UNIVERSIDAD GABRIELA MISTRAL

FACULTAD DE NEGOCIOS, INGENIERÍA Y ARTES DIGITALES

Control de identificación de rostros utilizando técnica de reconocimiento facial mediante cámara de seguridad para Informática Efasoft5

Modesto Elías Farfán Ascorbe

2021

UNIVERSIDAD GABRIELA MISTRAL

FACULTAD DE NEGOCIOS, INGENIERÍA Y ARTES DIGITALES

Control de identificación de rostros utilizando técnica de reconocimiento facial mediante cámara de seguridad para Informática Efasoft5

Trabajo de Titulación presentado en conformidad a los requisitos para obtener el Título de Ingeniero Civil en Informática.

Alumno: Modesto Elías Farfán Ascorbe

Profesor Guía: Gerardo Cerda Neumann

10 de Agosto de 2021

Agradecimientos

Dedico este proyecto especialmente a mi familia, ya que con su apoyo en todo ámbito puedo concluir la carrera de la mejor manera. A mi Tío, Job Ascorbe, por apoyarme y siempre guiarme cuando lo necesite. A mi Esposa, hijos por apoyarme en los momentos difíciles durante los años de Universidad, ya que con ellos pude seguir en la carrera. Y en general a todas las personas que apoyaron en todo momento ya sea familiares, como también al profesor guía Gerardo Cerda Neumman, colaborando en mi tesis hasta el final y muchos otros profesores que lamentablemente ya no se encuentran en la universidad. Gracias a todos.

Modesto Elías Farfán Ascorbe

Resumen

En el siguiente proyecto realizado por mí, como estudiante de la Universidad Gabriela Mistral, se estudiará, analizará e implementará un algoritmo para una solución tecnológica de un sistema de reconocimiento facial, para la Pyme Informática Efasoft5 ubicada en Avenida Recoleta 2390, comuna de Recoleta, el cual será utilizado con bibliotecas de OpenCV para integrar un trabajo más eficiente y exacto. Esto se realizará con el fin de dar un comienzo hacia un proyecto más avanzado y que más adelante sea en línea.

El sistema dispondrá de una computador que mediante la aplicación de solución tecnológica gestionara la cámara de seguridad, se crearan carpetas de las posibles personas a identificar localmente se permitirá almacenar las diferentes fotografías ya entrenadas sacadas al usuario para que en el paso siguiente se pueda reconocer ya el rostro de dicha persona, con esto podremos emitir una alerta sonora para los usuarios desconocido para tomar medidas sobre estos.

Palabras claves: Reconocimiento facial, bibliotecas OpenCV.

Abstract

In the following project carried out by me, as a student at the Gabriela Mistral University, an algorithm for a technological solution of a facial recognition system will be studied, analyzed and implemented for the Efasoft5 Computer Pyme located at av. Recoleta 2390, commune of Recoleta, which will be used with OpenCV libraries to integrate more efficient and accurate work. This will be done in order to give a start towards a more advanced project that will be online later.

The system will have a computer that, through the application of a technological solution, will manage the security camera. Folders will be created for potential people to be identified locally. The different trained photos taken from the user will be allowed to be stored so that in the next step it can be recognized. The face of said person, with this we can issue an audible alert for unknown users to take action on them.

Key words: Face recognition, OpenCV libraries.

INDICE

Capítulo N° 1: Introducción

Hoy en día la empresa **Efasoft5**, además del servicio de soporte técnico informático ofrece a su clientela artículos electrónicos, computacionales y de telefonía. Sin embargo, a la fecha no ha sido posible llevar un registro en la seguridad del establecimiento y lo que se busca es la propuesta tecnológica que apoye a esta necesidad.

El presente informe contempla el desarrollo de una solución tecnológica cuya funcionalidad principal es la identificación de personas mediante **el reconocimiento facial**. Esta solución será un gran aporte para la Empresa **Efasoft5,** ya que este reconocimiento se lleva a cabo de forma tradicional. La empresa cuenta con un local de **Centro de llamados e Internet** y se encuentra Ubicada en la Comuna de Recoleta.

La Pyme necesita implementar una solución tecnológica que pueda en un inicio registrar el rostro de las personas para posteriormente en un acto de robo se pueda identificar al delincuente y ser utilizado como medio de prueba a la justicia.

En los inicios de esta tecnología llamada “Reconocimiento facial” se usaba algoritmos de reconocimientos muy simples el cual daba mayor oportunidad a que los errores se produjeran, ya que al ser así el mismo reconocimiento se podría dar para 2 personas diferentes. En la actualidad y con los avances logrados, además de los algoritmos que han sido exponencialmente mejorados, los errores son mínimos ya que se han afinado la forma en cómo se reconoce cada rostro.

A continuación se explica cómo funcionan estas herramientas para la elaboración y construcción de un prototipo de una aplicación el cual es capaz en su versión final de hacer un reconocimiento facial. Este proyecto consta de varias secciones las cuales en complemento darán un buen funcionamiento del software que se implementará.

Capítulo N° 2: Objetivos generales y específicos del Trabajo

2.1.- Objetivo generales:

Control de identificación de rostros utilizando técnica de reconocimiento facial mediante cámara de seguridad para Informática Efasoft5.

2.2.- Objetivos específicos:

- Hacer un diagnóstico de la situación actual del proceso de la seguridad que se mantiene en la empresa.
- Investigar acerca de las diferentes técnicas que se utiliza para el reconocimiento facial.
- Investigar acerca de las bibliotecas existentes que se pueda utilizar con el lenguaje de programación.
- Investigar acerca de las cámaras de seguridad que se encuentran en el mercado y sea compatible con el software.
- Analizar la infraestructura física donde se implementará la aplicación para el reconocimiento de rostros.
- Investigar sobre los diferentes modelos o algoritmos, tanto de los procesos de detección y clasificación de rostros.
- Testear o someter a prueba cuál de los modelos o algoritmos seleccionados tienen mejor precisión.
- Realizar pruebas de las distintas herramientas de programación que sea compatible con el desarrollo del proyecto.

Capítulo N° 3: Marco Teórico

3.1.-Introducción

3.1.1.-Historia

Las caras son muchos más que claves de la identidad individual (Gan, 2018). Por otra parte, la expresión facial, es el camino directo para las personas que expresan el sentido intuitivo. El propósito fundamental de expresión de reconocimiento facial (FER).

Hasta hace poco, la tecnología de reconocimiento facial era comúnmente vista como algo sacado de la ciencia ficción. Pero en la última década, esta tecnología innovadora no solo se ha convertido en viable, se ha generalizado. De hecho, es difícil leer noticias sobre tecnología en estos días sin ver algo del reconocimiento facial.

3.1.2.-aplicaciones

Dentro de las aplicaciones usualmente tenemos: (Martin, 2016)

I. Área biometría (licencia de conducir, Programas de derecho, Inmigración, DNI, Pasaportes, Registro de volantes, Fraude).
II. Área seguridad de la información (inicio de sección, seguridad en aplicaciones, seguridad en bases de datos, cifrado de información, seguridad en internet, acceso a internet, registros médicos, terminales de comercio seguro, cajeros automáticos).
III. Área cumplimiento de la ley y vigilancia (video vigilancia avanzada, control CCTV, control portal, análisis post-event, hurto, seguimiento de sospechosos, investigación).
IV. Áreas tarjetas inteligentes (valor almacenado, autenticación de usuarios).
V. Área control de acceso (acceso a instalaciones, acceso a vehículos).

3.1.3.-aporte en seguridad

I. Seguridad Pública. - el primer campo donde se aplican los procesos de reconocimiento facial es en el ámbito policial. En China, (Vincent, 2018) por ejemplo, ya se usan gafas con capacidades de reconocimiento facial para capturar a criminales. Esta es una técnica con muy buenos resultados que ha hecho que la tecnología sea aplicada en toda el área

de Beijing. Esta tecnología vestible esta enlazada a una base de datos que contiene grabaciones de criminales que, en caso de que una de las imágenes coincida con alguna de las registradas en las lista, puede contribuir en el arresto del sospechoso.

II. Cerraduras inteligente.- las cerraduras inteligentes son dispositivos que parecen un timbre de puerta habitual, pero que contienen una pequeña cámara en su interior que, a través del reconocimiento facial o la voz, permite entrar al propietario en su casa, sin necesidad de ninguna llave. Airbnb utilizara tecnologías Latch de cerraduras inteligentes para asegurar los apartamentos de la marca que inauguraran esta primavera. Estas cerraduras inteligentes permitirán que Airbnb garantice la identidad de las personas instaladas en el apartamento (actualmente, solo puede confirmarse de quien hace la reserva) (aetecno, 2018).

III. Reconocimiento facial en aeropuertos.- el reconocimiento facial se puede utilizar para mejorar la seguridad en aeropuertos y acelerar el procesamiento en el check-in de un vuelo. Las autoridades aeroportuarias y federales de Estados Unidos presentaron nueva tecnología de reconocimiento facial diseñada para reemplazar las tarjetas de embarque y aumentar la velocidad en el proceso de abordaje en los vuelos internacionales del Aeropuerto Dulles, Washington. En lugar de presentar el pasaporte y la tarjeta de embarque, se toma una foto del rostro de cada pasajero y se compara la imagen con las fotos archivadas en las bases de datos del gobierno de Estados Unidos. Esto determina si la persona es un pasajero de ese vuelo y si está autorizada a abandonar el país. La verificación biométrica tarda menos de un segundo con una tasa de precisión del 99 por ciento y las autoridades aseguran que las fotos tomadas en la puerta de embarque se borran del sistema a las 12 horas. Inicialmente esta tecnología simplifica el proceso de a bordo para aquellos pasajeros que viajen fuera de Estados Unidos. En el caso de los extranjeros, la foto tomada por la cámara se almacena en el sistema del departamento de seguridad nacional, para rastrear la llegada y la salida de Estados Unidos (eficiencia, 2018).

3.1.4.-aporte reconocimiento facial

I. Aplicación para encontrar extraños.- con solo sacar una foto, quienes use ese sistema pueden llegar a saber cómo se llama una persona completamente desconocida, y acceder también a toda la información

pública que tenga en línea. Hablamos de Name Tag, de la empresa Facial Network, un servicio de reconocimiento visual capaz de tomar la fotografía de una persona desconocida, analizar su rostro y buscarlo en cualquier red social en línea. Este sistema presentado por Android, e incluso los lentes de realidad aumentada Google Glass, permiten que los usuarios puedan usar la cámara de sus dispositivos para identificar a las personas. Esta plataforma toma la imagen y la envía por internet a los servidores de la compañía. En este lugar, la captura es comparada con millones de contenido en línea hasta que el servicio encuentre una similar a la cara en la fotografía. Una vez que el servicio encuentra una coincidencia en el sistema, no solo envía de vuelta al usuario el posible nombre de la persona fotografiada, sino que también otras imágenes y sus perfiles en diferentes redes sociales como Facebook, Twitter e Instagram, donde la persona podría haber publicado sus capturas (Loffler, 2017).

3.2.-aplicación algoritmo para reconocimiento facial

3.2.1- descripción del algoritmo

Redes neuronales convolucionales:

Historia:

Los fundamentos de las redes neuronales convolucionales se basan en el Neocognitron, introducido por Kunihiko Fukushima en 1980. Este modelo fue más tarde mejorado por Yann LeCun et al. En 1998 al introducir un método de aprendizaje basado en la propagación hacia atrás para poder entrenar el sistema correctamente. En el año 2012, fueron refinadas por Dan Ciresan y otros, y fueron implementadas para una unidad de procesamiento grafico (GPU) consiguiendo así resultados impresionantes (Wenlai Zhao, 2018).

Fundamentos biológicos:

El trabajo realizado por Hubel y Wiesel en 1959 jugó un papel importante en la comprensión sobre cómo funciona la corteza visual, particularmente las células responsables de la selectividad de orientación y detección de bordes en los estímulos visuales dentro de la corteza visual primaria V1. Dos tipos de células principales fueron identificadas aquí, teniendo éstas campos receptivos alargados, con lo cual tienen una mejor respuesta a los estímulos visuales

alargados como las líneas y los bordes. Estas se denominan células simples y células complejas (Wiesel, 1981).

Arquitectura:

Las redes neuronales convolucionales consisten en múltiples capas de filtros convolucionales de una o más dimensiones. Después de cada capa, por lo general se añade una función para realizar un mapeo causal no-lineal.

Como redes de clasificación, al principio se encuentra la fase de extracción de características, compuesta de neuronas convolucionales y de reducción de muestreo. Al final de la red se encuentran neuronas de perceptrón sencillas para realizar la clasificación final sobre las características extraídas. La fase de extracción de características se asemeja al proceso estimulante en las células de la corteza visual. Esta fase se compone de capas alternas de neuronas convolucionales y neuronas de reducción de muestreo. Según progresan los datos a lo largo de esta fase, se disminuye su dimensionalidad, siendo las neuronas en capas lejanas mucho menos sensibles a perturbaciones en los datos de entrada, pero al mismo tiempo siendo estas activadas por características cada vez más complejas.

