de caras mayor tiempo de procesamiento y a su vez más tiempo para comprobar una persona. Ya que el método de Eigenfaces siempre da como resultado la persona más parecida dentro de la base de datos el autor incluye un nivel de confianza para saber el rostro detectado con que confianza acertó y estudia la posibilidad de incluir un etiquetado en los rostros detectados. El autor hace uso del entorno de desarrollo integrado de Microsoft Visual C++ en su versión Express y de la librería de código libre OpenCV logrando así tener un aplicativo de detección y reconocimiento de rostros con características como: entrenamiento online y offline, y la detección de rostros en tiempo real, usando un video pregrabado o cargando una imagen tomada, para el reconocimiento del rostro elegido por el usuario o reconocimiento del rostro en video que presenta la cámara web, entre otras. Con respecto al etiquetado de la persona, no se logra obtener debido a que si el nivel de confianza es muy alto se tendrán personas sin reconocer y si es muy bajo una persona tendrá varias etiquetas o etiquetas erróneas. (Valero, 2012)

Darío Villalón en el año 2012 hace uso para su tesis de grado en la universidad de Chile el lenguaje C++ para el desarrollo del código, la librería OpenCV para el manejo y procesamiento de imágenes y el algoritmo de análisis de componentes principales (PCA) basado en Eigenfaces para el reconocimiento facial. El aplicativo cuenta con una etapa de entrenamiento donde se cargan todas las imágenes del individuo a detectar y el aplicativo entrena el sistema, en la siguiente etapa se carga el video a evaluar y el sistema detecta los rostros, pero no los reconoce aún, por último, en la etapa de resultados el aplicativo muestra algunos datos estadísticos y la posibilidad de visualizar la escena donde se reconoció el rostro objetivo entrenado previamente. Como resultado el aplicativo tiene un 73% de efectividad en el caso que las fotos de entrenamiento fueron tomadas del mismo video que se usó para la evaluación y un 68% cuando las fotos eran diferentes a las del video, con lo cual el autor concluye que el sistema desarrollado es sensible a la calidad de las imágenes con las cuales se realiza el entrenamiento, debido al grado de luminosidad, inclinación y gestos presentados en la imagen. (Villalon de la Vega, 2012)

En el año 2019 en la Universidad Católica de Colombia David Castaño Saavedra desarrolla como trabajo de grado un sistema que reconoce el rostro de una persona para el ingreso a una vivienda, por la parte de hardware hace uso de una Raspberry Pi 3 modelo B+ con una cámara full HD Ecam 8000 y una pantalla táctil de 3,5 pulgadas, por otro lado, en cuanto a software utiliza Linux, como lenguaje de programación Python, la librería OpenCV para la manipulación de imágenes y Tkinter para la interfaz gráfica de usuario. Como resultado obtiene un sistema que permite grabar en su base de datos un mínimo de 4 rostros diferentes, en la parte de detección se detecta un único rostro en pantalla y da un acceso en pantalla si el rostro es conocido, todo lo anterior haciendo uso de técnica de Eigenfaces. El autor concluye que la técnica de Eigenfaces es muy eficiente y de rápido procesamiento, sin embargo, se vio afectada cuando el rostro tenía cierta inclinación, ya que todas las imágenes de entrenamiento eran sin inclinación, así mismo aclaró que ante diferentes posiciones y condiciones de luz la librería es consistente y asertiva. (Castaño & Sierra, 2019)

El ingeniero en sistemas Rafael Eduardo Pedroza egresado de la universidad de Cartagena en el año 2019 realizó un sistema de ubicación dinámica para localizar e identificar personas desaparecidas en Colombia. El proyecto emplea 2 servidores en la plataforma EC2, en cada uno de los servidores se ejecuta un servicio WEB, uno encargado de la gestión de información y otro encargado del análisis de datos biométricos. Además, se emplearon dos clientes para cada servicio web, uno encargado de la visualización y gestión de la información, y el segundo encargado de capturar imágenes con información biométrica de las personas con el fin de enviarlas al servicio web de análisis. El sistema tiene una etapa de registro de personas desaparecidas, en donde se envían las imágenes e información a un bucket, el cual responde enviando esa información a un servicio web. Posteriormente se debe entrenar el sistema con toda información de las personas desaparecidas, para ello la persona encargada del aplicativo da la orden mediante la interfaz de usuario, las imágenes son descargadas del bucket y el sistema se entrena con esta nueva información. Para el sistema de reconocimiento se captura la imagen del entorno, después es enviado a un cliente encargado de la etapa de detección facial, éste aísla el rostro y envía esta información a un servicio web encargado de comparar las imágenes, y en caso de que se presente similitud con la imagen de una persona desaparecida emitir una alerta. Se concluyó que el sistema tiene un grado de funcionalidad aceptable, una de las limitaciones de este proyecto fue que para entrenar el modelo se necesitan mínimo 10 imágenes para que la librería OpenCV realice un funcionamiento óptimo, y en algunos casos no se cuenta con la cantidad de imágenes suficientes, se observó que el uso de accesorios baja el nivel de confianza del sistema por lo tanto es de difícil implementación en un espacio cotidiano con todas las variables que representa, se determina que el proyecto tiene mayor viabilidad en ambientes controlados.(Pedroza, 2019)

El ingeniero Mauricio Saldarriaga Quintero egresado del programa de bioingeniería de la universidad de Antioquia desarrolló un sistema de reconocimiento orientado a la identificación de emociones de los sujetos que se encuentran registrados en el aplicativo. Para el desarrollo del proyecto se implementó la librería Tensorflow además del módulo face-api el cual permite obtener parámetros como edad, género y estimar expresiones faciales. Mediante estas librerías es posible obtener los puntos de referencia de la cara los cuales contribuyen a identificar al sujeto en caso de estar registrado, además se empleó la base de datos MySQL para almacenar tanto la información del usuario, como sus características faciales y hora del registro en el aplicativo. En las pruebas realizadas se determinó que los accesorios que más afectan la eficiencia del algoritmo son el uso de gorras y personas con el rostro hinchado pues los puntos de detección se ubican de forma diferente en la cara, el tiempo de respuesta del sistema es menor de 2 segundos. En cuanto a eficiencia de detección se concluyó que el sistema diseñado tiene un alto porcentaje de eficiencia en expresión de felicidad y neutralidad, obteniendo una precisión de 98%, en el caso del género tiene una precisión de 94% y la estimación de edad no es fiable por un alto margen de error.(Saldarriaga Quintero, 2020)

El ingeniero Julián David Pardo Morcote, egresado de la universidad Santo Tomás Seccional Tunja, desarrolló un sistema de validación de identidad orientado a usarse en audiencias legales evitando posibles casos de suplantación de identidad, y brindar transparencia en los procesos de audiencias legales en modalidad virtual. Para el desarrollo del proceso se emplearon las librerías Face_Recognition y dlib encargadas de las etapas de reconocimiento facial y machine Learning, mediante la librería PyAutogui se toma una captura de la pantalla y se selecciona el área de interés que sería desde el borde izquierdo hasta la mitad de la pantalla, esta sección representa el área en donde se encuentra el rostro de la persona. El algoritmo se encargaba de encontrar los bordes,

dentro de los bordes ubica el rostro y se colocan los puntos de referencia biométricos, las imágenes de caras conocidas son accedidas mediante la red local LAN y transmitidas por el puerto ethernet con un protocolo TCP/IP. Mediante el algoritmo euclidiano se halla la distancia entre los puntos de referencia del rostro que será evaluado y se determinará si las medidas de referencia calculadas corresponden con las de un rostro conocido, en caso de ser así en la pantalla se proyectará el aviso "CONOCIDO", y en caso contrario "DESCONOCIDO". La arquitectura implementada es cliente-servidor, en el cliente se encuentra la base de datos con la información de los usuarios registrados y en el servidor la interfaz para el reconocimiento facial. El proyecto se consiguió implementar de forma óptima, obteniendo alta fiabilidad, fácil ejecución y se concluyó que es potencialmente implementable, se determinó que el programa en ocasiones realiza reconocimientos incorrectos por lo cual se recomienda ejecutar varias veces el algoritmo durante la videollamada, para mejorar el rendimiento del algoritmo se debe mejorar la resolución dado a que a menor a resolución se presenta mayor pérdida de información esencial del rostro humano, otro elemento que potencia el algoritmo de reconocimiento es tener buena iluminación y tener el rostro despejado.(Morcote, 2020)

Juan Manuel Aldana Porras estudiante de la Universidad Abierta y a distancia y Escuela de Ciencias Básicas, Tecnología e Ingeniería realizó un proyecto de control de ingreso al CEAD de Ibagué implementando un sistema de seguridad biométrico. Para el desarrollo del proyecto se empleó una cámara logitecC310 de calidad HD y un computador portátil con sistema operativo Ubuntu y de alta capacidad de procesamiento, para detección de rostros se empleó la librería Face-Recognition. El proceso inicialmente consiste en adquirir la imagen y realizar una conversión a escala de grises, posteriormente se aplican descriptores HOG hallando gradientes en la imagen con respecto al entorno, mediante Face Land Mark Estimation se identifican 68 puntos esenciales del rostro humano y de esta manera aislar el rostro. Posteriormente se emplea una red convolucional la cual ya fue entrenada con información del personal autorizado, este algoritmo ubica 128 puntos de interés en el rostro y compara la información. Durante los 240 minutos de prueba del sistema se evidencio que la escala de detección, resolución de imagen afectan la eficiencia del sistema de detección, además se encontró que el sistema implementado obtuvo una sensibilidad de detecciones del 93% una especificidad y exactitud del 98% y una precisión de 91% determinando que es un sistema robusto y que puede ser implementado en sistemas de seguridad.(Porras Aldana, 2018)

En la universidad Distrital Francisco José de Caldas el ingeniero Edison Rene Caballero Barriga estudiante de especialización en ingeniería de software en el año 2017 realizó un prototipo de aplicación que permite entrenar una red neuronal, utilizando técnicas de visión artificial. El proyecto consta de emplear un contenedor EMGUCV para poder emplear la librería OpenCV en entornos .Net, se detectaron rostros mediante el algoritmo de Viola-Jones que hace parte de la librería OpenCV que emplea clasificadores cascada ya entrenados, el sistema se entrenó mediante el modelo Eigenfaces también de la librería OpenCV, mediante la cual se comparan imágenes ya almacenadas con la imagen de entrada y buscar un alto grado de similitud. Se concluye que emplear un proceso de aprendizaje de máquina tiene una buena relación costo, beneficio, pues el costo del cómputo es bajo y puede ser aplicado aun en celulares. Se evidencio que el reconocimiento facial usando el modelo Eigenfaces es altamente afectado por las condiciones variables del entorno.(Caballero, 2017)

En la Universidad Distrital Francisco José de Caldas el Ingeniero en control Donovan Camilo Platero Plazas en el año 2015, mediante la plataforma de desarrollo embebido Raspberry Pi 2B

realizó un sistema que emplea imágenes faciales para su uso en controles de acceso y sistemas de seguridad. Para la detección de rostros se implementó el algoritmo propuesto por Viola Jones el cual es uno de los más reconocidos en la actualidad para realizar extracción de rostros el cual unifica clasificadores débiles para formar un sistema de detección robusto, este algoritmo consta de rapidez y efectividad, Para el reconocimiento de personas se empleó la función Eigenfaces de la librería OpenCV, la cual funciona mediante el principio teórico de PCA (Análisis de componentes Principales). Para la adquisición de la imagen se implementó la cámara Pi NoIR que tiene una resolución de 5 Megapíxeles, la tarjeta de desarrollo Raspberry Pi 2B. Mediante el hardware y software implementado la reducción de costos en contraste con los sistemas tradicionales de seguridad fue amplia, el sistema tuvo un tiempo de detección de 0,18 segundos, el sistema tuvo un 94% en imágenes con rostros detectados, y acierto de 100% en imágenes sin rostros, al ser un sistema sensible a condiciones externas se recomienda entrenar el sistema en el mismo lugar en donde se ubicará el sistema de entrenamiento.(Plazas, 2015)

En la universidad distrital Francisco Jose de Caldas el ingeniero Edinson Cáceres Parra en el año 2018 empleando el SDK de JAVA y servicios de Amazon desarrolló un prototipo para el control el control ingreso de personas mediante identificación por reconocimiento facial, bajo en costos y evitando emplear sistemas tradicionales que son altamente susceptibles a suplantación. Debido al alto coste de procesamiento se determina emplear un computador, el proyecto consiste en emplear la librería OpenCV para realizar una captura de fotografía automática, posteriormente esta imagen se envía a un API de Amazon (compareFace) en la cual se comparan dos rostros y se obtiene un porcentaje de similitud, para ello también se hace necesario emplear Amazon S3 que provee el almacenamiento necesario para el proyecto. El tiempo de respuesta de la aplicación es de 0,5 segundos, durante las pruebas se obtuvo un 99% de coincidencia detectando de forma correcta si la persona que ingresaba a la empresa era la que se encontraba registrada en la base de datos. (Parra, 2018)

Luis Alejandro Parada Mojica para ese entonces estudiante de la Universidad distrital Francisco José de Caldas, emplea la librería OpenCV en el contenedor EMUG para emplear reconocimiento facial en una aplicación para el control y gestión de almuerzos en casinos empresariales, con la finalidad de evitar sobrecostos y ayudar al medio ambiente eliminado uso de tickets. De este proyecto se concluyó que el método de reconocimiento facial empleado se ve altamente afectado en exactitud por las condiciones de iluminación, calidad de cámara, y por lo tanto se recomienda que al emplear la aplicación las condiciones del entorno y la calidad de captura de video deben ser muy similares a las cuales con las que se entrenó el sistema.(Parada, 2019)

## 5.2. Marco teórico

### 5.2.1. Visión artificial.

La visión artificial también llamada visión por computador es una ciencia que emula como los seres vivos tienen la capacidad de adquirir imágenes, procesar y de acuerdo con la información ser capaz de actuar o tomar decisiones.

Es una disciplina que como lo demuestran investigaciones ha experimentado una constate evolución en los últimos años, algunas de las áreas sobre las cuales se implementa la visión artificial en la actualidad son; biología, medicina, cartografía, meteorología, agricultura, entre

otros. Implementar sistemas de visión artificial no es una tarea sencilla, debido a que para que sea implementado se necesita cumplir con ciertas características de precisión y velocidad de cómputo mediante las cuales las solución al problema sería eficiente además de viable económicamente.(García & Caranqui, 2015)

La visión artificial es un área compleja que para su desarrollo requiere de ciencias como ingeniería eléctrica, ingeniería electrónica, computación, además de ciencias básicas como matemáticas y física. (Pajares, 2008)

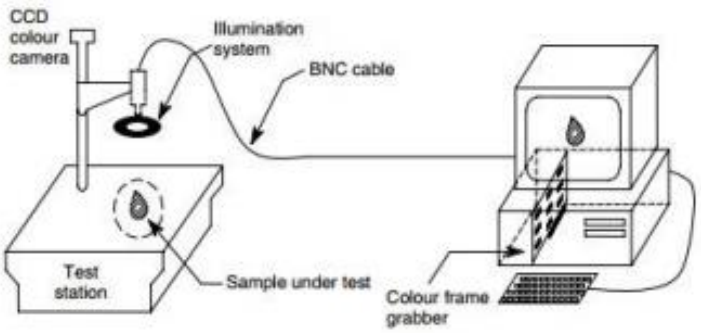


*Figura 1. Sistema basado en visión artificial*(Gonzáles & Zamara, 2014)

En la figura 1, se muestran un sistema con la función de realizar procesos de visión artificial, en este sistema se destacan dos componentes, un dispositivo de procesamiento (computador) y un dispositivo de adquisición de imagen (cámara).

La visión artificial tiene 6 procesos (García & Caranqui, 2015):

- Captura: Mediante un dispositivo de adquisición de imagen se realiza una captura de imagen.
- Pre-Procesamiento: Proceso mediante el cual se hace un tratamiento con la finalidad de eliminar ruido y aplicar filtros a la imagen que permitan resaltar ciertas características.
- Segmentación: Proceso mediante el cual se segmenta la imagen en regiones, las regiones pueden tener desde formas cuadradas, rectangulares o regiones más complejas.
- Descripción: Proceso que se encarga de obtener características de interés mediante las cuales se puede identificar un objeto
- Reconocimiento: Proceso que clasifica los objetos mediante los descriptores clasificando a todos los objetos que tengas características similares.
- Interpretación: Proceso que se asemeja al proceso humano de analizar resultados y tomar decisiones

### 5.2.2. Cámaras.

La cámara primitiva fue un invento que plasmaba imágenes sobre una superficie, pero su desarrollo al igual que el de cualquier otro invento no fue de la noche a la mañana, Para el desarrollo de este dispositivo influyeron decenas de inventores que con el paso del tiempo han conseguido crear dispositivos cada vez más compactos mejorando notablemente su calidad.(Rubal, 2018)

La primera cámara se desarrolló durante el siglo XIX, aunque anterior a eso ya habían inventos con funciones parecidas. En 1888 la empresa kodak que hasta hace unos años seguía en el mercado de las cámaras, desarrolló una cámara portable y automática de revelado. En la década de 1930 se crearon las cámaras instantáneas, mediante las cuales se podía obtener la foto toma en pocos segundos. El último gran salto de cámaras se vive en la época actual en la cual cualquier persona puede acceder a una cámara en cualquier momento pues los teléfonos móviles la traen integrada (Rubal, 2018)

El proceso mediante el cual las cámaras digitales adquieren una imagen consiste en dar una señal mecánica a través de un botón, este permite que la luz atraviese el lente y sea adquirida por un sensor. El sensor mide la intensidad de luz en pequeñas divisiones que son los llamados pixeles.(Serrano, 2016) Cada pixel convierte la cantidad de luz recibida en electrones, mediante un Conversor Análogo Digital (ADC), la señal adquirida por cada pixel se convierte en valor digital, y por ultimo toda esta información es enviada, procesada y proyectada ya sea en una superficie o una pantalla.

### 5.2.3. Procesamiento digital de imágenes.

El procesamiento digital de imágenes también por sus siglas PDI, es un área de investigación que se deriva del procesamiento digital de señales. Los dos objetivos del procesamiento digital de imágenes son mejorar la calidad de la imagen de modo que sea más entendible para el ojo humano y el segundo objetivo es dar la capacidad a una máquina para que pueda procesar la información que se presenta en una imagen.

La historia del procesamiento digital de imágenes inicia en 1920 cuando se implementaros sistemas de transmisión de imágenes para reducir el tiempo de enviar imágenes por el océano desde Londres a Nueva York. El procesamiento digital de señales computarizadas inicio en 1964 cuando el Ranger 7 transmitió imágenes de la luna y mediante procesamiento digital de imágenes buscaban corregir distorsiones de los sistemas de captura(Torres, 1996).

Entre los usos más frecuentes del Procesamiento Digital de imágenes se encuentran el uso en sistemas de cartografía automatizada mediante la cual se reciben información de los satélites que capturan la luz del espectro visible como también del no visible, mediante el espectro electromagnético los científicos tienen la capacidad de analizar el espectro infrarrojo para determinar el tipo y cantidad de plantas y árboles en determinada área. Otra área de aplicación es la medicina en donde con el paso de los años los médicos manipulan imágenes digitales de tomografías, resonancia magnética y TAC, las cuales emplean procesamiento digital de imágenes.

En las ciencias también se emplea procesamiento digital de imágenes, una aplicación es simulación del comportamiento de nubes y tormentas mediante simulaciones realistas(Torres, 1996).

Otras áreas de aplicación del procesamiento digital de imágenes son:

- Lectura de documentos.
- Astronomía.
- Percepción Remota.
- Video Vigilancia,
- Mapas climatológicos.
- Navegación autónoma

### 5.2.4. Reconocimiento Facial.

La finalidad de un sistema de reconocimiento facial es que mediante el uso de visión e inteligencia artificial, se pueda identificar de forma automática a una persona en un video o en tiempo real, las técnicas de reconocimiento facial en los últimos años se están empleado cada vez más en sistemas de video vigilancia, validación de identidad, acceso a áreas.

Existen 2 tipos de técnicas de reconocimiento facial (Ver figura 2): técnicas locales o geométricas y técnicas basadas en modelos.(Yang , 2002). Las técnicas locales o geométricas procesan la imagen facial y mediante un conjunto de operaciones buscan extraer características geométricas del rostro, ejemplo, la distancia entre las cejas. La técnica basada en modelos entrena un modelo mediante el cual se obtiene características faciales, pero no necesariamente geométricas.

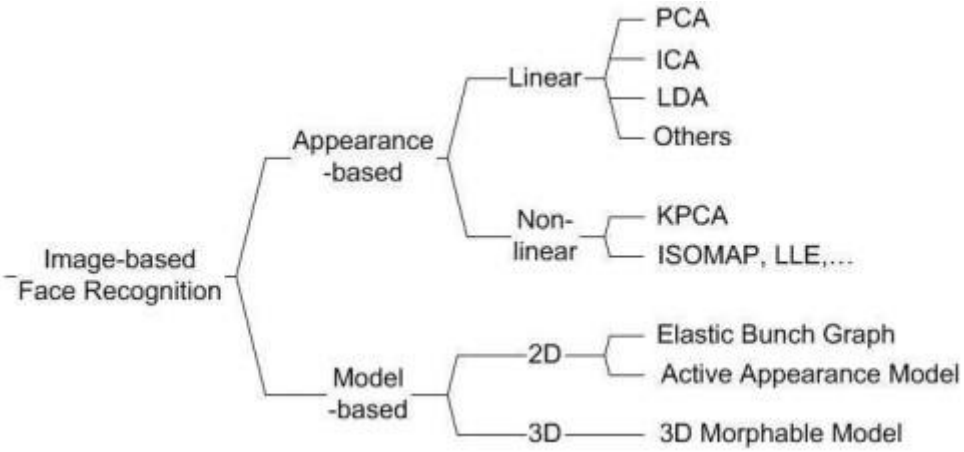


*Figura 2. Técnicas de Reconocimiento Facial* (Lu et al., 2003)

El reconocimiento facial tiene tres etapas (Castaño & Sierra, 2019).

- Detección de rostro: Es la etapa en la cual se detecta el rostro en una imagen, eliminado las áreas que no tiene el rostro.

- Extracción de características: Se extraen características geométricas del rostro, o parámetros únicos de determinado rostro.
- Reconocimiento: Clasificación del objeto acorde a las características propias del rostro.

#### 5.2.4.1. EigenFaces

Es una técnica que permite determinar qué vectores ofrecen más información a un conjunto de datos de dimensión N, mediante la ortogonalidad dimensional, pero toda esa información puede hacer que el sistema de clasificación sea de alto coso, para ello EigenFaces emplea una herramienta matemática llamada Análisis de Componentes Principales (PCA), la cual toma una cantidad menor de los vectores entregados por las imágenes de la base de datos pero con información necesaria para la reconstrucción de los rostros de las imágenes ingresadas.
Con esta técnica, el reconocimiento se realiza mediante la proyección de una imagen de un rostro en el subespacio formado por los EigenFaces y comparando su posición con la de los otros rostros conocidos. Los EigenFaces son un conjunto de vectores representados gráficamente, convirtiéndose en una especie de mapa de las variaciones entre imágenes. Estos vectores son el resultado de la aplicación de PCA a la matriz de covarianza de un conjunto de imágenes de rostros los cuales son denominados Eigenvectores, (Esparza et al., 2015)

Este método realiza las siguientes operaciones y pasos:

- Adquisición de una serie de imágenes de caras iniciales.
- Cálculo de los EigenFaces del conjunto de entrenamiento, almacenando únicamente las M-dimensiones que correspondan con los eigenvalores mayores.
- Cálculo de un conjunto de pesos basados en la imagen de entrada y las M-eigenfaces mediante la proyección de la imagen
- Determinar si la imagen de la cara pertenece o no al conjunto de entrenamiento, por medio de la distancia euclídea.

#### 5.2.4.2. FisherFaces.

Fisherface es una técnica de reconocimiento de rostros, que tiene en cuenta la luz y expresiones faciales, deriva de EigenFaces, ya que se basa en el mismo principio de funcionamiento, con la diferencia de que en este caso se pretende desarrollar una técnica de reconocimiento facial mejorada que sea insensible a la variación de iluminación y a las distintas expresiones faciales en las imágenes.

