Universidad Autónoma Metropolitana
Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Estudio del desempeño de algoritmos de reconocimiento facial mediante una plataforma basada en OpenCV

Proyecto Tecnológico

Trimestre 2022 Invierno

Presentado por:

Carlos Daniel Miranda Viloria

Para obtener el grado de:
Ingeniero en Computación

Dr. Ruslan Gabbasov
Profesor Asociado
Departamento de Sistemas

Dr. Jesús Isidro González Trejo
Profesor Titular
Departamento de Sistemas

Declaratoria

Yo, Carlos Daniel Miranda Viloria, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Carlos Daniel Miranda Viloria

Yo. Ruslan Gabbasov, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Dr. Ruslan Gabbasov

Yo. Jesús González Trejo, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Índice de imágenes

Índice de capítulos

Contenido

Resumen

En este proyecto de integración se estudió el rendimiento de diversos algoritmos estadísticos de reconocimiento facial con el fin de poder realizar una comparativa en diversos entornos no favorable de captura. Se estudiaron los algoritmos SIFT, Eigenfaces, y KNN aplicados a una base de rostros pública generadas por la compañía AT&T.

Durante la elaboración se desarrolló una interfaz gráfica con Python basada en algoritmos propios de OpenCV. El objetivo de esta interfaz es poder realizar pruebas y mediciones comparativas con todos los algoritmos implementados de una manera más visual para poder identificar factores clave que podrían jugar un papel fundamental en el rendimiento de cada uno de los algoritmos. Para poder simular distintas situaciones a las que se podría enfrentar un algoritmo de reconocimiento facial, se realizaron distorsiones a todas las imágenes originales para después ser evaluadas y comparadas con los resultados originales. De la misma manera, se creó un sistema automatizado para la realización de las pruebas con toda la base de datos de imágenes.

Como resultado principal de este proyecto, nuestro sistema ayudó a identificar ciertas aplicaciones para las cuales cada algoritmo estudiado podría ser útil. Podríamos decir que SIFT es uno de los mejores algoritmos para el reconocimiento facial, pero de la misma manera este algoritmo podría ser usado para la detección de objetos o incluso de otros componentes como lo podría ser un identificador de texto, etc. Por otro lado, Eigenfaces está diseñado principalmente para la detección y reconocimiento de rostros, y la ventaja que se tiene con este es que con un mayor entrenamiento los resultados serán mejores y mucho más rápidos si lo comparamos con otros algoritmos como lo podría ser el caso de SIFT. Finalmente, KNN es un algoritmo clasificador extremadamente versátil, que de la misma manera como lo hace SIFT, puede tener un gran abanico de aplicaciones desde clasificación de objetos, letras, números y muchos datos. Sin embargo, aún existe un enorme margen de mejora y experimentación para lograr precisiones y tiempos aún más cortos de reconocimiento de los que hemos visto en este proyecto.

Capítulo 1 Introducción

1.1 Antecedentes

Desde hace algunos años, el campo del reconocimiento facial ha ido evolucionando y mejorando considerablemente. Hoy en día, diversos algoritmos de reconocimiento tienen una amplia gama de aplicaciones gracias a que estos han ido mejorando su funcionamiento y desempeño. Se han desarrollado algoritmos bastante robustos ante baja iluminación, orientación de sujeto, etc. Sin embargo, las pruebas realizadas sobre estos solamente se han explorado con una cantidad de variaciones de ángulos e iluminación un tanto restringida por lo que aún existe un alto margen de error.

Es importante distinguir dos conceptos fundamentales dentro de la aplicación de la visión por computadora: detección facial y reconocimiento facial. Se puede definir la detección facial como un término más general dentro del cual se encuentra el reconocimiento facial, es por esto por lo que diversas técnicas para este último son usadas meramente para la detección de un rostro. La manera en la que funcionan estos algoritmos en un principio consta de la identificación de características en común de un rostro, esto se hace normalmente identificando algún punto clave como lo podrían ser los ojos y una vez teniendo esto como base genera una superficie la cual es analizada para encontrar otras características como lo son la nariz, la boca, bordes faciales, etc. Para el caso del reconocimiento facial se toman otras métricas las cuales, en primera instancia, toman rostros identificados mediante detección facial y una vez que se tiene identificado uno, se aplican diversos filtros para convertir las imágenes en vectores de atributos para posteriormente ser evaluado y determinar similitudes con algunas otras imágenes previamente almacenadas.

Con la creación de una plataforma visual la cual permite observar los resultados obtenidos de manera gráfica no sólo será más sencillo identificar con qué condiciones los algoritmos pierden su eficacia, sino, también se podrán realizar una serie de comparaciones como lo podría ser el desempeño global entre dos algoritmos obteniendo una lista de las imágenes que mejor se emparejan a una imagen dada. Desde el punto de vista del paradigma del reconocimiento facial actual, es necesario mejorar algoritmos existentes o desarrollar nuevos que permitan su uso en condiciones no óptimas. Adicionalmente, como señala J. Tanaka y D. Simonyi [1], dentro del

paradigma de identificación de rostro parcial, es importante la identificación de la identidad de un rostro con un cambio de sus componentes principales, lo cual está asociado con nuestras habilidades para reconocer un rostro.

1.2 Justificación

Si bien, los algoritmos de reconocimiento facial hoy en día se podrían considerar como eficientes y rápidos cada vez aprovechando los recursos de los equipos de mejor manera para conseguir mejores rendimientos. Muchos de estos a pesar de sus exhaustivas pruebas y entrenamientos suelen tener problemas en condiciones críticas las cuales podríamos encontrar día a día dentro de una de sus aplicaciones. La mayor parte del tiempo los conjunto de datos utilizados para entrenar estos algoritmos son generados en condiciones óptimas lo cual de cierta manera los destina a ser algo limitados en cuanto a rendimiento y eficacia al intentar hacer la búsqueda de un rostro en condiciones no óptimas y con variables ajenas a nosotros como lo podría ser la iluminación, tal como lo menciona el autor Moses et al., "las variaciones entre imágenes del mismo rostro debido a la iluminación y el ángulo de dirección son casi más grandes que las que se tienen con un cambio de identidad" [2] a tal grado que podrían llegar a considerarse "inutilizables" ciertos algoritmos ante ciertas variaciones. De la misma manera otro reto que se presenta para la implementación de sistemas de reconocimiento facial es la comparación y la búsqueda de rostros dentro de bases de datos masivas ya que si bien el tamaño de éstas aumenta, también lo hará el tiempo de búsqueda. Debido a esto los algoritmos de reconocimiento facial ahora no sólo deben de ser exactos, sino, también eficientes.

Para poder englobar todo lo mencionado anteriormente y tener mejores métodos de comparación es importante tener no sólo una manera cuantitativa de evaluar los resultados sino también una manera visual para poder interpretarlos y analizarlos comparándolos con los resultados obtenidos entre diversos algoritmos para determinar cuáles se desempeñan mejor en ciertas condiciones y cuales otros representan ciertas limitaciones.

1.3 Objetivos

1.3.1 Objetivo general:

Desarrollar una plataforma gráfica de reconocimiento de rostros para el estudio del desempeño de algoritmos estadísticos de clasificación.

1.3.2 Objetivos específicos:

1. Diseñar la estructura de la Base de Datos de imágenes a analizar.
2. Desarrollar un módulo para agregar el registro de la imagen a evaluar.
3. Diseñar y desarrollar un módulo encargado del reconocimiento facial.
4. Desarrollar un módulo encargado del despliegue de resultados al usuario.
5. Realizar un análisis de desempeño empleando los módulos anteriores.

Capítulo 2 Conceptos

2.1 Detección Facial

La detección facial usualmente llega a ser confundida con el reconocimiento facial, sin embargo, ambos son términos que pueden ser escuchados hoy en día con mucha frecuencia. Sin embargo, éstos difieren en lo que significa, el reconocimiento facial de cierta manera podríamos decir que es un derivado de la detección facial, ésta lo que hace es detectar rostros solamente dentro de alguna imagen o en video. La manera en la que funciona la detección facial es muy variada, existe la implementación de sistemas que hacen uso de redes neuronales las cuales son entrenadas con extensos conjuntos de datos para que el mismo algoritmo pueda discernir ciertos patrones en los pixeles de una imagen para determinar si estos son indicadores de lo que existe en cierta imagen sea un rostro. De la misma manera existen otras técnicas como con la comparación de modelos de rostros comunes mediante los cuales el algoritmo genera un porcentaje de probabilidad de lo que está evaluando se asimila a un rostro y si pasa cierto límite decide si es en efecto un rostro lo que hay presente dentro de una imagen (Fig. 1).

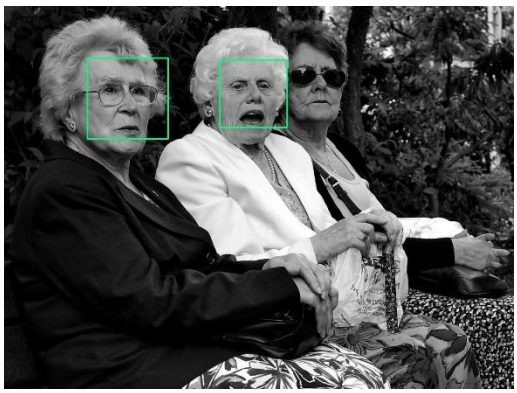


Figura 1. Rostros detectados dentro de una imagen.
Obtenido de: https://pythondiario.com/2017/06/deteccion-de-rostros-con-python-y.html

2.2 OpenCV

OpenCV [3] es una biblioteca de visión por computadora, de código abierto lo cual significa que es mantenido de manera activa por la comunidad. La importancia de esta biblioteca es que se ha convertido en un estándar a nivel internacional dentro del ámbito del reconocimiento facial y visión por computadora debido a que tiene la implementación de cerca de 2500 algoritmos para ambos propósitos los cuales han sido optimizados y generados para tener una compatibilidad en diversas

plataformas. La razón por la cual se optó por usar OpenCV como la biblioteca a usar es debido a que su vasta cantidad de algoritmos robustos los cuales nos pueden ayudar dentro del desarrollo del proyecto acortando los tiempos en la implementación de módulos de reconocimiento facial al contar con una gran cantidad de algoritmos útiles y prácticos para esta sección.

2.3 Reconocimiento Facial

Como bien se mencionaba en la sección anterior, el reconocimiento facial es algo distinto a la detección de rostros o la detección facial. Si bien, el reconocimiento es derivado de este último su funcionamiento es bastante distinto a lo que se tendría comúnmente con un sistema de detección facial. La manera en la que funciona el reconocimiento facial en términos generales es bastante sencilla, una cámara se encarga de capturar una imagen y esta imagen después es enviada a un sistema el cual implementa diversos métodos y algoritmos para poder extraer las características de mayor peso de un rostro. Por lo general estas características suelen estar conformadas por la distancia entre ambos ojos, la longitud y el ancho de un rostro, la longitud de los labios e incluso combinaciones de estas mismas. A diferencia de los seres humanos, las computadoras no pueden ver de la misma manera en la que nosotros lo hacemos debido a que estas no cuentan con dos ojos para poder ver de la manera estereoscópica y en espectro de colores en la que lo haría un humano. Éstos funcionan captando la luz que incide a ellos y esta es transformada en impulsos eléctricos que son transportados mediante nuestro sistema nervioso el cual los lleva directamente a nuestro cerebro dónde será interpretado y es así como podemos “ver” estas imágenes. Las computadoras tratan las imágenes como la descomposición de los valores en cada uno de los canales de color de la imagen. Estos valores son considerados como una matriz tridimensional donde cada uno de los colores es una matriz sobrepuesta con la anterior y combinar estos valores aplicando ciertas transformaciones se tienen valores numéricos que las computadoras pueden interpretar (Fig. 2). Para el reconocimiento facial existen diversos acercamientos y algoritmos. Dentro de los más populares están aquellos que usan técnicas de aprendizaje automático y los algoritmos de clasificación. Las aplicaciones de reconocimiento facial han aumentado drásticamente en los últimos años debido al gran avance que se ha tenido dentro de esta rama de la visión por computadora.

2.4 Visión por computadora

La visión por computadora es una rama respectiva de la inteligencia artificial. Para tener una mejor noción de lo que se realiza dentro de esta rama podemos tomar como punto de partida la visión humana. En el caso de las computadoras se tiene un funcionamiento similar, sin embargo, ciertas características cambian, principalmente la manera en la que las computadoras interpretan lo que ven. Para que una computadora pueda captar la luz es necesario contar con un sensor conectado el cual en este caso sería alguna cámara. Las imágenes recopiladas mediante ésta son enviadas al procesador de la computadora y usando diversos algoritmos y técnicas de procesamiento de imágenes es que una computadora puede "ver". Es importante remarcar que para que una computadora pueda ver como lo hacemos nosotros no sólo basta con que esta pueda captar las imágenes, sino, que la computadora debe de tomar estas imágenes y usarlas como información para interpretar qué es lo que está captando que en este caso es lo que hacemos nosotros de manera cotidiana que probablemente lo pasemos por alto al ser una tarea muy rutinaria e inconsciente.

Figura 2. Persona siendo detectada dentro de una imagen con una probabilidad de 0.688
Obtenido de: https://ingenierobeta.com/que-es-la-vision-artificial/

Capítulo 3 Marco teórico

3.1 PCA

PCA [4] (Principal Component Analysis por sus siglas en inglés) es un algoritmo estadístico de clasificación y de aprendizaje automático de tipo no supervisado, útil para la compresión de imágenes y para el desarrollo de sistemas de reconocimiento facial. Es la técnica básica para entender el funcionamiento de reconocimiento y además forma parte importante de técnicas más avanzadas. Como su nombre lo indica, el objetivo de éste es el interpretar datos y encontrar sus componentes principales, es decir a aquellos datos que se encarguen de describir de la manera más amplia y óptima posible el conjunto de datos. El PCA tiene una gran variedad de usos debido a la manera en la que funciona, y procesa los datos dando un razonamiento con el uso de vectores. El PCA está compuesto por dos componentes principales, el primero intenta describir la mayor cantidad de datos originales dentro de un vector mientras que el componente secundario funciona como un auxiliar para englobar a los datos restantes a partir de un vector ortogonal al componente principal. Los dos vectores forman un plano que representa una visualización de un subconjunto de datos.

De manera más detallada, el PCA busca una combinación lineal entre dos variables dadas por la siguiente ecuación:

(1)

$$\sum_{j=1}^{p} a_j \, x_j = Xa$$

Dentro de esta, a_j es un vector de constantes y x_j es el vector de observación de la variable j. Por otro lado, la varianza de una combinación lineal de este tipo está dado por la siguiente ecuación:

(2)

$$var(Xa) = a´Sa$$

Donde *S* es la covarianza de la muestra de la matriz asociada con el grupo de datos y "´" denota la transpuesta de esta matriz. Tomando en cuenta esto podemos decir que al obtener la combinación lineal con máxima variación obtenemos un vector p-dimensional *a* que maximiza la forma

cuadrática de $a´Sa$. A pesar de esto necesitamos agregar una restricción adicional para lo cual es necesario trabajar con vectores normales unitarios para obtener una solución precisa y bien definida. De manera general al agregar esta restricción y haciendo uso de un multiplicador de Lagrange definido por el símbolo λ, tenemos como resultante la siguiente ecuación la cual diferencia entre el vector *a,* el cual está igualado al vector nulo debido a que al usar vectores normales necesitamos que *a´a=1*.

(3)

$$Sa - \lambda a = 0 \Leftrightarrow Sa = \lambda a$$

Finalmente, de manera adicional, si quisiéramos saber la calidad de la aproximación, podemos hacer uso de la variabilidad asociada con los conjuntos almacenados de Componentes Principales (CP), meramente con la sumatoria de los elementos diagonales a la matriz de covarianza.

(4)

$$\pi_j = \frac{\lambda_j}{\sum_{j=1}^{p} \lambda_j} = \frac{\lambda_j}{tr(S)}$$

El algoritmo PCA permite determinar la variable más importante que describe el conjunto de datos, como, por ejemplo, características faciales de una persona.

3.2 Eigenfaces

[5] Es una técnica basada en el Análisis de Componentes Principales (PCA). Está creada a partir de dos fases, una de entrenamiento y otra de clasificación. En la primera, y por medio del PCA, se forma un espacio de facciones, conocido como eigenspace, a partir del uso de imágenes faciales de entrenamiento. El espacio de facciones es la matriz formada por una serie de vectores propios (eigenvectors o eigenfaces), que contienen la información de la variación de los valores de gris de cada píxel del conjunto de imágenes utilizadas al realizar el PCA.

Como bien sabemos, las imágenes son realmente una matriz de números la cual si es de forma NxN podríamos interpretarla como un vector lo cual nos permite representarla como una combinación lineal de sus bases ortonormales para poder ser comprimida la información lo que supondría una ventaja cuando estamos tratando imágenes con rostros.

La base de Karhonen-Lóeve (KL) es una alternativa para una base ortonormal, de esta manera podemos suponer que el vector de la imagen de un rostro está dado por un vector aleatorio *x* con una matriz de covarianza lo que se expresa en la siguiente ecuación:

(5)

$$C = E[xx^T]$$

Esto quiere decir que las bases KL están formadas por eigenvectores de C los cuales pueden ser demasiados, es aquí cuando hacemos uso de la técnica de PCA donde seleccionaremos solamente los eigenvectores más representativos los cuales son aquellos que presenten la mayor varianza como bien se describió anteriormente. Aquellos vectores que son seleccionados se les conoce como Eigenfaces.

Como se mencionaba anteriormente, Eigenfaces consta de dos etapas, una de entrenamiento y otra de clasificación. Durante la etapa de entrenamiento se usa el PCA para calcular el rostro promedio por cada individuo dentro del conjunto de datos, este rostro promedio también es conocido como el fantasma de Eigenfaces. Los pasos que seguir son calcular el rostro promedio por cada sujeto, normalizar las imágenes dentro del conjunto, obtener la matriz de covarianza *C*, calcular los eigenvectores y ordenarlos por los más altos para así poder obtener la combinación lineal del conjunto.