Neuronas convolucionales:

En la fase de extracción de características, las neuronas sencillas de un perceptrón son reemplazadas por procesadores en matriz que realizan una operación sobre los datos de imagen 2D que pasan por ellas, en lugar de un único valor numérico.

El operador de convolución tiene el efecto de filtrar la imagen de entrada con un núcleo previamente entrenado. Esto transforma los datos de tal manera que ciertas características (determinadas por la forma del núcleo) se vuelven más dominantes en la imagen de salida al tener estas un valor numérico más alto asignados a los pixeles que las representan. Estos núcleos tienen habilidades de procesamiento de imágenes específicas, como por ejemplo la detección de bordes que se puede realizar con núcleos que resaltan la gradiente en una dirección en particular. Sin embargo, los núcleos que son entrenados por una red neuronal convolucional generalmente son más complejos para poder estos extraer otras características más abstractas y no triviales.

Neuronas de reducción de muestreo:

Las redes neuronales cuentan con cierta tolerancia a pequeñas perturbaciones en los datos de entrada. Por ejemplo, si dos imágenes casi idénticas (diferenciadas únicamente por un traslado de algunos pixeles lateralmente) se analizan con una red neuronal, el resultado debería de ser esencialmente el mismo. Esto se obtiene, en parte, dado a la reducción de muestreo que ocurre dentro de una red neuronal convolucional. Al reducir la resolución, las mismas características corresponderán a un mayor campo de activación en la imagen de entrada.

Originalmente, las redes neuronales convolucionales utilizaban un proceso de subsampling para llevar a cabo esta operación. Sin embargo, estudio recientes han demostrado que otras operaciones, como por ejemplo max-pooling, son mucho más eficaces en resumir características sobre una región. Además, existe evidencia que este tipo de operación es similar a como la corteza visual puede resumir información internamente.

La operación de max-pooling encuentra el valor máximo entre una ventana de muestra y pasa este valor como resumen de características sobre esa área. Como resultado, el tamaño de los datos se reduce por un factor igual al tamaño de la ventana de muestra sobre la cual se opera.

Neuronas de clasificación:

Después de una o más fases de extracción de características, los datos finalmente llegan a la fase de clasificación. Para entonces, los datos han sido depurados hasta una serie de características únicas para la imagen de entrada, y es ahora la labor de esta última fase el poder clasificar estas características hacia una etiqueta u otra, según los objetivos de entrenamiento (Hernández-Mendo, 2014), La arquitectura de red neuronal se muestra en la figuras 1 y 2.

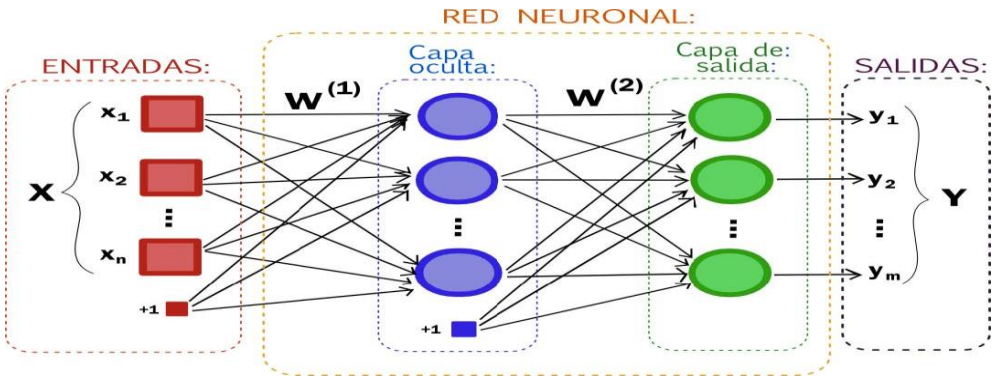


Figura: 1 Esquema de una red neuronal simple
Fuente: (Jimenez, 2019)

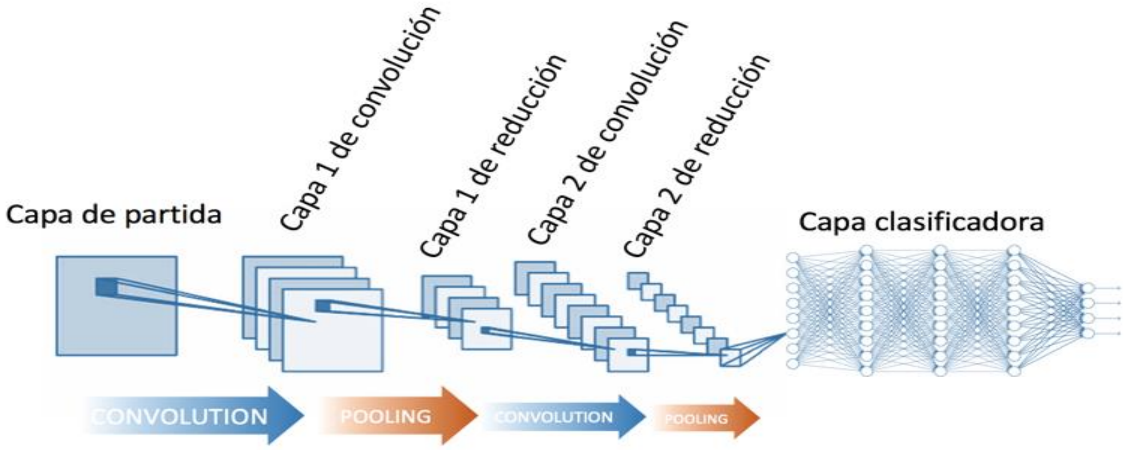


Figura: 2 Arquitectura de una red neuronal convolucional
Fuente: (Calvo, 2017)

Capítulo N° 4: Ámbito del problema

4.1.- Declaración del problema

La empresa cuenta con un Sistema de seguridad ya que esta se desarrolla de la siguiente manera.

En la actualidad se cuenta con una cámara de seguridad que se encuentra encendida las 24 horas diarias en donde graba todas las imágenes en un disco duro que se encuentra alojado en un servidor local donde cada mes hay que realizar su mantenimiento en el borrado de las grabaciones.

Se programo en el software que viene por defecto de la cámara en donde desde las 22:00 hasta la 08:00 horas una alarma en donde si ingresaban en ese horario alguna persona el sistema enviaría una alerta al celular del administrador de que algún intruso ingreso, pero no sabíamos con exactitud qué persona era la que había ingresado al local, a veces mandaba alarmas al celular en la madrugada donde había falsos ingresos la cual es un problema constante y fastidioso para el administrador donde se le interrumpía su sueño.
Además, usan tecnología bastante antigua ya que trabajan con un sistema de seguridad genérico que traen todas las cámaras.

El problema de	Seguridad en la Identificación de Personas
Efectos	Asaltos, Hurtos , clientes y usuarios expuestos al peligro
Cuál es el impacto	Alto costo de perdidas en productos y el ingreso de personas no autorizadas al local comercial
La solución satisfactoria debería ser	Se propone diseñar una solución tecnológica en donde se podrá registrar los rostros de personas que ingresan al local mediante la cámara de seguridad y reconocimiento facial.

4.2.- Declaración de la posición del producto

Pensar en el posicionamiento del artefacto de software es una tarea fundamental para desarrollar una estrategia, nosotros necesitamos llegar a las Pymes (pequeñas y medianas empresas) de la comuna en la cual se ven con la necesidad de obtener una solución tecnológica la cual registre los rostros de los delincuentes para ser utilizado como medio de prueba para la justicia.

Para	Pequeñas y medianas empresas de las comunas con menos recursos económicos.
Quien	Por necesidad requiera solucionar un problema de seguridad en los controles de accesos a las instalaciones de las empresas.
El (nombre del producto)	FaceCiber es un artefacto de software de Reconocimiento Facial.
Eso	Artefacto de software su principal beneficio es el costo de adquisición en el mercado.
A diferencia	Hikvision cámara de Reconocimiento Facial (IDS-2CD8426G0/F-I 4mm)
Nuestro Producto	Es un artefacto de software que es accesible en el mercado por su inversión inicial en el costo de adquisición.

4.3.- Descripción de clientes y usuarios

4.3.1.- Resumen de los involucrados

Los involucrados de los clientes y usuarios todos con sus respectivos roles de responsabilidad para poder apoyar al desarrollo del proyecto y dando todo su conocimiento se da a conocer en la siguiente tabla.

Nombre	Descripción	Responsabilidades
Loyda Layza	Administradora	El rol que tomara en el desarrollo es brindar información sobre las segmentación de clientes y funcionamiento de la seguridad

Elías Farfán	Analista y Desarrollador	Desarrollo del proyecto, encargado de las funcionalidades de la aplicación.

4.3.2.- Resumen de los Usuarios

Los usuarios que tendran la mision de familiarizarse a diario con el artefacto de software del proyecto es la administradora donde podra hacer uso del control del mismo y puesta en marcha.

Nombre	Descripción	involucrado	
Administrador	Administra el control al sistema y puesta en marcha	Loyda Layza	

Capítulo N° 5: Metodología empleada

5.1.- Resumen

El presente proyecto está enfocado a investigar una de las propuestas metodológicas que se utiliza en la actualidad para el desarrollo de proyectos de Software. La metodología RUP (Proceso Unificado de Rational) emplea varias de las mejores prácticas en el desarrollo de software de manera que es aplicable para un amplio rango de proyectos y organizaciones a la vez que provee a cada miembro de un equipo un fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades de desarrollo. Aplicando esta moderna metodología se va a desarrollar un sistema escolástico real, el mismo que será parametrizable para hacerlo adaptable en diferentes Instituciones Educativas, de esta manera se eliminarán los procesos que en la actualidad se llevan manualmente, se muestra la estructura de la metodología RUP en la Figura: 1

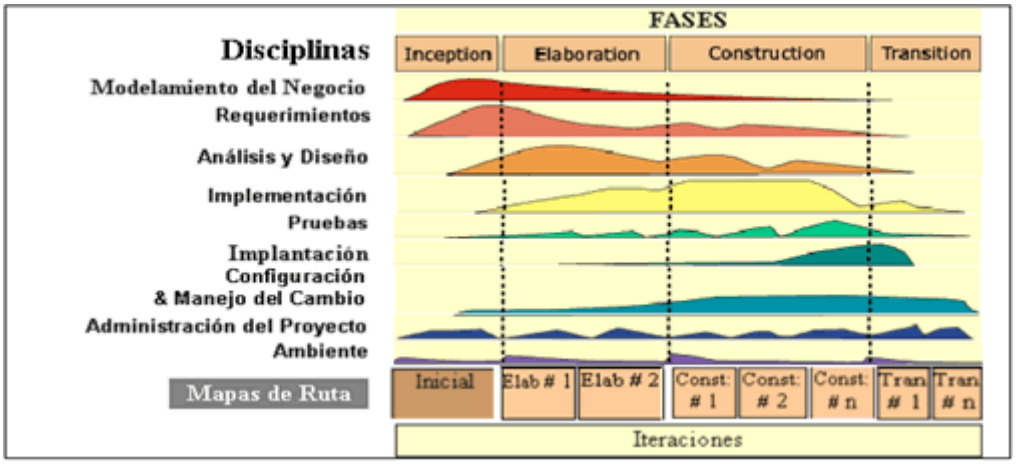


Figura: 1 Estructura de RUP
Fuente: (Carrillo, 2011)

Inicio:

Durante la fase de inicio se define el modelo del negocio y el alcance del proyecto. Se identifican todos los actores y Casos de Uso, y se diseñan los Casos de Uso más esenciales (aproximadamente el 20% del modelo completo). Se desarrolla, un plan de negocio para determinar qué recursos deben ser asignados al proyecto.

Elaboración:

El propósito de la fase de elaboración es analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos. En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final. Este prototipo debe contener los Casos de Uso críticos identificados en la fase de inicio. También debe demostrarse que se han evitado los riesgos más graves.

Construcción:

La finalidad principal de esta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante esta fase todos los componentes, características y requisitos deben ser implementados, integrados y probados en su totalidad, obteniendo una versión aceptable del producto.

Transición:

La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto.

5.2.- Diagrama de actividades.