Fisherface se encarga de clasificar y reducir las dimensiones de las caras usando el método FLD (Discriminant Lineal Fisher), intentando proyectar los datos de manera que su nueva dispersión sea óptima para la clasificación. Mientras PCA busca los vectores que mejor describen los datos, LDA (Discriminant Lineal Analysis) busca los vectores que proporcionan mejor discriminación entre clases después de la proyección. Fisherfaces realiza un LDA, donde se busca aprovechar la información disponible, sobre la clasificación de las imágenes de entrenamiento, para buscar una proyección que maximice la separación entre imágenes de diferentes personas (o clases) y minimice la distancia entre imágenes de una misma clase.(Costa Mari, 2020)

#### 5.2.4.3. Histograma de patrones binarios locales (LBPH).

El método de patrones binarios locales es un descriptor de textura, lo que quiere decir que describe características elementales como la forma, el color, la textura o el movimiento, entre otras, contenidas en una imagen o un video. El uso de descripciones locales en algunas regiones del rostro aporta más información que otras, ya que estos descriptores de texturas tienden a promediar la información que describen, lo cual no es conveniente al describir rostros ya que es importante mantener la información de las relaciones espaciales. Para obtener la descripción global, la imagen del rostro es dividida en diferentes regiones, a las que se les aplica un histograma con el que se obtiene el operador LBPH que describe información independiente por región, esas descripciones locales obtenidas son concatenadas para construir una descripción global del rostro. El método de LBPH asigna etiquetas a cada uno de los pixeles de la imagen tomando en cuenta la distribución de los vecinos. (Juárez Álvarez, 2018)

El método LBPH usa 4 patrones, el radio, vecinos, cuadricula X y cuadricula Y, los cuales se definen de la siguiente manera: (Esparza et al., 2015)

- Radio: Se usa para construir el patrón binario local circular, representa el radio que se encuentra alrededor del píxel central
- Vecinos: Es el número de puntos de muestras para la construcción del patrón binario local
- Cuadricula X: Es el número de celdas en dirección horizontal, entre más células más fina es la cuadricula y de esta manera mayor será la dimensionalidad del vector de características resultante.
- Cuadricula Y: Es el número de celdas en dirección vertical, entre más células más fina es la cuadricula y de esta manera mayor será la dimensionalidad del vector de características resultante.

### 5.2.5. Python.

Python es un lenguaje de programación que nace a finales de la década de los 80, cuando Guido van Rossum dio continuidad al desarrollo de un lenguaje de programación, en ese tiempo denominado ABC. Su creador lo diseño con la finalidad de que se usara y aprendiera de manera sencilla. Su nombre proviene de un grupo de cómicos llamado Monthy Python.

Las principales ventajas de usar Python para desarrollar proyectos son: que es open source, por lo cual no hay que pagar para hacer uso de él, es un lenguaje multiparadigma que lo hace flexible y fácil de aprender, tiene un amplio entorno de aplicación, pues sus desarrollos pueden ir desde creación y mantenimientos de páginas web ligadas a bases de datos como inteligencia artificial en modelos predictivos, puede ser usado desde diferentes plataformas como Windows, Linux o MAC, además tiene una comunidad gigante que brindan soporte para realizar proyectos.

Es un lenguaje de programación interpretado, su popularidad ha aumentado en los últimos años debido a que muchas de las herramientas que los ingenieros de datos usan se han desarrollado sobre este lenguaje de programación, entre sus principales usos se encuentran(Visus, 2020):

- Visión artificial
- Machine Learning.
- Deep Learning.
- Big data

### 5.2.6. Librería OpenCV.

Su significado es Open Computer Vision, lanzada formalmente en enero de 1999 se considera la librería más relevante cuando de visión artificial se trata, es una biblioteca libre, lo que quiere decir que es gratuitita para proyectos orientados a la parte académica y comercial(Marin, 2020).

Para su construcción se empleó el lenguaje de programación C++, aunque tiene librerías para realizar proyectos sobre Java, C, C++, Python. Además, se emplea para realizar proyectos sobre diferentes sistemas operativos como: Android, Linux, MAC, Windows. Debido a su practicidad y alcance a la hora de realizar proyectos relacionados con análisis de imágenes ha sido usada por grandes empresas como: Sony, Toyota, IBM, Intel, Google, Microsoft, entre otros (Gracia, 2013).

La librería es usada frecuente en proyectos que requieran funciones como(Gracia, 2013):

- Reconocimiento Facial.
- Detección facial.
- Trackig.
- Sistemas de detección de intrusos.
- Detector de emociones.
- Identificación de objetos.
- Eliminación ojos
- Sistema de seguridad.

### 5.2.7. Machine Learning.

El machine learning es una herramienta que deriva de la inteligencia artificial, ciencia que se fundamenta en conseguir que las computadoras puedan operar y actuar igual que nuestra red neuronal. Su historia comienza en 1950 Alan Mathison reconocido científico desarrolló “Test de Turing” prueba mediante la cual se buscaba determinar cuan inteligente era una computadora, test que consistía en desarrollar una conversación en la cual la computadora debería responder de manera racional como los seres humanos. Posteriormente, en 1952 Arthur Samuel creo un programa de cómputo que podía aprender, su desarrollo era un software que jugaba damas chinas, software que era capaz de aprender nuevas estrategias.(Hinestroza Ramirez, 2018) El avance del machine learning ha sido tan grande que en la actualidad el machine learning se destaca en áreas como:

- Robótica: Se emplea en robots de fábricas que realizan tareas que anteriormente eran desapañadas por humanos.
- Finanzas: Inicialmente se empleó en aplicaciones intuitivas que permitían a los usuarios realizar trámites de forma sencilla y cómoda. En la actualidad de emplea en los grandes mercados como Wall Street debido a que mediante estos modelos se realizan algunas de sus operaciones
- Construcción: Se emplea de la mano junto a la robótica en la automatización de fábricas para aumentar la producción y relevar de labores peligrosas a los humanos.
- Educación: Se emplea para desarrollar modelos de trabajos eficientes que permitan suplir y reforzar las falencias y vacíos educativos del estudiante

El machine Learning funciona bajo tres modelos de aprendizaje en la actualidad(Hinestroza Ramirez, 2018):

- Aprendizaje Supervisado: Se basa en detectar patrones de conjuntos de datos que anteriormente fueron etiquetados.
- Aprendizaje no Supervisado. Se basa en detectar patrones de conjuntos de datos lo que no han sido etiquetados, el sistema busca características que agrupen los conjuntos de datos.
- Aprendizaje de refuerzo: Se basa en detectar patrones de conjuntos de datos que no han sido etiquetados, pero en etapas posteriores el sistema recibirá retro alimentaciones.

### 5.3. Marco Legal

Los Sistemas de Videovigilancia o cámaras de seguridad implementadas con la finalidad de garantizar la seguridad de bienes o personas en un lugar determinado han venido incrementando su presencia al ser considerados como un medio idóneo para realizar el monitoreo y la observación de actividades en escenarios domésticos, empresariales, laborales y públicos.
Estas tareas de monitoreo y observación realizadas a través de los sistemas de videovigilancia implican la recopilación de imágenes de personas, es decir, de datos personales de acuerdo con la definición contenida en el literal c) del artículo 3 de la Ley 1581 de 2012, “Por la cual se dictan disposiciones generales para la protección de datos personales”, entendido como “cualquier información vinculada o que pueda asociarse a una o varias personas naturales determinadas o determinables”.
Ya que los Sistemas de Videovigilancia pueden implementarse en lugares como establecimientos de comercio, copropiedades, edificios, centros comerciales y parqueaderos, entre otros, en todos los casos se debe informar a los Titulares de datos personales que se encuentran en una zona de videovigilancia y obtener su autorización para el Tratamiento de estos. Para ello, se pueden utilizar señales o avisos distintivos en las zonas de videovigilancia, principalmente en las zonas de ingreso a los lugares que están siendo vigilados y monitoreados y al interior de estos. (I&C, 2016)

# 6. Metodología

En esta sección se describe el procedimiento que se realizó para la elaboración del sistema de videovigilancia residencial, dando una descripción del hardware que se implementó y del software desarrollado para dar cumplimiento a los requisitos del proyecto.

## 6.1. Identificación de métodos de detección, extracción y reconocimiento de Rostro

### 6.1.1. Métodos de detección de rostros.

Para el desarrollo del sistema se contó con la implementación de las librerías Face-Recognition y OpenCV, las cuales cuentan con tres algoritmos de detección facial. A continuación, se explicarán cada uno de los algoritmos

#### 6.1.1.1. Viola Jones

El algoritmo de Viola Jones se basa en la creación de una imagen que contiene la información de luminancia acumulada, desde el pixel inicial (0,0) hasta el pixel de la posición (x,y) como se muestra en la figura 3, esta imagen de intensidades recibe el nombre de imagen integral.(Barroso, 2014)

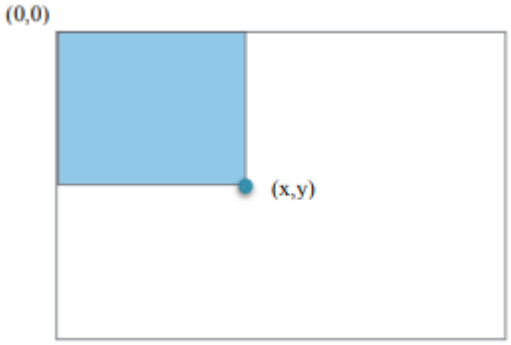


*Figura 3 Imagen Integral desde el pixel (0,0) hasta pixel (x,y)*(Barroso, 2014)

La ecuación 1 calcula la luminancia para el área azul de la figura 3, Siendo *ii(x, y)* la imagen integral, *i(x, y)* la luminancia de la imagen original.

$$ii(x,y) = \sum_{x' \leq x \, , y' \leq y} i(x', y') \tag{1}$$

La ecuación 1 se calcula a partir de las siguientes ecuaciones:

$$s(x,y) = s\,(x, y-1) + i(x,y) \tag{2}$$

$$ii(x,y) = ii\,(x-1, y) + s(x,y) \tag{3}$$

Donde *s (x, y)* es la acumulación de la fila, teniendo en cuenta que *s (x, -1) =0* y que, *ii (-1, y) =0*.

Gracias a la imagen integral se puede obtener las luminancias acumuladas en las regiones a analizar como se muestra en la figura 4.

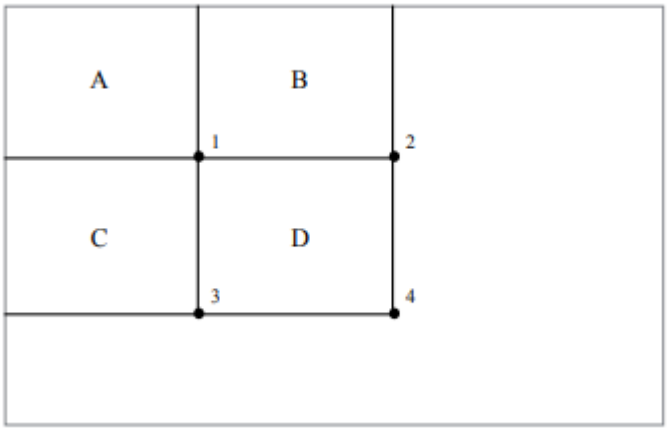


*Figura 4. Subregiones en una imagen integral*(Barroso, 2014)

La figura 4 muestra la imagen integral divida en subregiones, a través de cálculos algebraicos se puede obtener el valor de iluminancia de determinada región. Es decir, para obtener la iluminación de la subregión D, basta con calcular D+A-C-B, siendo la iluminación en la región C la suma de las subregiones A+C, de la región B la suma de A+B y de la región D la suma de la iluminación en las regiones A+B+C+D.

Mediante este procedimiento se consiguen evaluar las características requeridas por el clasificador. El algoritmo trabaja sobre características sencillas que se consiguen extraer fácilmente empleando diferentes escalas, como se muestra en la figura 5.

*Figura 5. Ejemplo de característica para evaluar la existencia del conjunto de ojos y nariz. a) Evaluación sobre una ventana que no es una cara; b) Evaluación sobre una ventana con una cara*(Barroso, 2014)

En la figura 5 se observa cómo trabaja el clasificador en diferentes imágenes, en la sección (a) se aplicó sobre un tren y en la sección (b) se aplicó en un rostro humano, para la evaluar en cada una de ellas la existencia de rasgos como nariz y ojos.

Para evaluar la existencia de los rasgos, se selecciona una región de pixeles teniendo en cuenta su respectiva luminancia, posteriormente, se calcula la diferencia entre las zonas negras y las zonas blancas mediante la ecuación 4 y de esta manera obtener un valor para compararse contra un umbral determinado en el clasificador(Barroso, 2014).

$$Valor\ de\ la\ caracteristica = \sum(Rectangulo\ blanco) - \sum(Rectangulo\ sombreado) \qquad (4)$$

Para la evaluación de características se emplean diferentes rectángulos, dependiendo la complejidad de la característica, complejidad que aumenta con el paso de las etapas.

*Figura 6. Características usadas por el clasificador*(Barroso, 2014).

En la figura 6 se pueden observar diferentes características que se pueden usar para detectar varios rasgos en una imagen. Para características básicas se emplean 2 rectángulos, a medida que a aparecen rasgos más complejos se aumenta la cantidad de rectángulos.

En la detección facial, una de las primeras etapas del clasificador, consiste en determinar si la imagen cuenta con determinadas características, como se puede observar en la figura 7, el clasificador determinó que esta imagen cuenta con las características de contener 2 ojos y 1 nariz.

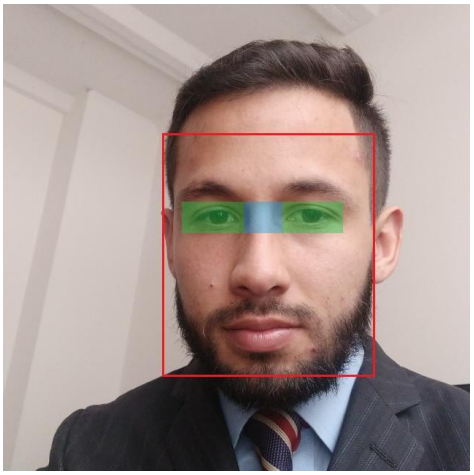

*Figura 7. Representación de características sobre una imagen. Fuente: Autor*

En la figura 7 y acorde a la ecuación (4) se suman la luminancia de las áreas sombreadas en verde y se resta con la sumatoria de las áreas sombreadas en azul, este valor se compara contra un umbral de clasificador, y de esta manera se sabe si la imagen contiene los rasgos solicitados.

Posteriormente se realiza la construcción de un clasificador mediante la selección de características empleando Boosting, ya que en cada subimagen el número total de posibles características es demasiado grande. Para optimizar la clasificación, el proceso de aprendizaje debe centrarse solo en características críticas.

*Figura 8. Características entrenadas con AdaBoost* (Barroso, 2014)

Hasta este punto el sistema cuenta con una alta tasa de detección, como también una alta tasa de falsos positivos, para mejorar el sistema se emplean clasificadores en cascada en el cual los clasificadores más complejos se emplearán a zonas de la imagen en donde los clasificadores más sencillos han detectado posibles características de interés. (Barroso, 2014)

#### 6.1.1.2. Histograma de gradientes orientados (HOG)

HOG es un algoritmo de detección facial de la librería dlib basado en Histogram of oriented Gradientes que consiste en dividir el rostro en varias celdas, realizar un histograma de cada celda y de esta manera obtener un único vector de características HOG.

Para la detección de rostros mediante HOG, se debe definir un conjunto de características que permitan determinar si dentro de una imagen se encuentra un rostro, este proceso se realiza mediante descriptores los cuales van a tener buen comportamiento aun en lugares de iluminación variable.(Alcántara Montiel et al., 2019)

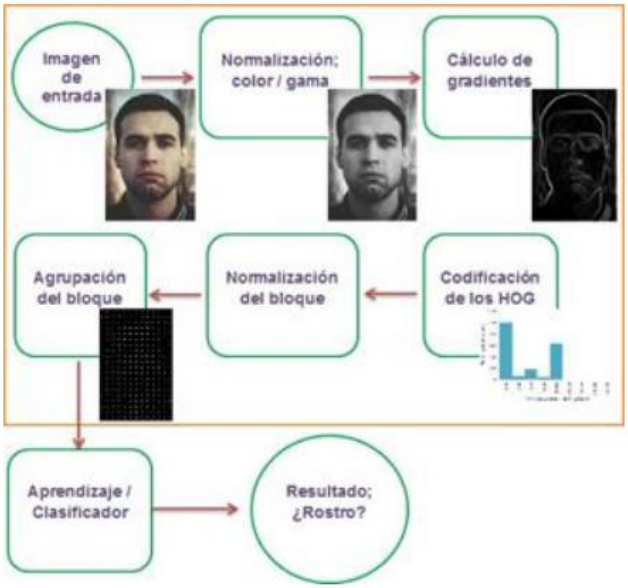


*Figura 9. Diagrama de funcionamiento HOG*(Alcántara Montiel, 2019)

En la figura 9 se presenta el diagrama de funcionamiento de la técnica de detección facial HOG. Para emplear esta técnica primero se debe obtener una imagen, a esta se le aplican métodos de normalización de gama y color, de la imagen resultante se hace un cálculo de gradientes para obtener las zonas que presentan contornos, luego se codifican mediante histogramas que presentan la información en subsecciones de la imagen normalizada, por último, se normaliza nuevamente y agrupan las codificaciones para obtener un clasificador que evalúa las características faciales, dando como resultado la validación de si en la imagen original se encontró algún rostro.

Para el proceso de detección facial mediante descriptores HOG se realiza una normalización de color o gamma aplicando un filtro de color para normalizar la imagen con el fin que se reduzca la influencia de condiciones externas como luminosidad, posteriormente se realiza el cálculo de gradientes, para ello se filtra la imagen aplicando una máscara y derivada discreta vertical y horizontal.

Luego se realiza una codificación de los histogramas locales de orientación en donde la representación gráfica se divide en pequeñas regiones espaciales rectangulares denominadas celdas, posteriormente para cada celda se genera un histograma basado en la orientación de los pixeles, cada píxel aporta información para un canal del histograma de la celda. Los histogramas son la representación del descriptor HOG, en donde se recomienda trabajar con histogramas de 9 canales, luego se realiza una normalización del bloque en donde se normalizan los gradientes, para ello las celdas se agrupan en grandes bloques, buscando unirlos de forma que se traslapan para buscar un mejor rendimiento en el descriptor final buscando reducir las influencias de iluminación y contraste.

Por último, se realiza la agrupación de lo descriptores, la cual consiste en la concatenación de los descriptores de los bloques, y de esta forma obtener el descriptor final que será empleado en un clasificador. (Alcántara Montiel , 2019)

#### 6.1.1.3. CNN

Las redes neuronales convolucionales tienen como objetivo extraer rasgos característicos de una imagen mediante el uso de filtros**.**

$$\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$$

*Figura 10. Filtro o kernel para extracción de características.* (Cowley, 2018)

La figura 10, muestra la matriz que representa el filtro o kernel para la extracción de características, observe que esta matriz en su primera columna está compuesta por -1, en su segunda columna por 2, en la tercera columna por -1, al aplicar este filtro sobre una imagen de entrada (Ver figura 11.a) Se obtiene una nueva imagen, como la que se presenta en la figura 11 b), la cual presenta los contornos más importantes.

*(a) Imagen de entrada* *(b) Imagen filtrada*

*Figura 11. Cambio de una imagen al aplicar un kernel.a) Imagen de entrada sin aplicar filtros.b)Imagen resultante al aplicar un filtro o kernel* (Cowley, 2018)

Para aplicar el filtro se debe realizar la sumatoria de los valores del producto escalar entre el filtro y la imagen de origen, Un ejemplo de este procedimiento se puede ver en la figura 12, donde se escoge una sección de la imagen total y se aplica el filtro. El procedimiento matemático consiste en realizar la multiplicación escalar entre la subimagen (Recuadro azul) con el filtro (Números de color azul) para posteriormente sumar todos los valores resultantes, esto da como resultado una matriz de 2x2, en la cual se presentará la información más relevante de la imagen.

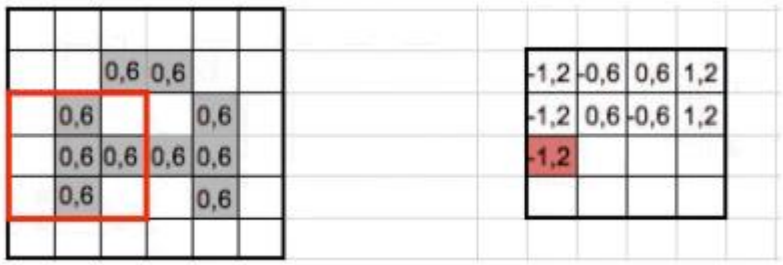


*Figura 12. Usos de filtro en una matriz 4 x4* (Bagnato, 2020)

Debido al alto coste computacional en la primera capa oculta que es el resultado de la primera convolución, se debe realiza un filtrado escogiendo las características más importantes que se detectaron en cada filtro, un método para realizar esta selección es el muestreo mediante Max-Pooling, buscando reducir la salida.

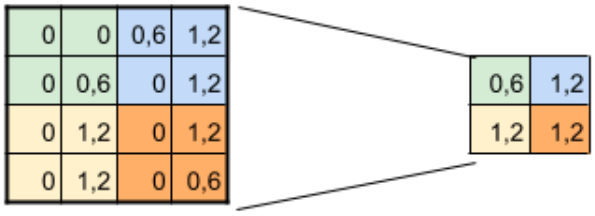


*Figura 13. Muestreo mediante Max Pooling* (Bagnato, 2020)

En la figura 13 se muestra como a una imagen se le aplica el filtro *Max-Pooling*, el procedimiento consiste en dividir la imagen en diferentes secciones del mismo tamaño, de cada sección se selecciona el número mayor y se crea una nueva matriz que contendrá los valores con la información más relevante, consiguiendo también reducir el tamaño de la imagen de salida.

Una vez realizada la primera convolución el sistema es capaz de detectar características simples como líneas o curvas, pero a medida que se realizan más convoluciones el sistema será capaz de reconocer patrones más complejos en la imagen.

Este algoritmo tiene buen desempeño en caras que no se encuentren de forma frontal al dispositivo de adquisición de imagen, además de detectar caras con gestos extraños, sin embargo, el coste computacional es alto además de omitir la detección de rostros pequeños.(Bagnato, 2020)

### 6.1.2. Métodos de extracción de rostros.

La técnica de extracción de rostros es una técnica que se emplea después de la técnica de detección de rostros, el algoritmo para la extracción de rostros recibe 4 coordenadas que delimitan el rostro, estas coordenadas provienen del algoritmo de detección de rostro. Las coordenadas recibidas son correspondientes Up, Down, Rigth y Left que son los puntos extremos de la imagen. (ver figura 14).

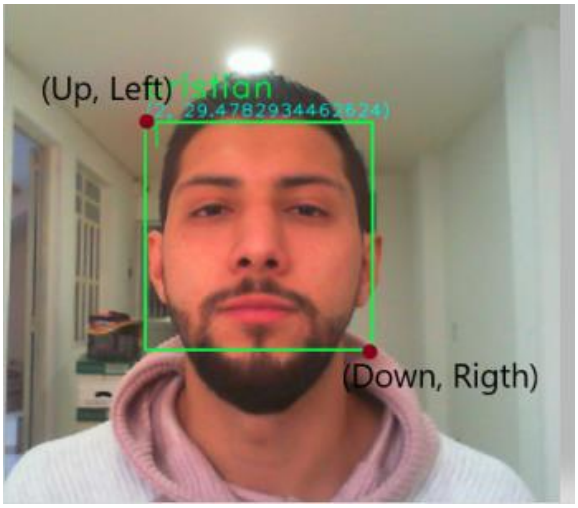


*Figura 14. Detección y extracción de las coordenadas del rostro, Fuente: Autor*

Para realizar la extracción del rostro se emplea la línea de código que se presenta a continuación, en donde la palabra sombreada de verde hace referencia al frame de video del cual se desea hacer la extracción, y la sección sombreada en rojo se refiere al rango de coordenadas de la región de interés

```
rostro = auxFrame[y:y+h,x:x+w]
```

*Figura 15. Línea de código de extracción de rostro. Fuente. Autor*

Como resultado, se obtiene la sección de interés obtenida del video, en este caso es el rostro que se encuentra delimitado por las coordenadas, en la figura 16 se puede observar la sección de imagen que se obtuvo al realizar la extracción.