Para la fase de reconocimiento se usa un proceso similar. Primero, la imagen de entrada en transformada en sus componentes de eigenfaces y se compara con los rostros promedios generados, después multiplicamos su diferencia por cada eigenvector, esto se puede expresar de la siguiente manera:

(1)

$$w_k = U_k^T(\Gamma - \sigma)$$

Dentro de esta ecuación σ denota la imagen de entrada, U_k^T cada uno de los eigenvectores generados y Γ siendo el rostro promedio. Los resultados de w_k se almacenarán dentro de una matriz con una columna para así poder calcular la distancia euclidiana mínima entre la imagen de entrada y la almacenada.

(2)

$$E_r = \arg min \ \|\Omega - \Omega_k\|$$

Siendo Ω_k cada uno de los valores de w_k. Para poder indicar que un rostro ha sido reconocido es importante tener en cuenta que debe de existir un umbral fijo.

3.3 SIFT

El algoritmo SIFT [6] (Scale-Invariant Feature Transform por sus siglas en inglés) es un método para extraer puntos característicos invariantes y distintivos de una imagen que pueden ser usados para mejorar la correspondencia entre dos vistas diferentes de un objeto o una escena. Los puntos característicos obtenidos por SIFT son invariantes a escala y rotación de la imagen, y son mostradas para proporcionar una identificación robusta a pesar de que exista un rango amplio de distorsiones afines, cambios en la vista 3D, adición de ruido a la escena y cambios en la iluminación. La razón por la cual SIFT es un algoritmo tan poderoso a pesar de las alteraciones que pueda presentar una imagen es debido a la manera en la que funciona y la lógica detrás para realizar el reconocimiento.

SIFT consta de cinco pasos para poder realizar el reconocimiento dentro de una imagen y cada uno es el responsable por el que el algoritmo posee una identificación robusta a pesar de variaciones dentro de la luminosidad orientación, tamaño, etc.

Antes de realizar los pasos es importante generar ciertas modificaciones dentro de la imagen para poder hacer uso del algoritmo SIFT, estas ya se contemplan dentro del proceso del mismo algoritmo, pero es importante mencionarlo. Lo que se debe de hacer para poder generar nuestro espacio de escalas es aplicar la técnica de suavizado gaussiano para reducir el ruido dentro de las imágenes (Fig. 3).

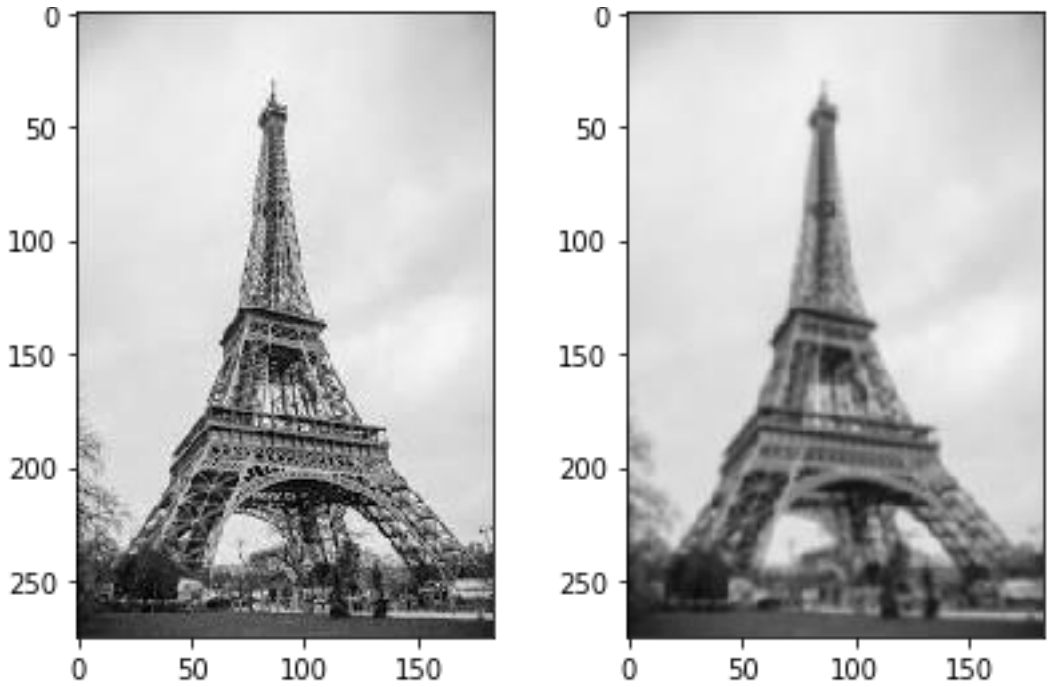


Figura 3. Imagen después de aplicar técnica de borrado
Obtenido de: https://laptrinhx.com/a-detailed-guide-to-the-powerful-sift-technique-for-image-matching-with-python-code-3921045966/

El espacio escalador es realmente una lista de imágenes las cuales tienen distintas escalas a partir de su imagen original. En este caso para el espacio partimos con las imágenes borrosas, por cada escalamiento de imágenes tomaremos la anterior y la reduciremos a la mitad, de la misma manera por cada imagen en la nueva escala también aplicaremos otro proceso de borrado como se ejemplifica con la imagen anterior (Fig. 3). Finalmente, este proceso se debe repetir alrededor de 19 veces, el numero óptimo de "octetos" es cuatro y dentro de este se deberían incluir 5 imágenes con cada una de las escalas de borrado obteniendo un resultado de la siguiente manera (Fig. 4).

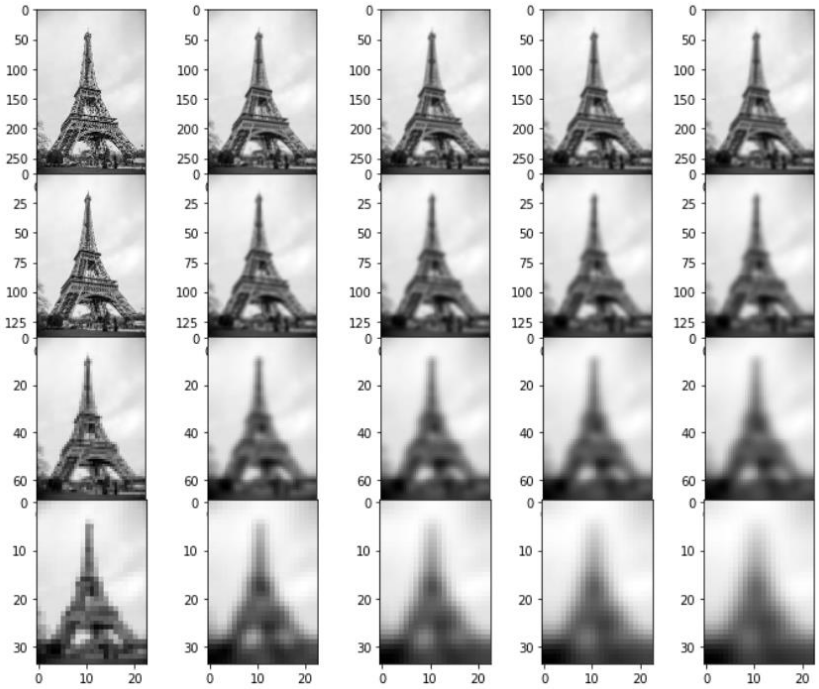


Figura 4. Octeto de imágenes con borrado gradual por cada grupo
Obtenido de: https://laptrinhx.com/a-detailed-guide-to-the-powerful-sift-technique-for-image-matching-with-python-code-3921045966/

i. Escalamiento espacial con detección de extremos: Para este paso se hace uso de la transformada de Laplace Gaussiana, también conocido como el algoritmo de Marr-Hildreth. Este se encarga de detectar bordes dentro de las imágenes lo cual se consigue evaluando la imagen y buscando curvas, es decir cambios fuertes en la iluminación dentro de alguna zona de la imagen.

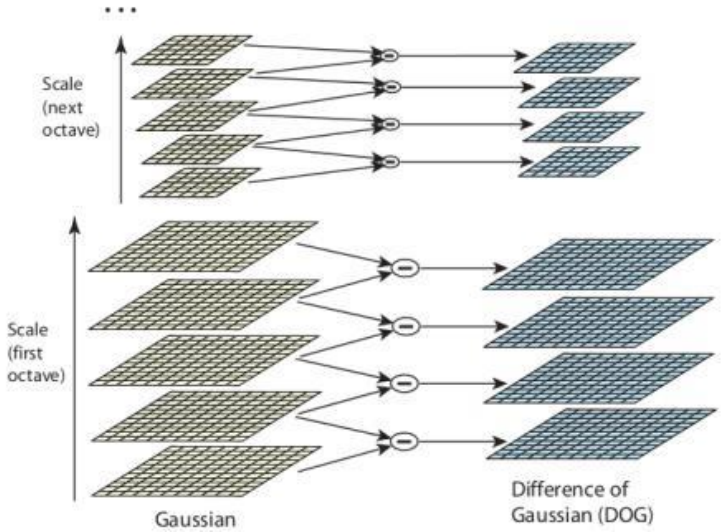


Figura 5. Representación de comparación con diferencia Gaussiana entre octetos
Obtenido de: https://www.researchgate.net/figure/Difference-of-Gaussian-is-obtained-as-the-difference-of-Gaussian-blurring-of-an-image_fig4_312107986

Finalmente se hace uso de la diferencia Gaussiana para determinar cuáles son los puntos clave de la imagen, mediante la búsqueda de puntos máximos locales. El proceso consiste en comparar cada píxel de la imagen con sus 8 vecinos aledaños, de la misma manera se compara con los 9 vecinos de la escala anterior y de la siguiente. Si es un extremo local entonces podemos decir que es un candidato a ser un punto clave.

ii. Localización de puntos clave: Una vez que hemos identificado los candidatos a puntos clave es necesario realizar un refinamiento para poder obtener resultados más precisos cuando realicemos la detección de rostros. Esto se realiza mediante una expansión en serie de Taylor dentro de nuestro espacio escalado para poder generar un resultado más preciso de los valores que representan los extremos de la imagen o del rostro en este caso. La manera en la que se seleccionan es que deben de sobrepasar un límite establecido, el cual normalmente es de 0.3. De esta manera se filtran los puntos clave de aquellos que son considerados débiles.

iii. Asignación de orientación: Este paso consiste en asignar una orientación a cada uno de los puntos clave detectados en el paso anterior. La razón por la cual se realiza esto es para poder conseguir que nuestro reconocimiento sea "a prueba" de rotaciones y diversos ángulos. La manera interna en la que funciona este proceso es la siguiente: por cada punto clave se selecciona algún vecino según la escala en la que se esté trabajando. Después se genera un histograma de 36 espacios para cubrir los 360 grados y los puntos que sobrepasen el 80% son considerados para decidir la dirección (Fig. 6).

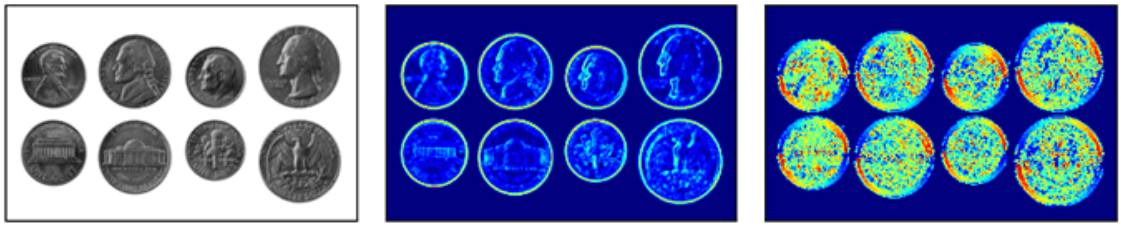


Figura 6. Representación de visión dentro de una imagen para asignación de orientación de puntos clave.
Obtenido de: https://pyimagesearch.com/2021/05/12/image-gradients-with-opencv-sobel-and-scharr/

iv. Descriptor de punto clave: Dentro de este paso se sigue trabajando sobre los puntos clave identificados dentro del segundo paso. Para este, se genera un área vecina de 16x16 alrededor de cada punto clave el cual se divide en 16 bloques de tamaños 4x4. Por cada uno de estos bloques se realizará otro histograma de orientación el cual es como el generado

en el punto anterior lo cual se puede observar en la siguiente imagen. Es importante notar que para ambos pasos se hace uso de una función Gaussiana (Fig. 7).Los valores generados dentro del histograma de orientación finalmente son usados para generar un vector el cual será nuestro descriptor del punto clave seleccionado.

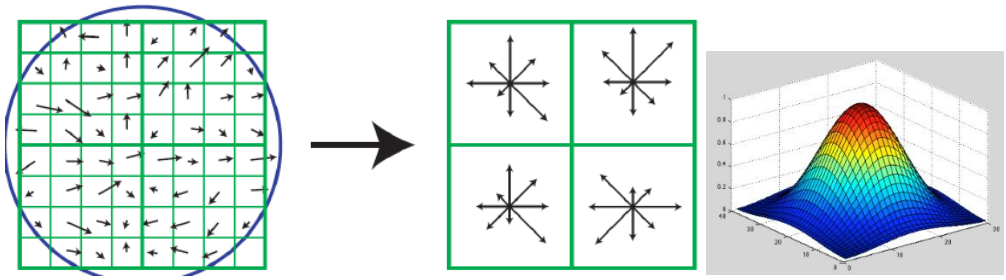


Figura 7. Representación de vectores descriptores de puntos clave y campana gaussiana.
Obtenido de: http://vision.stanford.edu/teaching/cs231b_spring1213/slides/hog_rafael_tao

v. Emparejamiento de puntos clave: Este paso consiste en emparejar los puntos clave de la imagen que se quiere identificar con alguna otra. En algunos casos es común que los resultados sean muy parecidos por lo que se evalúa también la distancia entre estos para descartar resultados erróneos o falsos hasta un 90% según *referencia* y tan sólo un 5% de resultados correctos (Fig. 8).

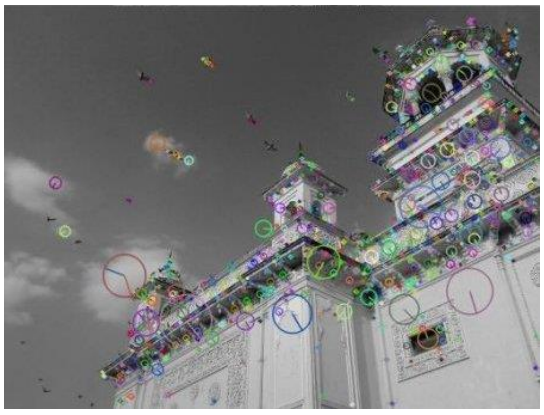


Figura 8. Visualización de radios de puntos clave
Obtenido de: https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html

3.4 KNN

El algoritmo KNN [7] (k-Nearest Neighbors por sus siglas en inglés), es un algoritmo de clasificación supervisado. Cuando hablamos de algoritmos supervisados nos referimos a aquellos a los que se les asigna un valor esperado para que pueda evaluar sobre sí mismo su eficacia y precisión. KNN es uno de los algoritmos de clasificación más antiguos que existe. Como su

nombre lo indica, si dos puntos tienen valores similares, probablemente deban de pertenecer a la misma clase.

La manera en que el algoritmo funciona es mediante la exploración de K vecinos más cercanos. Para la clasificación el algoritmo realiza una consulta sobre todos los datos y medirá la distancia de los puntos más cercanos. Una vez definidas las clases y todos sus puntos pertenecientes a ella, se mide la distancia de ambas clases en promedio con una distancia euclidiana para decidir la clasificación ganadora y esta es la seleccionada.

3.5 Clasificador en cascada Haar

Para que KNN pueda identificar a los sujetos dentro de una imagen, es necesario que primero identifique al menos un rostro que pueda evaluar. Para eso se usan los clasificadores en cascada los cuales como podría indicar su nombre, tiene la función de ir realizando un barrido sobre una sección de una imagen para identificar ciertas características las cuales se asimilen con el objeto que se busca clasificar. Dentro de los distintos clasificadores, está el clasificador de tipo Haar, el cual cuenta con 3 características principales las cuales son las que se buscan al momento de realizar el barrido por la imagen y estas son simplemente regiones dentro de rectángulos.
La manera en las que funcionan estas regiones y cómo es que se hace el procesamiento es bastante sencillo. Simplemente se suman los valores de cada píxel dentro de cada una de las zonas marcadas y se hace una diferencia entre ellas. Después, de manera iterativa se continúa con este proceso para poder generar cierto modelo que nos permita identificar imágenes que sigan de alguna manera este patrón. Para poder conseguir resultados precisos, es necesario entrenar estos modelos por lo que se realizan varias iteraciones con imágenes positivas y negativas para ir refinando los valores que nos indicarían si es que hay un rostro dentro de la imagen.

3.6 Preparación de imágenes

Debido a que se tiene considerado realizar pruebas con diversas imágenes en condiciones no óptimas, se decidió realizar ciertas alteraciones a las imágenes contenidas dentro de la misma base de datos y de esta manera poder obtener resultados más cercanos a los esperados dentro de las pruebas, para cada una de estas imágenes se decidió realizar 6 distorsiones con 4 tipos de alteraciones las cuales fueron el arqueo, debido a que de esta manera se puede jugar con las

proporciones y las curvas de las caras las cuales pueden dar un efecto de perspectiva cerca, y alteraciones de afinación con la perspectiva lo cual brinda cambios completos con la manera en la que la iluminación se hace presente dentro de la imagen así como ciertos cambios en la posición de características clave de los rostros por lo que se espera que con la combinación de estos se tenga una reducción del error al insertar imágenes distorsionadas de otra manera. Para poder realizar todas las modificaciones necesarias es necesario hacer uso de Wand. Esta es una biblioteca específicamente diseñada y desarrollada para Python con el objetivo de brindar un grupo de funciones que nos permitan realizar procesamiento de manera general a una imagen. A continuación, se hablará con más detalle sobre las funciones usadas y cómo funcionan, así como los valores usados y el porqué de estas distorsiones [8].