Se visualiza el diagrama de actividades de la propuesta, donde se muestra el proceso del sistema desde una vista general, una vez iniciada la aplicación, se recepciona la secuencia del video, luego se pasa a procesar cada fotograma y son enviadas a la red neuronal, se comparan las distancias para saber cuál está más cercano, una vez encontrado se envía una alerta para avisar que se pudo encontrar una respuesta, este diagrama explica la metodología que utilizaremos, a continuación se pasa a detalle cada una de estas actividades Figura:5.2

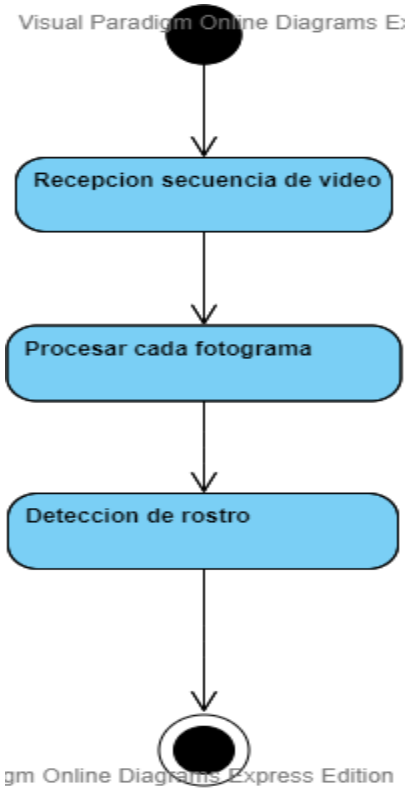


Figura 5.2 Diagrama de actividades
Fuente: Autoría propia

5.3.- Recepcionar secuencia de video

A continuación para poder recepcionar los fotogramas, necesitaremos una cámara para obtener el video que estará en un formato mp4, la cámara puede usar la resolución que esta misma posea, el sistema podrá leer este tipo de resolución, pero para poder hacer el proceso más rápido y adecuado usaremos una resolución mínima de 800 x 600 pixeles y 30 FPS, una vez preparado el sistema podrá obtener la secuencia de video que tendrá como entrada fotogramas todo esto lo logramos gracias a la biblioteca de OpenCV con Python.

5.4.- Procesar cada fotograma

Una vez obtenido la secuencia de video se proseguirá con la extracción de cada fotograma, el sistema va a extraer cada una de ellas cada 0.3 segundos, por ejemplo, en un video de formato mp4, el sistema recibirá la secuencia del video y extraerá los fotogramas se procede a extraer cada uno.

Luego de proceder con la extracción de los fotogramas; se tomará captura a una resolución de 800 x 600 pixeles, posteriormente el sistema procederá a escalar el fotograma obtenido a la mitad, con esto el proceso de detección de cada fotograma será rápido y optimo, sin perder información, dando como resultado una imagen de 400 x 300 píxeles.

5.5.- Detección del rostro

A continuación, utilizaremos la biblioteca de Dlib que usa algoritmos de aprendizaje de máquina, también usa una herramienta para detectar objetos basados en histogramas de gradiente orientado por sus siglas en inglés HOG, se encarga de describir una imagen de forma precisa, en esta representación se usa en la detección de puntos utilizando el modelo de aprendizaje y detección de máquinas de vectores de soporte por sus siglas en inglés (SVM).

En la Figura 5.6 se muestra el esquema de una imagen en escala de grises donde se agrupan histogramas de gradiente en celdas de NxN

pixeles, formando bloques, dando un conjunto de histogramas concatenados, estos agrupados se les conoce como descriptor.

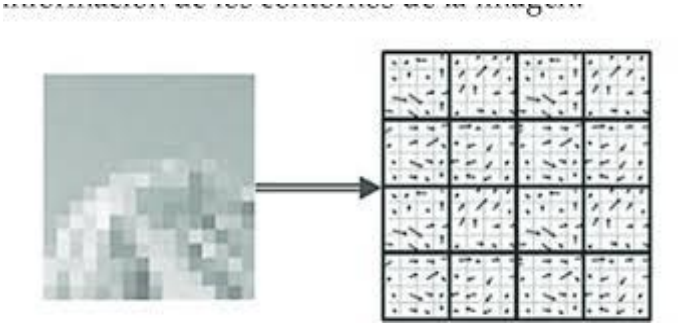


Figura 5.6 Esquema HOG, cálculo de gradiente

Fuente: (Merchan, 2014)

En la Figura 5.7 se muestra el resultado del procedimiento del cálculo del gradiente de la Figura 5.6 como resultado a una cara con una sonrisa, que pasa a una escala de grises y mostrando información de los rasgos de la cara en un descriptor.

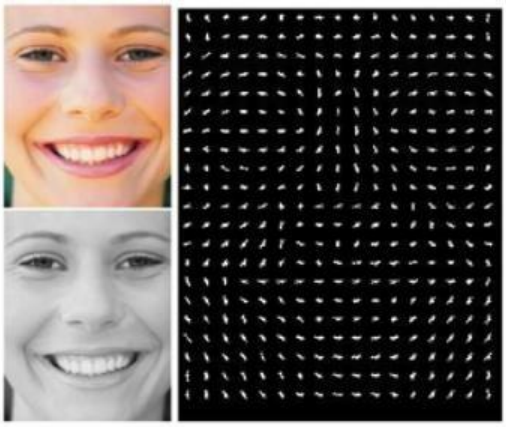


Figura 5.7 Esquema HOG, cálculo de gradiente para un rostro

Fuente: (Merchan, Mejoras en el entrenamiento de esquemas de detección de sonrisas basados en AdaBoost, 2014)

Es importante destacar, este algoritmo se encarga de ubicar los puntos que se quieren de un rostro mediante bloques de píxeles, cada una de las celdas brinda información importante en cómo separa los objetos de su fondo; Dlib también puede detectar puntos de referencia faciales lo que permite detectar regiones de la cara como ojos nariz, boca. Para nuestro caso utilizaremos esta biblioteca para la detección de rostro exclusivamente, en la Figura 5.8 se muestra como se hace la detección de rostro en Dlib.

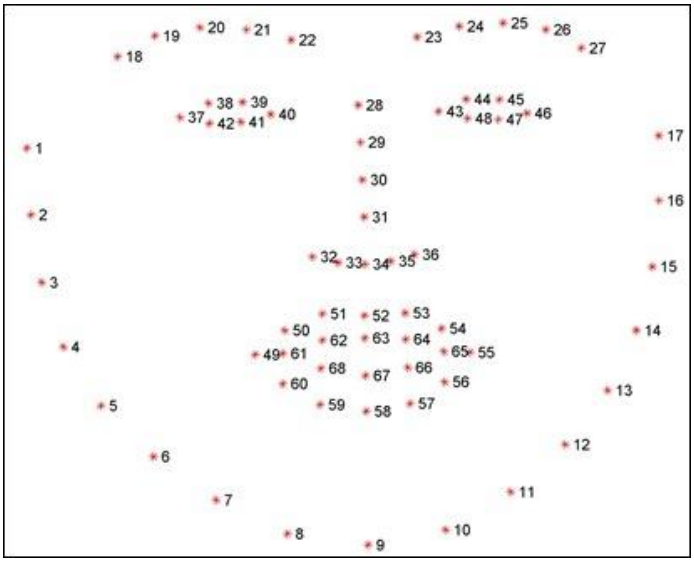


Figura 5.8 Dlib deteccion de rostro
Fuente: (Daza, 2017)

Todas estas herramientas hacen que funcione de la mejor forma la detección de rostros y alineación, por ejemplo, cada fotograma ingresado se procesa para detectar todos los rostros en ella, se transforma y se recortar el rostro encontrado dando una nueva imagen como resultado, como muestra la Figura 5.9.

Figura 5.9 Detección de rostro
Fuente: (Dahua y su cámara de reconocimiento facial disponible en Argentina, 2018)

5.7.- Feedforward a la red neuronal

En esta etapa los rostros obtenidos por Dlib se usan como entrada para luego ingresar a una red neuronal que utiliza OpenFace, esta red devuelve un vector de 128 dimensiones representados en una hiperesfera, esto hace que cada rostro este en 128 bytes, por ejemplo, en la Figura 5.10 se muestra cómo se representa la salida luego de ingresar a la red. Esta salida es usada para clasificar o agrupar

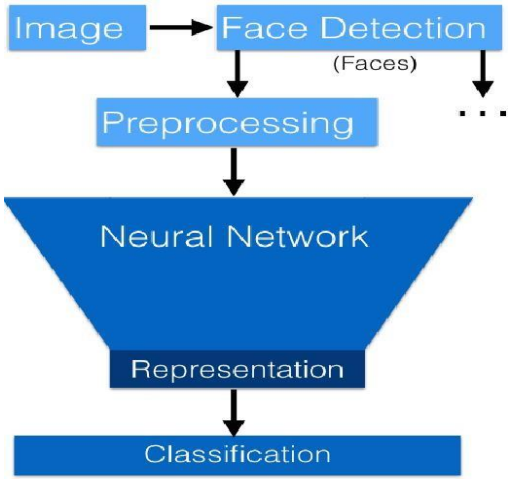


Figura 5.10 Clasificación OpenFace
Fuente: (Amos, 2006)

5.8.- Red neuronal

Openface está basada en la arquitectura FaceNet de Google, es una red neuronal convolucional que funciona bajo la función de pérdida de triplete, la red neuronal de Openface actúa bajo esta función para entrenar la red, consta básicamente en tomar una imagen de anclaje, se refiere a una muestra inicial de la imagen tomada, luego toma otra imagen con incrustaciones positivas de la imagen original y otra imagen adicional, pero con incrustaciones negativas. Entonces la disimilitud entre la imagen de anclaje y la imagen positiva debe ser menor a comparación de la disimilitud de la imagen de anclaje con la imagen negativa que debe ser alta.

Openface utiliza la red modificada de FaceNet, una red neuronal convolucional denominada modelo NN4, de este modelo es donde se obtienen las 128 características antes mencionadas como muestra en la figura 5.11

Figura 5.11 Ejemplo de valores de las 128 de un rostro

Fuente: (Calderon, 2018)

Adicionalmente se agrega que el modelo utilizado es la versión modificada del modelo de FaceNet, el cual es nn4.small2, este contiene 3733968 parámetros. La red neuronal se utiliza como un extractor de características que produce una representación de baja dimensión que caracteriza la cara de una persona. Cabe destacar que tener una representación de baja dimensión es clave para usarse eficientemente en clasificadores o técnicas agrupamiento.

5.9.- Comparar distancias

Como consecuencia del paso anterior obtuvimos la salida de la red neuronal de OpenFace con un vector resultante de 128 dimensiones, pasará a compararse con todas las fotos de la base de datos que tenemos, esta comparación se dará por la utilización de la fórmula de distancia euclidiana como se ve en la Ecuación 5.12, como definición la distancia entre 2 puntos Punto 1 y punto 2 cada uno con coordenadas en el plano (x1; y1) y (x2; y2)

$$d_{(p1,p2)} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Figura 5.12 Ecuación de la distancia Euclidiana
Fuente: (Silvercorp, 2012)

Podemos distinguir si una imagen coincide con otra por la medida de sus distancias de cada representación, por ejemplo, una vez encontrada resultados favorables y encontramos a una persona, entonces el sistema podrá enviar una alerta al usuario. Como muestra la Figura 5.13.

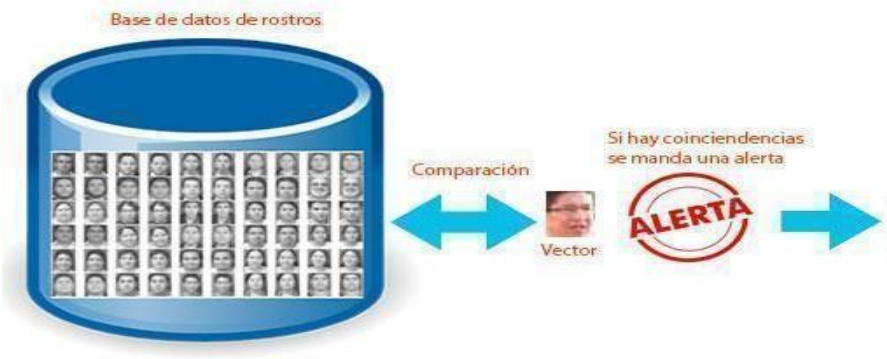


Figura: 5.13 Comparación y envió alerta.

Fuente: Autoría propia

Capítulo N° 6: Tecnologías utilizadas

6.1.- Inteligencia artificial

La Inteligencia artificial es el campo científico de la informática que se centra en la creación de programas y mecanismos que pueden mostrar comportamientos considerados inteligentes. En otras palabras, la IA es el concepto según el cual "las máquinas piensan como seres humanos".

Normalmente, un sistema de IA es capaz de analizar datos en grandes cantidades (big data), identificar patrones y tendencias y, por lo tanto, formular predicciones de forma automática, con rapidez y precisión. Para nosotros, lo importante es que la IA permite que nuestras experiencias cotidianas sean más inteligentes.