*Figura 16. Extracción de rostro. Fuente: Autor*

### 6.1.3. Métodos de reconocimiento de rostros.

De acuerdo con los requerimientos del proyecto, la búsqueda de algoritmos de reconocimiento facial se cerró a librerías que se puedan implementar en software libre. Principalmente se manejan las librerías OpenCV y FaceRecognition, debido a su bajo índice de error y velocidad de procesamiento.

#### 6.1.3.1. OpenCV

La librería OpenCV cuenta con tres algoritmos de reconocimiento facial de los cuales dos son de métodos holísticos y uno local o geométrico., estos algoritmos son EigenFaces, FisherFace e Histograma de patrones binarios locales (LBPH).

##### 6.1.3.1.1. EigenFaces

La técnica consiste en asumir *p-variables* en *n* elementos dispuestos en una matriz de dimensiones n x p, se desea encontrar un subespacio de menor dimensión que *p*, buscando perder la mínima información posible.

La ecuación (11) ejemplifica en dos dimensiones como aplicar el modelo.

$$min \sum_{i=1}^{n} {r_i}^2 = \sum_{i=1}^{n} |x_i - z_i a_1|^2 \qquad (5)$$

En donde *xi* es un punto en el espacio, $z_i a_1$ es el vector que representa la proyección en dicho punto y $r_i$ es la distancia entre el punto $x_i$ y la dirección $a_1$(Ver figura 17).

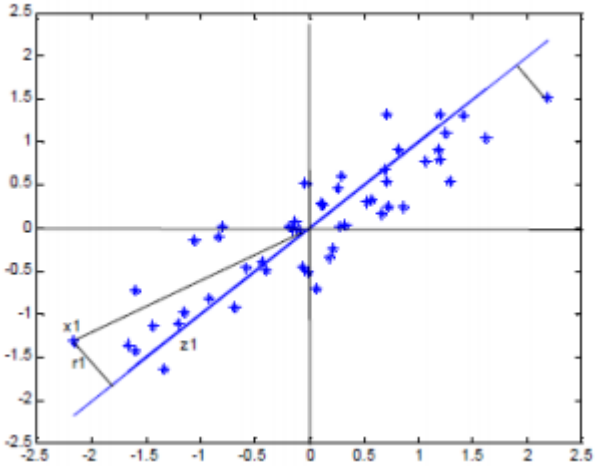


*Figura 17. Recta que minimiza distancias de los puntos*(Costa Mari, 2020).

Empleando el teorema de Pitágoras en la figura 17 la ecuación que caracteriza la distancia de todos los puntos es:

$$\sum_{i=1}^{n} x'_i x_i = \sum_{i=1}^{n} r_i^2 + \sum_{i=1}^{n} z_i^2 \qquad (6)$$

El primer componente principal de la ecuación 12 es la combinación lineal de las variables originales que presente mayor varianza.

El vector $z_1 = Xa_1$ da los valores del primer componente y su varianza está dada por la siguiente ecuación:

$$Var(z1) = a'_1 S a_1 \qquad (7)$$

Empleando Lagrange como método de optimización de funciones con restricciones, asumiendo que $a'_1\ a_1 = 1$ y sabiendo que S es la matriz de varianza y covarianzas, se obtiene:

$$Sa_1 = \lambda a_1 \qquad (8)$$

Se obtiene que $a_1$ es un vector propio de S y λ es su valor propio.

Para obtener el segundo componente se obtiene:

$$Sa_1 = \lambda_1 a_1 \qquad (9)$$

$$Sa_2 = \lambda_2 a_1 \qquad (10)$$

En dónde. $a_1$ y $a_2$ son vectores propios de la matriz S como también $\lambda_1$ y $\lambda_2$ son los dos mayores valores propios. Estos vectores definen el nuevo espacio.

El espacio *r*<*p* que mejor representa un conjunto de puntos viene definido por los vectores propios asociados a los *r* mayores valores propios de *S*, apareciendo nuevas variables que reciben el nombre de componentes principales.

Una imagen se puede considerar como un vector de dimensión $N^2$ si una imagen es de 250 x 250 se convierte en un vector de 62500 dimensiones. Si se habla de un conjunto de imágenes se representaría como un conjunto de puntos o vectores no distribuidos aleatoriamente debido a las similitud y características compartidas de los rostros humanos y mediante el estudio de PCA se obtendrá un subespacio de menor dimensionalidad.

Si se tiene un conjunto de imágenes $I_1 \ldots, I_M$, de estas se puede obtener una representación facial definida como $\varphi = \frac{1}{M}\sum_{I=1}^{M} I_I$.

El vector $\vartheta_i = I_i - \varphi$ caracteriza la distancia entre cada representación facial y la media para hallar la matriz de covarianza.

$$C = \frac{1}{M}\sum_{i=1}^{M} \vartheta_i {\vartheta_i}^T \tag{11}$$

De la ecuación anterior que representa la matriz de covarianza se puede obtener los vectores y los valores propios que definen el nuevo subespacio.

$$ca_i = \lambda_i a_i \qquad i = 1,2, \dots, M \tag{12}$$

De acuerdo con la matriz de covarianza se hallan los Eigenfaces. Para la identificación del rostro se proyectan las imágenes en el subespacio y la imagen que tenga menor distancia euclídea con la imagen de prueba es la que la identifica.

$$\boldsymbol{DIFS} = \|\Omega - \Omega_C\| \tag{13}$$

Siendo $\Omega$ la imagen proyectada y $\Omega_C$ la representación media de cada clase o población.(Costa Mari, 2020)

#### 6.1.3.1.2. FisherFace

El método de reconocimiento facial mediante Fisherface es una extensión del método EigenFace, por lo tanto, se debe aplicar la matemática expuesta en la sección anterior.

La ecuación 20 parte de un conjunto de imágenes N que esté compuesto por las imágenes $x_1$,.., $x_N$ en la cual cada imagen pertenece a una de las clases $X_1, \dots, x_C$ en un espacio n- dimensional se define $S_B$ que es la matriz de dispersión entre clases y $S_W$ que es la matriz de dispersión intra clases.(Costa Mari, 2020)

$$S_B = \sum_{i=1}^{C} N_i(\mu_i - \mu)\,(\mu_i - \mu)^T \tag{14}$$

$$S_W = \sum_{i=1}^{C} \sum_{X_K \in X_I} (x_k - \mu_i)\,(x_k - \mu_i)^T \tag{15}$$

Siendo $\mu$ la media total

$$\mu = \frac{1}{N}\sum_{I=1}^{N} x_i \tag{16}$$

Y $\mu_i$la media de cada clase $i \in \{1, \dots, c\}$

$$\mu_i = \frac{1}{|x_i|} \sum_{X_K \in X_I 1} x_k \tag{17}$$

A partir de estas matrices, se obtiene una ratio que relaciona $S_B$ y $S_w$.

$$W = argmax_W \frac{|W^T S_B W|}{|W^T S_W W|} \quad (18)$$

Al conseguir maximizar "el ratio" $W$ se logra aumentar la dispersión entre clases y disminuir la dispersión intra clases y de esta manera conseguir una mejor clasificación. (Costa Mari, 2020)

Para el proceso de identificación se repite el proceso de identificación de Eigenfaces, hallando la mayor similitud mediante la distancia euclidiana.

#### 6.1.3.1.3. Histograma de patrones binarios locales (LBPH)

El funcionamiento del algoritmo de Local Binary Pattern Histograms se basa en dividir la imagen en sub-imágenes de 3 x 3 pixeles, se compara el píxel central contra los que lo rodean, y determina si el pixel central es mayor que el siguiente, y se le asigna a este pixel el valor binario de 0 pero si por el contrario, el pixel central es menor que el siguiente se le asigna el valor de 1 (Ver figura 18). De la región resultante se le aplica una conversión a decimal siguiendo el orden de las manecillas del reloj, siendo este valor el descriptor LBP, para obtener la imagen LBP que enfatizara lo rasgos faciales.

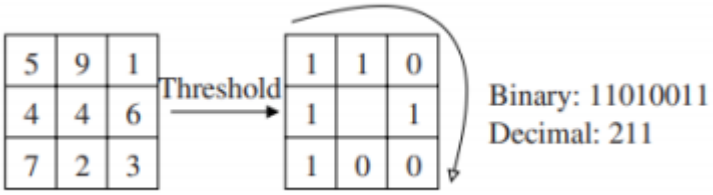


*Figura 18. Conversión de la región de una imagen a su resultante mediante LBPH.* (Costa Mari, 2020)

De esta forma se definen las características básicas en una imagen, características como puntos, líneas y bordes. En la figura 19 se muestran los tipos de patrones característicos que se obtiene al aplicar el procedimiento anterior y de esta forma obtener la información más relevante en una imagen al aplicar un filtro de tamaño 3x3.

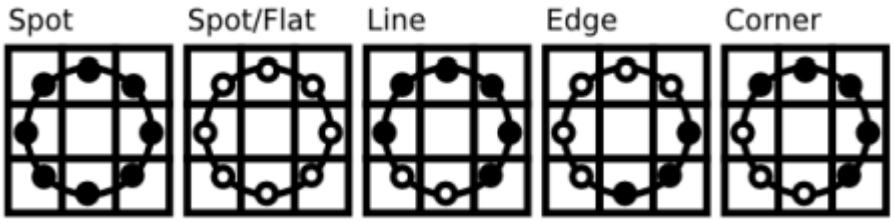


*Figura 19. Zonas definidas por operador BLP.* (Costa Mari, 2020)

El operador LBP tiene buenos resultados en situaciones en las cuales se presenta variación en el ambiente como se observa en la figura 20, en donde, al variar la intensidad de luz que se expone a una persona, el algoritmo aun así logra buenos resultados en la detección de los rostros.

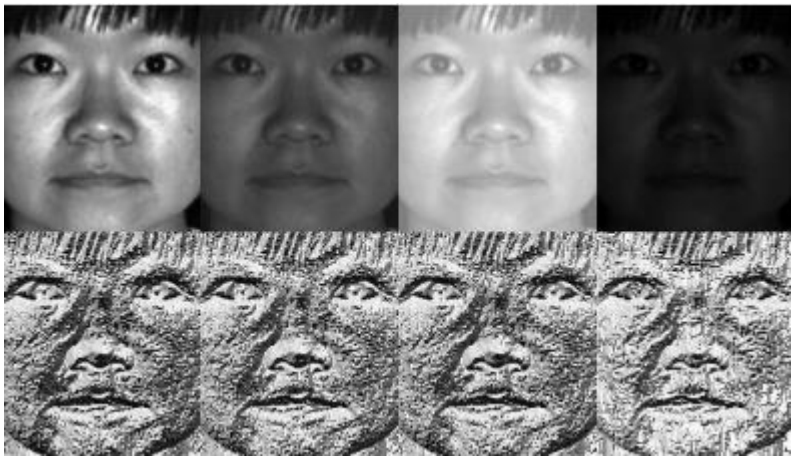

*Figura 20. Extracción de rasgos faciales mediante la variación de la iluminación de la imagen*(Costa Mari, 2020).

Ahora se divide la imagen en subregiones rectangulares, y de acuerdo con la información obtenida mediante el procedimiento LBP, se genera un histograma por subregión, para posteriormente concatenar todos los histogramas y obtener un histograma general que define la imagen completa.

Para identificar el rostro que se presenta en la imagen, se compara la base de datos de histogramas pertenecientes a los rostros identificados contra el histograma de la imagen de entrada, y de esta forma, los histogramas que tenga la menor distancia euclidiana son compatibles.(Costa Mari, 2020)

#### 6.1.3.1.4. Face Recognition

El algoritmo de Face-Recognition inicialmente busca encontrar los puntos de referencia del rostro humano, son 68 puntos como se muestran en la figura 21, para ellos se entrena un algoritmo de aprendizaje.

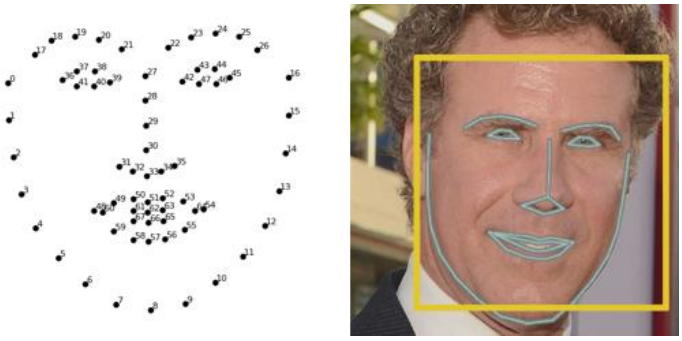

*Figura 21. Puntos de referencia del rostro humano.* (Geitgey, 2016)

El siguiente paso es recortar la imagen, para esto, se aplican movimientos y rotaciones básicas con la finalidad de que no se distorsione y se reduzca el ángulo de giro del rostro. (ver figura 22)

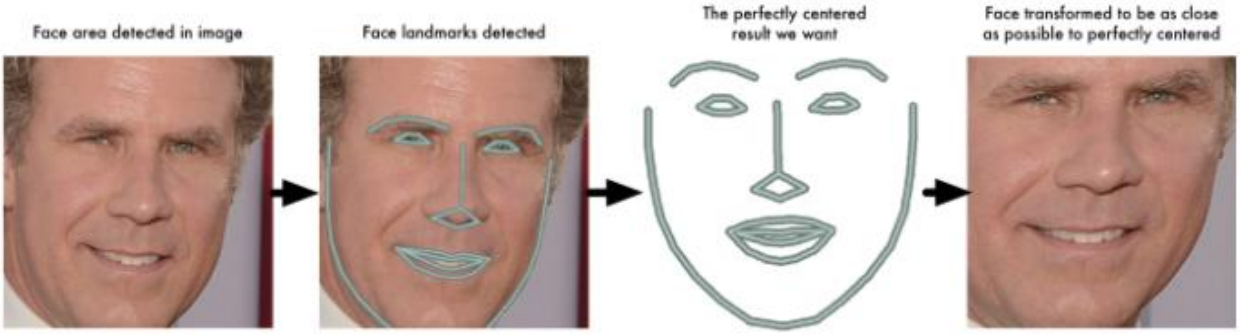


*Figura 22. Transformación de la imagen para realizar un centrado de imagen.* (Geitgey, 2016)

Por último, se entrena una red CNN (Red Neuronal Convolucional) para obtener un vector de 128 dimensiones que define las características de cada rostro.

Para mejorar el funcionamiento del algoritmo en el entrenamiento se cargan tanto imágenes de un rostro conocidos, como imágenes de un rostro desconocido, buscando que la red neuronal genere valores de características similares entre las imágenes del mismo sujeto, en cambio, entre la imagen del sujeto conocido y la del sujeto desconocido, los valores de características tendrán mayor diferencia en los datos.

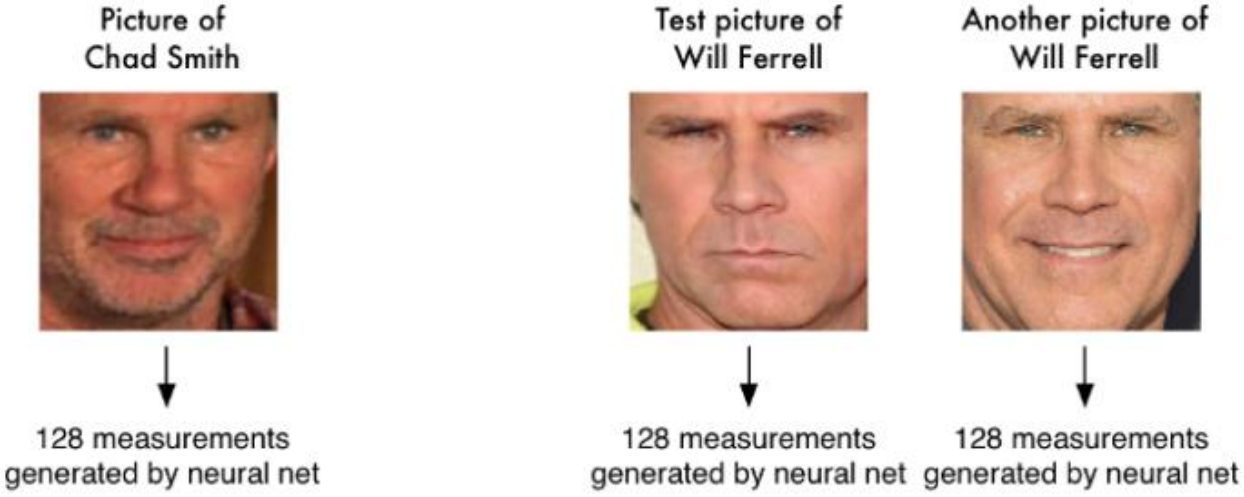


*Figura 23. Procedimiento para el entrenamiento óptimo de la red neuronal.*(Geitgey, 2016)

En la figura 23, se muestra el procedimiento para entrenar la red neuronal, se ingresan dos rostros conocidos, en este caso, el rostro de Chad Smith y Will Ferrell, el sistema de cada imagen obtendrá un vector de 128 dimensiones con información de cada rostro. Posteriormente se ingresa una imagen de prueba de Will Ferrer, la red neuronal de esta imagen obtendrá 128 datos característicos que tendrán valores muy similares a la primera imagen ingresada de Will Ferrell, en cambio, con relación a la imagen de Chad Smith los valores de los vectores presentarán una mayor dispersión.

Este procedimiento se repite millones de veces junto con millones de imágenes para que la red neuronal aprenda a calcular de forma correcta los valores de características que coinciden en el mismo rostro. (Geitgey, 2016)

Para realizar la comparación entre los rostros y determinar que rostro de la base de datos coincide con el rostro de prueba, se calcula la distancia euclidiana entre las marcas de referencia, para posteriormente comparar las distancias calculadas entre el rostro de entrada y los rostros referentes a la base de datos. (Morcote, 2020)

## 6.2. Desarrollo de sistema de detección, extracción y reconocimiento de rostros en escena de múltiples personas.

Las características de hardware para el desarrollo del sistema de videovigilancia son:

- Lenovo E430.
- Procesador: Intel Core I5 3ra generación.
- Memoria RAM: 6 GB.
- Sistema Operativo: Ubuntu.
- Cámara Web: LifeCam HD 3000, Resolución Máxima (1280x800).

El esquema que se presenta en la figura 24 demuestra cómo se organizó el sistema de videovigilancia residencial.

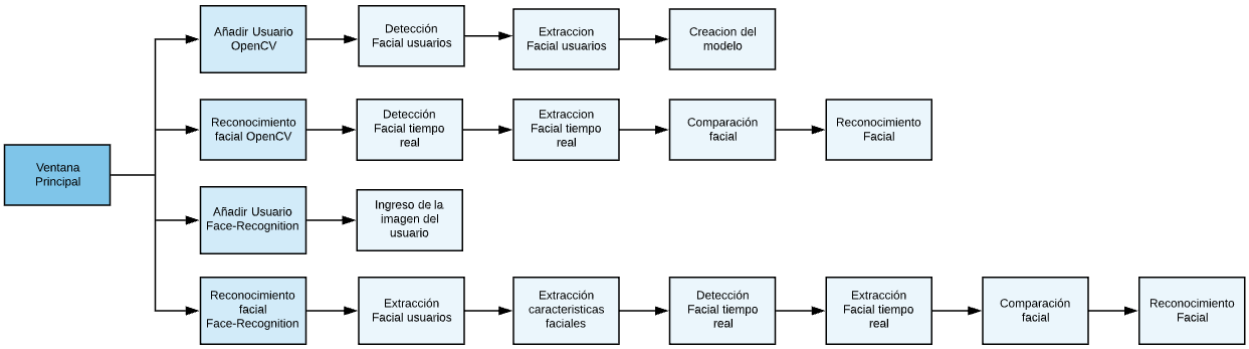


*Figura 24. Diagrama general del algoritmo desarrollado.*

Dicho sistema, está conformado por un bloque principal del cual se derivan 4 bloques en paralelo, dos bloques permiten trabajar con la librería OpenCV y los dos restantes trabajar con la librería Face-Recognition.

El primer bloque que recibe el nombre *"añadir usuario OpenCV",* se encarga del proceso de alimentar la base de datos con la información facial de los usuarios. Para realizar esta tarea se ejecutan las tareas de los tres bloques que se encuentran en serie, encargados de detectar y extraer los rostros para pasar al último paso, el cual es la creación de un modelo.

El segundo bloque que recibe el nombre de *"Reconocimiento facial OpenCV"* se encarga del proceso de Reconocer a los usuarios en tiempo real mediante la librería OpenCV. Para realizar esta función se debe realizar las tareas de los cuatro bloques en serie que derivan de este, las tareas de los primeros dos bloques son Detección y extracción facial en tiempo real, el tercer bloque compara los datos faciales, y el último bloque de esta sección identifica al usuario en caso de que haya una coincidencia.
El tercer bloque que recibe el nombre de *"Añadir Usuario Face Recognition"*, se encarga de almacenar la información de los residentes del lugar. Para realizar esta función se ejecuta la tarea del bloque que está en serie, que consiste en ingresar el nombre de usuario y almacenar su foto.

El último bloque con la etiqueta de *"Reconocimiento Facial Face-Recognition",* se encarga de las tareas relacionadas al reconocimiento facial haciendo uso de esta librería. Para realizar esta función se realizan las tareas de los 6 bloques que están en serie. Los primeros dos bloques se encargan de detectar y extraer las características faciales de los usuarios registrados, los siguientes dos bloques que están en serie se encargan de detectar y extraer las características faciales de los rostros en tiempo real, el quinto bloque compara las características faciales obtenidas en los bloques anteriores y el último bloque mostrar si hay coincidencias.

Para el desarrollo de un sistema de reconocimiento facial se deben implementar tres etapas, las cuales se presentan en el diagrama de la figura 25.

*Figura 25. Etapas sistema de reconocimiento facial*

Debido a que las librerías de OpenCV y FaceRecognition ya cuentan con los algoritmos de detección de rostro y reconocimiento facial, se desarrolló el sistema de videovigilancia empleando las dos librerías y se desarrollaron dos algoritmos uno propio para cada librería.

### 6.2.1. Reconocimiento facial mediante la librería Face-Recognition.

Para el algoritmo de reconocimiento facial que se implementó mediante la librería Face-Recognition se seleccionaron las técnicas que se observan en la figura 26.

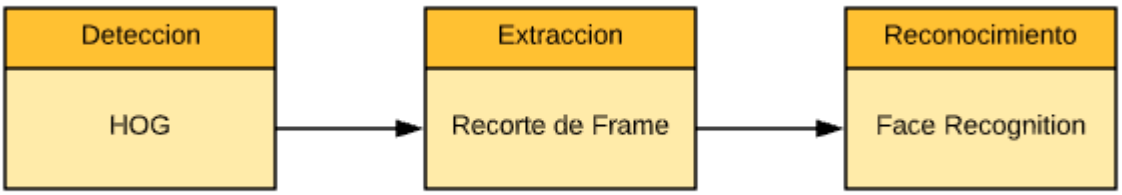


*Figura 26. Diagrama de flujo técnicas implementadas del algoritmo de Face-Recognition.*

Las etapas empleadas en el algoritmo desarrollado son**:**

- Alimentación base de datos: En el directorio de rostros se debe ingresar una imagen de cada residente de la vivienda.
- Extracción características faciales: Se obtiene un vector con las características únicas de cada rostro.
- Adquisición de imagen: Mediante una cámara web se obtiene la información de las personas que están ingresando a la zona residencial.
- Detección de rostro: Mediante la librería Face-Recognition se aplica un detector de rostros que emplea la técnica HOG, se escogió esta técnica debido a que la velocidad de detección es menor en comparación a la técnica CNN que también presenta la librería.
- Extracción de rostro: A través de programación extraer la información en la región de la imagen en la que se detectó un rostro.
- Comparación rostros: Calcula la distancia euclidiana entre los rostros en pantalla y los almacenados en la base de rostros conocidos.
- Reconocimiento Facial: Se muestra en pantalla que rostros en pantalla presentaron un alto nivel de coincidencia con el de los residentes de la casa.

El algoritmo desarrollado en base a la librería Face-Recognition tiene dos secciones principales en las cuales se desarrollará el sistema de reconocimiento facial.

La primera sección de código se encarga de añadir los usuarios a un directorio, para ellos se digita el nombre del usuario, posterior a eso se habilitan los dispositivos de adquisición de imagen para realizar una foto de la persona que se encuentra en pantalla.