3.6.1 Transformación Affine

La transformación afín realiza operaciones de cortado y deformación mediante las cuales podemos realizar distorsiones similares a la de una perspectiva. La manera en la que funciona esta distorsión es que toma los puntos de las coordenadas introducidas y las mueve hacia las nuevas coordenadas seleccionadas. De esta manera para hacer uso de esta distorsión se tienen que especificar 12 argumentos de entrada. Entre estos, las coordenadas de tres puntos en el plano XY y las siguientes 3 para especificar las coordenadas de los puntos a los cuales serán trasladados. Un ejemplo de esta transformación con parámetros (40,0,60,0,0,45,20,60,35,20,50,30) está dado en la figura 9.

Figura 9. Imagen original e imagen con distorsión affine

3.6.2 Transformación Arc

La distorsión arc, como su nombre lo podría sugerir, transforma la imagen en forma de un arco. Para hacer eso la biblioteca toma el centro de la imagen y sobre esta abre un arco tomando como parámetro el número de grados y dentro del cual nuestra imagen se estirará y deformará. De la

misma manera se puede pasar otro parámetro para seleccionar el punto dentro de la imagen sobre el cual se iniciará o se anclará el arco (ver Fig. 10).

Figura 10. Imagen original e imagen con distorsión arc con valor de 60

3.6.3 Transformación Scale

La distorsión scale escala las imágenes. Para que la función pueda lograrlo se necesita especificar al menos un parámetro el cual puede ser un porcentaje de escalado o bien, la nueva medida de la imagen en pixeles. Cabe señalar que la transformación degrada la nitidez de la imagen como se puede observar en la figura 11.

Figura 11. Imagen original e imagen con distorsión scale (40,40)

3.6.4 Transformación Brightness

Dentro de la biblioteca de Wand, también podemos trabajar y modificar el brillo de las imágenes, esto es una gran herramienta para poner a prueba los algoritmos ya que muchas veces el rendimiento y la precisión de estos se ve fuertemente afectada por la iluminación y el brillo de las imágenes. Para poder modificar ya sea el brillo o el contraste es necesario brindar como parámetros el valor del brillo, del contraste y en qué canales modificaremos estos valores (Fig. 12).

Figura 12. Imagen original con modificación birghtness_contrast con valores (-30, 20, ´all_channels´)

3.7 Generación de Base de Datos de imágenes

La generación de la base de datos con las imágenes de cada sujeto es un paso de vital importancia debido a que es nuestro principal recurso para poder trabajar con los algoritmos de reconocimiento facial y poder realizar el análisis de desempeño propuesto en este proyecto. Para la generación de este mismo se tomó como base la base de datos de AT&T. Esta está conformada por 400 imágenes en escala de grises y 40 sujetos. Lo que quiere decir que cada sujeto posee 10 imágenes tomadas desde distintos ángulos y con variaciones como lo son las expresiones faciales, el uso de lentes o la iluminación [9]. Para almacenar cada una de estas imágenes no es muy recomendable subir y almacenar la imagen dentro de nuestro servidor debido a que esto aumentaría de manera considerable la capacidad de almacenamiento necesaria para soportar un sistema de gran tamaño, es por esto mismo y después de investigar de manera extensiva que se encuentra la recomendación y se toma la decisión de dejar definida la estructura de la base de datos de una manera en la que se tenga la dirección en la que se encuentra la imagen dentro del sistema y un índice que nos permita saber de qué sujeto es la imagen que se está visualizando.

Para poder levantar de manera rápida y eficiente este servidor se usó una base de tipo MySQL ya que no sólo brinda la ventaja de ser ligera y rápida, sino que también nos brinda la posibilidad de montar una capa de servicio extra con Spark el cual se encargará de manejar y extraer la información necesaria por cada uno de los algoritmos empleados. De manera adicional se hace uso de la herramienta Docker para poder generar de manera automática los contenedores con las estructuras necesarias para el funcionamiento de la base de datos, así como también permite una administración más sencilla de diversas instancias en caso de que las haya.

3.8 Modificaciones a la base de datos

Para poder realizar pruebas más adelante, es necesario realizar modificaciones a las imágenes dentro de nuestra base de datos, esto para poder observar el comportamiento y el impacto que

tienen ciertas distorsiones sobre la precisión de los algoritmos. Las modificaciones que se realizaron son 6, todas tienen un objetivo de asimilar casos en la vida cotidiana a los que se podría enfrentar un algoritmo de reconocimiento facial.

3.8.1 Brightness

Como se explica anteriormente, esta distorsión se encarga de modificar la exposición, así como otros atributos relacionados con el brillo de una imagen. Para esta distorsión se usan como valores -30 en el brillo, 10 en el contraste y finalmente "all_channels" para indicar que esos valores se aplicaran dentro de todos los canales de color de la imagen. El resultado de esta transformación es ejemplificado en la figura 13. La razón de esta distorsión es el simular un entorno con poca iluminación lo cual espera tenga un impacto directo sobre la precisión de los algoritmos ya que podríamos inferir que al tener puntos más obscuros será mucho más difícil el determinar e identificar puntos clave.

Figura 13. Sujeto 1 con distorsión brightness.

3.8.2 Affine 1 y 2

Como observamos anteriormente, la distorsión affine nos permite trasladar ciertos puntos de una imagen a unos nuevos conservando su estructura, lo cual nos permite realizar ciertos ajustes y distorsiones con la finalidad de brindar una ilusión de perspectiva para la imagen. En este caso se realizan dos distorsiones, ambas con el propósito de simular una cámara sujeta desde un ángulo aproximado a 45%. Para conseguir esto, dentro de la primera distorsión se usaron los parámetros ([10,10,5,5,70,10,75,5,60,60,60,65]) los cuales son las coordenadas antiguas y las nuevas, mientras que para la segunda distorsión aplicada se usan los siguientes parámetros ([10,10,5,5,70,10,85,15,70,92,85,72]) (Fig. 14).

Figura 14. Sujeto1, ambas imágenes con distorsión de tipo affine. A la izquierda Affine 1 y a la derecha Affine 2

3.8.4 Arc 1 y 2

Las distorsiones de tipo arc brindan una enorme funcionalidad ya que, con esta, como se explica con anterioridad, podemos definir el ángulo con el que arquearemos la imagen y como dato extra podemos seleccionar un punto sobre el que se anclara la imagen. Con estas distorsiones se pretenden simular un lente cóncavo o un lente de pescado los cuales distorsionan de manera significativa las imágenes al alargar sus bordes y amplificar el centro. Para conseguir ambos efectos para la imagen se usaron los valores ([15,0, 'ideal radius', 'ideal radius']), mientras que para la segunda imagen fueron ([18, 'ideal radius', 'ideal radius']) (Fig. 15).

Figura 15. Sujeto 1, ambas imágenes con distorsión Arc. A la izquierda Arc 1 y a la derecha Arc 2.

3.8.6 Scale

Con la distorsión scale se simula la cercanía a la cámara, la razón de esta distorsión es el imitar una distancia que podría tener un impacto sobre la precisión de los algoritmos al cortar ciertos bordes del rostro, así como ciertas características principales como lo podría ser la silueta del rostro, sus orejas o incluso alguna parte del rostro. Para conseguir este efecto se utilizó la ampliación de la imagen con los valores (112x112,135%) (Fig. 16).

Figura 16. Sujeto 1 con distorsión de tipo scale. A la izquierda imagen original, a la derecha imagen amplificada.

Capítulo 4 Módulos

4.1 Descripción general de la plataforma

A continuación, en la figura 17 se muestra un diagrama del cómo está estructurado el proyecto con sus respectivos módulos de operación y el flujo de información y procesamiento que se realiza para la detección de rostros, así como el cómo se evalúa el desempeño de cada uno de los algoritmos.

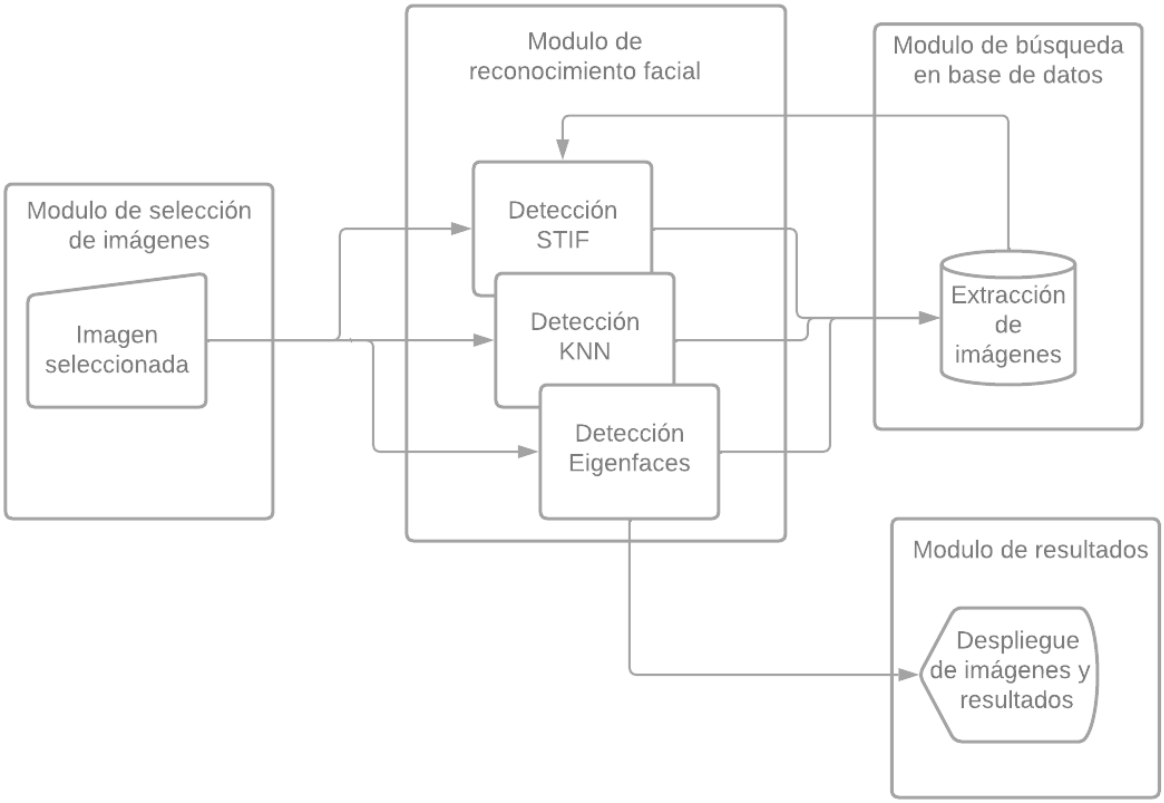


Figura 17. Diagrama de flujo del sistema integrado

4.2 Módulo de inserción de imagen a evaluar

Dentro de este módulo se selecciona la imagen que se quiera evaluar para poder ser enviada a través de los demás módulos por el flujo de trabajo que se tiene esquematizado en la figura 17. Dentro de la figura 18 podemos observar que en la interfaz gráfica tenemos un botón para seleccionar alguna imagen dentro de nuestro sistema de archivos, de la misma manera en la interfaz

de este módulo se implementa un botón de inicio para que se inicie la evaluación con todos los algoritmos con la imagen seleccionada.

Figura 18. Módulo de selección de imagen. El botón de selección despliega una ventana del buscador de archivos y el botón de inicio manda a iniciar la evaluación

4.3 Módulo de reconocimiento facial

Este módulo se encarga de la implementación de los algoritmos brindados por OpenCV para poder realizar la evaluación de la imagen ingresada dentro del módulo de inserción de imagen, así como también de la transformación de esta en vectores de atributos para poder ser evaluados en el módulo de búsqueda. Para poder asegurar el funcionamiento correcto de los algoritmos, las imágenes se escalan a una resolución de 112x112 píxeles procurando no modificar el rostro del sujeto en ninguna manera. Una vez hecho esto, se registra el tiempo de inicio y fin para poder llevar una métrica del tiempo elapsado en la evaluación de cada imagen. De la misma manera, para las pruebas automatizadas es aquí donde se almacenan las predicciones realizadas por cada uno de los algoritmos para poder ser evaluado una vez terminadas las pruebas. Éste modulo dentro del entorno gráfico envía datos relevantes como lo que sería la etiqueta y la imagen del sujeto al modulo de resultados para poder ser desplegados hacia el usuario.

4.4 Módulo de búsqueda en Base de Datos

Este módulo cuenta con la función de tomar los datos transformados por el módulo de reconocimiento facial para poder realizar la búsqueda de los rostros que más se asimilen a el que se está buscando y fue ingresado dentro del módulo de inserción. Este módulo funciona en

conjunto con el módulo de reconocimiento facial para poder obtener los datos relevantes para la imagen, dentro de estos está el nombre y la ubicación en el directorio para que el módulo de resultados pueda encontrarlos. De manera adicional, este módulo retorna la etiqueta del sujeto para poder desplegarlo junto con la imagen predecida del sujeto dentro del módulo gráfico. De la misma manera, este módulo es usado dentro del módulo de resultados para poder obtener y generar la comparación de las imágenes predecidas para su despliegue dentro de la pantalla de resultados detallados.

4.5 Módulo de resultados

La principal función de este módulo es la de desplegar los resultados obtenidos junto con los datos relevantes del procesamiento como lo son la imagen predecida, así como la etiqueta asociada con esta. La razón por la cual se tiene un módulo secundario es para desplegar una mayor información sin saturar al usuario ya que dentro de este se generarán las comparaciones entre la imagen ingresada y la predecida para poder ayudar al usuario a visualizar los puntos clave identificados por cada uno de los algoritmos. Este último modulo secundario trabaja en conjunto con el módulo de búsqueda, así como con el de reconocimiento facial, este último pasa las imágenes resultantes por una función que se encarga de identificar, marcar y unir por puntos y líneas los puntos clave identificados dentro de las imágenes (figuras 19 y 20).

Figura 19. Módulo inicial de despliegue de resultados, el botón “Mas detalles” despliega el segundo módulo de resultados.

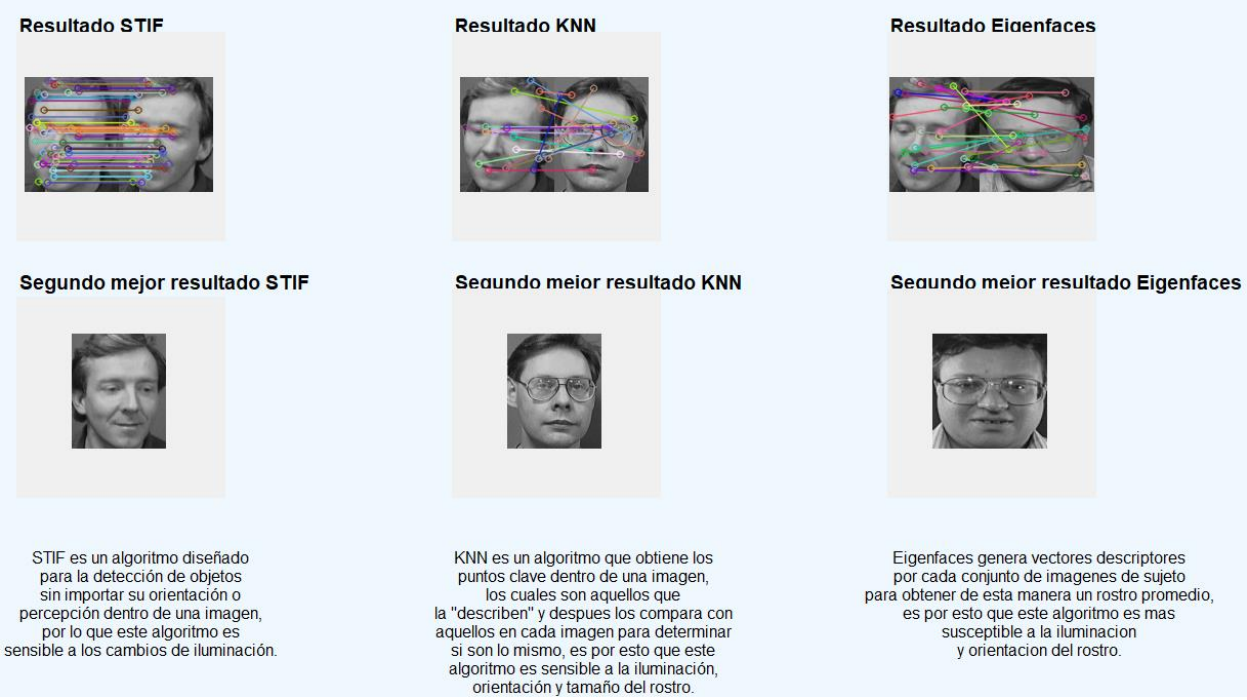


Figura 20. Módulo secundario de despliegue de resultados, se muestra los puntos clave identificados entre imágenes y la segunda mejor imagen clasificada.

4.6 Descripción del funcionamiento de algoritmos

Teniendo una comprensión sobre los algoritmos utilizados dentro del proyecto a continuación se realiza una explicación paso a paso sobre la manera en la que funcionan cada uno de los algoritmos en su implementación y finalmente cómo es que se integran dentro del sistema para realizar la evaluación de la imagen seleccionada por el usuario.

4.6.1 Eigenfaces

Eigenfaces es un algoritmo que consta de distintos pasos, esto se debe a que antes de poder iniciar con la evaluación de imágenes, es necesario realizar un entrenamiento previo lo cual le permita calcular los valores de eigen o los conocidos como eigenfaces, estos se almacenarán de manera local para poder compararlos con la imagen que deseemos evaluar y de esta manera seleccionar aquella que obtenga el puntaje más alto el cual será simplemente aquél sujeto que su imagen promedio tenga la menor distancia de los puntos descriptivos identificados. A continuación, en la figura 21, se muestra un diagrama de flujo sobre los pasos a seguir dentro de Eigenfaces.