- Siri funciona como un asistente personal, ya que utiliza procesamiento de lenguaje natural
- Facebook y Google Fotos sugieren el etiquetado y agrupamiento de fotos con base en el reconocimiento de imagen
- Amazon ofrece recomendaciones de productos basadas en modelos de canasta de compra
- Waze brinda información optimizada de tráfico y navegación en tiempo real

6.2.- Redes neuronales convolucionales

Las Redes Neuronales son los modelos de aprendizaje automático con mejor desempeño en la actualidad en una gran variedad de problemas. Son modelos generales y aproximadores universales. Con algoritmos de optimización basados en descenso de gradiente, pueden optimizar miles o millones de parámetros en base a una función de error. Se distinguen de otros modelos en que no requieren un diseño manual de características de los datos para funcionar; las características se aprenden automáticamente mediante el proceso de optimización, también llamado entrenamiento. Su diseño se organiza en capas que determinan su arquitectura. En los últimos años, se ha

conseguido entrenar Redes Neuronales con múltiples capas mediante un conjunto de técnicas que suelen denominarse Aprendizaje Profundo (Deep Learning). En particular, las Redes Convolucionales, es decir, Redes Neuronales que utilizan capas convolucionales, son el estado del arte en la mayoría de los problemas de visión por computadora, incluyendo la clasificación de imágenes. Las capas convolucionales permiten aplicar convoluciones con filtros aprendidos para un mejor desempeño y eficiencia. Muchos de los problemas para los cuales las Redes Convolucionales son el estado del arte requieren que los modelos se comporten de cierta manera ante transformaciones de su entrada. Existen dos propiedades fundamentales que capturan dicho requerimiento; la invarianza y la equivarianza. La invarianza nos dice que la salida del modelo no es afectada por las transformaciones. La equivarianza permite que la salida sea afectada, pero de una manera controlada y útil. Si bien los modelos tradicionales de Redes Convolucionales son equivariantes a la traslación por diseño, no son ni invariantes a dicha transformación ni equivariantes a otras en los escenarios usuales de entrenamiento y uso. Existen dos opciones principales para otorgar invarianza o equivarianza a un modelo de red neuronal. La tradicional ha sido modificar el modelo para dotarlo de esas propiedades. La otra opción es entrenarlo con aumentación de datos utilizando como transformaciones el mismo conjunto al que se desea la invarianza o equivarianza.

6.3.- Python

El lenguaje Python es un lenguaje de programación moderno, está orientado a objetos, es muy sencillo de usar a la vez potente y de código abierto. Toda la información relativa a este lenguaje es libre, el sitio donde puedes encontrar todo lo relativo a este lenguaje.

El Python es un lenguaje de programación que se le suele comparar con otros lenguajes como el TLC, Perl, Écheme, Java o Ruby. Este lenguaje fue creado por Guido van Rossum basándose en otro leguaje de programación, el ABC. El nombre de este lenguaje proviene de los humoristas británicos Monty Python que tanto le gustaban a Guido van Rossum.

El Python es un lenguaje de programación de scripting. Los lenguajes scripting son aquellos lenguajes que usan un intérprete en vez de ser compilados. Es opuesto al Perl, lenguaje con el que rivaliza amistosamente. La mayoría de los

usuarios del Python le consideran como un lenguaje más limpio y elegante a la hora de programar.

El Python nos permite separar el programa en módulos, este lenguaje tiene una gran variedad de módulos estándar que se pueden utilizar para programar, o incluso como una base para aprender a programar en Python.

El Python es un lenguaje interpretado lo que ahorra muchísimo tiempo en la creación de programas puesto que no es preciso compilar su código. El interprete que usa el Python se puede utilizar de modo interactivo lo que nos permite experimentar con este lenguaje mientras programamos.

Este lenguaje es un lenguaje de programación que permite que podamos programar en varios estilos: Programación orientada a objetos, programación estructurada, programación funcional y programación orientada a aspectos. A esto se le conoce como lenguaje de programación multiparadigma.

6.4.- Pycharm

PyCharm es un IDE o entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica, integración con VCS / DVCS y soporte para el desarrollo web con Django, entre otras bondades. PyCharm es desarrollado por la empresa JetBrains y debido a la naturaleza de sus licencias tiene dos versiones, la Community que es gratuita y orientada a la educación y al desarrollo puro en Python y la Professional, que incluye más características como el soporte a desarrollo web con varios precios.

Características:

Asistencia y análisis de codificación, con finalización de código, sintaxis y resaltado de errores, integración delinter y arreglos rápidos

- Navegación de proyectos y códigos: vistas de proyectos especializados, vistas de estructura de archivos y saltos rápidos entre archivos, clases, métodos y usos
- PythonRefactoring: incluye renombrar, extraer método, introducir variable, introducir constante y otros.

- Soporte para frameworks web: Django, web2py y Flask
- Depurador integrado de Python
- Pruebas unitarias integradas, con cobertura line-by-line (línea por línea)
- Desarrollo de Python de Google App Engine
- Integración de control de versiones: interfaz de usuario unificada para Mercurial, Git, Subversión, Perforce y CVS con listas de cambios y combinación

6.5.- TensorFlow

TensorFlow es una biblioteca de software de código abierto para computación numérica, que utiliza gráficos de flujo de datos. Los nodos en las gráficas representan operaciones matemáticas, mientras que los bordes de las gráficas representan las matrices de datos multidimensionales (tensores) comunicadas entre ellos.

TensorFlow es una gran plataforma para construir y entrenar redes neuronales, que permiten detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos.

La arquitectura flexible de TensorFlow le permite implementar el cálculo a una o más CPU o GPU en equipos de escritorio, servidores o dispositivos móviles con una sola API. TensorFlow fue desarrollado originalmente por investigadores e ingenieros que trabajaban en el equipo de Google Brain Team, dentro del departamento de investigación de Machine Intelligence, con el propósito de llevar a cabo el aprendizaje automático y la investigación de redes neuronales profundas.

Sin embargo, el sistema es lo suficientemente general como para ser aplicable a una amplia variedad de otros dominios igualmente.

6.6 Keras

Keras es una biblioteca de Redes neuronales de código abierto escrita en Python. Es capaz de ejecutarse sobre TensorFlow, o Theano.

Está especialmente diseñada para posibilitar la experimentación en más o menos poco tiempo con redes de Aprendizaje profundo. Sus fuertes se centran en ser amigable para el usuario, modular y extensible.

Inicialmente fue desarrollada como parte de los esfuerzos de investigación del proyecto ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System).

En 2017, el equipo de TensorFlow de Google decidió ofrecer soporte a Keras en la biblioteca de core de TensorFlow

Keras ha sido concebido para actuar como una interfaz en lugar de ser una framework de machine learning standalone. Ofrece un conjunto de abstracciones más intuitivas y de alto nivel haciendo más sencillo el desarrollo de modelos de aprendizaje profundo independientemente del backend computacional utilizado.

Microsoft añadió un backend en CNTK a Keras también, disponible desde la CNTK v2.0

6.7.- OpenCV

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. OpenCV significa Open Computer Visión (Visión Artificial Abierta). Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en una gran cantidad de aplicaciones, y hasta 2020 se la sigue mencionando como la biblioteca más popular de visión artificial. Detección de movimiento, reconocimiento de objetos, reconstrucción 3D a partir de imágenes, son sólo algunos ejemplos de aplicaciones de OpenCV.

Su popularidad se debe a que es:

- libre, publicada bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación.
- multiplataforma, para los sistemas operativos GNU-Linux, Mac Os X, Windows y Android, y para diversas arquitecturas de hardware como X86, X64 (PC), ARM (celulares)
- documentada y explicada: la organización tiene una preocupación activa de mantener la documentación de referencia para desarrolladores lo más

completa y actualizada posible, ejemplos de uso de sus funciones y tutoriales accesibles al público no iniciado en visión artificial, además de difundir y fomentar libros y sitios de formación.

El proyecto pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multinúcleo. OpenCV puede además utilizar el sistema de primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel específicas para procesadores Intel (IPP).

6.8.- Anaconda

Anaconda es una Suite de código abierto que abarca una serie de aplicaciones, bibliotecas y conceptos diseñados para el desarrollo de la Ciencia de Datos con Python. En líneas generales Anaconda Distribution es una distribución de Python que funciona como un gestor de entorno, un gestor de paquetes y que posee una colección de 720 paquetes de código abierto

Anaconda Distribution se agrupa en 4 sectores o soluciones tecnológicas, Anaconda Navigator, Anaconda Project, Las bibliotecas de Ciencia de datos y Conda. Todas estas se instalan de manera automática y en un procedimiento muy sencillo.

Capítulo N° 7: Desarrollo de Arquitectura de Software

7.1.- Introducción

En esta sección se detallará todo lo referente al desarrollo de la arquitectura desde el modelo de CNN (Redes Neuronales Convolucionales), el desarrollo del artefacto de software de reconocimiento facial consta de tres etapas, detección de caras, definición y entrenamiento, aplicación del conocimiento, dejando así toda la arquitectura a desarrollar correctamente

Propósito

Este documento proporciona una visión general y completa de la arquitectura del sistema, usando una serie de puntos de vista arquitectónicos diferentes para representar diferentes aspectos del sistema. Se tiene la intención de captar y transmitir las decisiones importantes que se han hecho sobre la arquitectura del sistema.

Alcance

Los procesos o procedimientos que van a ser considerados en el diseño de la arquitectura de software son:

- Detección de caras.
- Definición y entrenamiento.
- Aplicación del conocimiento.

7.2.- Descripción

En la primera etapa se captura y detecta el rostro en una imagen procediendo después al acondicionamiento y normalización. Una vez obtenida la cara, el segundo paso se encarga de extraer características representativas de esta, formando así, un vector patrón que se utiliza en el algoritmo de reconocimiento etapa de clasificación para la identificación o verificación de la identidad de una persona con respecto a las personas registradas en el sistema.

A partir de la interfaz de usuario desarrollada, se puede capturar un conjunto de imágenes conformando una base de rostros, con la cual se pueden llevaron a

cabo las pruebas del sistema. Luego, se desarrollaron las tres etapas para el reconocimiento

7.3.- Representación arquitectónica

El problema del reconocimiento facial se puede definir como la comparación de una escena o secuencia de imágenes contra una plantilla o modelo facial. De forma general, la solución al problema del reconocimiento implica una serie de procesos: 1) detección facial, 2) normalización facial y extracción de características, 3) identificación o verificación. La variabilidad del entorno, el ruido, la oclusión de las zonas faciales y los requisitos de velocidad hacen que las tres etapas anteriores puedan llegar a ser altamente complejas figura 1.

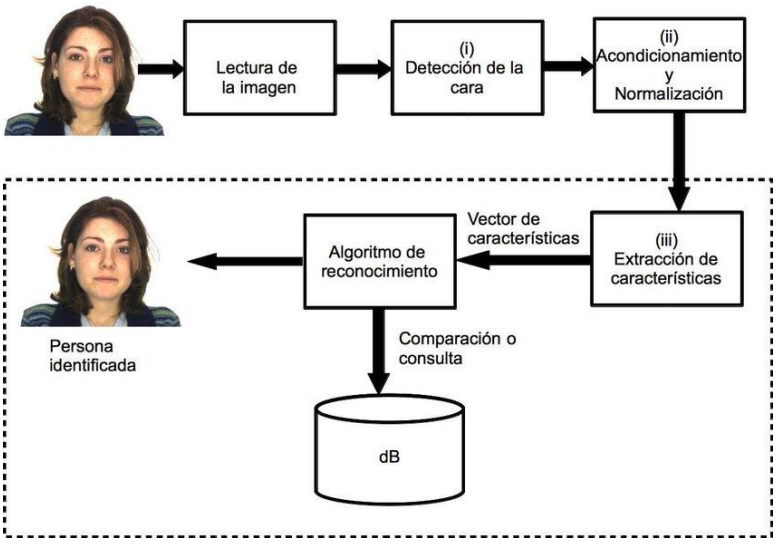


Figura: 1 Proceso del Reconocimiento Facial.
Fuente: (Carrero., 2010)

7.3.1.- Detección facial:

La detección del rostro supone el primer paso en cualquier sistema de reconocimiento facial automático. En este primer proceso existen dos objetivos principales localización de la región facial (si existe) y segmentación de la misma del rostro de la escena.

7.3.2.- Normalización Facial y extracción de características:

Es bien sabido que los sistemas de reconocimiento facial necesitan de forma precisa las posiciones de las características faciales El objetivo de los métodos de normalización y de extracción de características consiste en localizar de forma precisa las regiones faciales.

7.3.3.- Reconocimiento:

La última etapa del proceso hace referencia a la propia verificación o identificación del sujeto cuya imagen facial ha sido capturada. Es posible agrupar los métodos de reconocimiento en diferentes familias.

7.4.- Objetivos y Limitaciones de la Arquitectura

7.4.1.- Objetivo:

Con esta arquitectura se pretende solucionar un problema de seguridad con una solución tecnológica de reconocimiento facial.

7.4.2.- Limitaciones:

a) La arquitectura en su primera etapa de implementación no contara con la instalación de un motor de base de datos donde se almacenarán todos los rostros a ser registrados para las posibles consultas por parte del algoritmo.
b) La arquitectura solamente podrá funcionar bajo la plataforma de sistema operativo Windows 2010.