**AÑADIR USUARIO FACE-RECOGNITION**

```
Función añadir usuario face-recognition ()
    Si hay un nombre ingresado
        Habilitar cámara
        Capturar foto
        Guardar foto con nombre ingresado
    Si no
        Mostrar dialogo de alerta
```

La segunda sección es el bloque de reconocimiento facial, el funcionamiento de este algoritmo consiste en crear vectores que almacenaran los nombres de los usuarios, y la codificación de cada foto que se almaceno en el bloque de código anterior, posterior a eso se habilitan nuevamente el dispositivo de adquisición de imagen para obtener una imagen en tiempo real, de esta imagen se detecta y extra el rostro para pasarlo por un modelo pre-entrenado que extraerá 128 características faciales, estas características se compararan con las de los rostros almacenados y en caso de haber una coincidencia, en pantalla aparecerá el nombre del usuario con el cual se presentó la coincidencia.

**RECONOCIMIENTO FACIAL FACE-RECOGNITION**

***Función reconocimiento facial Face-recognition*** *()*

*Determinar cuántos usuarios existen*

*Vector_nombres = extraer nombre de los usuarios*

*Vector_fotos = importar fotos de los usuarios*

*Vector_caracteristicas = encontrar matriz de características para cada usuario*

*Habilitar Cámara*

*Capturar fotograma*

*Detectar rostros presentes en el fotograma*

*Extraer rostros*

*Extraer características de cada rostro encontrado*

*Comparar rostros encontrados con los usuarios*

*Mostrar nombre de usuario o desconocido*

### 6.2.2. Reconocimiento facial mediante OpenCV

El algoritmo de reconocimiento facial mediante la librería Open CV empleará las técnicas que se observan en la figura 27.

.

*Figura 27. Diagrama de flujo técnicas implementadas del algoritmo de OpenCV.*

Las etapas empleadas en el algoritmo desarrollado son**:**

- Alimentación base de datos: En el directorio de rostros se debe ingresar 300 imágenes de cada residente de la vivienda.
- Entrenamiento modelo: Se obtiene un modelo que caracteriza los rostros almacenados.
- Adquisición de imagen: Mediante una cámara web se obtiene la información de las personas que están ingresando a la zona residencial.
- Detección de rostro: Mediante la librería Open CV se aplica un detector de rostros que emplea la técnica Viola & Jones.
- Extracción de rostro: A través de programación extraer la información en la región de la imagen en la que se detectó un rostro.
- Comparación de rostros: Aplica el modelo a los rostros detectados, para esta etapa se escogió el algoritmo de LBPH debido a que presenta mayor eficiencia y tasa de éxito que los otros dos algoritmos de la librería OpenCV.
- Reconocimiento Facial: Se muestra en pantalla que rostros en pantalla presentaron un alto nivel de coincidencia con el de los residentes de la casa.

El algoritmo de reconocimiento facial que se desarrolló bajo la librería OpenCV tiene tres secciones principales de código, la primera se encarga de añadir usuarios, es decir, los habitantes de la zona residencial, para ello en un directorio se crea una carpeta con el nombre de la persona, posterior a eso se toman 300 fotos del usuario, en las cuales se detecta y recorta el rostro para su almacenamiento en el directorio.

**AÑADIR USUARIO OpenCV**

```
Función añadir usuario OpenCV ()
    Si hay un nombre ingresado
        Crear carpeta con nombre ingresado
        Habilitar cámara
        Mientras cantidad fotos almacenadas < 300
            Capturar foto
            Detectar rostro en la captura
            Extraer rostro
            Almacenar foto de rostro en la carpeta
    Función para entrenar modelo
    Si no
        Mostrar dialogo de alerta
```

Posterior a ello con la información ya organizada y almacenada se entrena un modelo, el cual se guarda en el mismo directorio del cual se extrajo la información, para su posterior comparación.

## ENTRENAR MODELO

***Función entrenar modelo*** *()*

*Importar nombres de los usuarios*

*Importar fotos*

*Importar modelo a usar*

*Entrenar modelo*

*Guardar .xml del modelo entrenado*

Para realizar el reconocimiento facial se importa el modelo creado en la sección anterior y se habilita la cámara para obtener una imagen en tiempo real, de esta imagen en tiempo real se detecta y extrae el rostro para pasarlo por el modelo, si el valor de correlación tiene una coincidencia mayor a 80, se genera en pantalla el nombre de la persona con la cual coincide el rostro.

## RECONOCIMIENTO FACIAL Open CV

***Función reconocimiento facial Open CV*** *()*

*Importar nombres de usuarios*

*Importar modelo .xml*

*Habilitar Cámara*

*Capturar fotograma*

*Detectar rostros presentes en el fotograma*

*Extraer rostros*

*Comparar cada rostro con el modelo entrenado*

***Si*** *Umbral > 80 el rostro es de un residente*

*Mostrar nombre de usuario*

***Si*** *Umbral < 80 el rostro es de un residente*

*Mostrar desconocido*

En la *figura 28,* se presenta lo lógica de funcionamiento del algoritmo final del sistema de videovigilancia residencial empleando las dos librerías de reconocimiento facial.

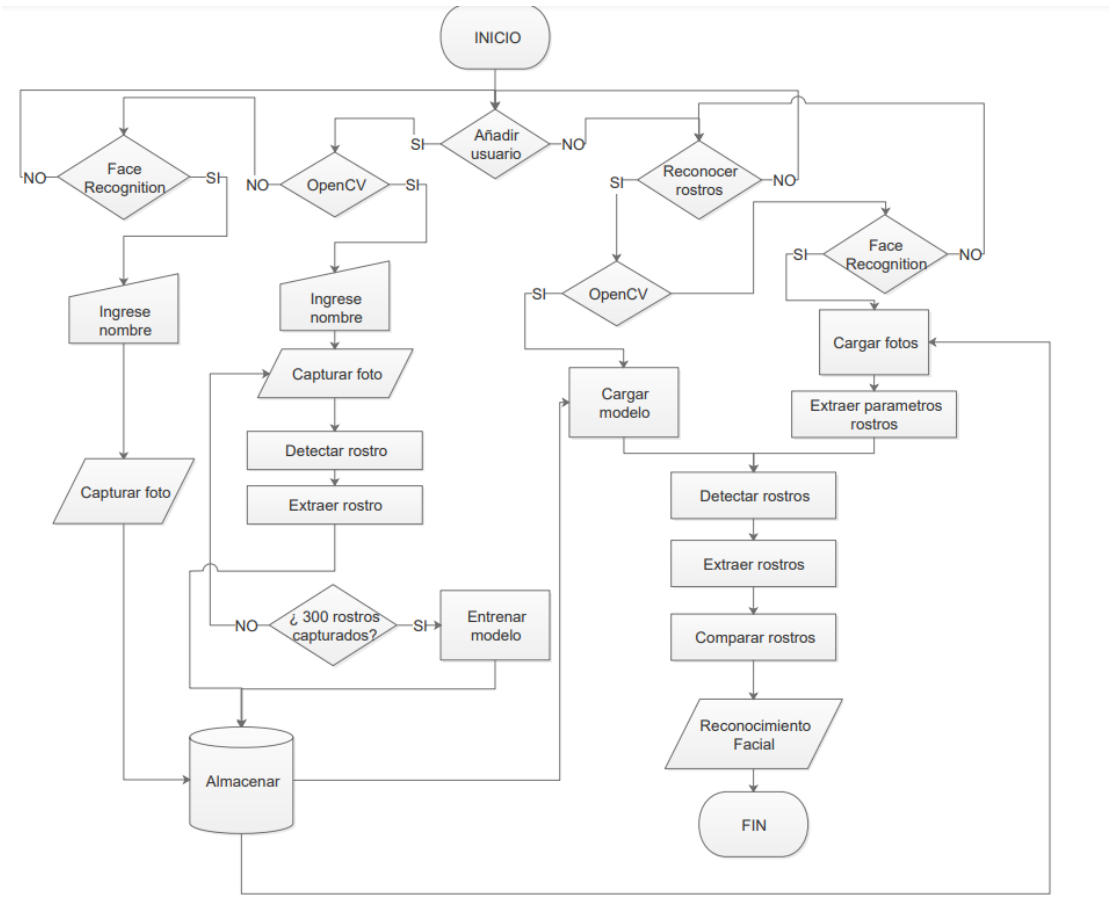


*Figura 28. Diagrama de flujo sistema de videovigilancia residencial Fuente: Autor*

El diagrama de flujo se divide principalmente en dos ramas, añadir usuarios o correr el sistema de reconocimiento facial.

Al dar inicio al programa lo primero es determinar si se desea añadir un usuario o no, si la respuesta es afirmativa se pregunta a que algoritmo desea añadir el usuario ya sea OpenCV o Face-Recognition, ya que cada librería tiene un almacenamiento diferente. Por la rama de Face-Recognition se espera el ingreso de un nombre de usuario y posteriormente se captura y almacena la foto. En la rama de OpenCV se espera el ingreso de un nombre, se captura la foto, se detecta el rostro en la foto, y extrae el rostro para posteriormente almacenarlo, este proceso de captura, detección, extracción y almacenamiento se repite 300 veces hasta tener un total de 300 fotos almacenadas por usuario, cuando se tienen las 300 capturas se procede a entrenar el modelo de LBPH. Estos procesos se repiten para cada usuario que se agregue.
Cuando la respuesta es negativa frente a si se desea añadir usuario, se debe decidir entre hacer reconocimiento facial con Face-Recognition u OpenCV, si se decide ir por la rama de Face-Recognition el sistema carga la foto de cada uno de los usuarios y posteriormente extrae los parámetros del rostro, a continuación, en cada fotograma detecta los rostros presentes, extrae las características de cada rostro encontrado, los compara con las características de los usuarios y finalmente menciona si es un usuario o un desconocido, en la rama de OpenCV el proceso es similar, en dado caso que se seleccione hacer reconocimiento facial con OpenCV el sistema carga el modelo entrenado de LBPH de la fase de añadir usuario, luego detecta rostros en cada fotograma,

extrae los rostros, los compara con el modelo entrenado y presenta en pantalla si es un usuario o un desconocido.

## 6.3. Diseño interfaz gráfica de usuario GUI.

La GUI fue desarrollada mediante la implementación de la librería Tkinter perteneciente a el lenguaje Python. En la figura 29 se observa la ventana principal.

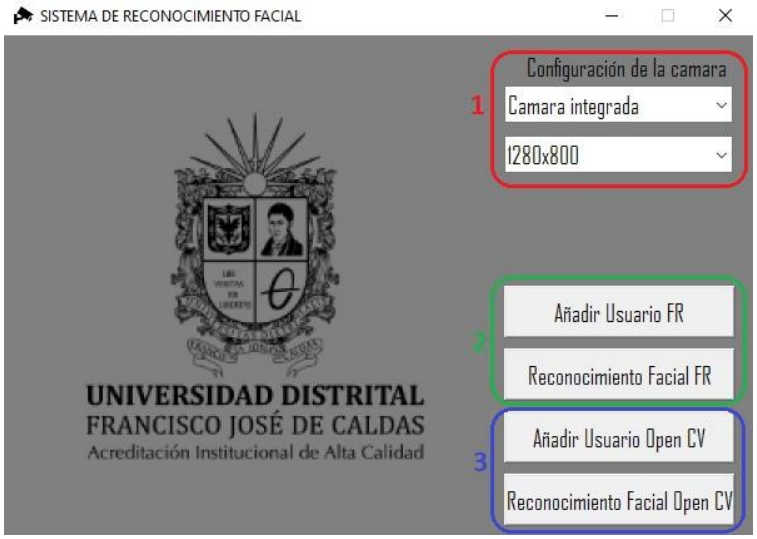


*Figura 29. Ventana principal Fuente: Autor*

La ventana principal se compone de 3 secciones, a continuación, se explicará cada una.

1. **Configuración de la cámara**

En esta parte de la interfaz se podrá realizar la configuración de la cámara que se usará ya sea para agregar un usuario o para realizar el reconocimiento facial, por defecto tiene configurada la cámara integrada del computador y la resolución de 1280x800.

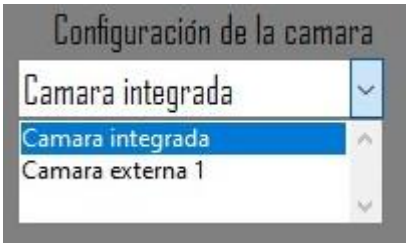


*Figura 30. Selección de cámara Fuente: Autor*

El sistema detecta que cámaras están conectadas al computador y las agrega a la lista que se puede ver en la figura 30, para que el usuario elija la que desee.

*Figura 31. Selección de la resolución Fuente: Autor*

En la figura 31 se despliega una lista con algunas resoluciones para que el usuario elija la que desee.

## 2. Manejo librería Face-Recognition

En esta sección se tienen dos botones uno para administrar los usuarios, llamado "*añadir usuario FR*", y el otro para correr el sistema de reconocimiento facial con la librería de Face-Recognition, llamado "*Reconocimiento Facial FR*".

- **Botón de Añadir usuario FR**

Al obturar el botón de "*añadir usuario FR*", se abrirá una ventana como la que se muestra en la figura 32.

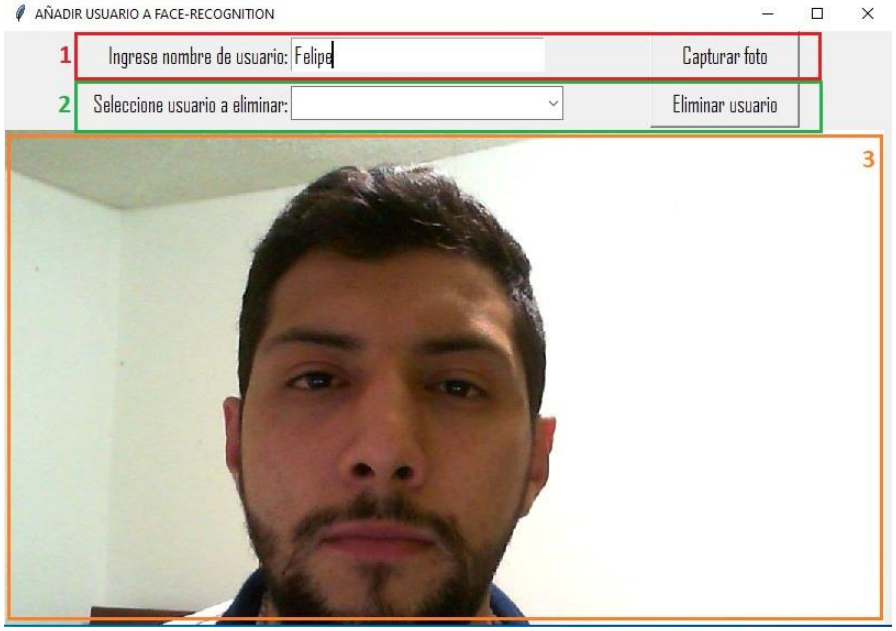


*Figura 32. Ventana administrar usuarios de Face-Recognition Fuente: Autor*

En esta ventana se tienen 3 zonas que se describen a continuación.

En la zona 3 se tiene la visualización en tiempo real de la cámara elegida anteriormente.

En dado caso que se desee eliminar un usuario de la librería Face-Recognition, en la zona 2 de color verde se puede elegir en la lista desplegable (Ver figura 33), el nombre del usuario que se desea eliminar y posteriomente con el boton contiguo eliminar al usuario seleccionado.

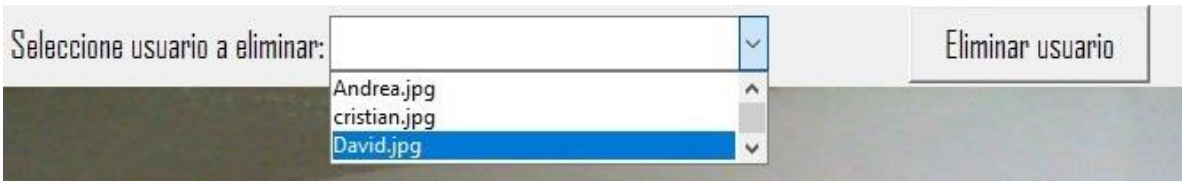


*Figura 33. zona para eliminar un usuario Fuente: Autor*

En la zona 1 demarcada por un recuadro de color rojo se puede ingresar el nombre de un nuevo usuario y con el boton situado a la derecha que lleva por nombre "Capturar foto", se tendrá una captura del video. Al obturar el boton se presentan en una nueva ventana (Ver figura 34) con la imagen capturada la cual será la imagen a archivar con el nombre ingresado.

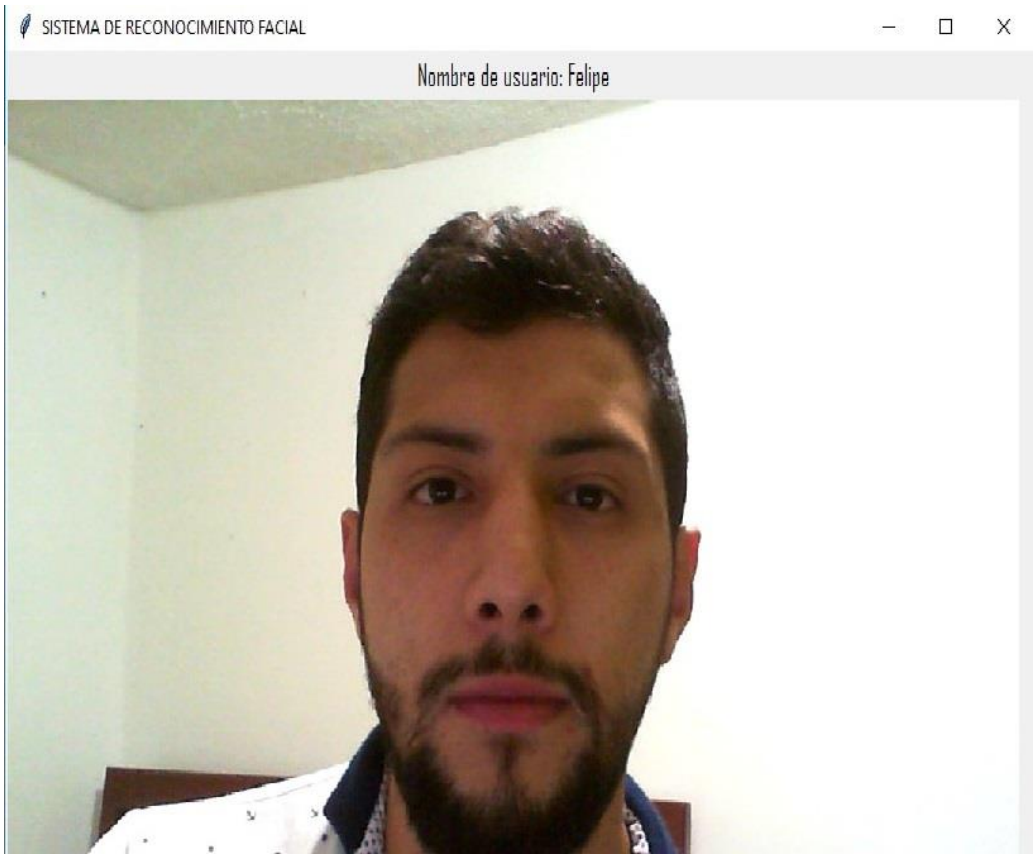


*Figura 34. Ventana nombre y foto a archivar Fuente: Autor*

Enseguida emerge un cuadro de dialogo como el que se muestra en la figura 35, donde se pregunta si se desea guardar o no el usuario.

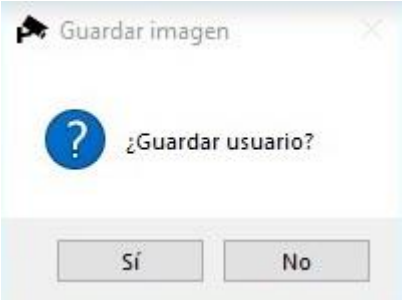


*Figura 35. Dialogo confirmando la decisión de añadir al usuario Fuente: Autor*

Dado el caso que se tome la decisión de no guardar el usuario, se cierran las ventanas, no se archiva el usuario y se regresa nuevamente a la ventana de ingresar un usuario. Si por el contario, se decide guardar el usuario, este será almacenado y se retornará a la ventana de ingresar usuario por si se desea archivar otro usuario.

- **Boton de reconocimento facial FR**

Al obturar el boton de "Reconocimiento Facial FR" se cargaran los usuarios y se abrirá una nueva ventana con un frame que permite ver la cámara en tiempo real y un boton en la parte superior que permite salir del modo de reconocimiento facial y retorna a la ventana a principal (Ver figura 36).

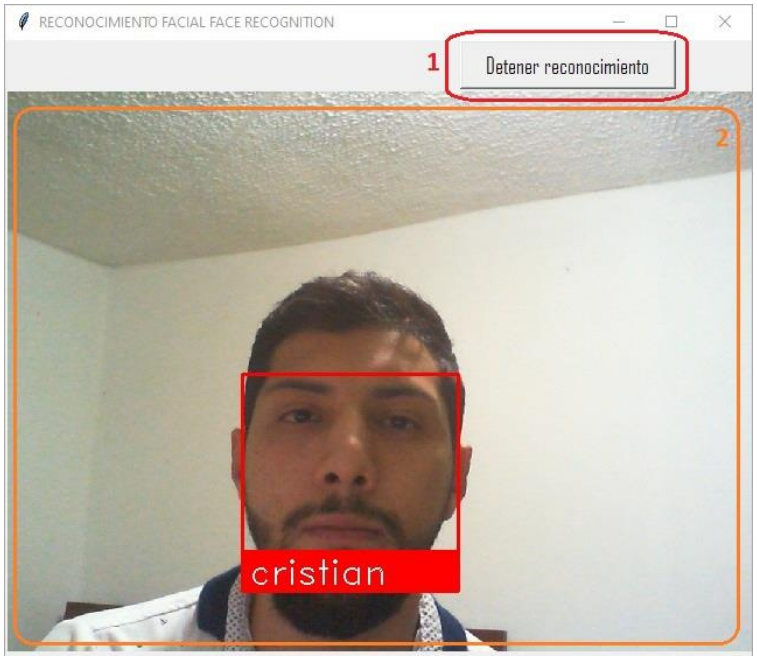


*Figura 36. Reconocimiento Facial con librería Face-Recognition Fuente: Autor*

### 3. Manejo de la librería OpenCV

Similar a la sección 2 se tienen en esta sección dos botones uno para añadir usuario a la librería OpenCV y otro para iniciar el reconocimiento facial con OpenCV.

- **Boton de añadir usuario OpenCV**

Al obturar el boton “Añadir usuario OpenCV” se abrira una ventana con 3 zonas (Ver figura 37).

*Figura 37. Ventana administrar usuarios de OpenCV Fuente: Autor*

En la zona 3 se tendrá la visualización en tiempo real de la cámara elegida.
En dado caso que se desee eliminar un usuario de la librería OpenCV en la zona 2 de color verde se podrá elegir en la lista desplegable, el nombre del usuario que se desea eliminar y posteriomente con el boton contiguo se podrá eliminar al usuario seleccionado.
En la zona 1 demarcada por un recuadro de color rojo se puede ingresar el nombre de un nuevo usuario y al oprimir el boton situado a la derecha que lleva por nombre “Capturar foto”, se abrirá un dialogo que informara que se no debe mover de enfrente de la cámara hasta que se indique (Ver figura 38).

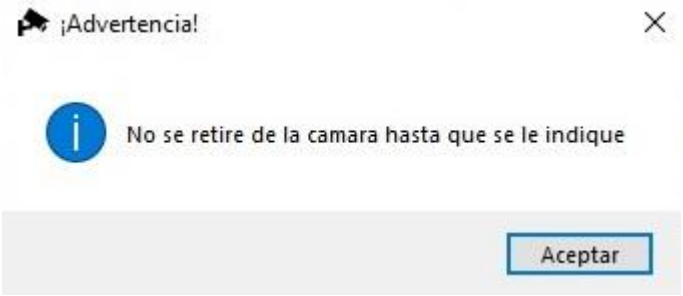


*Figura 38. Dialogo de advertencia Fuente: Autor*

Despues de oprimir “Aceptar” en el cuadro de dialogo se abrira un nuevo frame, como el que se muestra en la figura 39, en el cual empezaran a capturarse las 300 fotos necesarias para el algoritmo de OpenCV.