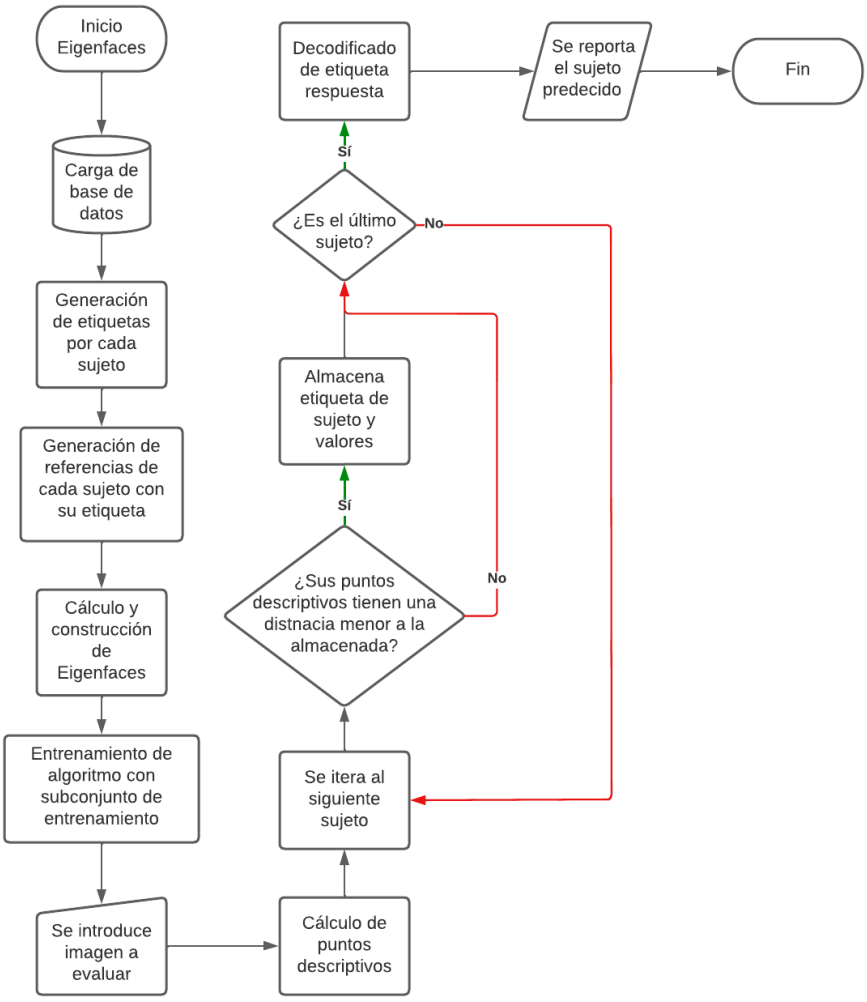


Figura 21. Diagrama de flujo para Eigenfaces.

4.6.2 SIFT

A diferencia de Eigenfaces, SIFT no tiene una etapa de procesamiento, por lo que no es necesario cargar la base de datos como un paso inicial. Sin embargo, esto puede repercutir una vez que se realiza la evaluación de las imágenes debido a que se realiza el cálculo de los puntos clave por cada iteración. Por otro lado, una parte del proceso de SIFT es similar al de Eigenfaces y KNN como veremos más adelante. Este proceso consiste en iterar por todos los sujetos mientras se realiza la comparación entre la imagen seleccionada por el usuario y aquellas almacenadas para determinar qué sujeto o qué imagen son los que arrojan los mejores resultados. A continuación, en la figura 22 se muestra un diagrama de flujo con los pasos que sigue este algoritmo.

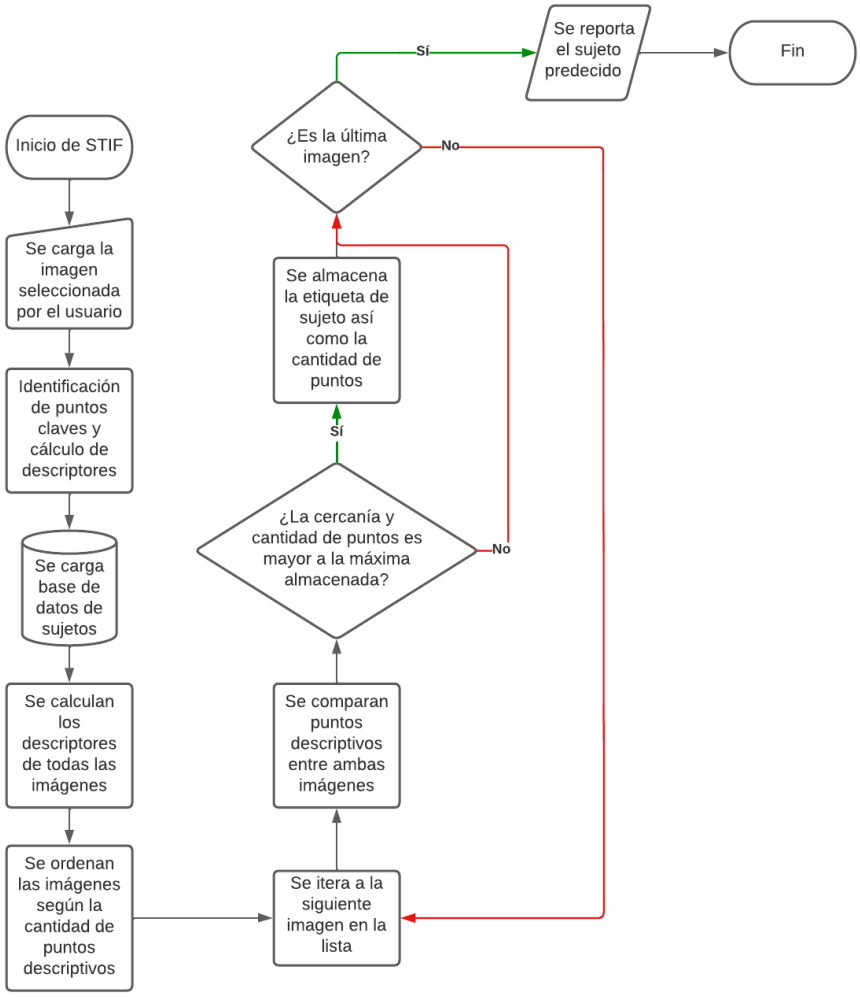


Figura 22. Diagrama de flujo para SIFT.

4.6.3 KNN

Como observamos anteriormente, los proceso para Eigenfaces y SIFT son muy similares. En el caso de KNN también, esto se debe a que todos los algoritmos deben de generar sus métricas para el reconocimiento, y una vez hechas se debe iterar por toda la base de datos. En algunos casos, estos valores pueden ser calculados previamente y almacenados para después cuando se esté intentando realizar la detección, sólo se haga una consulta a la base de datos con un rango de valores, lo cual acelera el tiempo de respuesta, así como el tiempo de procesamiento que se llevan los algoritmos en realizar la comparativa.

Una de las diferencias de KNN con SIFT y Eigenfaces es que este algoritmo realiza un barrido inicial de la imagen con un clasificador, esto para poder identificar los rostros dentro de una imagen y después cada rostro es recortado o escalado para poder realizar el análisis de puntos descriptivos,

estos son evaluados con los valores previamente almacenados por cada uno de los sujetos y es de esta manera que se realiza la comparación para determinar qué sujeto es el que estamos viendo. A continuación, en la figura 23 podremos observar el flujo para este proceso.

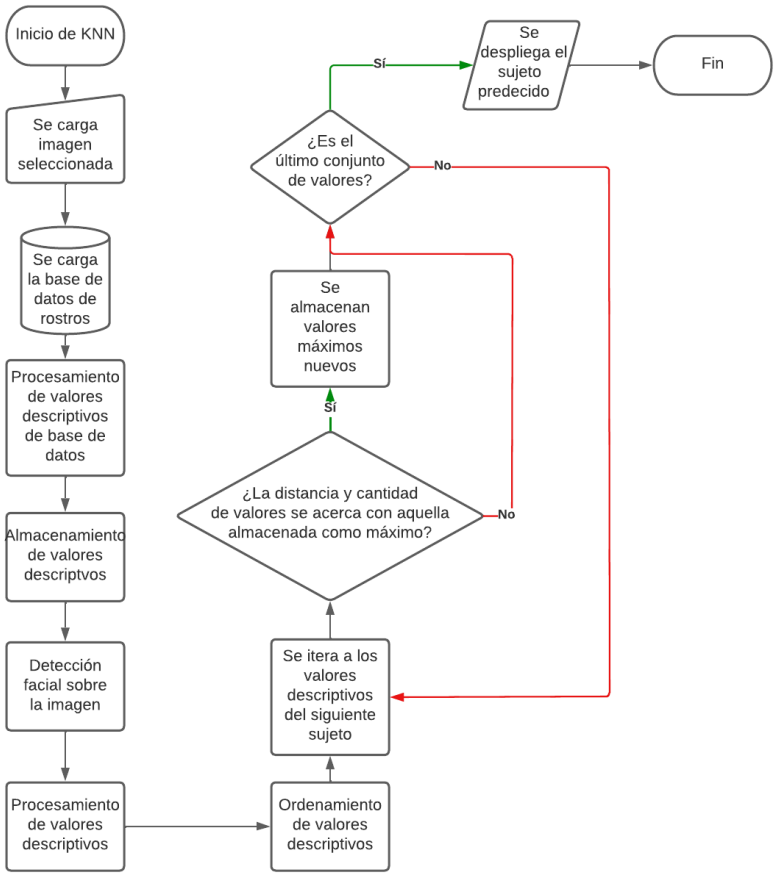


Figura 23. Diagrama de flujo para KNN.

4.6.4 Sistema de evaluación

Para poder realizar la ejecución de todos los algoritmos se realizó una interfaz gráfica la cual consiste de los módulos que hablamos en el capítulo 4. Este sistema se encarga de alimentar la imagen seleccionada por el usuario hacia cada uno de los algoritmos. De la misma manera, este es el encargado de desplegar los resultados de manera gráfica mediante un manejo de los datos que nos regresan cada uno de estos como se ha explicado en los puntos anteriores dentro de los diagramas de flujo. Este mismo sistema se usa para las pruebas automatizadas, sin embargo, los pasos son un poco distintos ya que dentro de este no se realiza el levantamiento de un entorno gráfico. A continuación, en la figura 24 se muestra el diagrama de flujo para el sistema de evaluación.

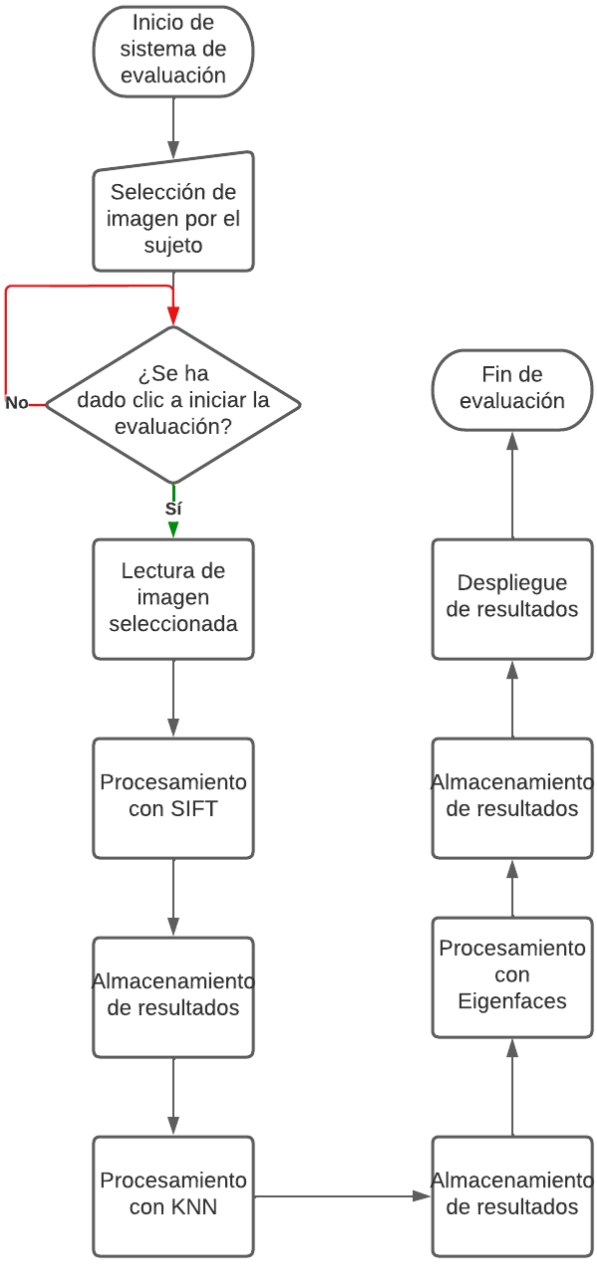


Figura 24. Diagrama de flujo del sistema de evaluación.

Capítulo 5 Desarrollo

5.1 Bibliotecas

Para el desarrollo de este proyecto se utilizaron diversas bibliotecas, dentro de las cuales caben destacar **OpenCV**, la cual nos ayudó dentro de la implementación de algoritmos de reconocimiento facial, al igual que **sklearn**. Por otro lado, también es importante resaltar el uso de la biblioteca **tkinter** con la cual se desarrolló la interfaz gráfica y fue la que permitió el despliegue de la información generada de manera visual. Dentro de la siguiente tabla se enlistan todas las bibliotecas usadas, así como una breve descripción sobre lo qué hacen y en qué se usó.

Biblioteca	Descripción
Tkinter [10]	Tkinter es la biblioteca por defecto de Python para generar interfaces gráficas. Esta está disponible en casi todas las plataformas de sistemas operativos por lo que su uso se hizo con la finalidad de tener una biblioteca que fuese lo suficientemente robusta para conseguir los resultados deseados, así como para conservar la portabilidad y la universalidad de esta. Tkinter es usado en este proyecto de manera amplia para generar la interfaz gráfica con la que interactúa el usuario durante el uso del sistema.
Matplotlib 3.5.1 [11]	Matplotlib es una biblioteca dedicada a la creación gráficos ya sea de tipo estático, interactivo o animado. Para este proyecto se hizo uso de la biblioteca por la gran cantidad de herramientas que tiene para la visualización de gráficos en tiempo real por lo que se tiene una gran utilidad para analizar el rendimiento de los algoritmos seleccionados.
Numpy 1.22.0 [12]	Numpy es una de las principales bibliotecas de análisis de datos dentro de Python. Se usa en una gran variedad de campos gracias a su gran manejo de arreglos numéricos, así como por todas las implementaciones de análisis con las que cuenta. Para el proyecto Numpy fue una gran herramienta ya que esta biblioteca está diseñada para el procesamiento de imágenes por lo que brinda una gran variedad de herramientas para esto mismo.
OpenCV 4.5.5.62	OpenCV es una biblioteca de código abierto especializado en la visión por computadora. Tiene integrado miles de algoritmos para la visión por computadora, el reconocimiento, y muchas otras herramientas. Esta

Biblioteca	Descripción
	biblioteca se puede definir como el núcleo del proyecto ya que se hizo uso de varias de sus funciones y herramientas para conseguir el reconocimiento facial con cada uno de los algoritmos.
Sklearn 1.0.2 [13]	Sklearn, o también conocido como scikit. Es una biblioteca diseñada para el análisis y procesamiento de datos dentro de los que destaca la clasificación y la aplicación de modelos. La biblioteca inició como un proyecto creado por Google. Dentro del proyecto se hace uso de la biblioteca para la clasificación y generación de ciertos datos los cuales son necesarios para ciertos algoritmos.
Imutils 5.4	Imutils es una biblioteca para realizar un procesamiento básico de imágenes. Hace uso de diversas funciones que son parte de matplotlib y opencv. Para el proyecto se hace uso de algunas de sus funciones para realizar principalmente modificaciones a las imágenes.
Pandas 1.4.0 [14]	Pandas es una biblioteca diseñada para el análisis y la manipulación de datos. Es algo parecido a lo que encontraríamos con sklearn, sin embargo, la biblioteca nos permite hacer uso de otros archivos para el análisis de datos como lo pueden ser los archivos xls por lo que se usa dentro del proyecto en conjunto con sklearn.
Wand 6.7	Wand es una de las principales bibliotecas dentro de Python para el procesamiento y manejo de imágenes. Cuenta con diversas herramientas para la modificación de valores propios de las imágenes. Dentro del proyecto se hacen uso diversas transformaciones incluidas dentro de sus herramientas para realizar diversas distorsiones.
Mysql 8.0.28	Mysql es una biblioteca creada originalmente por Sunmicrosystems la cual pasó a ser parte de Oracle. La biblioteca brinda lo necesario para poder realizar conexiones desde programas de Python a bases de datos por lo que era necesario en este proyecto para poder conectarse con la base de datos generada para almacenar las direcciones de todas las imágenes.

Tabla 1. Tabla de bibliotecas usadas

5.2 Software

5.2.1 Docker

Docker [15] es un proyecto de código abierto el cual automatiza el despliegue y la creación de contenedores de aplicaciones para así proporcionar una capa de abstracción y automatización de virtualización de aplicaciones dentro de diversos sistemas operativos. El papel de Docker dentro del proyecto es el de actuar como contenedor para nuestra base de datos y de esta manera poder hacer uso de ella para almacenar y leer datos dentro de esta.

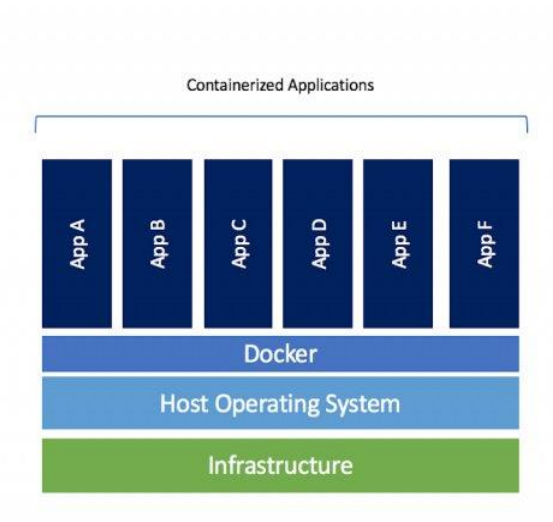


Figura 25. Diagrama de jerarquía para Docker.
Obtenido de: https://dev.to/ahmedgulabkhan/what-is-docker-and-why-do-we-need-it-5dje

5.2.2 Cascade Trainer GUI

Cascade Trainer GUI [16] es un software gratuito creado por Amin Ahmadi, autor y experto en visión por computadora el cual nos permite entrenar clasificadores en cascada con distintos conjuntos de imágenes, así como con distintos valores para su entrenamiento y evaluación. Fue necesario hacer uso de esta herramienta para poder generar nuestro clasificador en cascada de Haar ya que al intentar utilizar aquellos proporcionados por OpenCV para la detección de rostros, los resultados obtenidos dentro de KNN no eran nada favorables. Algunos de estos resultados de precisión general no rebasaban el 5%, por lo que era evidente que se necesitaba realizar una modificación.