7.5.- Vista de un Caso de Uso

El modelo de casos de uso describe la funcionalidad propuesta del nuevo sistema. Un Caso de Uso representa una unidad discreta de interacción entre un usuario (humano o máquina) y el sistema. Un Caso de Uso es una unidad de trabajo significativo; por ejemplo crear una solicitud y modificar una solicitud son todos Casos de Uso. Cada Caso de Uso tiene una descripción que especifica la funcionalidad que se incorporará al sistema propuesto. Un Caso de Uso puede

'incluir' la funcionalidad de otro Caso de Uso o puede 'extender' otro Caso de Uso con su propio comportamiento. (Sparks, 2010)

<table>
<tr><td>RF- 04</td><td colspan="2">Identificar persona</td></tr>
<tr><td>Versión</td><td colspan="2">1.0</td></tr>
<tr><td>Actores</td><td colspan="2">Administrador</td></tr>
<tr><td colspan="3">Descripción
Identifica a una persona a partir de una imagen de su rostro.</td></tr>
<tr><td>Precondición</td><td colspan="2">Debe estar localizado el rostro de la imagen</td></tr>
<tr><td rowspan="7">Secuencia

Normal</td><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>Se presiona el botón “identificar”.</td></tr>
<tr><td>2</td><td>El sistema captura una imagen.</td></tr>
<tr><td>3</td><td>Se localiza el rostro en la imagen.</td></tr>
<tr><td>4</td><td>Se extrae las características de la imagen.</td></tr>
<tr><td>5</td><td>Se clasifica el patrón mediante una red neuronal.</td></tr>
<tr><td>6</td><td>El sistema informa la identidad de la persona.</td></tr>
<tr><td>Postcondición</td><td colspan="2">El sistema regresa al menú principal</td></tr>
<tr><td rowspan="2">Excepciones</td><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>El sistema no informa la identidad de la persona</td></tr>
<tr><td colspan="3">Frecuencia esperada
200 veces/día</td></tr>
</table>

7.6.- Realizaciones de Caso de Uso

El proceso o secuencia de pasos en forma ordenada desde que el rostro de una persona es capturado por la cámara de seguridad hasta que sea identificada es un desarrollo secuencial que tiene que cumplirse paso a paso en forma lógica como se muestra en la figura 1.

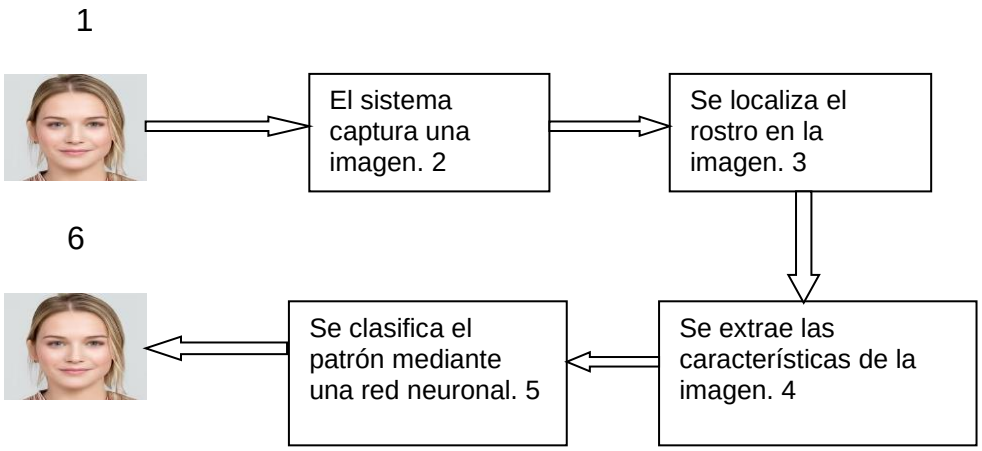


Figura: 1 Realización del caso de Uso Identificar Persona

Autoría: propia

7.7.- Vista Lógica

La vista lógica es la encargada de mostrar los principales elementos de diseño y sus relaciones en forma independiente de los detalles técnicos y de como la funcionalidad será implementada en la plataforma de ejecución (Londoño, 2015).

En la figura 1 se muestra la vista lógica de los procesos a realizar del proyecto.

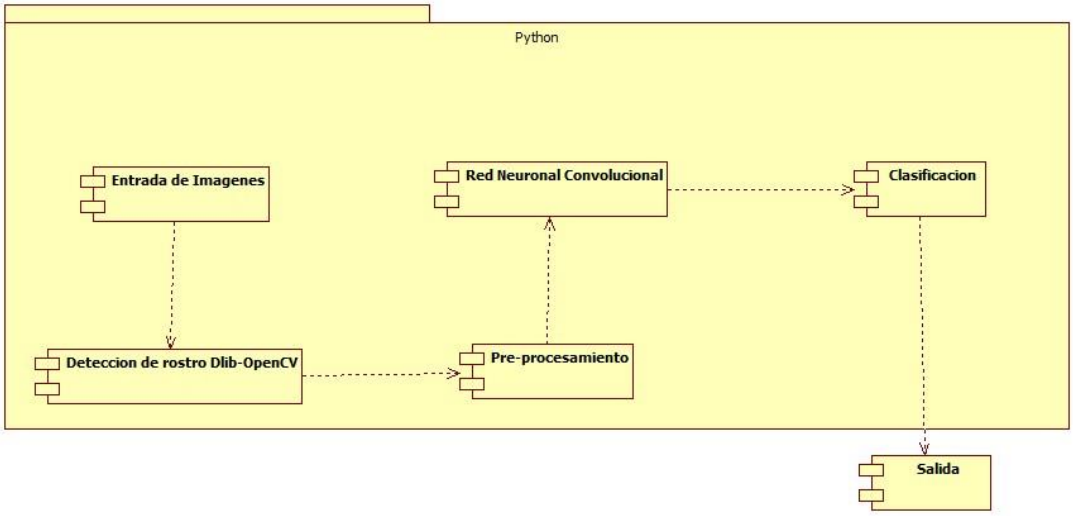


Figura: 1 Vista lógica de Reconocimiento Facial

Fuente: Autoría propia

7.8.- Descripción

7.8.1.-Entrada de imágenes: en esta vista lógica de la arquitectura, por medio de una cámara de seguridad realiza la captura del rostro de la persona.

7.8.2.-Detección de rostro: en esta vista lógica de la arquitectura, recibe la imagen y por medio de un algoritmo matemático pre-entrenado de la biblioteca OpenCV se detecta el rostro de la persona.

7.8.3.-Pre-procesamiento: en esta vista lógica de la arquitectura, el rostro detectado pasa por un proceso de trasformación en donde las imágenes vienen en tipo de calidad RGB o colores y como el proceso necesita pasar a blanco y negro para ser utilizada por CNN y luego se extraen las características o patrones del rostro.

7.8.4.-Red neuronal convolucional: en esta vista lógica de la arquitectura, las características o patrones del rostro, son enviadas al algoritmo CNN para que una vez definida la estructura y entrenamiento se pueda registrar

los datos de los patrones.

7.8.5.-Clasificación: en esta vista lógica de la arquitectura, se realiza la búsqueda de los patrones de los rostros que se encuentran registrados.

7.8.6.-Salida: en esta vista lógica de la arquitectura, se visualiza el rostro de la persona que coinciden con los parámetros de coincidencia.

7.9.- Paquetes de Diseño Arquitectónicamente Importantes

Los diagramas de arquitectura de software son una manera fantástica de comunicar cómo planea construir un sistema de software (diseño inicial) o cómo funciona un sistema de software existente (documentación retrospectiva, intercambio de conocimientos y aprendizaje)

El diagrama de paquetes es muy útil para darse una de la estructura del programa en este nivel solo estamos interesados en mirar por dentro, en la siguiente figura: 1 se muestra los paquetes de diseños arquitectónicos del proyecto de reconocimiento facial.

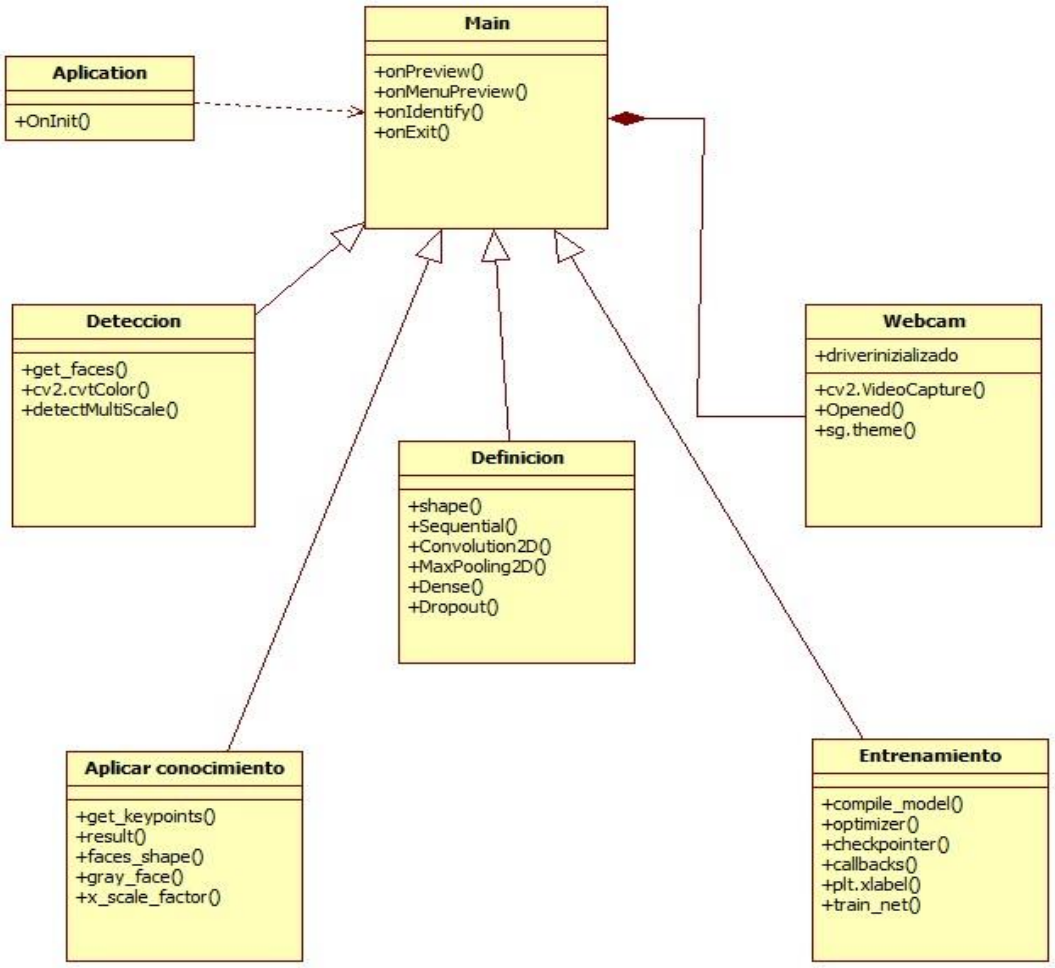


Figura: 1 Diagrama de Paquetes de Reconocimiento Facial
Autoria propia

En este diagrama de paquetes se muestran como la clase Main con sus respectivos procesos tiene una dependencia directamente con las demás clases y sus respectivos procesos, dentro de las clases importantes tenemos:

- Clase Aplication
- Clase webcam
- Clase detección
- Clase definición
- Clase entrenamiento

- Aplicar conocimiento

7.10.- Vista de Procesos

La vista de procesos representa los flujos de trabajo paso a paso del sistema, en la siguiente figura 1 se aprecia el diagrama de actividades para el sistema de reconocimiento facial, en el cual interactúa el administrador.

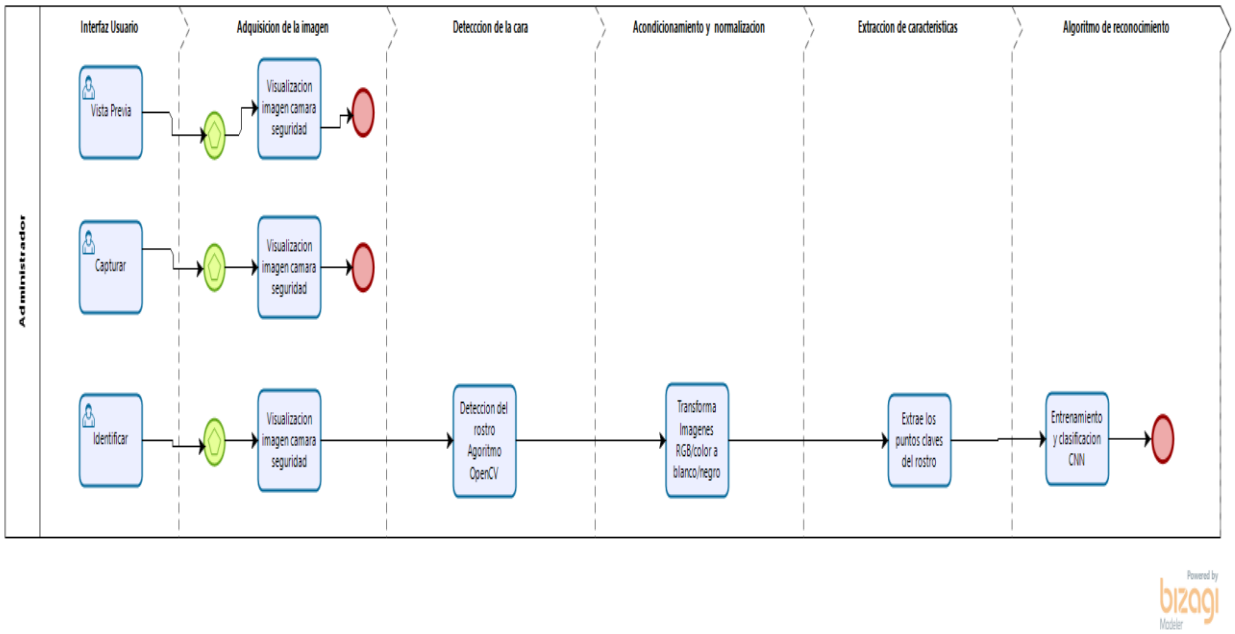


Figura: 1 Vista de proceso Reconocimiento Facial

Fuente: Autoría propia

7.11.- Ver Despliegue

El sistema de reconocimiento facial esta diseñado para que se ejecute en un terminal escritorio, la capa de presentacion(vista y controlador) se ejecutan en el computador local, las capas de negocio y persistencia que se ejecuta mediante el servicio de python tambien dentro del mismo proceso local según la figura 1.