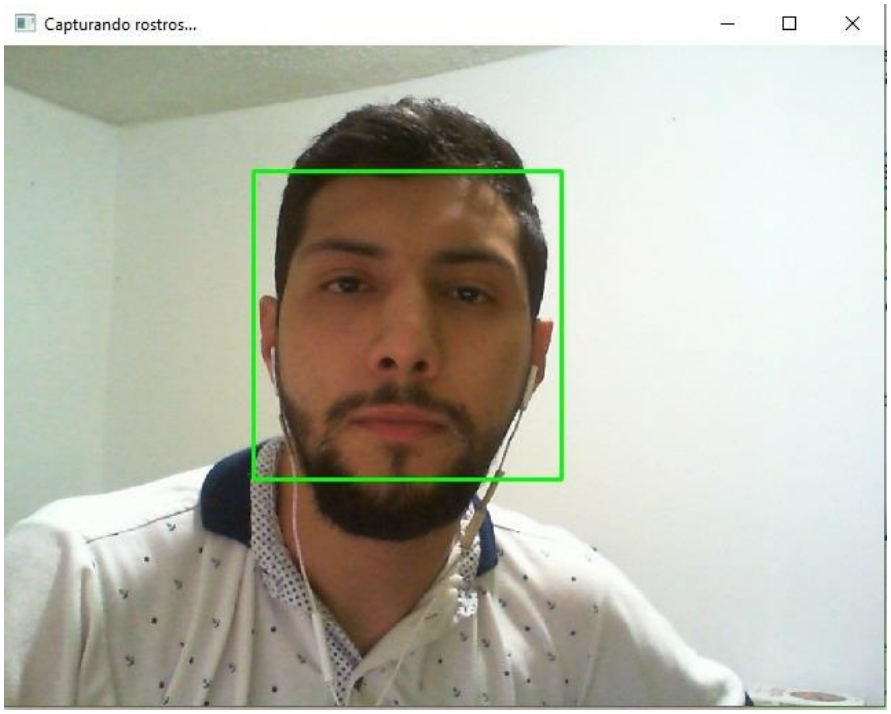


*Figura 39. Frame para captura de fotos de OpenCV Fuente: Autor*

Al terminar de tomarse las 300 fotos otro cuadro de dialogo emerge informando que ya se puede retirar de la cámara y preguntando si realemente se desea guardar el usuario (Ver figura 40).

*Figura 40. Dialogo para la confirmación de guardar usuario Fuente: Autor*

Si se selecciona “No” en el dialogo de la figura 40 se eliminan las fotos almacenadas y la carpeta con el nombre de usuario, si por el contrario se selecciona “Si” se almacena el usuario.

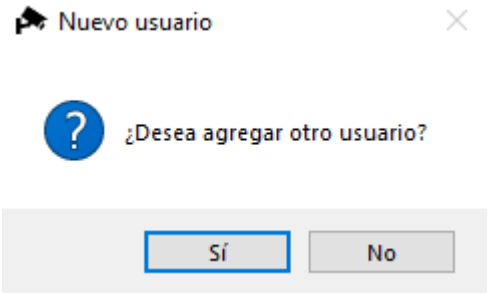


*Figura 41. Dialogo confirmar si desea añadir otro usuario Fuente: Autor*

Nuevamente emerge otro dialogo donde se pregunta si se quiere almacenar otro usuario, o no, en caso de ser afirmativa la respuesta se cierra el dialogo y se retorna a la ventana de la figura 37, si por el contrario es negativa la respuesta se entrena el modelo LBPH con los usuarios que hay hasta el momento y cuando el modelo termine de entrenarse se presenta un dialogo informando que ya se entrenó y almacenó el modelo para su posterior uso (Ver figura 42).

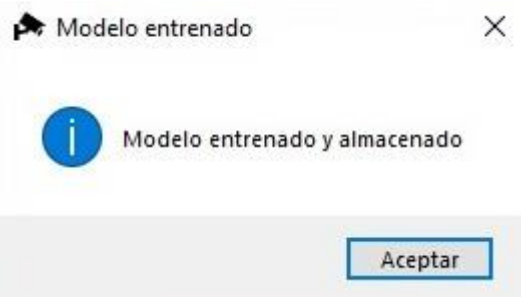


*Figura 42. Dialogo informativo acerca del modelo de entrenamiento Fuente: Autor*

- **Boton de reconocimento facial OpenCV**

Al obturar el boton de "Reconocimiento Facial Open CV" se cargará el modelo entrenado y se abrirá una nueva ventana (Ver figura 43)con un frame que permite ver la cámara en tiempo real y un boton en la parte superior que permite salir del modo de reconocimiento facial y retornar a la ventana principal.

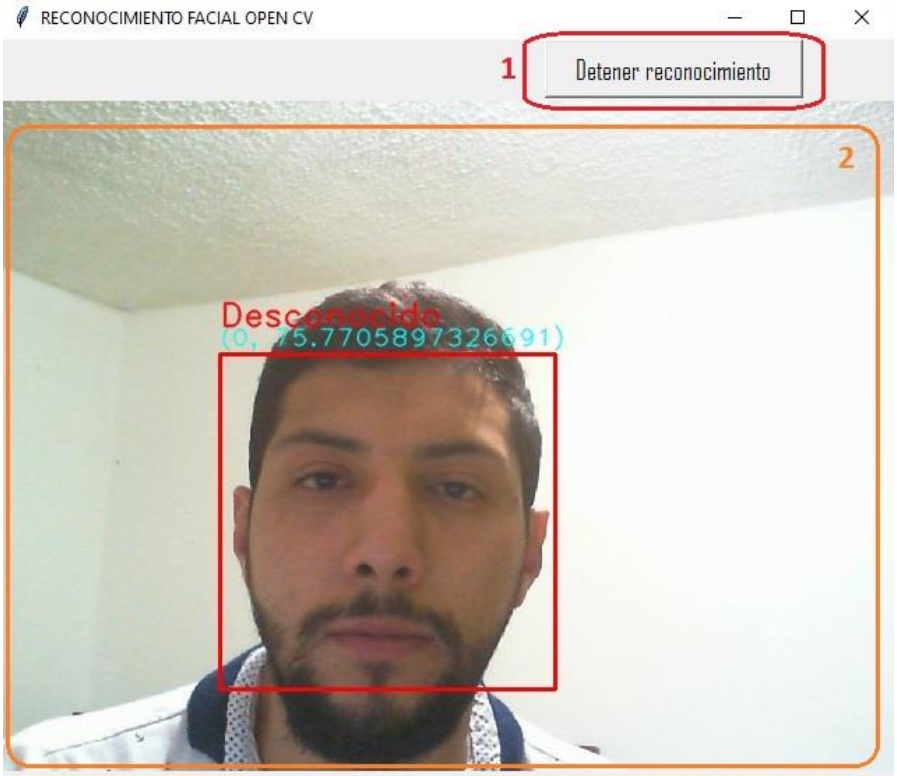


*Figura 43. Reconocimiento Facial con librería Open CV Fuente: Autor*

# 7. Resultados

En esta sección se muestran los resultados obtenidos al desarrollar el sistema de video vigilancia residencial de bajo costo, para la presentación de los resultados se desarrollarán cuatro etapas: Diseño definitivo, restricciones de la solución, validación de los resultados y Validación del funcionamiento del sistema de videovigilancia residencial.

## 7.1. Diseño definitivo.

Como resultado final se obtuvo un sistema de videovigilancia residencial en tiempo real mediante el cual se puede verificar si las personas que entran a una zona residencial son habitantes de esta o desconocidos, permitiendo al operador del sistema escoger entre dos algoritmos de reconocimiento facial de los cuales se caracterizó la distancia de operación, ángulo de detección y tiempo de operación, para escoger el más acorde al hardware que se disponga y las condiciones del lugar, con una interfaz de usuario de fácil manejo.

## 7.2. Restricciones a la solución.

Debido a que el sistema de videovigilancia implementa dos algoritmos de detección y reconocimiento facial de los cuales el usuario selecciona el más conveniente de acuerdo con su aplicación, las restricciones se dividirán de acuerdo con el algoritmo.

### 7.2.1. Restricciones algoritmo Face-Recognition.

- Distancia máxima de detección y reconocimiento facial 5.2 m.
- Máximo 4 personas en escena para detección y reconocimiento facial a una resolución de 1280 x 800 asumiendo que el tiempo del ciclo de video máximo es 1.5 s.
- Angulo de giro de rostro ± 60°.
- Resolución optima de funcionamiento 800x600.

### 7.2.2. Restricciones algoritmo OpenCV.

- Distancia máxima de detección y reconocimiento facial es 2.8 m.
- De acuerdo con las proyecciones 8 personas a la resolución de 1280 x 800 asumiendo que el tiempo máximo del ciclo de video es 1.5 s.
- Angulo de giro de rostro ± 30°.
- Resolución optima de funcionamiento 1280 x 800 o mayor.

## 7.3. Validación de los resultados.

### 7.3.1. Prueba de distancias.

En la prueba de distancias se busca establecer la distancia máxima en la cual el algoritmo reconoce un rostro correctamente, bajo condiciones específicas, y cómo dentro de ese rango de distancias varia la resolución con la cual se captura el rostro. Para esta prueba se planteó manejar las resoluciones de 160x120 (px), 352x288 (px), 800x500 (px) y 1280x800 (px).
La prueba consistió en realizar un recorrido en dirección frontal hacia cámara, durante el recorrido obtener los datos de distancia de operación y la cantidad de pixeles con los cual se extrae el rostro, variando la resolución de la cámara y la librería.

#### 7.3.1.1. Algoritmo Face-Recognition

Los resultados obtenidos en las pruebas de distancias de detección para 4 resoluciones de cámara (160x120, 352x288, 800x600, 1280x800) se presentan en la tabla 1.

| Resolución de la cámara | | | | | | | |
|---|---|---|---|---|---|---|---|
| 160x120 | | 352x288 | | 800x600 | | 1280x800 | |
| Distancia (cm) | Pixeles (px) | Distancia (cm) | Pixeles (px) | Distancia (cm) | Pixeles (px) | Distancia (cm) | Pixeles (px) |
| 10 | 197 x 197 | 10 | 266 x 266 | 20 | 460 x 460 | 30 | 554 x 554 |
| 30 | 62 x 62 | 40 | 129 x 129 | 50 | 223 x 223 | 100 | 186 x 186 |
| 60 | 44 x 44 | 80 | 75 x 75 | 100 | 129 x 129 | 150 | 129 x 129 |
| 90 | 36 x 36 | 120 | 52 x 52 | 150 | 75 x 75 | 200 | 89 x 89 |
| | | 160 | 36 x 36 | 200 | 62 x 62 | 250 | 74 x 74 |
| | | 200 | 36 x 36 | 250 | 52 x 52 | 300 | 62 x 62 |
| | | 230 | 36 x 36 | 300 | 44 x 44 | 350 | 52 x 52 |
| | | | | 350 | 36 x 36 | 400 | 43 x 43 |
| | | | | 400 | 36 x 36 | 450 | 36 x 36 |
| | | | | 460 | 36 x 36 | 520 | 36 x 36 |

*Tabla 1. Resultado prueba de distancias Face-Recognition*

De los datos mostrados en la tabla se puede observar como para la resolución de 160x120 pixeles la distancia de operación bajo las condiciones de prueba es de 90 cm, lo cual demuestra que trabajar en este algoritmo con una resolución tan baja, no es conveniente para sistemas de videovigilancia residencial. En la resolución de 352x288 la distancia de operación es de 230 cm, lo cual es un aumento significativo en la distancia operación de este algoritmo, esta resolución ya puede ser empleada en sistemas de video vigilancia, sin embargo, el posicionamiento de la cámara jugaría un papel crítico en la eficiencia del sistema. Configurando el sistema para trabajar con una resolución de 800x600, la distancia de operación del algoritmo aumento a 460 cm, aumento de distancia detección del doble con relación a la resolución anterior, la distancia es óptima para un sistema de videovigilancia residencial. Para la configuración del algoritmo con una resolución de 1280x800

se observó que se puede trabajar a una distancia de operación de 520 cm, distancia óptima para un sistema de videovigilancia residencial.

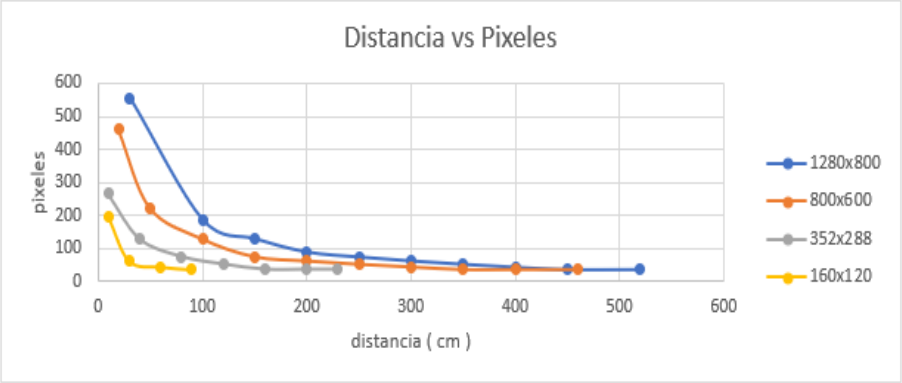


*Figura 44. Comportamiento resolución cámara vs distancia vs resolución detección Face-Recognition*

La figura 44, se puede observar que a medida que las resoluciones aumentan se puede ver un aumento en la distancia de operación, en el primer caso, el aumento de la distancia de operación fue del 155% que equivale 140 cm, el aumento en el segundo cambio de distancia fue de 100% que equivale a 230 cm, el tercer cambio de resolución el aumento distancia fue del 13 % que equivale a 60 cm por lo tanto se puede observar que aunque se presente un aumento notable en la resolución el aumento de la distancia de operación cada vez será menor.

#### 7.3.1.2. Algoritmo OpenCV

Los resultados obtenidos en las pruebas de distancias de detección para 3 resoluciones de cámara (352x288, 800x600, 1280x800) se presentan en la tabla 2, no se pudo tomar datos en la resolución de 160x120 debido a que la distancia máxima de reconocimiento del algoritmo era demasiado corta.

| Resolución de la cámara | | | | | |
|---|---|---|---|---|---|
| 352x288 | | 800x600 | | 1280x800 | |
| Distancia (cm) | Pixeles (px) | Distancia (cm) | Pixeles (px) | Distancia (cm) | Pixeles (px) |
| 15 | 244 x 244 | 15 | 399 x 399 | 25 | 698 x 698 |
| 20 | 204 x 204 | 20 | 323 x 323 | 50 | 367 x 367 |
| 40 | 155 x 155 | 50 | 235 x 235 | 100 | 166 x 166 |
| 50 | 131 x 131 | 100 | 129 x 129 | 150 | 112 x 112 |
| 70 | 92 x 92 | 150 | 97 x 97 | 200 | 94 x 94 |
| | | 200 | 74 x 74 | 240 | 74 x 74 |
| | | 230 | 64 x 64 | 280 | 62 x 62 |

*Tabla 2. Resultado prueba de distancias OpenCV*

De los datos presentados en la tabla 2 se observa como para la resolución de 352x288 la distancia de funcionamiento del algoritmo es de 70 cm, distancia que es demasiado corta para emplear en un

sistema de video vigilancia residencial. La distancia de operación para la resolución de 800x600 es de 230 cm distancia en la cual un sistema de video vigilancia funcionaria correctamente, sin embargo, sería bastante dependiente de la ubicación de la cámara. En la resolución de 1280x800 la distancia de operación del algoritmo es de 280 cm que representa una distancia que sigue quedando un poco corta para la aplicación, por lo cual, el sistema es dependiente de una correcta ubicación de la cámara y buena iluminación para lograr un mayor alcance.

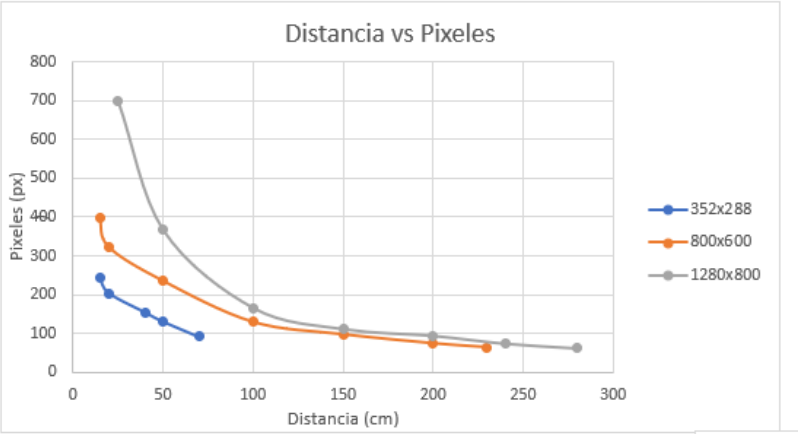


*Figura 45. Comportamiento resolución cámara vs distancia vs resolución detección OpenCV*

En la figura 45, se puede observar que el aumento de la distancia de operación desde el cambio de resolución de 352x288 a la resolución 800 x 600 es de 229% que equivale a 160 cm. El aumento distancia de operación de la resolución de 800 x 600 a la resolución 1280 x 800 es de apenas 21,7 % que equivale a 50 cm, en relación con los datos anteriores podemos observar que a mayor resolución, la distancia de reconocimiento será mayor, sin embargo, la curva de crecimiento es logarítmica, por lo tanto, al aumentar constantemente la resolución, la distancia de una resolución a otra será cada vez más pequeña.

### 7.3.2. Prueba de tiempo.

Para la prueba de tiempo se empleó un timer, con el fin de calcular el tiempo que se demora cada algoritmo en iterar el ciclo de video. Se realizaron 10 tomas de datos y se calculó el promedio variando la resolución de la cámara y la cantidad de personas que el sistema debió detectar y reconocer.

#### 7.3.2.1. Algoritmo Face-Recogintion.

Los resultados obtenidos en las pruebas de tiempos de detección y reconocimiento, para 4 resoluciones de cámara (160x120,352x288, 800x600, 1280x800) se presentan en la tabla 3.

| | Resolución de la cámara | | | |
|---|---|---|---|---|
| N° Personas | 160x120 | 352x288 | 800x600 | 1280x800 |
| 1 | 0,1878 s | 0,2337 s | 0,4560 s | 0,8088 s |
| 2 | 0,3647 s | 0,3888 s | 0,6235 s | 0,9741 s |
| 3 | 0,4838 s | 0,5380 s | 0,7843 s | 1,1383 s |
| 4 | 0,6366 s | 0,6839 s | 0,9337 s | 1,2793 s |

*Tabla 3. Resultados prueba de tiempo Face-Recognition*

De acuerdo a los datos presentados en la tabla 3, se puede observar como el aumento de la resolución, es factor crítico que afecta el tiempo de ejecución del algoritmo, pues, al detectar una persona para la resolución de 160x120 el tiempo de respuesta es de 0,2 segundos, tiempo que es bastante bajo y trabajable en sistemas de video vigilancia, pero para la resolución de 1280 x 800 al reconocer una persona el tiempo de respuesta del algoritmo fue de 0,809 segundos, que es trabajable para sistemas de video vigilancia, pero no es un tiempo óptimo. Cuando el sistema debe reconocer 4 personas con una configuración de resolución de 160x120 el algoritmo ya presenta un tiempo de respuesta de 0,637 segundos, tiempo que no es ideal para un sistema de video vigilancia, cuando se trabaja con una resolución de 1280x800 reconociendo 4 personas el tiempo de respuesta del algoritmo es 1,279 segundos, que es un tiempo relativamente alto, ya que si un individuo pasa corriendo el algoritmo no logrará detectar sus rasgos faciales, lo cual no es óptimo para sistemas de videovigilancia, sin embargo, es un tiempo con el cual se puede trabajar.

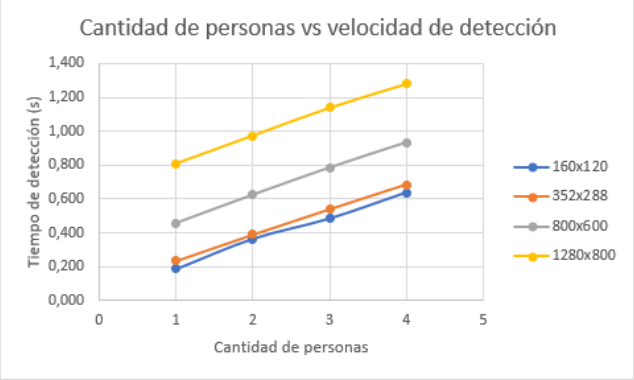


*Figura 46. Grafica de comportamiento Resolución vs personas vs tiempo Face-Recognition*

En la figura 46 se presenta el comportamiento del sistema de videovigilancia al detectar más de una persona es un comportamiento lineal de lo cual se puede deducir cómo se comportará el sistema con más de 4 personas, pero bajo las condiciones dadas se observa que por tiempos de respuesta no es conveniente trabajar con más de 4 personas pues el sistema ya presentaría demasiada perdida de información.

#### 7.3.2.2. Algoritmo OpenCV.

Los resultados obtenidos en las pruebas de tiempos de detección y reconocimiento, para 3 resoluciones de cámara (352x288, 800x600, 1280x800) se presentan en la tabla 4
.

| | Resolución de la cámara | | |
|---|---|---|---|
| N° Personas | 352X288 | 800X600 | 1280X800 |
| 1 | 0,1441 s | 0,1677 s | 0,1824 s |
| 2 | 0,2496 s | 0,2778 s | 0,2904 s |
| 3 | 0,3863 s | 0,3980 s | 0,4450 s |
| 4 | 0,5175 s | 0,5492 s | 0,5901 s |

*Tabla 4. Resultados prueba de tiempo OpenCV*

En la tabla 4,se presentan los datos obtenidos para este algoritmo, se observa que los tiempos de reconocimiento son mucho más bajos que los de la librería de Face-Recognition, los tiempos de respuesta del sistema al cambiar las resolución son bastantes bajos pues máximo se adiciona un retardo de 0,060 segundos, en cuanto a la variación de acuerdo a la cantidad de personas que el sistema debe detectar, los tiempos son bajos al detectar y reconocer una persona para la resolución de 1280x800, que presenta mayor coste computacional, el sistema tiene un tiempo de ejecución de 0,18 segundos y al detectar y reconocer 4 personas el sistema tiene un tiempo de respuesta de 0,59 segundos, tiempo bajo y óptimo para un sistema de video vigilancia.

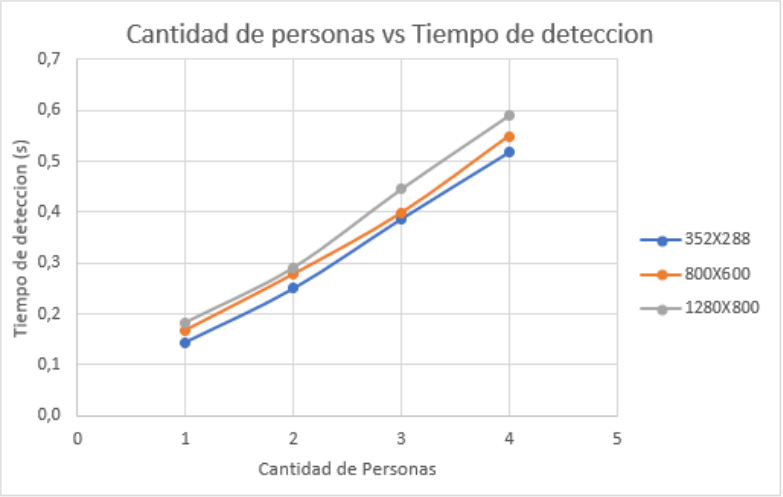


*Figura 47. Grafica de comportamiento Resolución vs personas vs tiempo OpenCV*

En la figura 47, se observa el comportamiento del sistema al detectar y reconocer determinado número de personas, el sistema presenta un comportamiento lineal, mediante el cual se puede observar que el sistema en función del tiempo puede funcionar para realizar detecciones y reconocimientos faciales para más de 4 personas.

### 7.3.3. Prueba de ángulo.

Se realizó una prueba de ángulo a la distancia intermedia de la distancia máxima hallada en la sección 7.3.1 para cada resolución, con la finalidad de observar variaciones en el ángulo de reconocimiento facial.