Los pasos para generar el clasificador fueron relativamente rápidos. Primero debíamos generar dos conjuntos, uno de imágenes positivas y otro de negativas. Para poder realizar este conjunto, primero se tuvo que analizar qué imágenes estábamos clasificando, en este caso rostros, por lo que

no sería del todo correcto usar imágenes de automóviles o edificios como negativos ya que si bien, no son rostros, estos negativos no ayudarían a corregir los valores que nos permitan identificar aquellos rostros dentro de nuestra base de datos. Como solución, se usa la base de datos de FACES la cual contiene varias imágenes de personas con distintas expresiones y posiciones para ser nuestro conjunto de imágenes negativas.

El siguiente paso fue separar ambos conjuntos en distintas carpetas, una llamada “p” y otra “n” como identificador de positivos y negativos. Después, estas se seleccionarán dentro de la interfaz para ser cargadas y usadas dentro de los cálculos necesarios.

En este punto se pueden hacer modificaciones sobre cuántas etapas se quieren realizar, mientras más etapas se especifiquen mayor será la precisión de nuestro clasificador, pero tomará más tiempo. La recomendación dentro de la página oficial de la interfaz gráfica es que seleccionemos un número intermedio alrededor del 15, por lo que para nuestro caso especificaremos 20 inclinándonos por la precisión, pero sin sacrificar por completo la rapidez.

De la misma manera, podemos especificar un tamaño de memoria la cual permite realizar un precálculo de valores relacionado con los índices de las imágenes y el número de hilos que usaremos para acelerar el tiempo de procesamiento del entrenamiento, este valor podría cambiar dependiendo de los recursos disponibles dentro del equipo.

Finalmente, una vez que termina el entrenamiento obtendremos nuevas carpetas dentro de nuestro espacio de trabajo donde encontraremos la carpeta “classifier” (Fig. 26).

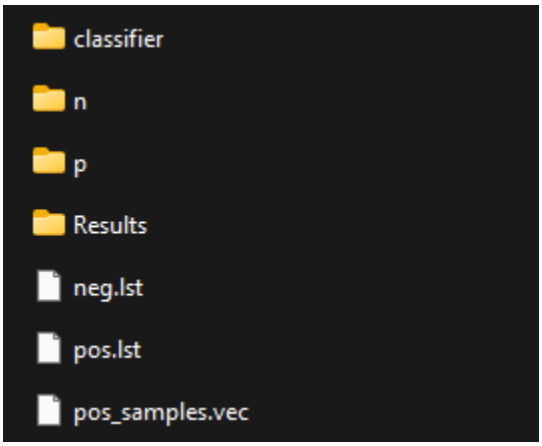


Figura 26. Carpetas resultantes de entrenamiento de clasificador.

Aquí buscaremos el archivo llamado classifier.xml el cual contiene los valores esperados dentro de cada una de las características de Haar y las etapas generadas con la interfaz gráfica. Dentro de estas podemos ver atributos denominados nodos internos los cuales simplemente indican los

puntos de cada uno de estos y los valores esperados. Esto se observa de la siguiente manera (Fig. 27).

```
<stages>
  <!-- stage 0 -->
  <_>
    <maxWeakCount>2</maxWeakCount>
    <stageThreshold>-9.0107358992099762e-02</stageThreshold>
    <weakClassifiers>
      <_>
        <internalNodes>
          0 -1 13 1.1076328158378601e-01</internalNodes>
        <leafValues>
          -7.2972971200942993e-01 9.2660552263259888e-01</leafValues></_>
      <_>
        <internalNodes>
          0 -1 51 -3.7666976451873779e-02</internalNodes>
        <leafValues>
          6.3962239027023315e-01 -9.0753984451293945e-01</leafValues></_></weakClassifiers></_>
  <!-- stage 1 -->
  <_>
    <maxWeakCount>3</maxWeakCount>
    <stageThreshold>-6.9954150915145874e-01</stageThreshold>
    <weakClassifiers>
      <_>
        <internalNodes>
          0 -1 22 4.9641523510217667e-02</internalNodes>
        <leafValues>
          -7.4311923980712891e-01 9.0990990400314331e-01</leafValues></_>
      <_>
        <internalNodes>
          0 -1 1 4.7481250017881393e-02</internalNodes>
        <leafValues>
          -5.8218377828598022e-01 8.4878224134445190e-01</leafValues></_>
      <_>
        <internalNodes>
```

Figura 27. Vista de archivo XML de clasificador generado con Cascade Trainer GUI.

Finalmente, el archivo XML generado será especificado dentro del código de KNN para su uso y como era de esperarse los resultados de la precisión del algoritmo han mejorado significativamente como se observará más adelante en la recopilación de resultados para su análisis.

Capítulo 6 Resultados

6.1 Análisis y discusión de resultados

En este capítulo se describen los resultados de aplicación de algoritmos de detección de los individuos en las imágenes originales y distorsionadas. La tasa de detección está dada como promedio sobre los 40 sujetos y en fracciones de unidad, en donde la unidad corresponde a la identificación correcta de todos los sujetos. Las tablas 2, 3, y 4 a continuación resumen las mediciones realizadas para los algoritmos SIFT, KNN y Eigenfaces, respectivamente.

Sujetos	Sin distorsión	brightness	affine1	affine2	arc1	arc2	scale
Sujeto 1	0.9	1	0.5	0.3	0.9	0.7	0.6
Sujeto 2	0.9	0.7	0.8	0.3	0.9	0.7	0.5
Sujeto 3	0.9	1	1	0.8	1	0.9	1
Sujeto 4	0.9	1	1	0.2	0.8	0.9	0.9
Sujeto 5	0.9	1	1	0.5	1	0.6	0.9
Sujeto 6	0.9	0.8	0.6	0.8	0.7	0.8	0.6
Sujeto 7	0.9	1	1	0.9	0.9	1	1
Sujeto 8	0.9	0.7	0.8	0.2	1	0.8	0.5
Sujeto 9	0.9	0.8	0.9	0	0.6	0.6	0.7
Sujeto 10	0.9	0.8	0.8	0	0.7	1	0.8
Sujeto 11	0.9	0.9	1	1	0.9	0.8	1
Sujeto 12	0.9	0.9	1	0.6	0.8	0.8	1
Sujeto 13	0.9	0.8	0.9	0.6	0.9	0.5	0.8
Sujeto 14	0.9	0.9	0.9	0.9	1	1	1
Sujeto 15	0.9	0.7	1	0.3	1	1	0.9
Sujeto 16	0.9	1	1	0.6	0.9	1	0.8
Sujeto 17	0.9	0.9	0.8	0.3	1	0.8	1
Sujeto 18	0.9	1	1	0.6	0.8	1	0.8
Sujeto 19	0.9	1	1	1	0.8	1	1
Sujeto 20	0.9	1	0.9	0.8	1	1	0.7
Sujeto 21	0.9	0.8	0.6	0.7	0.8	0.9	0.6
Sujeto 22	0.9	0.2	1	1	0.9	0.7	1
Sujeto 23	0.9	0.7	0.9	0.1	0.8	0.7	0.8
Sujeto 24	0.9	0.7	0.8	0.2	0.9	1	0.4
Sujeto 25	0.9	0.6	0.9	0.2	1	0.9	0.8
Sujeto 26	0.9	0.7	0.9	0.6	1	1	0.8
Sujeto 27	0.9	1	1	0.7	1	1	1
Sujeto 28	0.9	1	1	0.8	1	1	0.9
Sujeto 29	0.9	0.7	1	0.5	1	1	1

Sujetos	Sin distorsión	brightness	affine1	affine2	arc1	arc2	scale
Sujeto 30	0.9	0.4	0.5	0.5	0.9	0.6	0.8
Sujeto 31	0.9	0.8	0.8	0.3	1	0.7	0.8
Sujeto 32	0.9	0.9	1	1	0.8	0.8	1
Sujeto 33	0.9	0.7	0.3	0.3	0.8	0.7	0.4
Sujeto 34	0.9	1	1	1	1	0.9	1
Sujeto 35	0.9	0.8	1	0.1	0.7	1	1
Sujeto 36	0.9	0.6	0.8	0.7	0.8	1	0.6
Sujeto 37	0.9	0.9	0.9	0.8	0.7	0.8	0.8
Sujeto 38	0.9	0.5	0.9	0.3	1	0.8	1
Sujeto 39	0.9	0	0.6	0.3	0.6	0.8	0.7
Sujeto 40	0.9	0.9	0.9	0.3	1	1	1
Promedio	0.9	0.795	0.8675	0.5275	0.8825	0.855	0.8225

Tabla 2. Tabla de precisiones de SIFT en pruebas automatizadas

Sujetos	Sin distorsión	brightness	affine1	affine2	arc1	arc2	scale
Sujeto 1	0.9	0	0.3	0.6	0.7	0.8	0.3
Sujeto 2	1	0	0.2	0.9	1	1	0
Sujeto 3	0.9	0	0	0	1	1	0
Sujeto 4	1	0	0.1	0	1	1	0
Sujeto 5	1	0	1	0.6	1	0.6	0.9
Sujeto 6	1	0	1	0.8	1	1	0.8
Sujeto 7	1	0	0.7	0	0.9	0.8	0.1
Sujeto 8	1	0	0.7	0	1	1	0.4
Sujeto 9	0	0	0	0	0	0	0
Sujeto 10	0.9	0	0.5	0	0.5	0.4	0.2
Sujeto 11	1	0	0	0	1	0.8	0
Sujeto 12	1	0	0.2	0	0.8	0.9	0
Sujeto 13	1	0	0.7	0.2	1	0.9	0.2
Sujeto 14	1	0	0.1	0	1	1	0
Sujeto 15	1	0	1	0	1	1	0.2
Sujeto 16	1	0	0.2	0.2	0.4	0.7	0
Sujeto 17	1	0	0.9	0.2	0.9	0.7	0.5
Sujeto 18	0.6	0	0.9	0.3	0.7	0.8	0.1
Sujeto 19	0.9	0	0.1	0	0.6	0.5	0.1
Sujeto 20	0.7	0	0.4	0	0.5	0.3	0.3
Sujeto 21	1	0	0.1	0.2	1	1	0
Sujeto 22	1	0	1	1	1	0.9	1
Sujeto 23	0	0	0	0	0	0	0
Sujeto 24	1	0	0.8	0	1	1	0.3
Sujeto 25	0.8	0	0.5	0.1	1	1	0
Sujeto 26	0.7	0	0.2	0.9	0.7	0.9	0.1

Sujetos	Sin distorsión	brightness	affine1	affine2	arc1	arc2	scale
Sujeto 27	0.9	0	0.5	0	0.6	0.6	0.2
Sujeto 28	0.9	0	0	0	0.7	0.6	0
Sujeto 29	1	1	0.5	0.3	1	1	1
Sujeto 30	1	0	1	0.8	1	0.9	1
Sujeto 31	1	0	0.7	0.5	0.5	0.2	0.2
Sujeto 32	1	0	0.4	0.6	0.8	0.8	0.6
Sujeto 33	0.8	0	1	0	0.7	0.6	0.5
Sujeto 34	1	0	1	1	1	1	0.6
Sujeto 35	1	0	0.2	0	0.7	0.8	0
Sujeto 36	0.9	0	0	0	0.5	0.5	0
Sujeto 37	1	0	0.2	1	1	1	0.2
Sujeto 38	0	0	0	0	0	0	0
Sujeto 39	1	1	0.6	0.3	1	1	0
Sujeto 40	0.6	0	0.4	0	0.9	1	0.1
Promedio	0.8625	0.05	0.4525	0.2625	0.7775	0.75	0.2475

Tabla 3. Tabla de precisiones de KNN en pruebas automatizadas

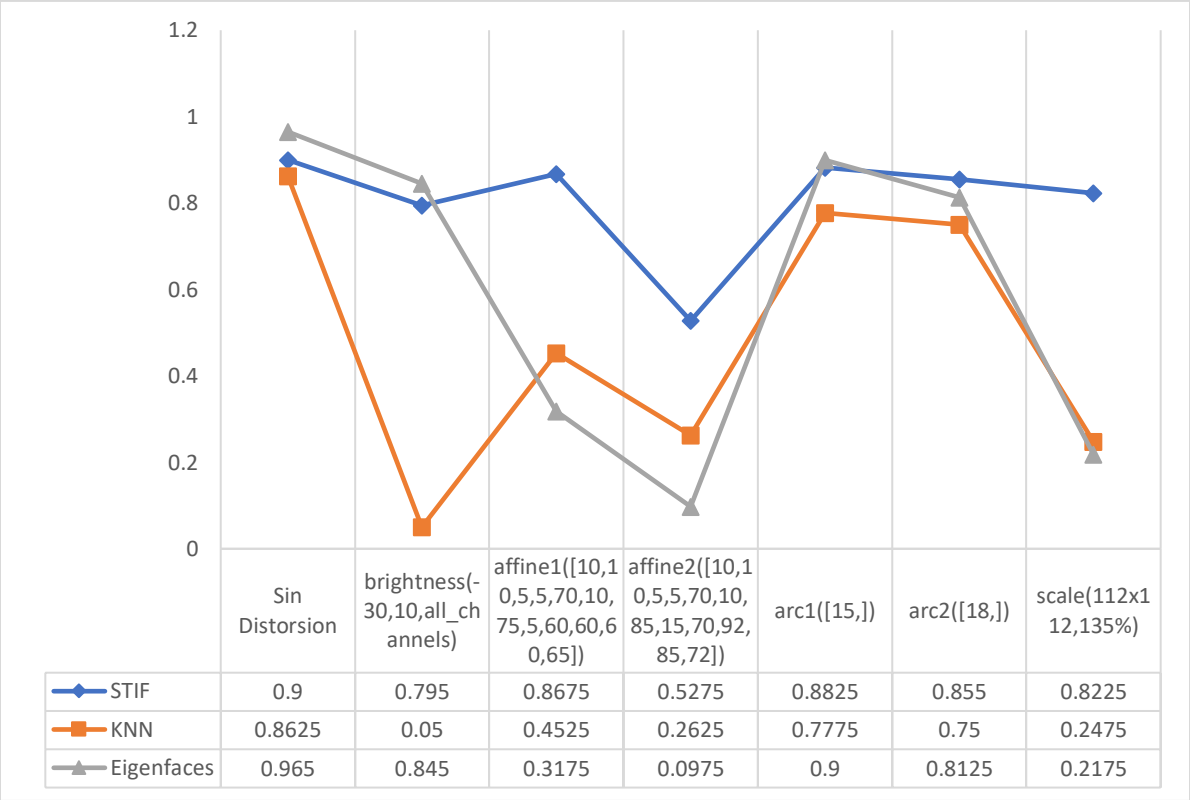
Sujetos	Sin distorsión	brightness	affine1	affine2	arc1	arc2	scale
Sujeto 1	0.8	0.4	1	0	1	1	0.8
Sujeto 2	1	0.9	0	0	1	1	0
Sujeto 3	1	0.9	0.1	0	1	1	0
Sujeto 4	0.9	1	0	0	0.8	0.9	0
Sujeto 5	1	0.9	0.7	0.1	0.9	0.6	0.5
Sujeto 6	1	0.4	1	0.8	1	1	1
Sujeto 7	1	0.9	0.6	0	0.9	0.8	0
Sujeto 8	1	1	0.4	0	1	1	0.5
Sujeto 9	0.9	0.3	0	0	0.9	1	0.4
Sujeto 10	1	1	1	0	0.7	1	0.7
Sujeto 11	1	1	0	0	0.9	0.4	0
Sujeto 12	1	1	0.5	0	1	0.9	0.2
Sujeto 13	1	1	0	0	1	0.6	0
Sujeto 14	1	0.9	0	0	1	0.8	0.4
Sujeto 15	1	1	0.4	0	1	0.9	0.3
Sujeto 16	1	0.8	0.2	0.2	0.9	0.7	0
Sujeto 17	1	1	0.1	0.1	0.7	0.6	0
Sujeto 18	1	1	0.3	0	1	1	0
Sujeto 19	1	0.8	0.1	0	0.9	0.5	0
Sujeto 20	0.9	0.9	0.2	0	1	0.8	0
Sujeto 21	1	1	0.1	0	1	1	0
Sujeto 22	0.9	1	0	0	0.8	0.8	0
Sujeto 23	1	1	0.5	0	1	1	0.3

Sujetos	Sin distorsión	brightness	affine1	affine2	arc1	arc2	scale
Sujeto 24	1	0.8	0.5	0	1	0.7	0.5
Sujeto 25	1	0.9	0.4	0	1	1	0.1
Sujeto 26	1	0.8	0.8	1	1	1	0.6
Sujeto 27	1	0.5	0.4	0	0.6	0.2	0
Sujeto 28	0.8	0.7	0	0	0.4	0.6	0
Sujeto 29	0.9	1	0.1	0	1	1	0.5
Sujeto 30	1	0.6	0	0	0.9	0.8	0
Sujeto 31	1	0.8	0.3	0	0.5	0.3	0
Sujeto 32	0.9	0.8	0.4	0.1	0.8	0.8	0.1
Sujeto 33	1	0.7	1	0.6	0.9	0.9	1
Sujeto 34	1	1	0.3	0	1	1	0
Sujeto 35	1	0.8	0.1	0	1	1	0.2
Sujeto 36	0.9	0.9	0	0	0.8	0.5	0.2
Sujeto 37	1	1	0	0.7	1	1	0
Sujeto 38	1	0.6	0.8	0	0.9	0.8	0
Sujeto 39	0.8	1	0.3	0.3	0.9	0.6	0.3
Sujeto 40	0.9	0.8	0.1	0	0.9	1	0.1
Promedio	0.965	0.845	0.3175	0.0975	0.9	0.8125	0.2175

Tabla 4. Tabla de precisiones de Eigenfaces en pruebas automatizadas

- Para las pruebas automatizadas se generaron 6 variaciones de imágenes distintas que se incluyeron en la base de datos. Las imágenes pertenecientes a cada sujeto eran modificadas con cada una de las distorsiones mencionadas en capítulos anteriores para ser evaluadas una por una sobre la base de datos de imágenes originales. La razón de este tipo de pruebas unitarias tiene el objetivo de demostrar la precisión de cada uno de los algoritmos bajo distintas condiciones sobre las que se podría encontrar un algoritmo de reconocimiento facial.
- La manera en la que se generan los resultados de precisión es de la siguiente manera: Se toman las diez imágenes pertenecientes al sujeto a ser evaluado, y se pasan por cada una de las distorsiones, y se genera una base de datos nueva, finalmente se recopilan todas las imágenes, se les asigna una etiqueta de sujeto, y esta lista de rostros junto con sus etiquetas se pasan por cada uno de los algoritmos y se recopilan las respuestas de cada uno de estos para después ser comparadas con las etiquetas de cada imagen las cuales son nuestras respuestas esperadas.
- Finalmente, obtenemos una lista de cuantos aciertos se obtienen por cada uno de los algoritmos por sujeto, realizamos un promedio sobre cuantos aciertos hay y de esta manera

se obtiene la precisión por sujeto. Una vez que se obtiene la precisión por cada uno de los sujetos, se obtiene un promedio armónico entre todos los sujetos. El resultado final se resume en la figura 28.