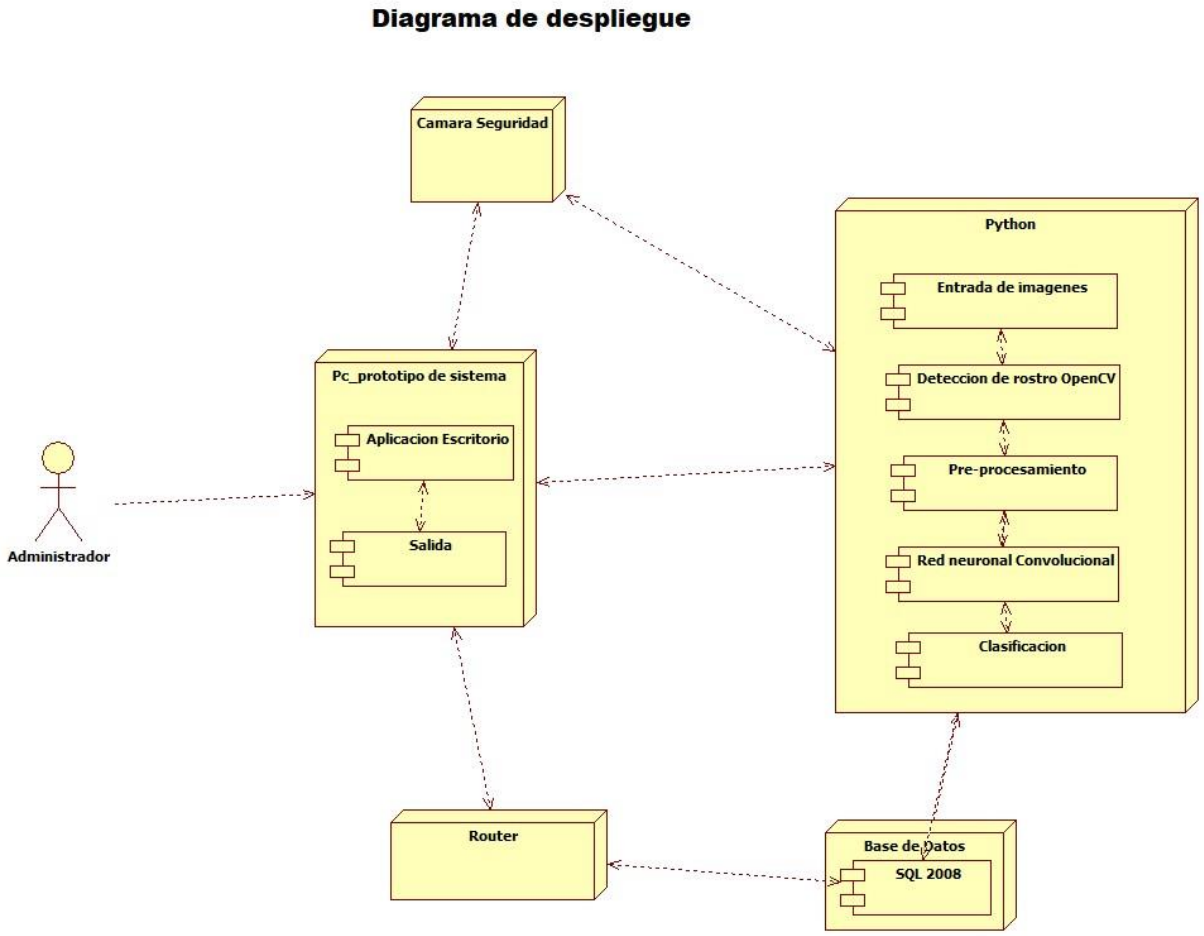
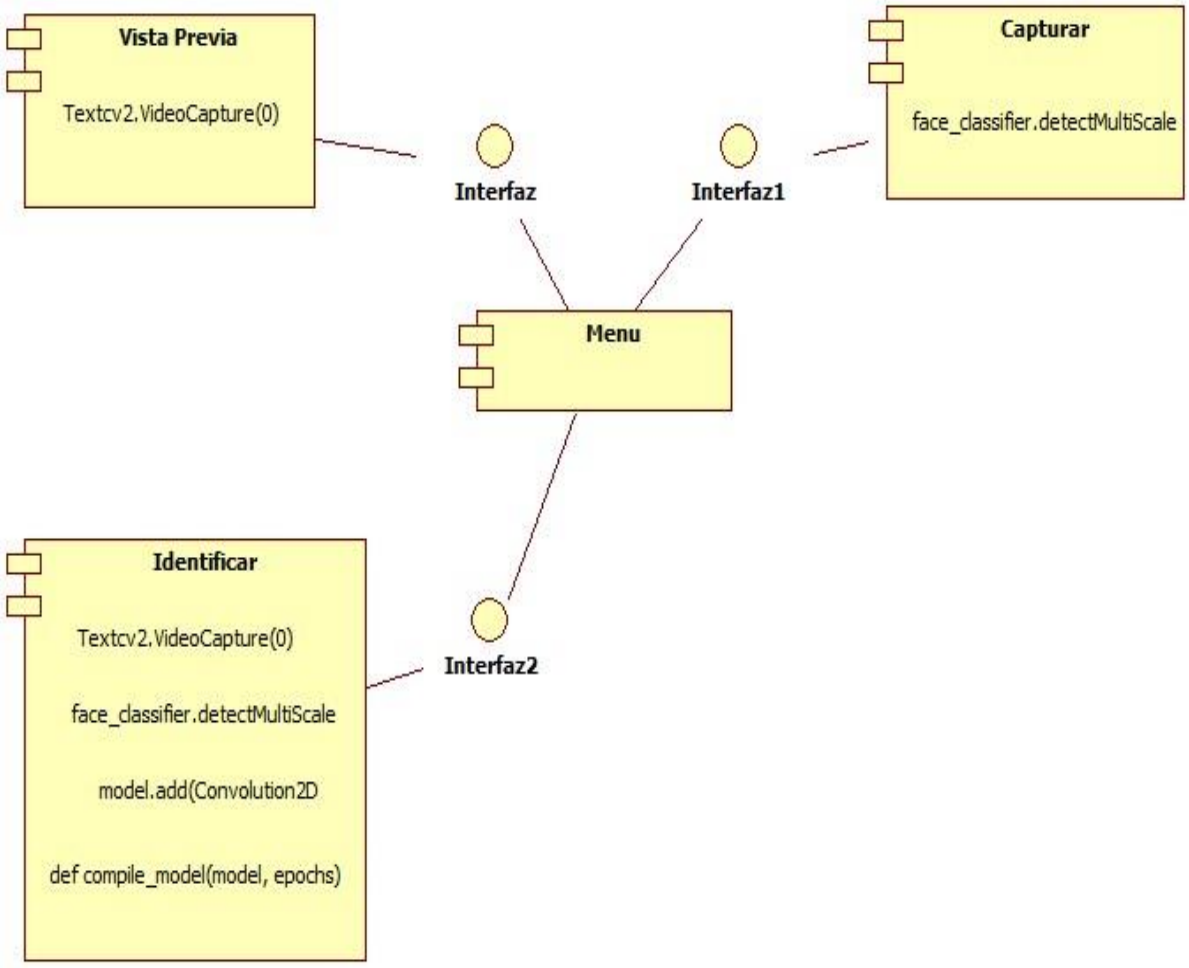


Figura: 1 Diagrama de despliegue Reconocimiento Facial.
Fuente: Autoría propia

7.12.- Descripción

La descripción de un modelo de despliegue en la cual se modela la arquitectura en tiempo de ejecución, el sistema de reconocimiento facial. Esto muestra la configuración de los elementos de software en nodos como son:
La vista previa, la captura y la identificación la cual tiene interfaz con el menú.



7.13.- Capas

La arquitectura del sistema de reconocimiento facial se basa en capas en donde se enfoca la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades, el rol indica el modo y tipo de interacción con otras capas como son:

- La capa se presentación.
- Lógica de negocio.
- Capa de datos.

7.13.1.- Capa de Presentación

```
import PySimpleGUI as sg
from datetime import date
import cv2
def main():
   #Conectamos a la webcam
   camara = cv2.VideoCapture(0)
   #Elegimos un tema de PySimpleGUI
   sg.theme('GreenTan')
   #Definimos los elementos de la interfaz grafica
   layout = [[sg.Image(filename='', key='-image-')],
    [sg.Button('Vista       Previa'),sg.Button('Capturar'),sg.Button('Identificar')]]
   #Creamos la interfaz grafica
   window = sg.Window('Sistema de Reconocimiento Facial',
                      layout,
                      no_titlebar=False,
                      location=(0, 0))

   image_elem = window['-image-']
   numero=0
   while camara.isOpened():
         #Obtenemos informacion de la interfaz grafica y video
         event, values = window.read(timeout=0)
         ret, frame = camara.read()

        #Si salimos
          if event in ('Exit', None):
```

```
            break

        #Si tomamos foto
        elif event == 'Capturar':
            numero=numero+1
            ruta = sg.popup_get_folder(title='Guardar Fotografia',
message="Carpeta destino")
            cv2.imwrite(ruta + "/" + str(date.today()) + "_" + str(numero) + ".png",
frame)
        elif event == 'Vista Previa':
            print ("Esto es Una Prueba")

        elif event == 'Identificar':
            print ("Falta módulo identificar")

        #Mandamos el video a la GUI
        imgbytes = cv2.imencode('.png', frame)[1].tobytes()  # ditto
        image_elem.update(data=imgbytes)
        #numero = numero + 1
```

7.13.2.- Lógica de Negocio

a) Detectar las caras a partir de modelos definidos en OpenCV

```
    def get_faces(image):
        """
        It returns an array with the detected faces in an image
        Every face is defined as OpenCV does: top-left x, top-left y, width and
    height.
        """
        # To avoid overwriting
        image_copy = np.copy(image)

        # The filter works with grayscale images
        gray = cv2.cvtColor(image_copy, cv2.COLOR_RGB2GRAY)
```

```
    # Extract the pre-trained face detector from an xml file
    face_classifier =
cv2.CascadeClassifier('detectors/haarcascade_frontalface_default.xml')

    # Detect the faces in image
    faces = face_classifier.detectMultiScale(gray, 1.2, 5)

def draw_faces(image, faces=None, plot=True):

def plot_image_with_keypoints(image, image_info)
```

b) Definición y entrenamiento de una Red Neuronal Convolucional

```
shape = (96,96)
model = Sequential()
model.add(Convolution2D(16,(2,2),padding='same',input_shape=(96,96, 1),
activation='relu'))
model.add(MaxPooling2D(pool_size=3))
model.add(Convolution2D(32,(3,3),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=3))
model.add(Dropout(0.2))
model.add(Convolution2D(64,(3,3),padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=3))
model.add(Dropout(0.2))
model.add(Convolution2D(128,(3,3),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=3))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(256,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(30))
def compile_model(model, epochs):
def show_training_validation_loss(hist, epochs):
```

c) Aplicar el conocimiento de la CNN

```
def get_keypoints(image, faces=None):

    # list of pairs (face, keypoints)
    result = []

    if faces is None:
        faces = get_faces(image)

    # Same size than training/validation set
    faces_shape = (96, 96)

    # To avoid overwriting
    image_copy = np.copy(image)

    # For each face, we detect keypoints and show features
    for (x,y,w,h) in faces:

        # We crop the face region
        face = image_copy[y:y+h,x:x+w]

        # Face converted to grayscale and resize (our CNN receives images of
96x96x1)
        gray_face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        resize_gray_face = cv2.resize(gray_face, faces_shape) / 255

        # Formatting x inputs. Inputs will have format of (1, 96, 96, 1)
        inputs = np.expand_dims(np.expand_dims(resize_gray_face, axis=-1),
axis=0)

        # Get keypoints result
        predicted_keypoints = model.predict(inputs)

        # All keypoints in a single flat array. We will retrieve keypoints as (x,y) with
(idx, idx+1) values.
        predicted_keypoints = np.squeeze(predicted_keypoints)
```

```
        keypoints = []
        for idx in range(0, len(predicted_keypoints), 2):
            # Scale factor (revert scale)
            x_scale_factor = face.shape[0]/faces_shape[0]
            y_scale_factor = face.shape[1]/faces_shape[1]

            # Offset of the center of the scatter
            x_center_left_offset = predicted_keypoints[idx] * faces_shape[0]/2 +
faces_shape[0]/2
            y_center_left_offset = predicted_keypoints[idx + 1] * faces_shape[1]/2 +
faces_shape[1]/2

            x_center = int(x + (x_scale_factor * x_center_left_offset))
            y_center = int(y + (y_scale_factor * y_center_left_offset))

            keypoints.append([x_center, y_center])

        result.append([(x,y,w,h), keypoints])

    return result

def show_image_and_features(image_path):
```

7.13.3. - Capa de Datos

```
image = read_image('images/breaking_bad.jpg')
faces = get_faces(image)
print("Faces detected: {}".format(len(faces)))
draw_faces(image, faces)
```

Capítulo N° 8: Descripción General del Producto

Por estos problemas identificados es que se decidió empezar a desarrollar un artefacto de Software de reconocimiento facial, la cual en su primera etapa solo se desarrollarán dos módulos, el primero visualiza la imagen de la persona y da la posibilidad de tomarle una foto y almacenarla , el segundo módulo detecta el rostro de la persona y la identifica, en una siguiente etapa se incluirán más funcionalidades como por ejemplo tener un motor de base de datos y ser conectada a diferentes bases de datos de instituciones.