*Figura 48. Prueba ángulo de reconocimiento Fuente: Autor*

#### 7.3.3.1. Algoritmo Face-Recognition.

Los resultados de las mediciones de ángulo de detección para 4 resoluciones de cámara (160x120,352x288, 800x600, 1280x800) se encuentran en la tabla 5:

| Face Recognition | | | | |
|---|---|---|---|---|
| Distancia | 45 cm | 115 cm | 230cm | 260 cm |
| Resolución | 160X120 | 352X288 | 800X600 | 1280x800 |
| Angulo + | 60° | 54° | 53° | 60° |
| Angulo - | 55° | 60° | 60° | 55° |

*Tabla 5. Resultados prueba de ángulo Face Recognition*

En la tabla 5, se presentan los resultados obtenidos en la prueba de medición de ángulos en la librería Face-Recognition, se observó que el ángulo detección e identificación del rostro no varía de acuerdo a la resolución, dicho ángulo se encuentra entre -55° y +60°, estos ángulos podrían mejorar si se hubiera implementado el método de detección facial CNN pero debido a que este acarrea un mayor coste computacional se determinó trabajar con el método HOG.

*Figura 49. Prueba de ángulo Face-Recognition Fuente: Autor*

La figura 49 muestra el resultado final de las pruebas de ángulo de detección de rostro para el algoritmo Face-Recognition implementado en la investigación.

#### 7.3.3.2. Algoritmo Open CV

Los resultados de las mediciones de ángulo de detección para 3 resoluciones de cámara (352x288, 800x600, 1280x800) se encuentran en la tabla 6.

| Open CV | | | |
|---|---|---|---|
| Distancia | 35 cm | 115 cm | 140 cm |
| Resolución | 342x288 | 800X600 | 1280x800 |
| Angulo + | 30° | 30° | 33° |
| Angulo - | 28° | 30° | 29° |

*Tabla 6. Resultados prueba de ángulo OpenCV*

En la tabla 6 se observan los datos de ángulo obtenidos para este algoritmo, en donde se puede observar que el ángulo de giro, para detección y reconocimiento facial de este algoritmo, es de aproximadamente -30° a +33°, además que la resolución no afecta el ángulo de detección facial.

*Figura 50. Prueba de ángulo OpenCV Fuente: Autor*

La figura 50 muestra el resultado final de las pruebas de ángulo de detección de rostro para el algoritmo OpenCV implementado en la o investigación.

## 7.4. Validar el funcionamiento del sistema de videovigilancia residencial.

Para validar el funcionamiento del sistema de videovigilancia residencial se realizaron 2 pruebas en diferentes lugares, evaluando el desempeño del sistema de videovigilancia residencial y analizando que factores afectan la efectividad del sistema

### 7.4.1. Validación en la entrada de una residencia.

Se instaló el sistema de videovigilancia en la entrada de una residencia familiar, con la finalidad de evaluar el desempeño de los algoritmos del sistema, la residencia contaba con amplio garaje (Ver figura 51) a través del cual ingresaban los residentes, debido a que era un lugar cerrado las condiciones externas se minimizaban notablemente.

*Figura 51. Residencia en la cual se validó el funcionamiento Fuente: Autor*

Las pruebas de validacion consistian en observar el comportamiento del sistema cuando ingresaba a la residencia una, dos, tres y cuatro personas, personas que ingresaban en 2 direcciones diferentes.

En la figura 52 se observa el trayecto realizado por la persona, ingresando en la izquierda y finalizando en la derecha.

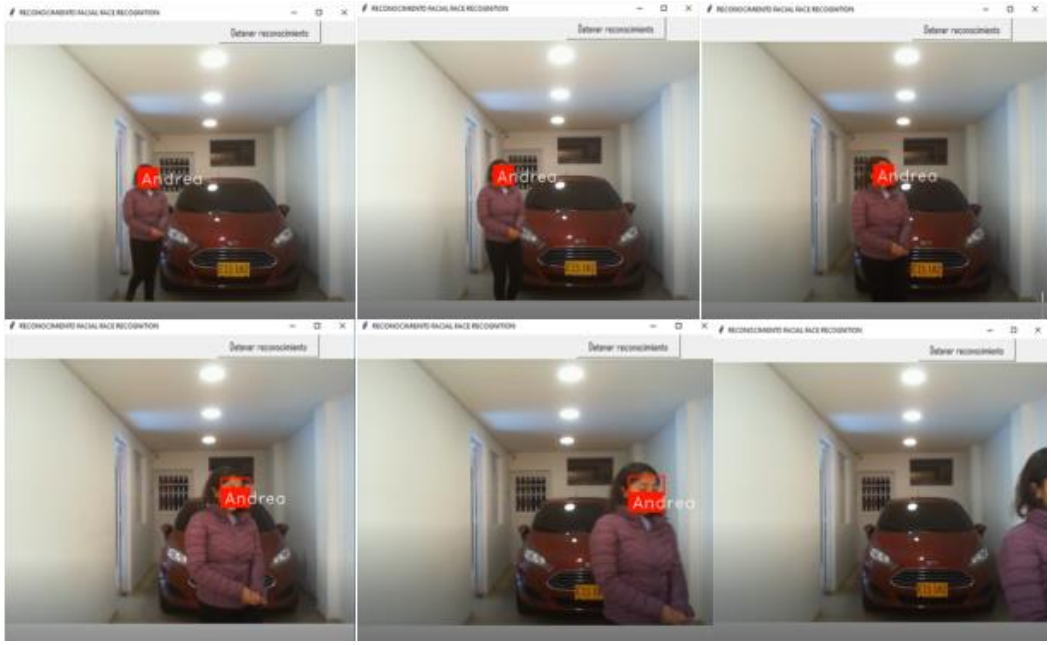


*Figura 52. Validación algoritmo de Face-Recognition reconociendo la identidad de una persona en un lugar residencial recorrido izquierda a derecha Fuente: Autor*

En la figura 52 se puede observar que el algotritmo Face recognition esta reconociendo correctamente la identidad de la persona en este caso "Andrea" durante un recorrido de 4 metros con orientacion de izquierda a derecha.

En la figura 53 se observa el trayecto realizado por la persona, ingresando desde la derecha y finalizando en la izquierda.

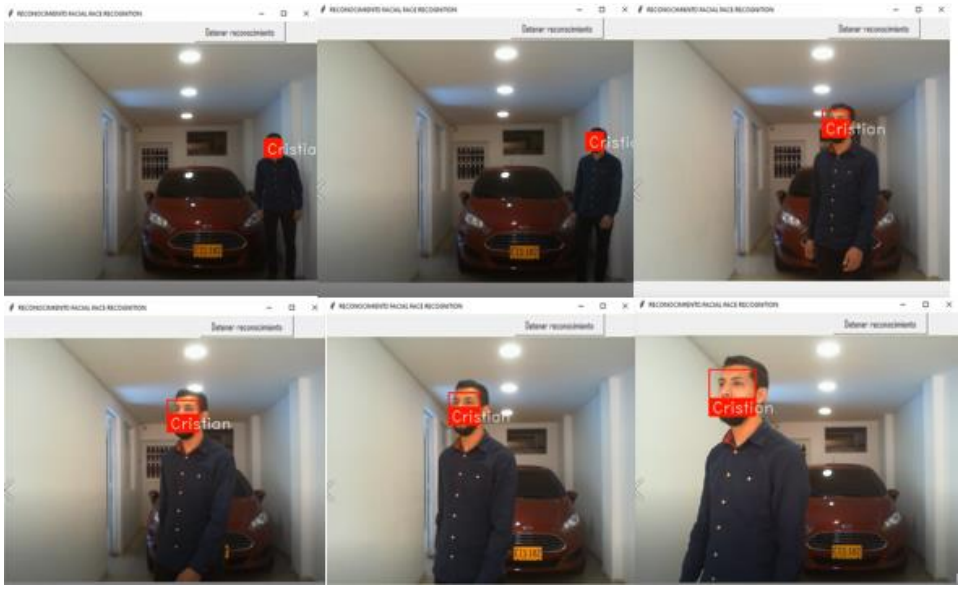


*Figura 53. Validación algoritmo de Face-Recognition reconociendo la identidad de una persona en un lugar residencial recorrido derecha a izquierda Fuente: Autor*

En la figura 53 se puede observar como el algoritmo Face-Recognition reconoce correctamente a la persona, en este caso “Cristian”, durante un recorrido de aproximadamente 4 metros en dirección derecha a izquierda.

Posterior a estas pruebas que evaluaban la distancia y el ángulo de detección se realizó una prueba para verificar la capacidad del sistema de identificar a dos personas.

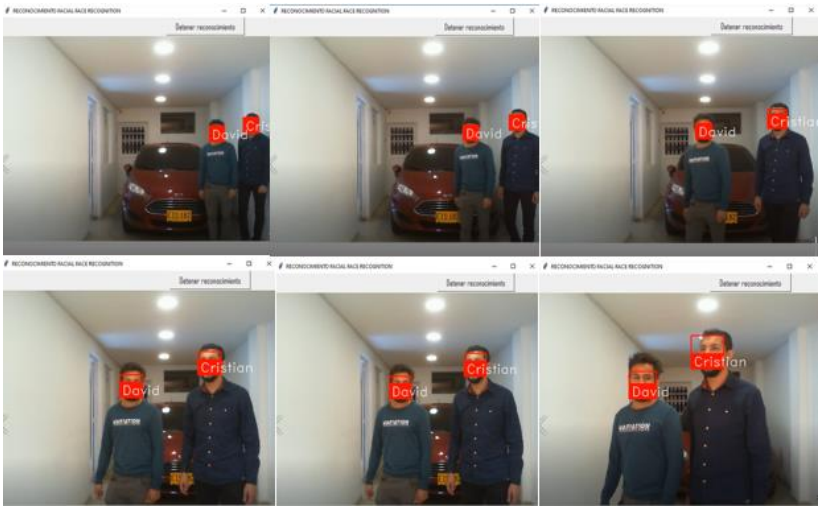


*Figura 54. Verificación del reconocimiento de 2 personas mediante la librería Face-Recognition en residencia Fuente: Autor*

En la figura 54, se puede observar que este algoritmo detecto e identifico correctamente a las 2 personas durante el recorrido que realizaron. Posterior a eso se repitió la prueba, pero esta vez 3 persona realizarían el recorrido.

*Figura 55. Verificación del reconocimiento de 3 personas mediante la librería Face-Recognition en residencia. Fuente: Autor*

El algoritmo detecto y reconoció correctamente a las personas que estaban ingresando a la residencia, en la figura 55 se puede observar la etiqueta que el sistema coloco a cada uno a medida que realizan el recorrido.

Como última prueba de la librería Face-Recognition en la residencia se realizó nuevamente el recorrido, pero esta vez lo realizaron 4 personas, de estas 4 personas 3 están registradas como residentes de la vivienda las cuales son “David”, “Cristian” Y “Andrea” y a la persona restante el sistema debe asignarle la etiqueta de “Unknown”.

*Figura 56. Verificación del reconocimiento de 4 personas mediante la librería Face-Recognition en residencia Fuente: Autor*

Como se puede observar en la figura 56 el algoritmo reconoció correctamente a las 3 personas registradas como residentes de la vivienda, y a la persona que no estaba registra le asigno correctamente la etiqueta.

Posterior a eso se realizaron pruebas en la residencia, pero mediante el algoritmo de la librería OpenCV, en estas pruebas se validó el funcionamiento del algoritmo con relación al ángulo, distancia y cantidad de personas reconocidas.

Inicialmente se realizó una prueba en la cual persona caminaba al costado izquierdo de la cámara.

*Figura 57. Prueba de ángulo librería OpenCV en zona residencial Fuente: Autor*

En la figura 57 se puede observar como el algoritmo OpenCV no reconoce correctamente el rostro de una persona al tener un pequeño ángulo de giro de la cara.

Debido a estos resultados se repitió la prueba, aunque ahora el usuario se ubicó de forma frontal a la cámara.

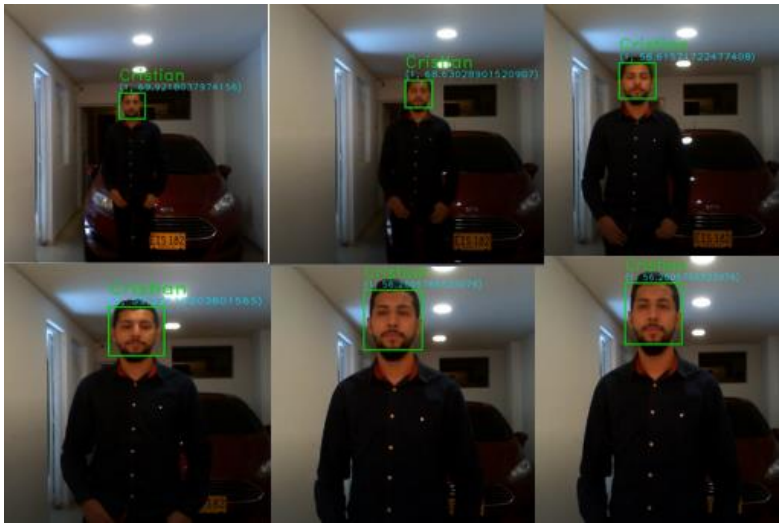

*Figura 58. Prueba de reconocimiento sin ángulo de giro. Fuente: Autor*

En la figura 58, se observa cómo se comportó el sistema en la prueba realizada, el sistema de reconocimiento facial tuvo buenos resultados debido a que durante la trayectoria realizada por "Cristian" el sistema lo reconoció correctamente.

Se realizó una prueba con la finalidad de detectar y reconocer dos personas mediante la librería OpenCV, debido a los resultados de las pruebas anteriores, los usuarios realizaron el recorrido de forma frontal a la cámara.

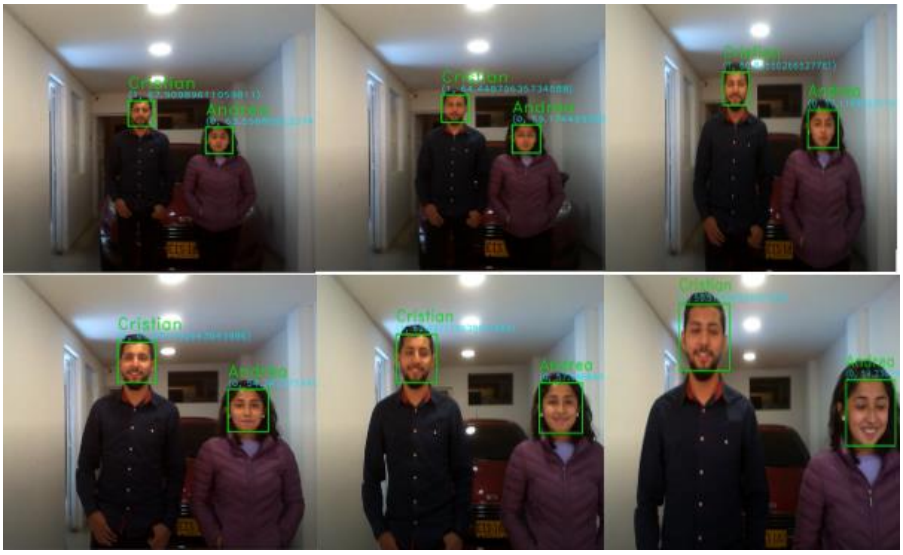


*Figura 59. Prueba de reconocimiento facial de dos personas en una residencia. Fuente: Autor*

En la figura 59 se observa como el sistema realizo el reconocimiento facial de forma satisfactoria reconociendo de forma correcta a “Cristian” y a “Andrea”.

Para la siguiente prueba tres personas realizaron el recorrido, y el sistema debería reconocer a “David”, “Cristian” y “Andrea”.

*Figura 60. Prueba de reconocimiento facial de tres personas en una residencia. Fuente: Autor*

Como se muestra en la figura 60 el sistema en esta prueba no trabajo correctamente, pues, aunque a “Andrea” y “Cristian” fueron identificados satisfactoriamente en gran parte del recorrido, el tercer usuario “David” solo fue reconocido de forma correcta en la última parte del recorrido, en el recorrido inicial no fue reconocido o su reconocimiento no fue correcto.

La última prueba que se realizó a esta librería en una zona residencial consistió en que 4 personas realizarían un recorrido de forma frontal a la cámara de estas 4 personas, 3 eran residentes de la vivienda y 1 era un desconocido.

*Figura 61. Prueba de reconocimiento facial de tres personas en una residencia y un desconocido. Fuente: Autor*

Como se muestra en la figura 61 el comportamiento del sistema de videovigilancia residencial no fue el esperado, "Cristian" y "Andrea" fueron reconocidos de forma correcta durante la mayoría del trayecto, pero "David" en una parte del recorrido fue reconocido como Cristian y la cuarta persona que no se registró como residente de la vivienda el sistema le dio la etiqueta de "Desconocido" de forma correcta durante todo el trayecto.

### 7.4.2. Validación en un centro comercial.

Para la realizar la segunda validación se instaló el sistema de videovigilancia residencial en un centro comercial, se desarrollaron 4 pruebas de reconocimiento facial, en las cuales varían el número de personas que el sistema debería reconocer.

Para la primera prueba, una persona que estaba registrada en el sistema se colocaría en frente de la cámara esperando ser reconocida.

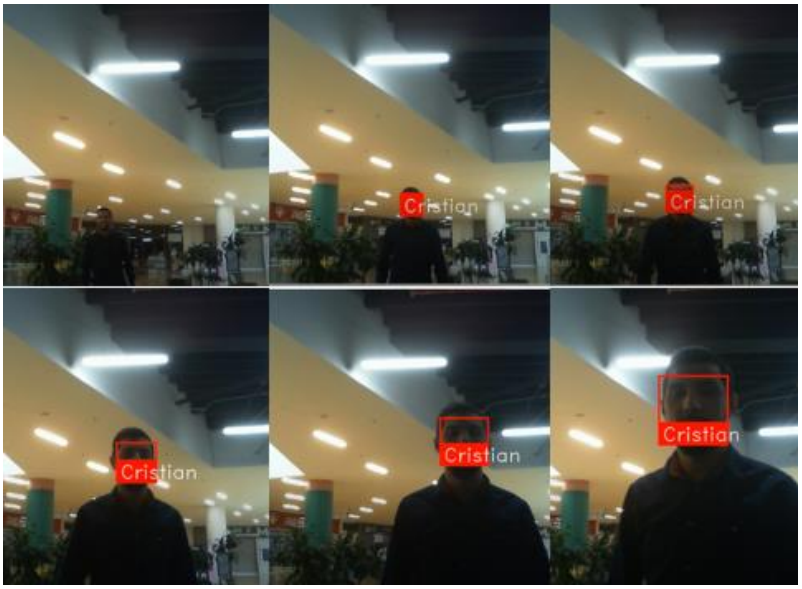


*Figura 62. Validación del algoritmo de Face-Recognition en un centro comercial reconociendo 1 persona Fuente: Autor*

En la figura 62 se pueden observar los resultados de esta primera prueba, el sistema detecto y reconoció correctamente a la persona en pantalla, le asigno la etiqueta "Cristian", aunque el reconocimiento fue correcto, en ocasiones el sistema no detectaba que enfrente de él se encontraba un rostro.

Para la segunda prueba, dos personas registradas se pararon enfrente del sistema, realizaron un recorrido de forma frontal a la cámara, acercándose.

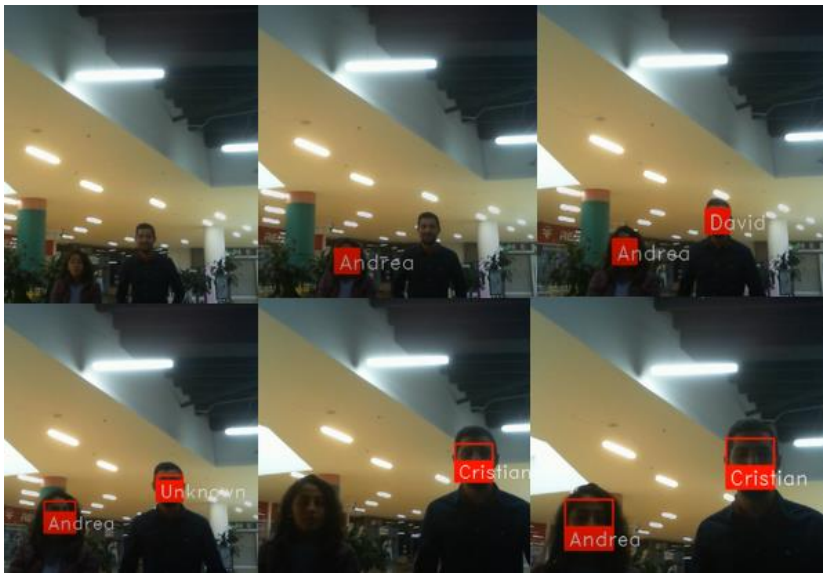


*Figura 63. Validación del algoritmo de Face-Recognition en un centro comercial reconociendo 2 personas Fuente: Autor*

Como se muestra en la figura 63 el sistema no funciono de la forma esperada, “Andrea” fue reconocida de forma correcta, aunque en tramos del recorrido su rostro no fue detectado para un posterior reconocimiento, en el caso de “Cristian” solo fue reconocido cuando se encontraba cerca de la cámara, en algunos tramos el sistema no le detecto el rostro, no lo reconoció o reconoció de forma errónea.

Para la tercera prueba se situaron tres personas registradas en el sistema de forma frontal a la cámara.

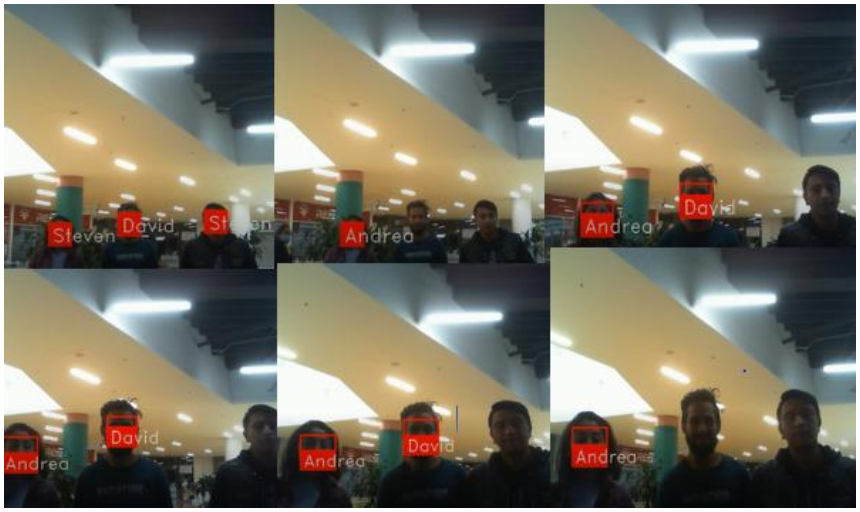


*Figura 64. Validación del algoritmo de Face-Recognition en un centro comercial reconociendo 3 personas. Fuente: Autor*

En la figura 64 se muestra como el sistema no se comportó de la forma esperada, a “Andrea el sistema la reconoció de forma correcta casi en todo el tramo recorrido, excepto comenzando, pues la reconoció como “Steven”, a “David“ el sistema en tramos lo reconoció de forma correcta o no lo reconoció  y a Steven solo lo reconoció comenzando el recorrido, en el resto del recorrido no le reconoció el rostro.

Para la última prueba realizada sobre este algoritmo, 4 personas todas registradas en el sistema realizaron un recorrido hacia la cámara.