	Sin Distorsion	brightness(-30,10,all_channels)	affine1([10,10,5,5,70,10,75,5,60,60,60,65])	affine2([10,10,5,5,70,10,85,15,70,92,85,72])	arc1([15,])	arc2([18,])	scale(112x112,135%)
STIF	0.9	0.795	0.8675	0.5275	0.8825	0.855	0.8225
KNN	0.8625	0.05	0.4525	0.2625	0.7775	0.75	0.2475
Eigenfaces	0.965	0.845	0.3175	0.0975	0.9	0.8125	0.2175

Figura 28. Gráfica con las precisiones de cada uno de los algoritmos obtenidos con pruebas automatizadas.

De primera instancia, podemos identificar que los tres algoritmos tienen precisiones muy buenas para el conjunto de imágenes sin distorsiones, esto con la finalidad de tener un punto de comparación con las demás etapas de evaluación.

Para la distorsión de brillo algo que llama la atención a simple vista es el bajo rendimiento del algoritmo KNN. Si bien, todos los algoritmos se vieron impactados ante la presencia de esta modificación, ninguno de los otros dos se vio tan fuertemente afectado como lo fue en el caso de "brightness". Otro punto que es de llamar la atención es que cuando uno indaga un poco más dentro de las predicciones realizadas por el algoritmo KNN, muchas de estas son de manera repetitiva hacia un mismo sujeto.

Figura 29. Comparación entre sujeto 1 (izquierda), sujeto 22 (centro) y sujeto 40 (derecha).

Para identificar cuál podría ser el desviador encontramos algo muy particular. En la figura 29, podemos observar a 3 sujetos, y aquellos que están en los costados (Sujeto 1 y Sujeto 40) son una de las imágenes introducidas a los algoritmos dentro de las pruebas, mientras que sujeto central presenta su imagen sin distorsión, este mismo sujeto fue nuestro principal desviador, para ser exactos, representa aproximadamente el 70% de las predicciones erróneas para las pruebas dentro de KNN con distorsión de tipo brightness. Esto podría parecer que estamos observando en acción un algoritmo que tiene un cesgo por respuesta a una persona de tes oscura siempre que la imagen sea de bajo brillo. Una de las posibles causas de este comportamiento, principalmente dentro de KNN la encontraremos en uno de sus componentes vitales, el clasificador de objetos en cascada de tipo Haar. Si recordamos, anteriormente se explica cómo funcionan estos clasificadores de objetos y cómo es que estos mismos almacenan los puntos clave de cada una de las imágenes. Si consideramos que nuestro clasificador hace la suma de los valores de los pixeles dentro de ciertas áreas y realiza una diferencia entre la imagen evaluada y los datos que tenemos de las sumatorias de otras imágenes, es de cierta manera esperado que cuando estos valores disminuyan, ya que el tono oscuro dentro de nuestra escala rgb son los valores más cercanos al 0, la sumatoria también lo haga. Esto significa que las sumatorias obtenidas se acercarán más a los datos que tenemos almacenados para una imagen de una persona de tes oscura.

De manera adicional, tenemos que considerar que dentro de la base de datos que se está usando, predominan las personas de tes clara, por lo que nuestro clasificador muy probablemente tenga problemas para discernir entre personas de tes oscura, lo cual no implica que se tenga un algoritmo racista. Lo que realmente significa es un algoritmo con el entrenamiento insuficiente lo cual se corrige teniendo una mejor variación de etnias y sexos dentro del mismo.

El siguiente punto que llama la anteción es que se esperaba que en la mayor parte los algoritmos mantuvieran su precisión de una manera estable, sin embargo, al realizar ambas pruebas de tipo affine, podemos observar que Eigenfaces tuvo una caida un tanto drástica colocándolo como el

peor algoritmo en cuanto a estas distorsiones. La razón, nuevamente, puede que este relacionada a la manera en la que Eigenfaces realiza la evaluación de los rostros. Si bien recordamos, para que podamos identificar un rostro, el algoritmo evalua la imagen de entrada con una imagen o una serie de valores que conforman un promedio de las imágenes relacionadas con un sujeto, y si la imagen que estamos evaluando presenta modificaciones en la posición de sus puntos, muy probablemente el algoritmo tenga problemas para poder discernir entre qué sujeto deberá predecir puesto que las características principales del rostro muy probablemente estén más cercanas a las de otro sujeto.

Como tercer punto, hacemos referencia a los resultados obtenidos con la distorsión de tipo scale. Si bien, en todas las distorsiones se esperaba tener un impacto directo sobre la precisión de los algoritmos, algo que podemos ver que es cierto tanto para KNN como para Eigenfaces. Sin embargo, cuando vemos la gráfica, podemos observar que SIFT a pesar de tener una disminución en su precisión, esa fue mínima si lo comparamos con nuestros valores base. Esto puede que se deba a la manera en la que SIFT realiza la evaluación de sus imágenes. Si recordamos, este algoritmo busca puntos descriptivos así como su orientación, por lo que en caso de que desaparezcan ciertas características importantes, aún tiene otras sobre las que se puede valer. Sin duda, este sería un punto a tener en cuenta cuando se realiza la elección del algoritmo de reconocimiento facial, especialmente cuando pueden existir situaciones en las que el sujeto no se encontrará en un lugar óptimo todas las veces.

Finalmente, de manera general, se puede decir que el algoritmo SIFT durante todas las pruebas tuvo un mejor desempeño en comparación con KNN y Eigenfaces. Si bien en un comienzo la precisión de este último supero la de SIFT, fue en el momento de las distorsiones cuando empezamos a experimentar ciertas variaciones, principalmente en aquellas relacionadas con distorsión en los ángulos de las imágenes así como posición del sujeto, por lo que en aplicaciones dentro de la vida cotidiana podría significar un impacto notable en su uso.

Capítulo 7 Conclusiones

7.1 Conclusiones

Al concluir este proyecto se lograron varios puntos. Primero, se obtiene un sistema visual el cual permite evaluar imágenes dentro de una base de datos de rostros y despliega los resultados obtenidos con 3 algoritmos distintos: Eigenfaces, SIFT, y KNN. En segundo, se pudo identificar qué algoritmos se desempeñan de mejor manera a pesar de existir ciertas distorsiones lo que nos lleva a las siguientes observaciones.

- De manera general, podríamos decir que SIFT fue el algoritmo que mejor se desempeñó sobre los otros dos, esto puede ser debido a la manera en la que SIFT funciona y para lo que fue hecho. Es importante recordar que SIFT tiene la posibilidad de identificar imágenes u objetos a pesar de que presenten rotaciones, distorsiones y cambios en la luz por lo cual era de esperar que se tuvieran buenos resultados. Esto es un punto de vital importancia a considerar especialmente cuando se esté realizando el análisis para la selección de alguno de estos algoritmos teniendo en cuenta consideraciones del entorno que podría tener un impacto notable sobre el rendimiento de estos mismos. Por poner un ejemplo, en caso de tener un sistema de control de accesos mediante reconocimiento facial, donde la iluminación no sea la misma en todas las situaciones, podríamos escoger a SIFT o incluso Eigenfaces como nuestros algoritmos óptimos, esto principalmente por la buena precisión que demostraron durante las pruebas. Sin embargo, una de las razones por las cuales podríamos descartar a Eigenfaces es que su precisión disminuyó de manera considerable en las distorsiones que simulaban distintos ángulos de posición de sujeto. Dentro del escenario planteado anteriormente esto sería una enorme desventaja, ya que, si el sujeto que se pretende reconocer no se coloca en una posición completamente frente a la cámara o si por su distancia, alguno de los elementos del rostro llega a ser cortado, el algoritmo muy probablemente arroje resultados erróneos.
- Como segundo mejor algoritmo tenemos a Eigenfaces, a pesar de que en situaciones como las expuestas anteriormente presente problemas en su precisión, considero que no debería ser descartado por completo dentro de los sistemas de reconocimiento facial. La razón de esto es que realmente es un algoritmo bastante rápido y eficaz en situaciones controladas, por lo que, si los temas de posición no son una variable preocupante, probablemente los

resultados sean incluso mejores que los de SIFT como lo fue en el caso de las pruebas iniciales. Otro punto a favor que podríamos encontrar para Eigenfaces es el hecho de que el modelo se puede entrenar cuantas veces se necesite con más datos, lo cual tendría muy probablemente un impacto positivo en cuanto a la precisión al momento de realizar el reconocimiento facial. Continuando con el ejemplo anterior, si el sistema de control de accesos mediante reconocimiento facial, tuviese una función que asegure que ninguna de las características de los rostros sea cortada en la imagen, el tema de la precisión sería resuelto, De la misma manera si se tuviese un mecanismo para que cada imagen evaluada que tuvo un resultado erróneo sea almacenado y usado para entrenar nuevamente el algoritmo, la precisión mejoraría incluso en presencia de variaciones como la posición y orientación del sujeto así como la presencia de otras distorsiones como lo podría ser gafas o algún otro accesorio que podría cambiar la forma del rostro.

- Como tercer algoritmo tenemos a KNN, que, si bien su precisión en cuanto a la presencia de baja luminosidad no fue la óptima, recuperó su desempeño en las demás pruebas. De la misma manera que con Eigenfaces, este algoritmo puede volverse a entrenar cuantas veces sea necesario, así como podemos generar clasificadores más precisos siempre y cuando se tengan los datos suficientes. El único punto que podría mencionar en contra para KNN a pesar de la posibilidad de refinar su funcionamiento seguirá siendo su uso en situaciones de baja luminosidad. Debido a la manera en la que realiza su evaluación esta de cierta manera atado a tener resultados erróneos en presencia de estas condiciones. Sin embargo, una posible solución sería el realizar un procesamiento inicial de la imagen antes de ser enviada a la evaluación, esto sería realizando ajustes en el tiempo de exposición, así como en el contraste de cada imagen para obtener resultados más claros y discernibles. El único inconveniente que se presentaría con esta solución es el tiempo de respuesta, el cual muy probablemente lo descartaría como una opción viable dentro del mismo ejemplo de control de acceso mediante reconocimiento facial ya que para este tipo de sistemas la precisión y la velocidad son factores clave que no se pueden sacrificar uno por el otro.

Finalmente, como se observó dentro de este proyecto, nuestro sistema nos ayudó a identificar ciertas aplicaciones para las cuales cada algoritmo estudiado podría ser útil. Si bien, a pesar de que en algunos escenarios ciertos algoritmos se hayan desempeñado de mejor manera a comparación de los otros, no significa que estos deban de ser descartados. Cabe destacar que existen muchos

algoritmos diferentes disponibles para el problema de reconocimiento de rostros. En este proyecto se han seleccionado los más populares. Como resultado podríamos decir que SIFT es uno de los mejores algoritmos para el reconocimiento facial, pero de la misma manera este algoritmo podría ser usado para la detección de objetos o incluso de otros componentes como lo podría ser un identificador de texto, etc. Por otro lado, Eigenfaces está diseñado principalmente para la detección y reconocimiento de rostros, y la ventaja que se tiene con este es que con un mayor entrenamiento los resultados serán mejores y mucho más rápidos si lo comparamos con otros algoritmos como lo podría ser el caso de SIFT. De la misma manera ya existen ciertas variantes que traen respuesta a ciertas problemáticas o limitaciones como lo hace el algoritmo Fisherfaces. Finalmente, KNN es un algoritmo clasificador extremadamente versátil, que de la misma manera como lo hace SIFT, puede tener un gran abanico de aplicaciones desde clasificación de objetos, letras, números y muchos datos. Sin duda alguna la tecnología en términos de algoritmos ha avanzado de manera drástica, así como la tecnología que se posee para explotarlos a su máximo potencial. Sin embargo, aún existe un enorme margen de mejora y experimentación para lograr precisiones y tiempos aún más cortos de reconocimiento de los que hemos visto en este proyecto.

Instalación de Python

Python es uno de los lenguajes de programación más populares en la actualidad, especialmente dentro de la rama de la inteligencia artificial, así como todas sus ramas dentro de las cuales encontramos la visión por computadora. Este proyecto fue realizado usando Python principalmente por su implementación de bibliotecas de visión por computadora de código abierto. A continuación, se muestran los pasos para su instalación.

i. Deberemos ingresar a la página web oficial de Python, a continuación, la liga: https://www.python.org/downloads/

ii. Una vez dentro, seleccionaremos el sistema operativo para el que instalaremos nuestro interpretador y compilador.

Figura 30. Página de descarga de instalador de Python.

Obtenido de: https://www.python.org/downloads/

iii. Seleccionaremos el instalador o el paquete de binarios que se adecuen en nuestro sistema. La arquitectura en bits de nuestros equipos para el caso de Windows la podremos encontrar en información acerca de nuestro equipo. (Windows > Configuración > Sistema > Acerca de).

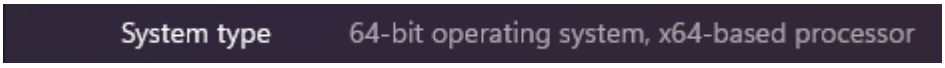


Figura 31. Arquitectura de sistema operativo Windows

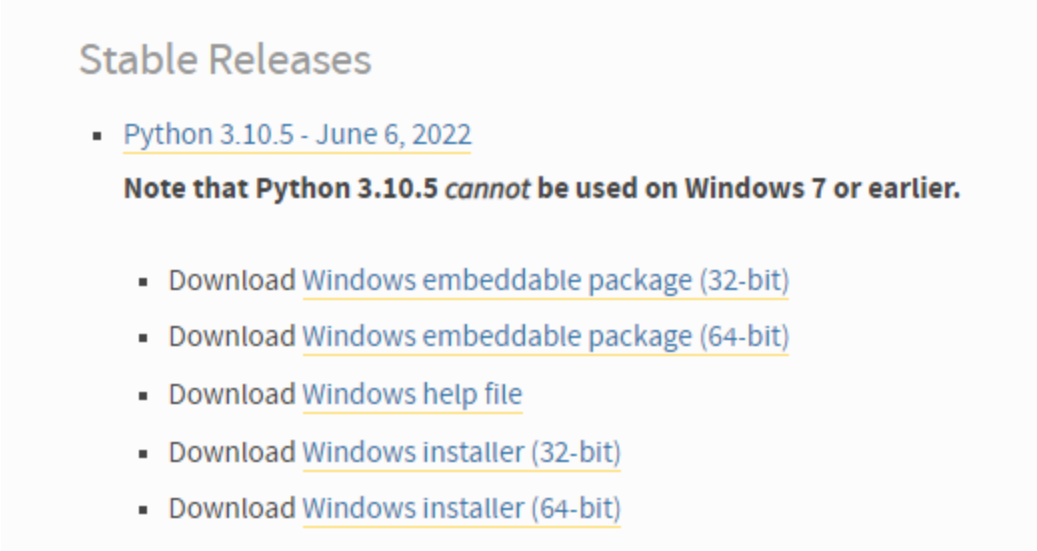


Figura 32. Listado de versiones y tipos de instaladores de Python.

Obtenido de: https://www.python.org/downloads/windows

iv. Una vez descargado ejecutaremos el instalador o los binarios según sea el caso y seguiremos la guía de instalación correspondiente que se encuentra dentro de ambos.

Instalación de herramienta pip

Para poder instalar todas las bibliotecas necesarias para ejecutar todo el proyecto, es necesario contar con la herramienta de manejo de dependencias de Python llamado Pip. Esta herramienta por lo general se incluye dentro de la instalación de Python, para verificar su correcta instalación introduciremos el siguiente comando en una terminal: pip --version.

Si la herramienta esta correctamente instalada obtendremos como resultado una respuesta como la que se muestra en la figura 33.

```
PS C:\Users\Carlos Miranda> pip --version
pip 22.1.1 from C:\Users\Carlos Miranda\Ap
```

Figura 33. Captura de terminal al ejecutar el comando pip --version.

En caso de obtener un error, primero es necesario revisar que el comando *pip* se encuentra dentro de nuestras variables de sistema. Si el caso es que el directorio de la herramienta no se encuentra dentro del directorio de Python ejecutaremos el siguiente comando dentro de este mismo:

```
python get-pip.py
```

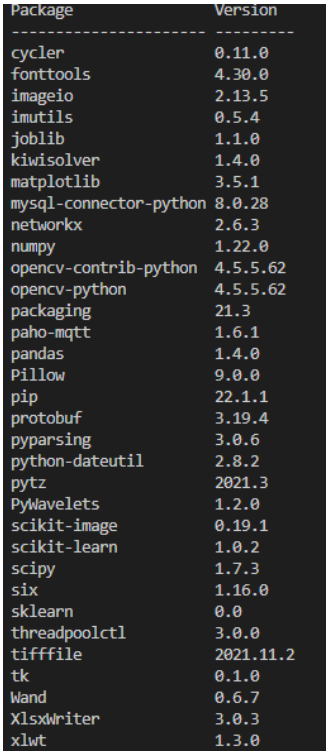
Para comprobar que nuestra herramienta de manejo de dependencias ha sido instalada correctamente volvemos a introducir el comando que se usó para verificar la versión de la herramienta.