Todo esto se realizará en una interfaz sencilla y básica en un comienzo para que así facilitar su uso a los usuarios que lo utilicen.

La propuesta de solución se hace en función de las causas identificadas y que originan el problema, por lo anterior, es que se propone como solución:

- El desarrollo de una aplicación que registre la presencia de personas en el local mediante la toma de fotos y posibilidad de almacenarlas. Solución que ha de ser integrada a la actual infraestructura de cámaras de vigilancia que hoy existe.
- El desarrollo de una aplicación que reconozca las personas autorizadas que ingresan al local.

a).- Alternativas y Competencias

Una alternativa a la solución son las cámaras de seguridad, las cuales están grabando en todo momento y almacenando el video en un medio de persistencia, incluso puede que algunos casos estén agentes de seguridad monitoreando las cámaras en casos potencialmente peligrosos para la organización. Pero todos estos sistemas que hay ahora, no dan una solución real ya que todos o casi todos necesitan que el administrador sea el que este monitoreando y en muchos casos estos ven el teléfono o se ponen a hacer otras funciones que al final en la mayoría solo se dan cuenta una vez ocurrido el evento delictivo, y lo único que resulta con esto es poder tener grabado el hecho.

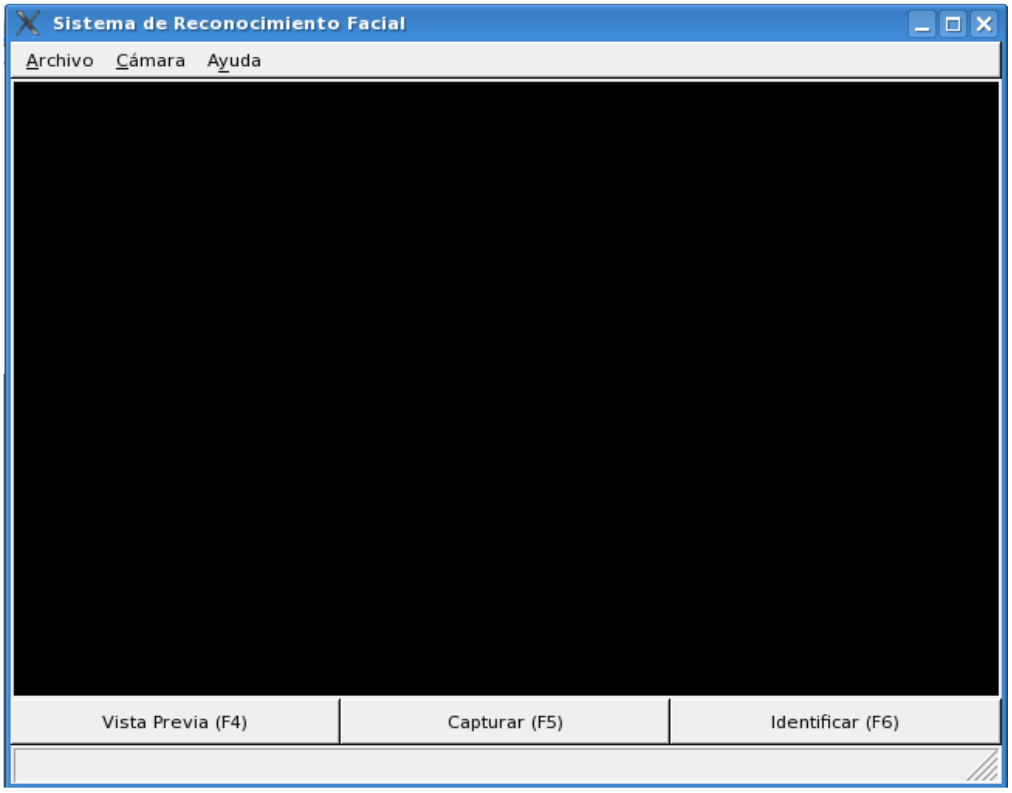
La solución nace para poder ser un aporte para la sociedad que posee un grado de desconfianza y temor a hechos delictivos, otorgando una mitigación a estos hechos.

8.1.- Características del Producto

Vista Previa : inicializa la cámara, y reproduce en el recuadro negro lo que esta captura.

Capturar : captura una imagen de la vista previa visualizándola en otra ventana, en la cual se puede optar por guardar la imagen tomada.

Identificar : captura una imagen de la vista previa, aplica las tres etapas para el reconocimiento e indica quien es el usuario que se encuentra frente a la cámara.



8.2.- Otros Requisitos del Producto

a).- Implementación de algoritmo para la detección de rostro.

b).- Implementación de algoritmo para el reconocimiento facial.

c).- Añadir y eliminar imágenes desde la base de datos.

d).- Mostrar en una pantalla de Windows la cámara web para la prueba de software.

e).- Limitar el ingreso al sistema mediante una clave y contraseña.

f).- Obtener una cámara IP con buenas resoluciones de imagen.

Capítulo N° 9: Análisis y discusión de resultados

Se realizo un plan de prueba en donde se aplico el estándar IEEE 829 lo que se describe a continuación.

9.1.- Módulos a probar:

MÓDULO ENTRADA

1) Importación de bibliotecas necesarias y ser cargadas en memoria.
2) Conectar la cámara de seguridad.
3) Elegir un tema de característica de biblioteca de interfaz.
4) Definir los elementos de la interfaz gráfica.
5) Crear la interfaz gráfica.

MÓDULO PROCESO

1) Información de la interfaz gráfica y video.
2) Captura de fotografía del rostro.
3) Aumenta la variable acumulador.

MÓDULO SALIDA

1) Indica la ruta donde tenemos que guardar.
2) Guarda la fotografía con la fecha actual y numerada.

9.2.- Características a probar Interfaz de Usuario.

- ✓ Clara: La claridad de la interfaz.
- ✓ Concisa: En perfecta armonía con la cualidad anterior.
- ✓ Coherente: La coherencia ayudará a los usuarios a desarrollar patrones de uso.
- ✓ Legible: Usar un lenguaje simple para ayudar a la rápida lectura.
- ✓ Interactiva: Una buena interfaz gráfica tiene que ser rápida y además ofrecer información.
- ✓ Flexible: Una buena interfaz debería posibilitar al usuario restaurar.
- ✓ Familiar: El usuario debe sentirse familiarizado con la mayoría de los elementos.

9.3.- Criterio paso/fallo para cada elemento que se probara.

Se aplicará el siguiente criterio para la siguiente prueba de paso/fallo.

Paso	Pasos de prueba	Datos de prueba	Resultado esperado	Resultado actual	Estado paso/fallo
1	Importación de bibliotecas necesarias y ser cargadas en memoria				Pasar
2	Conectar la cámara de seguridad	Botón vista previa	El usuario debe poder visualizar imagen	El usuario debe poder visualizar imagen	Pasar
3	Elegir un tema de característica de biblioteca de interfaz				Pasar
4	Definir los elementos de la interfaz gráfica				Pasar
5	Crear la interfaz gráfica				Pasar
6	Información de la interfaz gráfica y video				Pasar

7	Captura de fotografía del rostro	Botón capturar	El usuario debe poder tomar una fotografía	El usuario debe poder tomar una fotografía	Pasar
8	Aumenta la variable acumulador				Pasar
9	Indica la ruta donde tenemos que guardar				Pasar
10	Guarda la fotografía con la fecha actual y numerada	Botón guardar	El usuario debe poder almacenar la fotografía	El usuario debe poder almacenar la fotografía	Pasar

9.4.- Ejecución Prueba Grafos:

La justificación de la funcionalidad para poder validarla donde se calcula la complejidad ciclomatica.

a) Código a probar.

```
1: import PySimpleGUI as sg
1: from datetime import date
1: import cv2
1: def main():
       #Conectamos a la webcam
1:     camara = cv2.VideoCapture(0)
       #Elegimos un tema de PySimpleGUI
1:     sg.theme('DarkGreen5')
       #Definimos los elementos de la interfaz grafica
1:     layout = [[sg.Image(filename='', key='-image-')],
           [sg.Button('Tomar Fotografia'),sg.Button('Salir')]]
       #Creamos la interfaz grafica
1:     window = sg.Window('Fotografia',
                   layout,
                   no_titlebar=False,
```

```
                location=(0, 0))

1:   image_elem = window['-image-']
1:   numero=0
2:  while camara.isOpened():
          #Obtenemos informacion de la interfaz grafica y video
3:         event, values = window.read(timeout=0)
4:         ret, frame = camara.read()

          #Si salimos
5:        if event in ('Exit', None):
6a:            break

           #Si tomamos foto
6b:       elif event == 'Tomar Fotografia':
7:            numero=numero+1
8:                          ruta  =  sg.popup_get_folder(title='Guardar  Fotografia',
message="Carpeta destino")
9:            cv2.imwrite(ruta + "/" + str(date.today()) + "_" + str(numero) + ".png",
frame)

10:       if not ret:
11:           break
          #Mandamos el video a la GUI
12:       imgbytes = cv2.imencode('.png', frame)[1].tobytes()  # ditto
13:       image_elem.update(data=imgbytes)
```

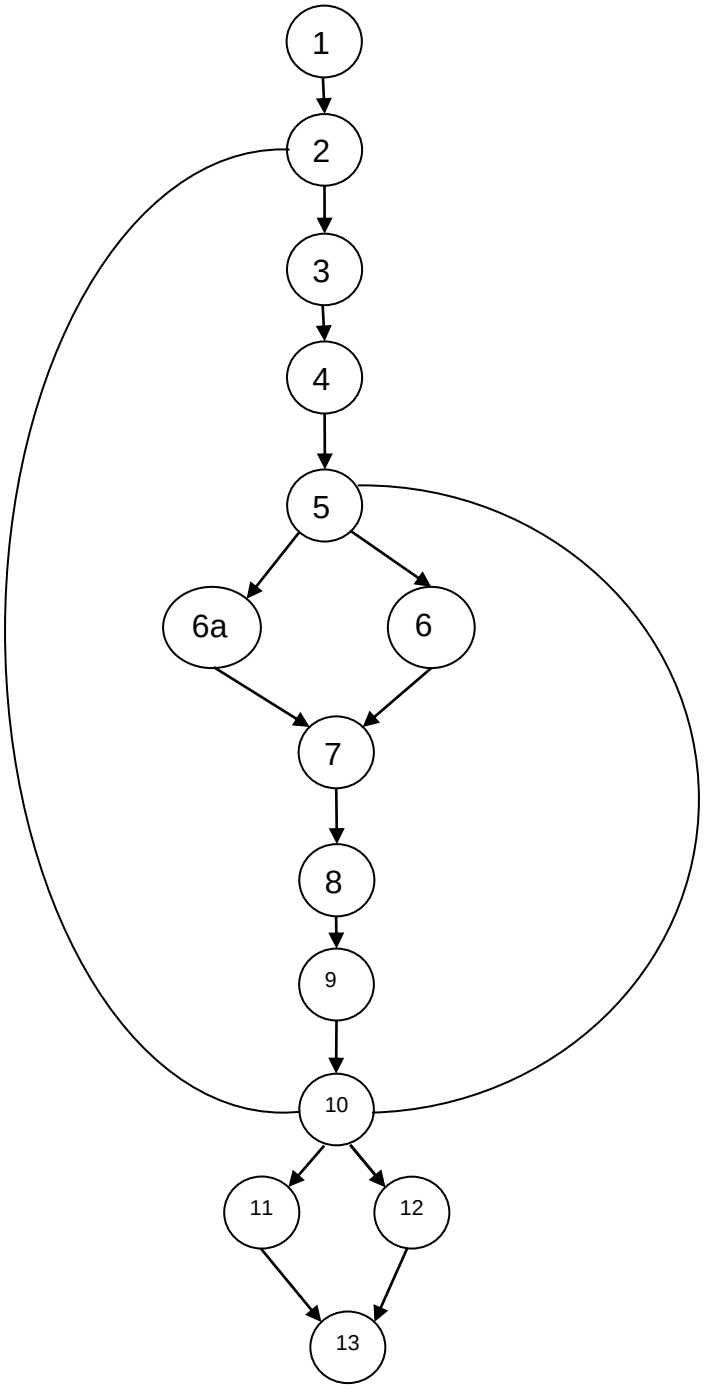
b) Caminos ejecución del código.