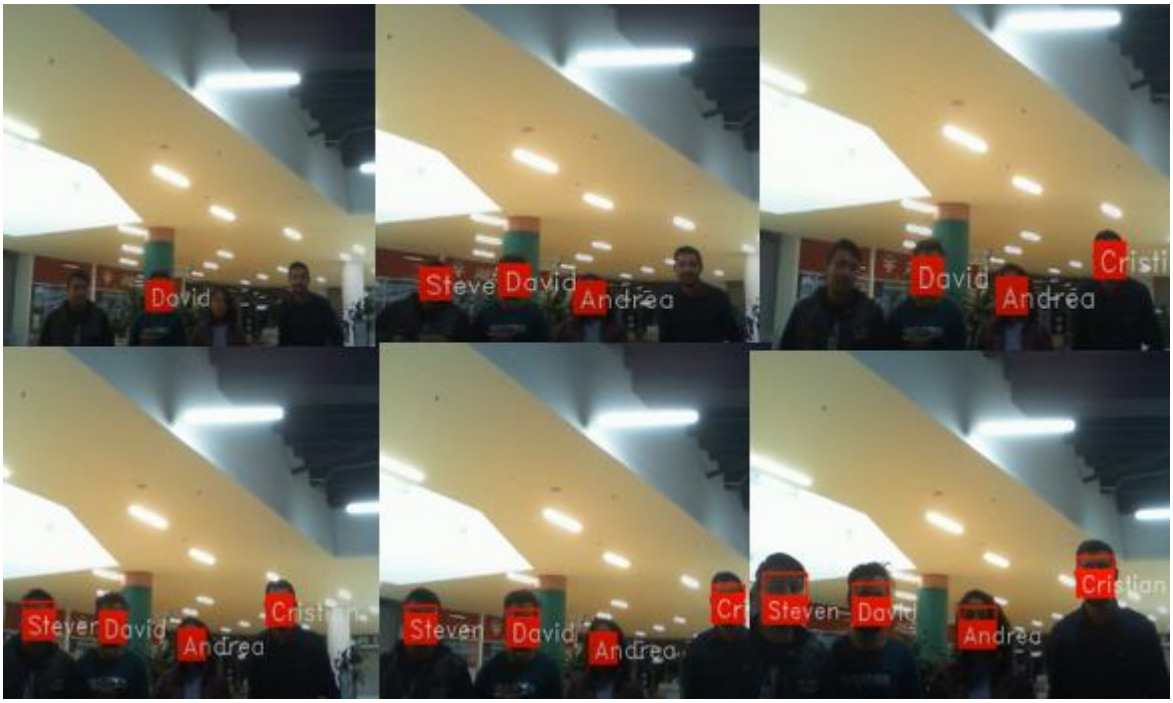

*Figura 65. Validación del algoritmo de Face-Recognition en un centro comercial reconociendo 4 personas. Fuente: Autor*

En la figura 65, se observa como el sistema mejoro su rendimiento, aunque presento ciertas fallas, en el comienzo del recorrido, el sistema no detecto el rostro de “Steven” y “Cristian”, sin embargo despues de la mitad del recorrido hacia la camara, el sistema funciono de forma correcta detectando y reconiendo las 4 personas que estaban situadas delante de el.

Para validar elfuncionamiento del algoritmo de OpenCV en el centro comercial, se realizaron 4 pruebas, en las cuales se observa el comportamiento del sistema en lugares con un ambiente mas variable.

La primera prueba consiste en observar el comportamiento del sistema cuando una persona resgistrada en el sistema , se ubica frente a el.

*Figura 66. Validación del algoritmo de OpenCV en un centro comercial reconociendo 1 persona. Fuente: Autor*

De acuerdo a lo observado en la figura 66 el sistema solo reconoce a “David” cuando este se enuentra ubicado cerca a la cámara, cuando se encuentra lejos el sistema no detecta su rostros o no encuentra la suficiente compatibilidad en la informacion de su rostro con relacion al modelo cargado.

La segunda prueba consiste en evaluar como se comporta el sistema, cuando dos usuarios previamente registrados se ubican en frente de la cámara.

*Figura 67. Validación del algoritmo de OpenCV en un centro comercial reconociendo 2 personas Fuente: Autor*

En la figura 67, se evidencia el funcionamiento del sistema, el cual comporto de manera similar en las dos pruebas, cuando “ Andrea” y “David” se ubican frente al sistema, el sistema solo los reconoce de forma correcta cuando se ubican cerca, por el contrario, cuando se encuentra a distancia mediana y lejana del sistema, el no reconoce su rostro o no detecta el rostro.

La tercera prueba consistio en ubicar a tres personas que se han registrado previamente en el sistema, y observar si el sistema los reconoce de forma correcta.

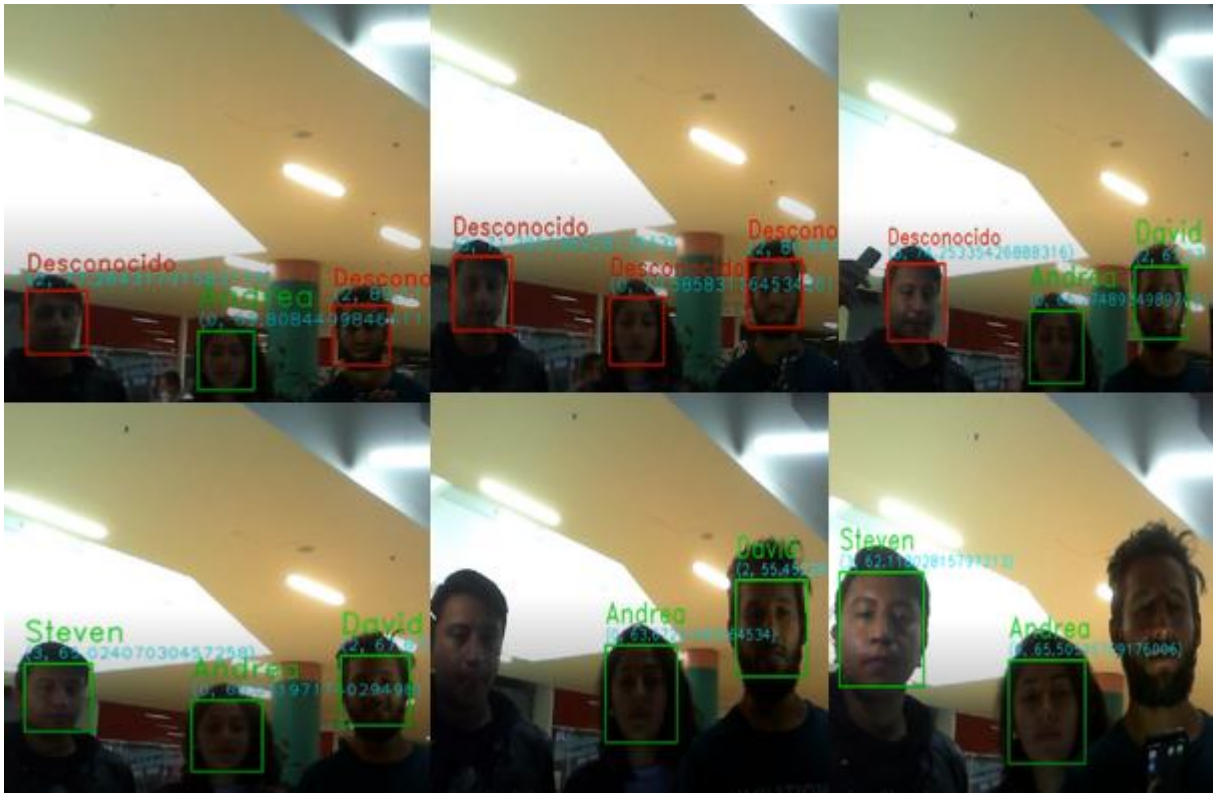


*Figura 68. Validación del algoritmo de OpenCV en un centro comercial reconociendo 3 personas. Fuente: Autor*

Como se observa en la figura 68, cuando los usuarios se encuentra a una distancia relativamente lejana el sistema no reconoce sus rostros , a medida que ellos se acercan al dispostivo de captura el sistema los reconoce mas facilmente, aunque en ocaciones no detecto el rostro de “Steven” y “David”.

La ultima prueba para validar el funcionamietno de este algoritmo en el centro comercial, consitio en ubicar a 4 personas frente al dispositivo de adquisicion de imagen, y observar si el sitema los reconoce correctamente.

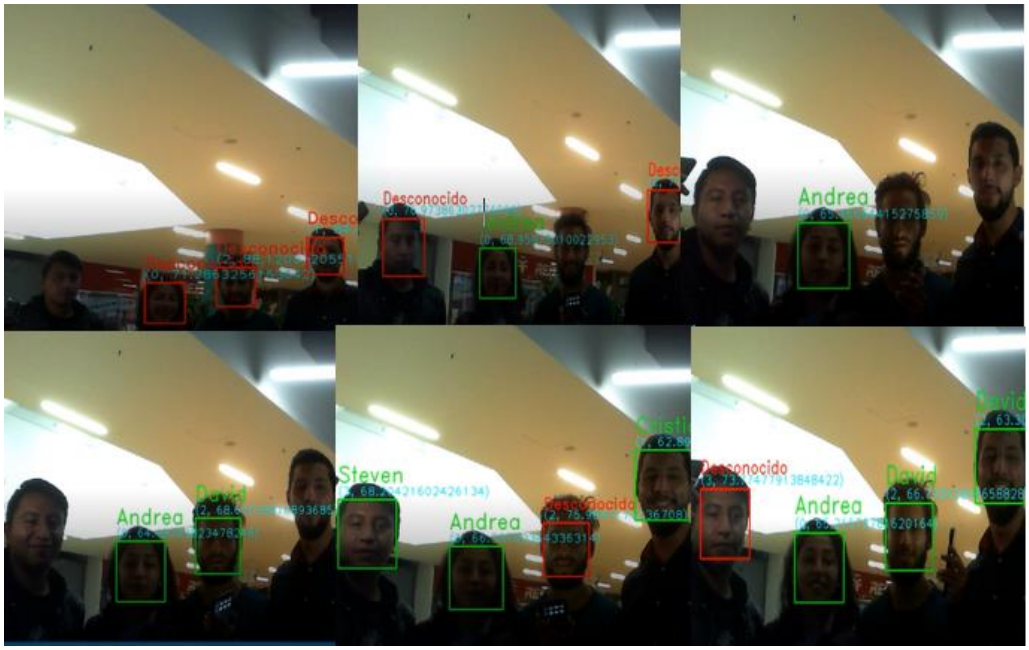

*Figura 69. Validación del algoritmo de OpenCV en un centro comercial reconociendo 4 personas. Fuente: Autor*

De acuerdo a la figura 69 cuando los usuarios se encuentran  a una distancia lejana de la camara el sistema no reconoce los rosotros o no los detecta, pero a medida que se acercan al dispositvo de adquisison de imagen el sistema los reconce de mejor forma, a “Andrea” el sistema la reconoció de forma optima en casi todo el recorrido hacia la camara,  a “Steven” , “ David” y “Cristian” el sistema los reconoció solo en un corto tramo del recorrido.

# 8. Conclusiones y Recomendaciones

Por medio de técnicas de procesamiento digital de imágenes se logró realizar un sistema de videovigilancia residencial, con algoritmos de detección, extracción e identificación de rostros, para identificar las personas que ingresan a una zona residencial bajo condiciones específicas.

En los dos algoritmos de reconocimiento facial implementados en el desarrollo del sistema de video vigilancia, un factor común presente, es la relación que existe entre la resolución y la distancia máxima de detección, si es verdad que a mayor resolución mayor distancia, se evidencia que la curva es logarítmica, después de una resolución de 800x600 por más resolución que se logre tener en la cámara el aumento en la distancia máxima está representado en muy pocos centímetros.

Implementar sistemas de video vigilancia más robustos que permitan ampliar el rango de distancias de detección y a su vez mantener tiempos, incrementa los costos, al necesitar hardware de mejores especificaciones, tanto en etapa de procesamiento como en etapa de adquisición de imágenes, lo ideal es tener un hardware destinado únicamente al procesamiento del sistema ,ya que la librería Face-Recognition al usar técnicas de reconocimiento facial basada en CNN tiene buenos resultados pero un alto coste computacional.

La resolución de la cámara del sistema de video vigilancia es un factor importante para seleccionar, pues en caso de que la resolución sea demasiado baja, la distancia de detección es demasiado corta y si se desea detectar y reconocer 3 o 4 personas la resolución sobre los rostros no será la mínima requerida, por lo tanto, no detectará los rostros.

Al realizar la validación final del sistema en un ambiente no residencial, como lo es un centro comercial, se evidencia que la iluminación presente en el recinto es uno de los factores más importantes a la hora de implementar estas técnicas de reconocimiento. La librería de OpenCV se ve muy afectada, a tal punto de no reconocer ninguno de los rostros presentes ya que la cámara estaba a contraluz. Por otra parte, la librería de Face-Recognition bajó su distancia máxima de detección a casi la mitad.

Mediante la investigación realizada se pudo determinar cuáles son los algoritmos desarrollados en Python para aplicaciones relacionadas al reconocimiento de rostros, encontrando que los algoritmos más usados son los de las librerías Face-Recognition y OpenCV debido a su fácil implementación y alta efectividad en condiciones controladas.

De los dos algoritmos de reconocimiento facial implementados en el sistema de video vigilancia, el algoritmo hecho con la librería perteneciente a Face-Recognition presenta mejor comportamiento en cuanto a mayor distancia de operación y detección facial en un mayor ángulo de giro del rostro, sin embargo, el algoritmo hecho con la librería perteneciente a OpenCV presenta menos perdida de información al ser un algoritmo que necesita menor procesamiento, en el cual es viable implementar mayores resoluciones de cámara.

El desarrollo de la interfaz gráfica de usuario le da más facilidad al usuario para hacer uso del sistema de videovigilancia, pues mediante la GUI, el usuario puede interactuar de una mejor

manera con el sistema, haciendo selección de la configuración de reconocimiento que más se ajuste a las exigencias o necesidades que presenta el ambiente en donde requiere realizar la vigilancia

## 9. Trabajo futuro

Hay optimizar los cálculos y modelos de la librería Face-Recognition ya que esta si se presta para realizar reconocimiento facial con un muy buen porcentaje de acierto, pero el nivel de procesamiento es demasiado alto y se pierden fotogramas.

Se deben mezclar las técnicas de reconocimiento facial en un mismo sistema para permitir obtener un mayor asertividad en el reconocimiento de personas, es decir, una librería apoyando la otra para que el error se disminuya.

## 10. Referencias


Alcántara Montiel, C. V., Pedraza Ortega, J. C., Ramos Arreguín, J. M., Gorrostieta Hurtado, E., Tovar Arriaga, S., & Vargas Soto, J. E. (2019). Detección efectiva de rostros en imágenes utilizando descriptores basados en HOG. *Research in Computing Science*, 371–385. https://doi.org/10.13053/rcs-148-8-28

Angel, B. (2021). *Hurto a residencias* . Policía Nacional de Colombia. https://www.policia.gov.co/delitos-de-impacto/hurto-residencias

Arellano, L. R., & Cortijo, R. E. (2020). Sistema de video vigilancia mediante visión por computador para el centro de educación inicial n°1 del ministerio de educación [Universidad Tecnológica Israel]. In *Journal of Chemical Information and Modeling* (Issue 1). https://doi.org/10.1016/j.tmaid.2020.101607%0Ahttps://doi.org/10.1016/j.ijsu.2020.02.034%0Ahttps://onlinelibrary.wiley.com/doi/abs/10.1111/cjag.12228%0Ahttps://doi.org/10.1016/j.ssci.2020.104773%0Ahttps://doi.org/10.1016/j.jinf.2020.04.011%0Ahttps://doi.o

Bagnato, J. (2020). Aprende Machine Learning en Español Teoría + Práctica Python. In *LeanPub* (1ra ed.). Leanpub.

Barroso, C. (2014). *Implementación del algoritmo de detección de caras de viola y jones sobre una FPGA*. Universidad Carlos III de Madrid.

Caballero, E. R. (2017). *Aplicación práctica de la visión artificial para el reconocimiento de rostros en una imagen, utilizando redes neuronales y algoritmos de reconocimiento de objetos de la biblioteca OpenCV*. Universidad distrital Francisco José de Caldas.

Castaño, D. L., & Sierra, J. D. (2019). Sistema de reconocimiento facial para control de acceso a viviendas. In *Αγαη*. Universidad Catolica de Colombia.

Costa Mari, D. (2020). *Análisis De Un Sistema De Reconocimiento Facial a Partir De Una Base De Datos Realizado Mediante Python* [Universitat Politécnica de Catalunya]. http://hdl.handle.net/2117/331277

Cowley, J. (2018). *Redes neuronales convolucionales* . IBM Developer. https://developer.ibm.com/es/articles/cc-convolutional-neural-network-vision-recognition/

Encalada, J. C. (2015). *Maestría en control y al reconocimiento de rostros en vídeo vigilancia del Sis-Ecu-911*. Universidad politécnica salesiana.

Esparza, C., Tarazona, C., Sanabria, E., & Velazco, D. (2015). Reconocimiento facial basado en eigenfaces, LBPH y fisherfaces en la Beagleboard-XM. *Revista Tecnológica Colombiana de Avanzada*, *10*(20).

García, I., & Caranqui, V. (2015). La visión artificial y los campos de aplicación. *Tierra Infinita*, *1*, 94–103. http://revistasdigitales.upec.edu.ec/index.php/tierrainfinita/article/view/76

Geitgey, A. (2016, July). *Machine Learning is Fun!* . Medium. https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Gonzáles, C. E., & Zamara, D. F. (2014). *Aplicación de sistemas de visión por computador (VIR/NIR) para la inspección visual de uchuvas* [Pontificia universidad Javeriana]. http://journal.stainkudus.ac.id/index.php/equilibrium/article/view/1268/1127

Gracia, L. M. (2013). *¿Qué es OpenCV?* . Un Poco de Java. https://unpocodejava.com/2013/10/09/que-es-opencv/

Hinestroza Ramirez, D. (2018). *El machine learning a través de los tiempos, y los aportes a la humanidad* (Vol. 1, Issue 1) [Universidad libre seccional Pereira]. http://www.ncbi.nlm.nih.gov/pubmed/7556065%0Ahttp://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC394507%0Ahttp://dx.doi.org/10.1016/j.humpath.2017.05.005%0A

https://doi.org/10.1007/s00401-018-1825-z%0Ahttp://www.ncbi.nlm.nih.gov/pubmed/27157931

I&C, S. intendencia. (2016). Protección de datos personales en sistemas de videovigilancia. *SuperIntendencia de Industria y Comercio*.

Juárez Álvarez, U. (2018). *Reconocimiento de objetos y rostros con técnicas de visión por computadora e inteligencia artificial.*

Lu, X., Lu, X., & Lansing, E. (2003). Image analysis for face recognition. *Dept. of Computer Science & Engineering Michigan*, *1*, 7. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.4026&rep=rep1&type=pdf

Marin, R. (2020). *OpenCV, Instalación en Python y ejemplos básicos*. Revista Digital. https://revistadigital.inesem.es/informatica-y-tics/opencv/

Meneses, M. D. (2021). *Diseño y desarrollo de un sistema de video vigilancia basado en dispositivos embebidos, técnicas de visión artificial y algoritmos inteligentes*. Universidad politécnica Salesiana.

Meritxell, P. (2011, April 6). *¿Las videocámaras reducen la delincuencia?* La Vanguardia. https://www.lavanguardia.com/vida/20110406/54136649079/las-videocamaras-reducen-la-delincuencia.html

Morcote, J. D. (2020). Reconocimiento facial en tiempo real orientado a video llamadas o live stream para autenticar identidades durante una audiencia legal. In *Orphanet Journal of Rare Diseases* (Vol. 1, Issue 1). Universidad Santo Tomás Seccional Tunja.

Pajares, G. (2008). *Visión por Computador, imágenes digitales y aplicaciones* (2da ed.). Alfa y Omega.

Parada, A. (2019). Desarrollo de aplicación para el control y gestión de almuerzos en casinos empresariales utilizando sistema de reconocimiento facial. *Universidad Distrital Fráncico José de Caldas*, *1*(1), 1–6.

Parra, E. C. (2018). *Prototipo para el control de ingreso de personal por reconocimiento facial*. Universidad distrital Francisco José de Caldas.

Pedroza, R. E. (2019). Diseño e implementación de un sistema de biometría facial para la búsqueda e identificación de personas desaparecidas en Colombia. In *Αγαη*. Universidad de Cartagena.

Pereyra, P. A. (2018). *Reconocimiento Facial Mediante Imágenes Estereoscópicas Para Control de Ingreso Tesis de Grado de Ingeniería en Informática*. Universidad de Buenos Aires.

Plazas, D. C. (2015). *Reconocimiento de imágenes faciales orientado a controles de acceso y sistemas de seguridad* (Vol. 1). Universidad Distrital Fráncico José de Caldas.

Porras Aldana, J. M. (2018). Diseño de un sistema de reconocimiento facial como medio de control de acceso biométrico mediado por técnicas de inteligencia artificial como herramienta base de seguridad del CEAD Ibagué [Universidad Nacional abierta y a distancia]. In *Director*. http://awsassets.wwfnz.panda.org/downloads/earth_summit_2012_v3.pdf%0Ahttp://hdl.handle.net/10239/131%0Ahttps://www.uam.es/gruposinv/meva/publicaciones jesus/capitulos_espanyol_jesus/2005_motivacion para el aprendizaje Perspectiva alumnos.pdf%0Ahttps://ww

Rubal, M. (2018, April 13). *Los orígenes de la fotografía*. La Vanguardia. https://www.lavanguardia.com/vida/junior-report/20180411/442459480247/historia-fotografia-camara-kodak-polaroid-digital-daguerrotipo.html

Saldarriaga Quintero, M. (2020). *Reconocimiento de sujetos y caracterización de sus emociones a partir de sus expresiones faciales usando algoritmos de reconocimiento facial*. Universidad de Antioquia.

Sanches, C. R. (2017). *Análisis de dos algoritmos para detección de rostros e implementación de uno de ellos utilizando en analítica de video para sistemas de videovigilancia*. Escuela Politécnica Nacional.

Sánchez Calle, Á. (2005). *Aplicaciones de la visión artificial y la biometría informática* (1a ed.). S.L. - Dykinson.

Serrano. (2016). *Partes de tu cámara: Fundamentos de fotografía 2*. https://www.curso-fotografia-digital.com/blogs-de-fotografia/tips-y-consejos/128-partes-de-tu-camara.html

Torres, D. (1996). Procesamiento digital de imágenes. *Perfiles Educativos*, *1*.

Valero, M. (2012). *Sistema de vigilancia y control de personas basado en la detección y reconocimiento de caras mediante visión por computador*. Universidad Zaragoza.

Villalon de la Vega, D. E. (2012). *Diseño e implementación de una plataforma de software para reconocimiento facial en video*. Universidad de Chile.