Instalación de dependencias necesarias

Una vez que se ha instalado tanto Python como *pip*, ahora será necesario instalar todas las dependencias necesarias para poder ejecutar el proyecto. Para asegurar que las dependencias no generen errores se generó un documento con los valores de todas las dependencias, para ejecutarlo sólo es necesario posicionarse dentro del directorio del proyecto terminal y ejecutar el siguiente comando:

```
pip install -r requirements.txt
```

Finalmente, para verificar que las dependencias fueron instaladas de manera correcta, dentro de la terminal ejecutaremos el comando: *pip list* y deberíamos obtener un resultado igual al de figura 34.



Package	Version
cycler	0.11.0
fonttools	4.30.0
imageio	2.13.5
imutils	0.5.4
joblib	1.1.0
kiwisolver	1.4.0
matplotlib	3.5.1
mysql-connector-python	8.0.28
networkx	2.6.3
numpy	1.22.0
opencv-contrib-python	4.5.5.62
opencv-python	4.5.5.62
packaging	21.3
paho-mqtt	1.6.1
pandas	1.4.0
Pillow	9.0.0
pip	22.1.1
protobuf	3.19.4
pyparsing	3.0.6
python-dateutil	2.8.2
pytz	2021.3
PyWavelets	1.2.0
scikit-image	0.19.1
scikit-learn	1.0.2
scipy	1.7.3
six	1.16.0
sklearn	0.0
threadpoolctl	3.0.0
tifffile	2021.11.2
tk	0.1.0
Wand	0.6.7
XlsxWriter	3.0.3
xlwt	1.3.0

Figura 34. Listado obtenido al ejecutar el comando pip list.

Instalación de Docker

Docker es una herramienta de gestión de contenedores. La manera más sencilla para ver un contenedor es verlo como una instancia de una máquina virtual. A diferencia de las máquinas virtuales los contenedores poseen una serie de ventajas sustanciales dentro de las que destacan su peso, su velocidad al ser tan ligeros y la versatilidad de orquestar y manejar múltiples contenedores dentro de un equipo a diferencia de una máquina virtual. Dentro del proyecto se hace uso de una base de datos para el almacenamiento y la ubicación de las imágenes pertenecientes a la base de datos que se usará. Para poder ejecutar el proyecto es necesario tener una instancia de Docker arriba en la que se esté ejecutando una base de datos MySQL. A continuación, los pasos para la instalación tanto de Docker como de nuestra instancia.

i. Lo primero que hay que hacer es instalar Docker en nuestro equipo, para esto accederemos al sitio web oficial el cual se encuentra en la siguiente liga: https://www.docker.com/products/docker-desktop/
ii. Una vez dentro, seleccionaremos la versión de sistema operativo que estamos usando para bajar el instalador.

Figura 35. Página de descarga de instalador de Docker.
Obtenido de: https://www.docker.com/products/docker-desktop/

iii. Una vez descargado el instalador lo ejecutaremos y seguiremos los pasos de instalación dentro de este. Probablemente Docker pida permisos dentro de nuestro firewall por lo que

será necesario asignárselos ya que sin estos no nos podremos conectar a través de alguno de los puertos de nuestro equipo.

iv. Una vez instalado el software se abrirá y se desplegará una página tutorial. Podemos hacer caso omiso si ya se tiene un conocimiento previo, de lo contrario se recomienda leer la documentación o alguno de los videos proporcionados aquí mismo. Es importante recordar que Docker es completamente gratuito para fines no comerciales.

v. Lo siguiente que haremos será obtener nuestra imagen de MySQL para poder hacer uso de la base de datos, para eso seguiremos el comando dentro de la siguiente liga: https://hub.docker.com/_/mysql

vi. Una vez copiado, abriremos una instancia de nuestra terminal donde pegaremos el comando y la descarga de la imagen de MySQL comenzará.

vii. Para levantar una instancia de MySQL dentro de Docker conforme lo configurado dentro del proyecto, es necesario ingresar el siguiente comando en la terminal: `docker run –name terminalProject -e MYSQL_ROOT_PASSWORD=uamAzc2022! -d mysql:latest`

viii. Una vez iniciada la base de datos se puede usar algún gestor como DBeaver para conectarse con ella y será necesario crear una base de datos con el nombre *terminalProjectV2*.

ix. Cuando la base haya sido creada y la conectividad funcione de manera adecuada, se podrá ejecutar el archivo *modifyImagesV2.py* para el poblado y la generación de todas las imágenes.

modifyImagesV2.py

```
import sys
import os.path
import random
import cv2
from wand.image import Image
import mysql.connector
from mysql.connector import Error
import dbConnection

class Distortion:
    distortion = ''
    def __init__(self,distortion,arguments,distortionName):
        self.distortion = distortion
        self.arguments = arguments
        self.distortionName = distortionName

distortion = [Distortion('affine',[10,10,5,5,70,10,75,5,60,60,60,65],'affine1'),
Distortion('affine',[10,10,5,5,70,10,85,15,70,92,85,72],'affine2'),
Distortion('arc',[15,],'arc1'),
Distortion('arc',[18,],'arc2')]

# Esta clase se encarga de probar la conexion e insercion de datos al extraer
# todos los campos dentro de la base.
def testClass():
    for i in range(40):
        records = dbConnection.main(i+1)
        print("valores extraidos con el sujeto %s" % (i+1))
        for record in records:
            print(record)

# Este metodo se encarga de realizar la conexion, modificacion de imagenes y su
# insercion dentro de la base de datos.
try:
    #Se inicializa la conexion con la base de datos.
    connection =
mysql.connector.connect(host='localhost',database='terminalProjectV2',user='root',password='uamAzc2022!')
    if connection.is_connected():
        db_info = connection.get_server_info()
        print("Connected to MySQL Server version ", db_info)
        cursor = connection.cursor()
        # Se crean las tablas dentro de la base con su respectivas llaves.
        cursor.execute("CREATE TABLE IF NOT EXISTS terminalProjectV2.facesDatabase  (id int NOT NULL
AUTO_INCREMENT, subject VARCHAR(255), PRIMARY KEY(id));")
```

```
    cursor.execute("CREATE TABLE IF NOT EXISTS terminalProjectV2.imagesPath (id int NOT NULL
AUTO_INCREMENT, imagePath VARCHAR(255), subjectId int, modified BOOL, PRIMARY KEY(id));")
    print("Se ha generado de manera exitosa la tabla facesDatabase e imagesPath")
    for i in range(40):
        # Se inician las distorsiones mediante la iteracion de los 40 sujetos.
        print("Modificando sujeto " + str(i+1))
        path = "./archive/s" + str(i+1)
        cursor.execute("INSERT INTO terminalProjectV2.facesDatabase (subject) VALUES('%s');" % ("sujeto" +
str(i+1)))
        for modify in distortion:
            for j in range(10):
                img = random.randint(1,10)
                imgPath = path + "/" + str(img) + ".pgm"
                print(imgPath)
                with Image(filename=imgPath) as imageSelection:
                    # Se realiza la distorsion seleccionada en el momento junto con sus
                    # parametros
                    imageSelection.distort(modify.distortion, modify.arguments)
                    print("Modificando la imagen: " + str(img) + " con el metodo: " + modify.distortion)
                    saveName = "./archive/modifications/" + str(i+1) + "_distortion_" + modify.distortionName + "_" +
str(j+1) + ".pgm"
                    print("Guardando la imagen con en: " + saveName)
                    imageSelection.save(filename=saveName)
                    # Se realiza la insercion de la imagen modificada dentro de la base de datos.
                    cursor.execute("INSERT INTO terminalProjectV2.imagesPath (imagePath, subjectId, modified)
VALUES('%s',%s, TRUE);" % (saveName, str(i+1)))
    for i in range(40):
        print("Modificando sujeto " + str(i+1))
        path = "./archive/s" + str(i+1)
        # Este ciclo repetira los pasos del anterior pero realizando la transformacion de ampliacion.
        for j in range (10):
            imgPath = path + "/" + str(j+1) + ".pgm"
            print(imgPath)
            with Image(filename=imgPath) as imageSelection:
                imageSelection.transform('112x112','135%')
                imageSelection.crop(width=112, height=112, gravity='center')
                saveName = "./archive/modifications/" + str(i+1) + "_scale_" + str(j+1) + ".pgm"
                imageSelection.save(filename=saveName)
                cursor.execute("INSERT INTO terminalProjectV2.imagesPath (imagePath, subjectId, modified)
VALUES('%s',%s, TRUE);" % (saveName, str(i+1)))
        # Este ciclo de la misma manera repetira los ciclos anteriores aplicando cambios en el brillo.
        for j in range(10):
            imgPath = path + "/" + str(j+1) + ".pgm"
            with Image(filename=imgPath) as imageSelection:
                imageSelection.brightness_contrast(-30,10,'all_channels')
                saveName = "./archive/modifications/" + str(i+1) + "_brightness_" + str(j+1) + ".pgm"
                imageSelection.save(filename=saveName)
```

```
                cursor.execute("INSERT INTO terminalProjectV2.imagesPath (imagePath, subjectId, modified)
VALUES(\"%s\",%s, TRUE);" % (saveName, str(i+1)))
            for imageNumber in range(10):
                saveName = path + "/" + str(imageNumber + 1) + ".pgm"
                cursor.execute("INSERT INTO terminalProjectV2.imagesPath (imagePath, subjectId, modified)
VALUES('%s',%s, FALSE);" % (saveName, str(i+1)))
except Error as e:
    print("Error while connecting to MySQL", e)
finally:
    if connection.is_connected():
        connection.commit()
        cursor.close()
        connection.close()
        testClass()
        print("MySQL connection is closed")
```

Eigenfaces.py

```
from numpy.lib.npyio import load
import sklearn
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from skimage.exposure import rescale_intensity
from sklearn.metrics import accuracy_score
import numpy as np
import argparse
import imutils
import time
import cv2
import os
import loadingFaces

# Este metodo se encarga del manejo general del metodo, toma como argumentos
# el nombre o la ubicacion del archivo que se estara evaluando.
# Este metodo retorna las imagenes predecidas asi como la etiqueta ligada
# con la imagen que se predecide. Esto acompañado del tiempo de ejecucion.
def main(args, fileName):
    print("Inicializando deteccion con algoritmo Eigenfaces")
    # Se carga el conjunto de datos y sus etiquetas, respectivamente
```

```
(faces, labels) = loadingFaces.loadDataSet()
print("Las imagenes de la base de datos han sido cargadas")
pcaFaces = np.array([f.flatten() for f in faces])
print("Longitud de pcaFaces")
print(len(pcaFaces))
le = LabelEncoder()
labels = le.fit_transform(labels)
labels.sort()
# Se divide el conjunto de datos extraidos en subconjuntos para realizar el entrenamiento
# de nuestro algoritmo
split = train_test_split(faces, pcaFaces, labels, test_size=0.25, stratify=labels, random_state=42)
(origTrain, origTest, trainX, testX, trainY, testY) = split
print(testY)
print("Cantidad de imagenes en test %s" % (len(origTest)))
print("Construyendo eigenfaces")
pca = PCA(svd_solver="randomized", n_components=300, whiten=True)
start = time.time()
# Se realiza entrenamiento con subconjunto.
trainX = pca.fit_transform(trainX)
end = time.time()
print("El tiempo del calculo de eigenfaces tomo {:.4f} segundos".format(end-start))
images = []
print("Numero de componentes: %s Numero de features: %s" % (pca.n_components_,pca.n_features_))
for (i, component) in enumerate(pca.components_[:40]):
    component = component.reshape((112, 276))
    component = rescale_intensity(component, out_range=(0,255))
    component = np.dstack([component.astype("uint8")] * 3)
    images.append(component)

print("Numero de imagenes %s" % (len(images)))
display = imutils.build_montages(images, (56, 138), (10,4))[0]

media = pca.mean_.reshape((112, 276))
media = rescale_intensity(media, out_range=(0,255))

print("Iniciando entrenamiento de clasificador")
model = SVC(kernel="linear", C=10.0, gamma=0.001, random_state=42)
model2 = SVC(kernel="sigmoid", C=10.0, gamma=0.001, random_state=42)
model.fit(trainX,trainY)
model2.fit(trainX,trainY)
imageToEvaluate = []
# Se carga la imagen seleccionada por el usuario para su evaluacion
# y prediccion.
imageToEvaluate.append(cv2.imread(fileName))
imageToEvaluate = np.array([image.flatten() for image in imageToEvaluate])
imageToEvaluate.reshape(1,-1)
imageToEvaluate = imageToEvaluate[:, :300]
print("Forma de imagen a evaluar ", imageToEvaluate.shape)
```

```
    start = time.time()
    response = model.predict(imageToEvaluate)
    response2 = model2.predict(imageToEvaluate)
    end = time.time()
    # Se realiza la descomposicion de los valores obtenidos.
    print("Valor de respuesta %s" % (str(response[0])))
    responseName = le.inverse_transform(response)[0]
    responseName2 = le.inverse_transform(response2)[0]
    responseSubjectImage = cv2.imread("./archive/s%s/1.pgm" % (str(response[0])))
    responseSubjectImage2 = cv2.imread("./archive/s%s/2.pgm" % (str(response2[0])))
    responseImage = images[int(str(response[0]))]
    responseImage2 = images[int(str(response2[0]))]
    responseSubjectImageResized = cv2.resize(responseSubjectImage, (112,112))
    responseSubjectImageResized2 = cv2.resize(responseSubjectImage2, (112,112))
    responseImageResized = cv2.resize(responseImage, (112,112))
    responseImageResized2 = cv2.resize(responseImage2, (112,112))
    expectedImage = cv2.imread(fileName)
    expectedImage = cv2.resize(expectedImage, (112,112))

    print("prediccion: {}, valor esperado: {}".format(responseName, "Sujeto 1"))

    print("Evaluando el modelo")
    predicciones = model.predict(pca.transform(testX))
    print("Reporte de precision para conjunto sin alteraciones")

    resultingTime = end-start
    return (responseSubjectImageResized, resultingTime, ("Sujeto " + str(response[0])),
responseSubjectImageResized2, ("Sujeto " + str(response2[0])))

distortions = ['_brightness_','_distortion_affine1_', '_distortion_affine2_', '_distortion_arc1_', '_distortion_arc2_',
'_scale_']

# Este metodo se encarga de la realizacion de pruebas automatizadas asi como el registro
# y el reporte de resultados. No necesita ningun parametro de entrada puesto que las pruebas
# se realizan sobre todo el conjunto de datos generados. El proceso es el mismo que el principal,
# sin embargo, este carga junto con el conjunto normal los modificados.
# Este metodo solo retorna una lista con las predicciones realizadas y el calculo general de
# precisiones tanto por sujeto como por distorsion.
def testingMatrix():
    generalScores = []
    print("Inicializando deteccion con algoritmo Eigenfaces")
    (faces, labels) = loadingFaces.loadDataSet()
    pcaFaces = np.array([f.flatten() for f in faces])
    print("Las imagenes de la base de datos han sido cargadas")
    print(len(faces))
    le = LabelEncoder()
    labels = le.fit_transform(labels)
    labels.sort()
```

```
    split = train_test_split(faces, pcaFaces, labels, test_size=0.50, stratify=labels, random_state=42)
    (origTrain, origTest, trainX, testX, trainY, testY) = split
    pca = PCA(svd_solver="randomized", n_components=200, whiten=True)
    trainX = pca.fit_transform(trainX)
    for (i, component) in enumerate(pca.components_[:40]):
            component = component.reshape((112, 276))
            component = rescale_intensity(component, out_range=(0,255))
            component = np.dstack([component.astype("uint8")] * 3)
    model = SVC(kernel="linear", C=10.0, gamma=0.001, random_state=42)
    model.fit(trainX,trainY)
    print("Evaluando el modelo")
    predicciones = model.predict(pca.transform(pcaFaces))
    print("Precision sin alteraciones")
    print(accuracy_score(labels,predicciones))
    generalScores.append(accuracy_score(labels,predicciones))
    for distortion in distortions:
        (faces2, labels2) = loadingFaces.loadModifiedDataSetNew(distortion)
        print("Tamanio de caras cargadas %s" % (len(faces2)))
        pcaModiefiedFaces = np.array([f.flatten() for f in faces2])
        labels2 = le.fit_transform(labels2)
        labels2.sort()
        prediccionesMod = model.predict(pca.transform(pcaModiefiedFaces))
        print("Precision con distorsion %s" % (distortion))
        print(accuracy_score(labels2,prediccionesMod))
        generalScores.append(accuracy_score(labels2,prediccionesMod))
        for x in prediccionesMod:
            predicciones = np.append(predicciones, x)
    return generalScores, predicciones

if __name__ == '__main__':
    main()
```