Camino 1: 1- 2 - 3 - 4 – 5 – 6b – 7 – 8 – 9 - 10
Camino 2: 1- 2 - 3 - 4 – 5 – 6a – 7 – 8 – 9 - 10
Camino 3: 10 – 12 - 13
Camino 4: 10 – 11 - 13

c) Calculo complejidad ciclomática usando la fórmula V (G)= A – N + 2.

V (G)= A – N + 2
V (G)= 17 – 14 + 2
V (G)= 5

A = ARISTAS
N = NODOS
V (G) = 4 NODOS PREDICADO + 1V
(G) = 5



Capítulo N° 10: Conclusiones

10.1.- Objetivos propuestos:

1) Captura: captura una imagen de la vista previa visualizándola en otra ventana, en la cual se puede optar por guardar la imagen tomada.

   Sí se logó este objetivo, construyendo el módulo la cual es un requerimiento solicitado por el cliente lo que puede ser revisado en la sección 7.13.1:(capa de presentación)

2) Detección: detectar rostros de personas

   Sí se logró el objetivo, construyendo el módulo lo que puede ser revisado en la sección 7.13.2: (capa de negocio)

3) Definición y entrenamiento: la estructura de una red neuronal convolucional.

   Sí se logró el objetivo, construyendo el módulo lo que puede ser revisado en la sección 7.13.2: (capa de negocio)

4) Aplicar el conocimiento: red neuronal convolucional.

   Sí se logró el objetivo, construyendo el módulo lo que puede ser revisado en la sección 7.13.2: (capa de negocio).

5) Reconocimiento de rostro de personas

   Sí se logró el objetivo, construyendo el módulo lo que puede ser revisado en el Anexo.

10.2.- Valoración Personal

La realización de este proyecto ha resultado ser una labor costosa. No obstante, también ha resultado ser una gran experiencia a nivel de aprendizaje propio.

La cantidad de conceptos nuevos de inteligencia artificial y aprendizaje automático que se han tenido que analizar para llevar a cabo la realización del proyecto han resultado muy de mi interés.

Este tipo de conceptos, durante la carrera, no han sido estudiados con profundidad. Pues pertenecen a una especialización completamente lejana a la que durante estos años he cursado.

Respecto a las redes neuronales convolucionales, mi valoración es que tienen un potencial muy elevado pero un coste computacional alto que con mis recursos ha sido difícil manejar. Para empresas con ordenadores más potentes, este campo puede agilizar muchos procesos y también reinventar algunos de ellos.

El sistema realizado cumple con los objetivos esperados y propuestos logrando un rendimiento aceptable dado el ambiente en el cual se presentó el problema. Este presenta una interfaz gráfica sencilla y básica sin menús que permitan operar fácilmente tanto a un administrador del sistema a la hora de tomar fotos como a un usuario que intenta identificarse. A través de los experimentos se demostró que el clasificador se desempeñó con los estándares básicos, logrando un reconocimiento aceptable, en las pruebas sometidas.

10.3.- Futuras Ampliaciones y Mejoras

Con la idea de mejorar el desempeño general del sistema, una mejora estaría dada por un montaje dedicado en el lugar de instalación del mismo. Esto implicaría colocar un fondo fijo de color blanco y un flash como fuente de luz para lograr luminosidad constante permitiendo, de esta manera, obtener una mejor calidad en las imágenes capturadas.

Por otro lado, una ampliación del sistema estaría dada por el desarrollo de un módulo de administración de usuarios a fin de automatizar el proceso de carga de los mismos. Este incluiría funcionalidades para el registro de nuevos

usuarios, dar de baja aquellos que ya no se requieran en el sistema, así como también asociar las imágenes capturadas a los usuarios correspondientes.

Bibliografía

1. Dahua y su cámara de reconocimiento facial disponible en Argentina. *Dahua y su cámara de reconocimiento facial disponible en Argentina.* [En línea] 24 de 03 de 2018. [Citado el: 17 de 07 de 2020.] https://infosertec.com.ar/2018/03/24/dahua-y-su-camara-de-reconocimiento-facial-disponible-en-argentina/.

2. **Amos, Brandon.** OpenFace: A general-purpose face recognition library with mobile applications. *OpenFace: A general-purpose face recognition library with mobile applications.* [En línea] 11 de 10 de 2006. [Citado el: 10 de Julio de 2020.] https://www.semanticscholar.org/paper/OpenFace%3A-A-general-purpose-face-recognition-with-Amos-Ludwiczuk/82e66c4832386cafcec16b92ac88088ffd1a1bc9.

3. **Calderon, Marks.** Prototipo robótico para autenticación por comparación de proporciones faciales . *Prototipo robótico para autenticación por comparación de proporciones faciales .* [En línea] 16 de 10 de 2018. [Citado el: 17 de 07 de 2020.] http://revistas.uni.edu.pe/index.php/tecnia/article/view/561/1081.

4. **Calvo, Diego.** Red Neuronal Convolucional CNN. *Red Neuronal Convolucional CNN.* [En línea] 20 de 07 de 2017. [Citado el: 17 de 07 de 2020.] https://www.diegocalvo.es/red-neuronal-convolucional/.

5. **Carlsson, Gunnar.** Using Topological Data Analysis to Understand the Behavior of Convolutional Neural Networks. *Using Topological Data Analysis to Understand the Behavior of Convolutional Neural Networks.* [En línea] 21 de 06 de 2018. [Citado el: 17 de 07 de 2020.] https://www.ayasdi.com/blog/artificial-intelligence/using-topological-data-analysis-understand-behavior-convolutional-neural-networks/.

6. **Daza, Stefan.** Face Landmarks Detector con Dlib y OpenCV. *Face Landmarks Detector con Dlib y OpenCV.* [En línea] 09 de 11 de 2017. [Citado el: 17 de 07 de 2020.] http://acódigo.blogspot.com/2017/11/face-landmarks-detector-con-dlib-y.html.

7. **Merchan, Fernando.** Histograma de Gradientes Orientados (HOG) y características tipo Haar para la detección de sonrisas. En este contexto real... *Histograma de Gradientes Orientados (HOG) y características tipo Haar para la detección de sonrisas. En este contexto real...* [En línea] 30 de 12 de 2014. [Citado el: 12 de Julio de 2020.] https://www.researchgate.net/figure/Figura-6-

Calculo-de-gradientes-en-esquema-HOG-para-un-rostro-con-sonrisa-abierta-bien_fig6_272784648.

8. —. Mejoras en el entrenamiento de esquemas de detección de sonrisas basados en AdaBoost. *Mejoras en el entrenamiento de esquemas de detección de sonrisas basados en AdaBoost.* [En línea] 12 de 12 de 2014. [Citado el: 17 de 07 de 2020.] https://www.researchgate.net/publication/272784648_Mejoras_en_el_entrenamiento_de_esquemas_de_deteccion_de_sonrisas_basados_en_AdaBoost.

9. **Silvercorp.** Distancia euclidiana en Python. *Distancia euclidiana en Python.* [En línea] 30 de 10 de 2012. [Citado el: 17 de 07 de 2020.] https://silvercorp.wordpress.com/2016/09/15/distancia-euclidiana-en-python/.

10. **Zaragoza, Omar Emilio Contreras.** https://pdfs.semanticscholar.org/f379/d8a0922b090b6bc631aa4da3ac17a27c983d.pdf. *Desarrollo de una red.* [En línea] 27 de 01 de 2018.

11. **Martin, Miguel Vargas.** The ACM Digital Library is published by the Association for Computing Machinery. *Detection of Subconscious Face Recognition Using Consumer-Grade Brain-Computer Interfaces.* [En línea] 01 de 08 de 2016. [Citado el: 21 de 07 de 2020.] https://dl.acm.org/doi/10.1145/2955097#d1274085e1.

12. **Vincent, James.** The Verge. *Chinese police are using facial recognition sunglasses to track citizens.* [En línea] 08 de 02 de 2018. [Citado el: 21 de 07 de 2020.] https://www.theverge.com/2018/2/8/16990030/china-facial-recognition-sunglasses-surveillance.

13. **Loffler, Gunter.** Sciencie Direct. *Contributions of individual face features to face discrimination.* [En línea] 12 de 08 de 2017. [Citado el: 21 de 07 de 2020.] https://www.sciencedirect.com/science/article/pii/S0042698917301165.
14. Wenlai Zhao. *The ACM Digital Library is published by the Association for Computing Machinery. .* [En línea] 12 de 03 de 2018. [Citado el: 21 de 07 de 2020.] https://dl.acm.org/doi/10.1145/3177885.

15. **Wiesel, Torsten.** The Neural Basis of Visual Perception. *The Rockefeller University .* [En línea] 10 de 01 de 1981. [Citado el: 21 de 07 de 2020.] http://centennial.rucares.org/index.php?page=Neural_Basis_Visual_Perception.

16. **Hernández-Mendo, Pastrana-Brincones y A.** Algoritmos de clasificación y redes neuronales en la observación automatizada de registros. *Universidad de Murcia.* [En línea] 18 de 07 de 2014. [Citado el: 21 de 07 de 2020.]

http://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S1578-84232015000100003.

17. **Jimenez, Lic. Geol. Francisco Crespo.** ¿Deep Learning… y clasificación de imágenes? [En línea] 28 de 08 de 2019. [Citado el: 21 de 07 de 2020.] https://idecor.cba.gov.ar/introduccion-a-las-redes-neuronales-convolucionales-para-clasificacion-de-imagenes/.

18. **Carrillo, Anay.** Multimedia de Apoyo a la Enseñanza de la Metodología Rup. *Multimedia de Apoyo a la Enseñanza de la Metodología Rup.* España : Editorial Academica Espanola, 2011, pág. 92.

19. **Carrero., D.** Prestaciones de la Normalización del Rostro en el Reconocimiento Facial. *Prestaciones de la Normalización del Rostro en el Reconocimiento Facial.* [En línea] 13 de 05 de 2010. [Citado el: 23 de Julio de 2020.] http://jrbp10.unizar.es/papers/S5.C3.pdf.

20. **Sparks, Geoffrey.** Introducción al modelado de sistemas de software. [En línea] 15 de 12 de 2010. [Citado el: 24 de 07 de 2020.] http://www.sparxsystems.com.ar/downloads/whitepapers/El_Modelo_de_Casos_de_Uso.pdf.

21. **Londoño, Jesús Andrés Hincapié.** MDSD multi-plataforma: más allá de la vista funcional*. [En línea] 15 de 12 de 2015. [Citado el: 24 de 07 de 2020.] http://www.scielo.org.co/pdf/rium/v15n29/1692-3324-rium-15-29-00141.pdf.

Anexos

Código Reconocimiento de rostro

```
import os
import cv2
import imutils
import argparse
import sys
from glob import glob
# Importing from the opencv_face_recognition module in the folder
from opencv_face_recognition import face_recognition

'''
Here we load the face recogniser
'''
face_recogniser = face_recognition()

# Create a list of training images and labels
labels = []
images = glob('./photos/*/*', recursive=True)
for filename in images:
    labels.append(filename.split(os.path.sep)[-2].title())

# Train using this dataset
face_recogniser.train(images, labels)

# Now start video feed
print('Checking camera..')

# Edit 0 to './filename.mov' to perform facial recognition on video streaming
cam = cv2.VideoCapture('http://192.168.0.22:1200/video.cgi')
if ( not cam.isOpened() ):
    print ("no cam")
    sys.exit()
print ("cam: present")
```

```
# Video frame parameters
cv2.namedWindow('AI Project', cv2.WND_PROP_FULLSCREEN)
cv2.setWindowProperty('AI Project', cv2.WND_PROP_FULLSCREEN,
cv2.WINDOW_FULLSCREEN)

# Loop until q is pressed
while True:
    ret, frame = cam.read()
    labelled_frame = face_recogniser.recognise(frame)
    cv2.imshow('AI Project', labelled_frame)
    if cv2.waitKey(20) & 0xFF == ord('q'):
        break

# When everything done, release the capture
cam.release()
cv2.destroyAllWindows()
```