Visus, A. (2020, October). *¿Para qué sirve Python? Razones para utilizarlo* . ESIC. https://www.esic.edu/rethink/tecnologia/para-que-sirve-python

Yanez, J. C. (2018). *Analíticas de video inteligentes, la tendencia de la videovigilancia* . Revista Innovación Seguridad. https://revistainnovacion.com/nota/10211/analiticas_de_video_inteligentes_la_tendencia_de_la_videovigilancia/

Yang, M. H., Kriegman, D. J., & Ahuja, N. (2002). Detecting Faces In Image : A Survey - Presentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(1), 1–25. https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.210.3372

# 11.Anexos.

## Anexo A. Código implementado para el sistema de reconocimiento

```
import tkinter as tk
from tkinter import messagebox
from tkinter import ttk
from PIL import Image
from PIL import ImageTk
import cv2
import os
import numpy as np
import face_recognition
import shutil



def iniciarCV():
    def seleccion_camara(combo_camaras):
        cam = 0
        cam = combo_camaras.get()

        if cam == "Camara integrada":
            camara = 0
        elif cam == "Camara externa 1":
            camara = 1
        elif cam == "Camara externa 2":
            camara = 2
        elif cam == "Camara externa 3":
            camara = 3
        elif cam == "Camara externa 4":
            camara = 4

        return camara

    def seleccion_resolucion(combo_resoluciones):
        ancho = 1280
        alto = 800
        res = combo_resoluciones.get()
        resolución = res.split('x')
        ancho = resolución[0]
        alto = resolución[1]
        return ancho, alto
```

```
def ver_videoCV():

    global cap
    global frame
    if cap is not None:
        ret, frame = cap.read()
        if ret == True:

            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            img = ImageTk.PhotoImage(Image.fromarray(frame))
            lblVideo.configure(image=img)
            lblVideo.image = img
            lblVideo.after(10, ver_videoCV)
        else:
            lblVideo.image = ""
            cap.release()

def capturar_fotos():

    global cap

    def entrenar_modelo():
        global cap
        peopleList = os.listdir(dataPath)
        print('lista de personas : ', peopleList)

        labels = []
        facesData = []
        label = 0

        for nameDir in peopleList:
            personPath = dataPath + '/' + nameDir


            for fileName in os.listdir(personPath):

                labels.append(label)
                facesData.append(cv2.imread(personPath + '/' + fileName, 0))

            label = label + 1


        face_recognizer = cv2.face.LBPHFaceRecognizer_create()

        face_recognizer.train(facesData, np.array(labels))
```

```
            face_recognizer.write('modeloLBPHFace.xml')
            print("Modelo almacenado...")

        if txtnombreFR.get(1.0, 'end-1c') != "":

            nombres_usuarios = os.listdir('/home/linux/Escritorio/Tesis/usuariosC
V')
            nombre_ingresado = txtnombreFR.get(1.0, 'end-1c')
            if len(nombres_usuarios)==0:
                existe_usuario=False
            else:

                for nombre_usuario in nombres_usuarios:
                    if nombre_ingresado.lower() == nombre_usuario.lower():
                        existe_usuario = True
                        break
                    else:
                        existe_usuario = False

            if existe_usuario:
                messagebox.showinfo("¡Advertencia!", "El usuario ingresado ya se
encuentra registrado")
            else:
                messagebox.showinfo("¡Advertencia!",
                                    "No se retire de la camara hasta que se le in
dique. \nPara lograr mejores resultados, mantenga el rostro a una distancia de 50
 a 70 cm de la cámara y evite ambientes que sean demasiado brillates o demasido t
enues durante la inscripción")


                personName = txtnombreFR.get(1.0, 'end-1c')
                dataPath = '/home/linux/Escritorio/Tesis/usuariosCV'
                personPath = dataPath + '/' + personName

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

                cap = cv2.VideoCapture(camara)
                cap.set(3, int(resolucion[0]))
                cap.set(4, int(resolucion[1]))
                faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades + 'haar
cascade_frontalface_default.xml')
                count = 0
```

```
                cap = cv2.VideoCapture(camara)
                cap.set(3, int(resolucion[0]))
                cap.set(4, int(resolucion[1]))

                while True:
                    ret, frame = cap.read()
                    if ret == False:
                        break

                    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                    auxFrame = frame.copy()

                    faces = faceClassif.detectMultiScale(gray, 1.3, 5)

                    for (x, y, w, h) in faces:
                        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0),
 2)
                        rostro = auxFrame[y:y + h, x:x + w]
                        cv2.imwrite(personPath + '/rotro_{}.jpg'.format(count), r
ostro)
                        count = count + 1

                    cv2.imshow('Capturando rostros...', frame)

                    k = cv2.waitKey(1)
                    if k == 27 or count >= 300:
                        break

                cv2.destroyAllWindows()
                dglpregunta1 = messagebox.askquestion("Guardar imagen",
                                                      "¡Ya se puede retirar de la
 camara! \n ¿Desea Guardar usuario?")

                if dglpregunta1 == "no":
                    shutil.rmtree(personPath)

                dglpregunta2 = messagebox.askquestion("Nuevo usuario", "¿Desea ag
regar otro usuario?"

                if dglpregunta2 == "yes":
                    cap.release()
                    txtnombreFR.delete("1.0", "end")
                    cap = cv2.VideoCapture(camara)
                    cap.set(3, int(resolucion[0]))
                    cap.set(4, int(resolucion[1]))
```

```
                else:
                    cap.release()
                    entrenar_modelo()
                    messagebox.showinfo("Modelo entrenado", "Modelo entrenado y a
lmacenado")
                    usuarioCV.destroy()

        else:
            messagebox.showinfo("¡Advertencia!", "Debe ingresar un nombre de usua
rio")

    def usuarios():
        users = os.listdir('/home/linux/Escritorio/Tesis/usuariosCV')
        return users

    def delete_usuario():
        global combo_usuario
        global users

        def entrenar_modelo():
            dataPath = '/home/linux/Escritorio/Tesis/usuariosCV'
            global cap
            peopleList = os.listdir(dataPath)
            print('lista de personas : ', peopleList)

            labels = []
            facesData = []
            label = 0

            for nameDir in peopleList:
                personPath = dataPath + '/' + nameDir


                for fileName in os.listdir(personPath):

                    labels.append(label)
                    facesData.append(cv2.imread(personPath + '/' + fileName, 0))

                label = label + 1

            face_recognizer = cv2.face.LBPHFaceRecognizer_create()


            face_recognizer.train(facesData, np.array(labels))
```

```
            face_recognizer.write('modeloLBPHFace.xml')
            print("Modelo almacenado...")

        nombre = combo_usuario.get()
        if nombre != "":
            usuario_eliminar = '/home/linux/Escritorio/Tesis/usuariosCV' + '/' +
nombre
            shutil.rmtree(usuario_eliminar)

            messagebox.showinfo("¡Información!", "Usuario eliminado")

            if len(os.listdir('/home/linux/Escritorio/Tesis/usuariosCV'))==0:
                pass
            else:
                entrenar_modelo()
                messagebox.showinfo("Modelo entrenado", "Modelo entrenado y almac
enado")
        else:
            messagebox.showinfo("¡Advertencia!", "Debe seleccionar un usuario par
a eliminar")


    global cap
    global combo_camaras
    global combo_resoluciones
    global combo_usuario

    camara = seleccion_camara(combo_camaras)
    resolucion = seleccion_resolucion(combo_resoluciones)
    users = usuarios()
    if cap is not None:
        cap.release()
    cap = cv2.VideoCapture(camara)
    cap.set(3, int(resolucion[0]))
    cap.set(4, int(resolucion[1]))

    if int(resolucion[0]) == 1280 and int(resolucion[1]) == 800:
        tamaño = '1290x820+0+0'
        tamaño_letra = 6
        ancho = 30
        ancho_btn = 20
    elif int(resolucion[0]) == 800 and int(resolucion[1]) == 600:
        tamaño = '810x700+10+10'
        tamaño_letra = 7
        ancho = 25
```

```
        ancho_btn = 18
    elif int(resolucion[0]) == 352 and int(resolucion[1]) == 288:
        tamaño = '360x370+10+10'
        tamaño_letra = 7
        ancho = 13
        ancho_btn = 16

    usuarioCV = tk.Toplevel(raiz)
    usuarioCV.geometry(tamaño)
    usuarioCV.title("AÑADIR USUARIO A OPEN CV")
    usuarioCV.grab_set()


    lblinfo1 = tk.Label(usuarioCV, text="Ingrese nombre de usuario:", font=("Agen
cy FB", tamaño_letra))
    lblinfo1.grid(column=0, row=0, sticky="e")
    txtnombreFR = tk.Text(usuarioCV, width=ancho, height=1, font=("Agency FB", ta
maño_letra), padx=2, pady=2,
                          selectbackground="gray")
    txtnombreFR.grid(column=1, row=0, sticky="w")
    btnfoto = tk.Button(usuarioCV, text="Capturar fotos", width=ancho_btn, height
=1, command=capturar_fotos,
                        font=("Agency FB", tamaño_letra))
    btnfoto.grid(column=2, row=0, sticky="w")

    lblinfo2 = tk.Label(usuarioCV, text="Seleccione usuario a eliminar:",
                        font=("Agency FB", tamaño_letra))
    lblinfo2.grid(column=0, row=1, sticky="e")
    combo_usuario = ttk.Combobox(usuarioCV, values=users, state="readonly", width
=ancho, height=1,
                                 font=("Agency FB", tamaño_letra))
    combo_usuario.grid(column=1, row=1, sticky="w")
    btndelete = tk.Button(usuarioCV, text="Eliminar usuario", width=ancho_btn, he
ight=1, command=delete_usuario,
                          font=("Agency FB", tamaño_letra))
    btndelete.grid(column=2, row=1, sticky="w")

    lblVideo = tk.Label(usuarioCV)
    lblVideo.grid(column=0, row=2, columnspan=3)

    ver_videoCV()


def iniciarFR():
```

```
def seleccion_camara(combo_camaras):
    camara = 0
    cam = combo_camaras.get()

    if cam == "Camara integrada":
        camara = 0
    elif cam == "Camara externa 1":
        camara = 1
    elif cam == "Camara externa 2":
        camara = 2
    elif cam == "Camara externa 3":
        camara = 3
    elif cam == "Camara externa 4":
        camara = 4

    return camara

def seleccion_resolucion(combo_resoluciones):
    ancho = 1280
    alto = 800
    res = combo_resoluciones.get()
    resolución = res.split('x')
    ancho = resolución[0]
    alto = resolución[1]
    return ancho, alto

def ver_video():
    global cap
    global frame
    if cap is not None:
        ret, frame = cap.read()
        if ret == True:

            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            img = ImageTk.PhotoImage(Image.fromarray(frame))
            lblVideo.configure(image=img)
            lblVideo.image = img
            lblVideo.after(10, ver_video)
        else:
            lblVideo.image = ""
            cap.release()

def capturarfoto():
    if txtnombreFR.get(1.0, 'end-1c') != "":
```

```
            nombres_usuarios=os.listdir('/home/linux/Escritorio/Tesis/usuariosFR'
)
            nombre_ingresado = txtnombreFR.get(1.0, 'end-1c')+'.jpg'
            if len(nombres_usuarios)==0:
                existe_usuario=False
            else:

                for nombre_usuario in nombres_usuarios:
                    if nombre_ingresado.lower() == nombre_usuario.lower():
                        existe_usuario = True
                        break
                    else:
                        existe_usuario = False

            if existe_usuario:
                messagebox.showinfo("¡Advertencia!", "El usuario ingresado ya se
encuentra registrado")
            else:

                global cap
                global usuariosFR
                global tamaño
                global resolucion

                guardar_foto = tk.Toplevel(usuario)
                guardar_foto.geometry(tamaño)
                usuario.title("AÑADIR USUARIO FACE RECOGNITION")
                usuario.grab_set()
                nombre_usuario = txtnombreFR.get(1.0, 'end-1c')
                lblnombreusuario = tk.Label(guardar_foto, text=f'Nombre de usuari
o: {nombre_usuario}',
                                            font=("Agency FB", 16))
                lblnombreusuario.grid(column=0, row=0)
                lblfoto = tk.Label(guardar_foto)
                lblfoto.grid(column=0, row=1)

                ret, frame = cap.read()

                frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                img = ImageTk.PhotoImage(Image.fromarray(frame))
                lblfoto.configure(image=img)
                lblfoto.image = img

                dglpregunta = messagebox.askquestion("Guardar imagen", "¿Guardar
usuario?")
```

```
                if dglpregunta == "yes":
                    path = '/home/linux/Escritorio/Tesis/usuariosFR'
                    usuariosFR.append(nombre_usuario)
                    cv2.imwrite(os.path.join(path, nombre_usuario + '.jpg'),
                                frame)
                    txtnombreFR.delete("1.0", "end")
                    guardar_foto.destroy()
                else:

                    txtnombreFR.delete("1.0", "end")
                    guardar_foto.destroy()


        else:
            messagebox.showinfo("¡Advertencia!", "Debe ingresar un nombre de usua
rio")

    def usuarios():
        users = os.listdir('/home/linux/Escritorio/Tesis/usuariosFR')
        return users

    def delete_usuario():
        global combo_usuario
        global users
        nombre = combo_usuario.get()

        if nombre != "":
            usuario_eliminar = '/home/linux/Escritorio/Tesis/usuariosFR' + '/' +
nombre
            os.remove(usuario_eliminar)
            messagebox.showinfo("¡Información!", "Usuario eliminado")
        else:
            messagebox.showinfo("¡Advertencia!", "Debe seleccionar un usuario par
a eliminar")

    global cap
    global combo_camaras
    global combo_resoluciones
    global combo_usuario
    global tamaño
    global resolucion
    if cap is not None:
        cap.release()
```

```
    camara = seleccion_camara(combo_camaras)
    resolucion = seleccion_resolucion(combo_resoluciones)
    users = usuarios()

    cap = cv2.VideoCapture(camara)
    cap.set(3, int(resolucion[0]))
    cap.set(4, int(resolucion[1]))

    if int(resolucion[0]) == 1280 and int(resolucion[1]) == 800:
        tamaño = '1290x820+0+0'
        tamaño_letra = 6
        ancho = 30
        ancho_btn = 20
    elif int(resolucion[0]) == 800 and int(resolucion[1]) == 600:
        tamaño = '810x700+1+1'
        tamaño_letra = 8
        ancho = 25
        ancho_btn = 18
    elif int(resolucion[0]) == 352 and int(resolucion[1]) == 288:
        tamaño = '360x370+10+10'
        tamaño_letra = 6
        ancho = 13
        ancho_btn = 16

    usuario = tk.Toplevel(raiz)
    usuario.geometry(tamaño)
    usuario.title("AÑADIR USUARIO A FACE-RECOGNITION")
    usuario.grab_set()


    lblinfo1 = tk.Label(usuario, text="Ingrese nombre de usuario:",
                        font=("Agency FB", tamaño_letra))
    lblinfo1.grid(column=0, row=0, sticky="e")
    txtnombreFR = tk.Text(usuario, width=ancho, height=1, font=("Agency FB", tama
ño_letra), padx=2, pady=2,
                          selectbackground="gray")
    txtnombreFR.grid(column=1, row=0, sticky="w")
    btnfoto = tk.Button(usuario, text="Capturar foto", width=ancho_btn, height=1,
 command=capturarfoto,
                        font=("Agency FB", tamaño_letra))
    btnfoto.grid(column=2, row=0, sticky="w")

    lblinfo2 = tk.Label(usuario, text="Seleccione usuario a eliminar:",
                        font=("Agency FB", tamaño_letra))
    lblinfo2.grid(column=0, row=1, sticky="e")
```

```
    combo_usuario = ttk.Combobox(usuario, values=users, state="readonly", width=a
ncho, height=1,
                                font=("Agency FB", tamaño_letra))
    combo_usuario.grid(column=1, row=1, sticky="w")
    btndelete = tk.Button(usuario, text="Eliminar usuario", width=ancho_btn, heig
ht=1, command=delete_usuario,
                          font=("Agency FB", tamaño_letra))
    btndelete.grid(column=2, row=1, sticky="w")

    lblVideo = tk.Label(usuario)
    lblVideo.grid(column=0, row=2, columnspan=3)

    ver_video()
    usuario.mainloop()


def reconocimientoFR():

    def seleccion_camara(combo_camaras):
        camara = 0
        cam = combo_camaras.get()

        if cam == "Camara integrada":
            camara = 0
        elif cam == "Camara externa 1":
            camara = 1
        elif cam == "Camara externa 2":
            camara = 2
        elif cam == "Camara externa 3":
            camara = 3
        elif cam == "Camara externa 4":
            camara = 4

        return camara

    def seleccion_resolucion(combo_resoluciones):
        ancho = 1280
        alto = 800
        res = combo_resoluciones.get()
        resolución = res.split('x')
        ancho = resolución[0]
        alto = resolución[1]
        return ancho, alto

    def detener_rec():
```

```
        cap.release()
        mostrar_rec.destroy()

    def mostrar():
        global known_face_encodings
        global known_face_names
        global cap


        ret, frame = cap.read()

        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        face_locations = face_recognition.face_locations(rgb_frame)
        face_encodings = face_recognition.face_encodings(rgb_frame, face_location
s)


        for (top, right, bottom, left), face_encoding in zip(face_locations,
                                                              face_encodings):
            matches = face_recognition.compare_faces(known_face_encodings, face_e
ncoding)

            name = "Desconocido"


            face_distances = face_recognition.face_distance(known_face_encodings,
 face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = known_face_names[best_match_index]

                cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2
)

                cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 25
5, 0), cv2.FILLED)
                font = cv2.FONT_HERSHEY_DUPLEX
                cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255,
 255, 255), 1)

            if not matches[best_match_index]:

                cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2
)
```

```
                cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0,
255), cv2.FILLED)
                font = cv2.FONT_HERSHEY_DUPLEX
                cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255,
255, 255), 1)


        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        img = ImageTk.PhotoImage(Image.fromarray(frame))
        lblVideo1.configure(image=img)
        lblVideo1.image = img
        lblVideo1.after(10, mostrar)


    global cap
    global combo_camaras
    global combo_resoluciones
    global known_face_encodings
    global known_face_names
    if cap is not None:
        cap.release()

    algun_usuario = os.listdir(
        '/home/linux/Escritorio/Tesis/usuariosFR')
    if len(algun_usuario) > 0:

        camara = seleccion_camara(combo_camaras)
        resolucion = seleccion_resolucion(combo_resoluciones)

        cap = cv2.VideoCapture(camara)
        cap.set(3, int(resolucion[0]))
        cap.set(4, int(resolucion[1]))

        if int(resolucion[0]) == 1280 and int(resolucion[1]) == 800:
            tamaño = '1290x790+0+0'
            tamaño_letra = 8
            ancho_btn = 24
        elif int(resolucion[0]) == 800 and int(resolucion[1]) == 600:
            tamaño = '810x660+1+1'
            tamaño_letra = 8
            ancho_btn = 22
        elif int(resolucion[0]) == 352 and int(resolucion[1]) == 288:
            tamaño = '360x340+10+10'
            tamaño_letra = 8
```

```
            ancho_btn = 20

        mostrar_rec = tk.Toplevel(raiz)
        mostrar_rec.geometry(tamaño)
        mostrar_rec.title("RECONOCIMIENTO FACIAL FACE RECOGNITION")
        mostrar_rec.grab_set()

        btncerrar_rec = tk.Button(mostrar_rec, text="Detener reconocimiento", wid
th=ancho_btn, height=1,
                                  command=detener_rec,
                                  font=("Agency FB", tamaño_letra))
        btncerrar_rec.grid(column=0, row=0, columnspan=2)

        lblVideo1 = tk.Label(mostrar_rec)
        lblVideo1.grid(column=0, row=1, columnspan=2)

        known_face_names = []
        known_face_encodings = []
        imagenes = []

        cantidad_fotos = os.listdir(
            '/home/linux/Escritorio/Tesis/usuariosFR'
        for foto_usuario in cantidad_fotos:
            nombre = foto_usuario.split(".")
            known_face_names.append(nombre[0])

        for image in cantidad_fotos:
            imagenes.append(face_recognition.load_image_file('/home/linux/Escrito
rio/Tesis/usuariosFR/' + image))

        for x in imagenes:

            known_face_encodings.append(face_recognition.face_encodings(x)[0])

        mostrar()
    else:
        messagebox.showinfo("¡Advertencia!", "Debe tener como minimo un usuario r
egistrado")


def reconocimientoCV():
    def seleccion_camara(combo_camaras):
        camara = 0
        cam = combo_camaras.get()
```

```
        if cam == "Camara integrada":
            camara = 0
        elif cam == "Camara externa 1":
            camara = 1
        elif cam == "Camara externa 2":
            camara = 2
        elif cam == "Camara externa 3":
            camara = 3
        elif cam == "Camara externa 4":
            camara = 4

        return camara

    def seleccion_resolucion(combo_resoluciones):
        ancho = 1280
        alto = 800
        res = combo_resoluciones.get()
        resolución = res.split('x')
        ancho = resolución[0]
        alto = resolución[1]
        return ancho, alto

    def detener_recCV():

        global cap
        cap.release()
        mostrar_recCV.destroy()

    def mostrarCV():
        global faceClassif
        global face_recognizer
        global imagePaths
        global cap

        ret, frame = cap.read()

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        auxFrame = gray.copy()

        faces = faceClassif.detectMultiScale(gray, 1.3, 5)

        for (x, y, w, h) in faces:
            rostro = auxFrame[y:y + h, x:x + w]
            rostro = cv2.resize(rostro, (150, 150), interpolation=cv2.INTER_CUBIC
)
```

```
            result = face_recognizer.predict(rostro)

            cv2.putText(frame, "", (x, y - 5), 1, 1.3, (255, 255, 0), 1, cv2.LINE
_AA)


            if result[1] < 60:

                left = x
                top = y
                right = x + w
                bottom = y + h
                cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2
)
                cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 25
5, 0), cv2.FILLED)
                font = cv2.FONT_HERSHEY_DUPLEX
                cv2.putText(frame, '{}'.format(imagePaths[result[0]]), (left + 6,
 bottom - 6), font, 1.0,
                            (255, 255, 255), 1)

            else:

                left = x
                top = y
                right = x + w
                bottom = y + h
                cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2
)
                cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0,
 255), cv2.FILLED)
                font = cv2.FONT_HERSHEY_DUPLEX
                cv2.putText(frame, 'Desconocido', (left + 6, bottom - 6), font, 1
.0, (255, 255, 255), 1)

        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        img = ImageTk.PhotoImage(Image.fromarray(frame))
        lblVideo1.configure(image=img)
        lblVideo1.image = img
        mostrar_recCV.after(10, mostrarCV)


    global cap
    global combo_camaras
    global combo_resoluciones
```

```
    global faceClassif
    global face_recognizer
    global imagePaths
    if cap is not None:
        cap.release()
    algun_usuarioCV = os.listdir('/home/linux/Escritorio/Tesis/usuariosCV')

    if len(algun_usuarioCV) > 0:

        camara = seleccion_camara(combo_camaras)
        resolucion = seleccion_resolucion(combo_resoluciones)

        cap = cv2.VideoCapture(camara)
        cap.set(3, int(resolucion[0]))
        cap.set(4, int(resolucion[1]))

        if int(resolucion[0]) == 1280 and int(resolucion[1]) == 800:
            tamaño = '1290x790+0+0'
            tamaño_letra = 8
            ancho_btn = 24
        elif int(resolucion[0]) == 800 and int(resolucion[1]) == 600:
            tamaño = '810x660+1+1'
            tamaño_letra = 8
            ancho_btn = 22
        elif int(resolucion[0]) == 352 and int(resolucion[1]) == 288:
            tamaño = '360x340+10+10'
            tamaño_letra = 8
            ancho_btn = 20

        mostrar_recCV = tk.Toplevel(raiz)
        mostrar_recCV.geometry(tamaño)
        mostrar_recCV.title("RECONOCIMIENTO FACIAL OPEN CV")
        mostrar_recCV.grab_set()

        btncerrar_rec = tk.Button(mostrar_recCV, text="Detener reconocimiento", w
idth=ancho_btn, height=1,
                                  command=detener_recCV,
                                  font=("Agency FB", tamaño_letra))
        btncerrar_rec.grid(column=0, row=0, columnspan=2)

        lblVideo1 = tk.Label(mostrar_recCV)
        lblVideo1.grid(column=0, row=1, columnspan=2)

        dataPath = '/home/linux/Escritorio/Tesis/usuariosCV'
        imagePaths = os.listdir(dataPath)
```

```
        print('imagePaths=', imagePaths)


        face_recognizer = cv2.face.LBPHFaceRecognizer_create()

        face_recognizer.read('modeloLBPHFace.xml')

        faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_
frontalface_default.xml')

        mostrarCV()
    else:
        messagebox.showinfo("¡Advertencia!", "Debe tener como minimo un usuario r
egistrado")


global combo_camaras
global combo_resoluciones
camaras = range(5)
count = []
band = 0
nombres = []
for x in camaras:
    cap = cv2.VideoCapture(x)

    if cap.isOpened():

        count.append(1)
    else:

        count.append(0)



for idx, val in enumerate(count):
    if val == 1 and band == 0:
        nombres.append('Camara integrada')
        band = 1
    elif val == 1:
        nombres.append('Camara externa ' + str(idx))


cap = None
usuariosFR = []
raiz = tk.Tk()
raiz.geometry("600x400+10+10")
```

```
raiz.title("SISTEMA DE RECONOCIMIENTO FACIAL")
raiz.config(bg="gray")


imagenud = tk.PhotoImage(file="escudo_ud.png")
imagenud = imagenud.subsample(1, 1)
lblImagen = tk.Label(raiz, image=imagenud, bg="Gray").place(x=50, y=50)
raiz.resizable(0, 0)

combo_camaras = ttk.Combobox(raiz, values=nombres, state="readonly", width=20, he
ight=1, font=("Agency FB", 7))
combo_camaras.place(x=400, y=40)
combo_camaras.current(0)
BtnadduserFR = tk.Button(raiz, text="Añadir Usuario FR", command=iniciarFR, width
=25, height=1,
                         font=("Agency FB", 7)).place(x=400, y=200)
BtnrecfacialFR = tk.Button(raiz, text="Reconocimiento Facial FR", command=reconoc
imientoFR, width=25, height=1,
                           font=("Agency FB", 7)).place(x=400, y=250)
BtnadduserCV = tk.Button(raiz, text="Añadir Usuario Open CV", command=iniciarCV,
width=25, height=1,
                         font=("Agency FB", 7)).place(x=400, y=300)
BtnrecfacialCV = tk.Button(raiz, text="Reconocimiento Facial Open CV", command=re
conocimientoCV, width=25, height=1,
                           font=("Agency FB", 7)).place(x=400, y=350)
lblinforesolucion = tk.Label(raiz, text="Configuración de la cámara", bg='gray',
font=("Agency FB", 7)).place(x=410,

                             y=10)
combo_resoluciones = ttk.Combobox(raiz, values=["1280x800", "800x600", "352x288"]
, state="readonly", width=20, height=1,
                                  font=("Agency FB", 7))
combo_resoluciones.place(x=400, y=80)
combo_resoluciones.current(0)

raiz.mainloop()
```