knn.py

```
import cv2
import numpy as np
import pandas as pd
import os.path
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from skimage.exposure import rescale_intensity
from sklearn.metrics import accuracy_score
import time
import loadingFaces
```

```
from sklearn.preprocessing import LabelEncoder

fileName = "faceData.csv"
classifier = cv2.CascadeClassifier("./Tailored_cascade.xml")

# Este metodo se encarga del manejo general del algoritmo. Dentro de este se realiza la lectura
# asi como la deteccion de rostros para su procesamiento y almanacenamiento dentro de un archivo csv
# el cual es necesario para la clasificacion de las imagenes.
def main(args, fileName):
    for subject in range(40):
        subjectName = str(subject + 1)
        facesAsList = []
        print("Generando valores correspondientes a el sujeto %s" % (str(subject + 1)))
        print("Detectando rostros")
        for image in range(10):
            selection = cv2.imread("./archive/s%s/%s.pgm" % (str(subject+1),str(image+1)))
            # Se realiza la deteccion de rostros dentro de una imagen,
            # en caso de que sean varias, se cortan los rostros para ser
            # almacenados dentro de una lista.
            detectedFaces = classifier.detectMultiScale(selection, 1.5, 5)
            sorted(detectedFaces, key = lambda x: x[2]*x[3],reverse = True)
            if len(detectedFaces) == 1:
                x, y, w, h = detectedFaces[0]
                imageFrame = selection[y:y + h, x:x + w]
                imageToSave = cv2.resize(imageFrame, (100,100))
                facesAsList.append(imageToSave.reshape(-1))
            else:
                x, y, w, h = [0,0,92,112]
                imageFrame = selection[y:y + h, x:x + w]
                imageToSave = cv2.resize(imageFrame, (100,100))
                facesAsList.append(imageToSave.reshape(-1))
        print("Ordenando rostros detectados")
        print("Almacenando datos en archivo csv")
        # Se almacenan los datos propios de los rostros generados para poder usarlos dentro
        # del reconocimiento facial
        saveDataInCSV(subjectName, np.array(facesAsList))
    print("Iniciando reconocimiento facial")
    # Se manda a llamar al metodo encargado del reconocimiento para regresar todos los valores
    # relacionados con la prediccion por parte del algoritmo.
    return recognizeFace(fileName)

# Este metodo se encarga de almacenar los valores numericos junto con las etiquetas del sujeto.
# El archivo generado se usa para buscar valores descriptivos de cada sujeto con el fin de buscar
# los mas cercanos a los identificados dentro de la imagen ingresada.
def saveDataInCSV(name, data):
    if os.path.isfile(fileName):
        df = pd.read_csv(fileName, index_col=0)
        latest = pd.DataFrame(data, columns=map(str,range(30000)))
```

```
        latest["name"] = name
        df = pd.concat((df,latest), ignore_index=True, sort=False)
    else:
        df = pd.DataFrame(data, columns=map(str,range(30000)))
        df["name"] = name
    df.to_csv(fileName)

# Este metodo se encarga del reconocimiento de la imagen ingresada. Dentro de estas se usa la
# evaluacion con knn para poder comparar los valores generados en un inicio y los detectados
# dentro de una imagen. Este metodo retorna lo relacionado con el sujeto predecido, su etiqueta
# asi como la imagen identificada por parte del sujeto.
def recognizeFace(imageToRecognizePath):
    start = time.time()
    data = pd.read_csv(fileName).values
    X, Y = data[:, 1:-1], data[:, -1]
    print(X,Y)
    print("Generando modelo de clasificador K-NN")
    model = KNeighborsClassifier(n_neighbors=5)
    model2 = KNeighborsClassifier(n_neighbors=8)
    print("Ajustando modelo")
    model.fit(X,Y)
    model2.fit(X,Y)
    imageToEvaluate = cv2.imread(imageToRecognizePath)
    print("Detectando rostro en imagen proporcionada")
    facesInImage = classifier.detectMultiScale(imageToEvaluate, 1.5, 5)
    XTest = []
    if len(facesInImage) > 1:
        for face in facesInImage:
            x, y, w, h = face
            foundFace = imageToEvaluate[y:y + h, x:x + w]
            foundFace = cv2.resize(foundFace, (100,100))
            XTest.append(foundFace.reshape(-1))
        response = model.predict(np.array(XTest))
        response2 = model2.predict(np.array(XTest))
        for i, face in enumerate(facesInImage):
            x, y, w, h = face
    else:
        x, y, w, h = [0,0,92,112]
        foundFace = imageToEvaluate[y:y + h, x:x + w]
        foundFace = cv2.resize(foundFace, (100,100))
        XTest.append(foundFace.reshape(-1))
        response = model.predict(np.array(XTest))
        response2 = model2.predict(np.array(XTest))
    print("Sujeto predecido es: %s" % (response[0]))
    end = time.time()
    predictedImage = cv2.imread("./archive/s%s/1.pgm" % (response[0]))
    predictedImage2 = cv2.imread("./archive/s%s/2.pgm" % (response2[0]))
    imageToEvaluate = cv2.resize(imageToEvaluate, (112,112), interpolation = cv2.INTER_AREA)
```

```
    sideBySide = np.concatenate((imageToEvaluate, predictedImage), axis=1)
    cv2.putText(img=sideBySide, text="KNN", org=(0,0), fontFace=cv2.FONT_HERSHEY_TRIPLEX, fontScale=1,
color=(0, 255, 0),thickness=3)
    print("Eliminando archivo csv para actualizacion futura...")
    #os.remove("./%s" % (fileName))
    print("Eliminacion exitosa, cuando se vuelva a ejecutar el algoritmo se volvera a generar este archivo")

    return predictedImage, (end-start), response[0], predictedImage2, response2[0]

# Este metodo trabaja en conjunto junto con el metodo de pruebas automatizadas.
# Funciona de la misma manera que el anterior pero este se encarga especificamente de
# leer los datos a diferencia del anterior que se encarga de generarlos tambien.
def evaluatingPhoto(imageToEvaluate):
    start = time.time()
    data = pd.read_csv(fileName).values
    X, Y = data[:, 1:-1], data[:, -1]
    model = KNeighborsClassifier(n_neighbors=5)
    model.fit(X,Y)
    facesInImage = classifier.detectMultiScale(imageToEvaluate, 1.5, 5)
    XTest = []
    if len(facesInImage) > 1:
        for face in facesInImage:
            x, y, w, h = face
            foundFace = imageToEvaluate[y:y + h, x:x + w]
            foundFace = cv2.resize(foundFace, (100,100))
            XTest.append(foundFace.reshape(-1))
        response = model.predict(np.array(XTest))
        for i, face in enumerate(facesInImage):
            x, y, w, h = face
    else:
        x, y, w, h = [0,0,92,112]
        foundFace = imageToEvaluate[y:y + h, x:x + w]
        foundFace = cv2.resize(foundFace, (100,100))
        XTest.append(foundFace.reshape(-1))
        response = model.predict(np.array(XTest))
    return response[0]-1

distortions = ['_brightness_','_distortion_affine1_', '_distortion_affine2_', '_distortion_arc1_', '_distortion_arc2_',
'_scale_']

# Este metodo se encarga de la realizacion de pruebas. Este itera sobre todas las
# imagenes de la base de datos para poder generar las estadisticas relacionadas con
# la precision o rendimiento general de los algoritmos.
def testingMatrix():
    generalScores = []
    (faces, labels) = loadingFaces.loadDataSet()
    le = LabelEncoder()
    labels = le.fit_transform(labels)
```

```
    labels.sort()
    predicciones = []
    for face in faces:
        result = evaluatingPhoto(face)
        print("Imagen predecida %s" % (result))
        predicciones.append(result)
    print("Precision sin distorsion")
    print(accuracy_score(labels, predicciones))
    generalScores.append(accuracy_score(labels, predicciones))
    for distortion in distortions:
        (faces2, labels2) = loadingFaces.loadModifiedDataSetNew(distortion)
        labels2 = le.fit_transform(labels2)
        labels2.sort()
        prediccionesMod = []
        for face in faces2:
            result = evaluatingPhoto(face)
            print("Imagen predecida %s" % (result))
            prediccionesMod.append(result)
            predicciones.append(result)
        print("Precision con distorsion %s" % (distortion))
        print(accuracy_score(labels2, prediccionesMod))
        generalScores.append(accuracy_score(labels2, prediccionesMod))
    return generalScores, predicciones



if __name__ == '__main__':
    main()
```

SIFT.py

```
import cv2
import time
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
import loadingFaces

# Este metodo se encarga del manejo general del algoritmo. Toma como argumento la ubicacion de la
# imagen que se pretende evaluar. Dentro de este se inicializa nuestro identificador SIFT para
# retornar todos los valores relacionados con la imagen, esto incluye la imagen del sujeto que
# se esta prediciendo, asi como su etiqueta, tiempo de ejecucion, segunda imagen mejor calificado
# y su etiqueta.
def main(args, fileName):
    img1 = cv2.imread(fileName)
```

```
    # Se inicializa nuestro identificador SIFT
    sift = cv2.SIFT_create()
    keypoints_1, descriptors_1 = sift.detectAndCompute(img1, None)
    imagesMatches = []
    imageDescriptors = []
    imageKeypoints = []
    images = []
    start = time.time()
    imageIndex = 0
    # Se realiza la identificacion y almacenamiento de puntos descriptores de la base de datos
    for subject in range(40):
        for imageNumber in range(10):
            selectedImage = cv2.imread('./archive/s%s/%s.pgm' % (str(subject+1), str(imageNumber+1)))
            selectedImgKP, selectedImgDsp = sift.detectAndCompute(selectedImage, None)
            bf = cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)
            imageMatches = bf.match(descriptors_1, selectedImgDsp)
            imageMatches = sorted(imageMatches, key=lambda x:x.distance)
            images.append(selectedImage)
            imageKeypoints.append(selectedImgKP)
            imageDescriptors.append(selectedImgDsp)
            imagesMatches.append(imageMatches)

    matchedImage = -1
    secondMatchedImage = -1
    segundoSujeto = -1
    matchedImageMatches = -1
    sujeto = -1
    # Se realiza la comparacion de los puntos descriptores almacenados de la base de datos
    # y se va almacenando la mejor comparativa como nuestro resultado predecido.
    for imageNumberToEvalute in range(len(images)):
        if(len(imagesMatches[imageNumberToEvalute]) > matchedImageMatches):
            secondMatchedImage = matchedImage
            segundoSujeto = sujeto
            matchedImage = imageNumberToEvalute
            matchedImageMatches = len(imagesMatches[imageNumberToEvalute])
            sujeto = (matchedImage+1)//10

    print('La imagen %s contiene el mayor numero de coincidencias que es %s.' % (matchedImage,
matchedImageMatches))

    matches = imagesMatches[matchedImage]
    end = time.time()
    print("Tiempo de ejecucion para la deteccion del rostro %s" % (end-start))
    matched_img = np.concatenate((img1, images[matchedImage]), axis=1)
    sujetoTexto = "Sujeto " + str(sujeto+1)
    segundoSujetoTexto = "Sujeto " + str(segundoSujeto+1)
    return images[matchedImage], (end-start), sujetoTexto, images[secondMatchedImage], segundoSujetoTexto
```

```
# Este metodo es similar al principal, la unica diferencia es que este fue generado especificamente
# para las pruebas ya que esta maneja tambien la base de datos de imagenes distorsionadas. De la misma
# manera, este metodo solo retorna la etiqueta del sujeto predecido para poder almacenarlo en listas
# para las metricas.
def evaluatingPhoto(image):
    img1 = image
    sift = cv2.SIFT_create()
    keypoints_1, descriptors_1 = sift.detectAndCompute(img1, None)
    imagesMatches = []
    imageDescriptors = []
    imageKeypoints = []
    images = []
    start = time.time()
    imageIndex = 0
    for subject in range(40):
        for imageNumber in range(10):
            selectedImage = cv2.imread('./archive/s%s/%s.pgm' % (str(subject+1), str(imageNumber+1)))
            selectedImgKP, selectedImgDsp = sift.detectAndCompute(selectedImage, None)
            bf = cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)
            imageMatches = bf.match(descriptors_1, selectedImgDsp)
            imageMatches = sorted(imageMatches, key=lambda x:x.distance)
            images.append(selectedImage)
            imageKeypoints.append(selectedImgKP)
            imageDescriptors.append(selectedImgDsp)
            imagesMatches.append(imageMatches)
    matchedImage = -1
    matchedImageMatches = -1
    sujeto = -1
    for imageNumberToEvalute in range(len(images)):
        if(len(imagesMatches[imageNumberToEvalute]) > matchedImageMatches):
            matchedImage = imageNumberToEvalute
            matchedImageMatches = len(imagesMatches[imageNumberToEvalute])
            sujeto = (matchedImage+1)//10

    return sujeto

distortions = ['_brightness_','_distortion_affine1_', '_distortion_affine2_', '_distortion_arc1_', '_distortion_arc2_',
'_scale_']

# Este metodo se encarga de las pruebas sobre todas las bases de datos, tanto las originales como
# las modificadas. Como resultado el metodo retorna todas las metricas almacenadas las cuales son
# la precision, y el rendimiento general del algoritmo a manera de lista.
def testingMatrix():
    generalScores = []
    print("Iniciando evaluacion en matriz")
    (faces, labels) = loadingFaces.loadDataSet()
    le = LabelEncoder()
    labels = le.fit_transform(labels)
```

```
    labels.sort()
    predicciones = []
    for face in faces:
        result = evaluatingPhoto(face)
        print("Imagen predecida %s" % (result))
        predicciones.append(result)
    print("Precision sin distorsiones")
    print(accuracy_score(labels, predicciones))
    generalScores.append(accuracy_score(labels, predicciones))
    for distortion in distortions:
        (faces2, labels2) = loadingFaces.loadModifiedDataSetNew(distortion)
        labels2 = le.fit_transform(labels2)
        labels2.sort()
        prediccionesMod = []
        for face in faces2:
            result = evaluatingPhoto(face)
            print("Imagen predecida %s" % (result))
            prediccionesMod.append(result)
            predicciones.append(result)
        print("Precision con distorsion %s" % (distortion))
        print(accuracy_score(labels2, prediccionesMod))
        generalScores.append(accuracy_score(labels2, prediccionesMod))
    return generalScores, predicciones

if __name__ == '__main__':
    main()
```

guiv4.py

```
from email.mime import image
import tkinter as tk
from tkinter import ttk
from tkinter import *
from tkinter import filedialog
from PIL import Image, ImageTk
from os.path import exists
from os import remove
import matplotlib.pyplot as plt
import numpy as np
import STIF
import knn
import Eigenfaces
import cv2
import DetailedMatches
```

```
global knnSecImgRes
global knnSecSujeto
global eigenSecImageRes
global eigenSecSujeto
global stifSecondImage
global stifSegundoSujeto
global stifResImgPath
global stifSecResImgPath
global knnResImgPath
global eigenfacesResImgPath
global eigenSecResImgPath
eigenSecResImgPath = './eigenSecRes.png'
stifResImgPath = './stifResult.png'
stifSecResImgPath = './stifSecImg.png'
knnResImgPath = './knnResult.png'
knnSecResImgPath = './knnSecRes.png'
eigenfacesResImgPath = './eigenfacesResult.png'
global stifExecTime
global knnExecTime
global eigenfacesExecTime
stifExecTime = []
knnExecTime = []
eigenfacesExecTime = []
global eigenClassReport
eigenClassReport = ''

global stifExp
global eigenExp
global knnExp

stifExp = "STIF es un algoritmo diseñado\npara la detección de objetos\nsin importar su orientación o\npercepción dentro de una imagen,\npor lo que este algoritmo es\nsensible a los cambios de iluminación."
knnExp = "KNN es un algoritmo que obtiene los\npuntos clave dentro de una imagen,\nlos cuales son aquellos que\nla \"describen\" y despues los compara con\naquellos en cada imagen para determinar\nsi son lo mismo, es por esto que este\nalgoritmo es sensible a la iluminación,\norientación y tamaño del rostro."
eigenExp = "Eigenfaces genera vectores descriptores\npor cada conjunto de imagenes de sujeto\npara obtener de esta manera un rostro promedio,\nes por esto que este algoritmo es mas\nsusceptible a la iluminacion\ny orientacion del rostro."

# Este metodo se encarga de actualizar el recuadro de imagenes para la que haya sido seleccionada.
def uploadImage():
    global fileName
    fileName = filedialog.askopenfilename(initialdir="./", title="Seleccione la imagen a evaluar", filetypes=(("all files",
"*.*"),("pgm files", "*.pgm"),("png file","*.png")))
    root.picture_file = PhotoImage(file = fileName)
    root.imageToEvaluate.create_image(100, 100, anchor=CENTER, image=root.picture_file)
    root.imageToEvaluate.update()
    root.update()
```

```
# Este metodo se encarga de graficar los valores obtenidos de tiempo por cada uno de los metodos
# principales de cada algoritmo.
def plotValues():
    stifX = [x for x in range(len(stifExecTime))]
    stifY = stifExecTime
    knnX = [x for x in range(len(knnExecTime))]
    knnY = knnExecTime
    eigenX = [x for x in range(len(eigenfacesExecTime))]
    eigenY = eigenfacesExecTime
    plt.title("T de ejecucion")
    plt.plot(stifX, stifY)
    plt.plot(knnX, knnY)
    plt.plot(eigenX, eigenY)
    plt.savefig('./timeGraph.png')
    #plt.show()

# Este metodo se encarga de actualizar los recuadros de imagenes resultantes con cada uno de los
# algoritmos, de la misma manera se actualizan las etiquetas para poder agregar la etiqueta del
# sujeto que fue predecido en cada uno de los casos.
def updateResultImages():
    root.stifImage = PhotoImage(file = stifResImgPath)
    root.knnImage = PhotoImage(file = knnResImgPath)
    root.eigenfacesImage = PhotoImage(file = eigenfacesResImgPath)
    root.stifCanva.create_image(100, 100, anchor=CENTER, image=root.stifImage)
    root.stifCanva.update()
    root.update()
    root.knnCanva.create_image(100, 100, anchor=CENTER, image=root.knnImage)
    root.knnCanva.update()
    root.update()
    root.eigenfacesCanva.create_image(100, 100, anchor=CENTER, image=root.eigenfacesImage)
    root.eigenfacesCanva.update()
    root.update()

# Este metodo se manda a llamar cuando se presiona el boton de incio para evaluar la imagen seleccionada.
# De la misma manera este se encarga de proveer los valores necesarios como la ubicacion de la imagen para
# cada algoritmo.
def startEvaluation():
    print("Iniciando evaluacion")
    stifImageRes, stifExecutionTime, stifSujeto, stifSecondImage, stifSegundoSujeto = STIF.main(args="",
fileName=fileName)
    stifExecTime.append(stifExecutionTime)
    knnImageRes, knnExecutionTime, knnSujeto, knnSecImgRes, knnSecSujeto = knn.main(args="",
fileName=fileName)
    knnExecTime.append(knnExecutionTime)
    eigenfacesImageRes, eigenfacesExecutionTime, eigenSujeto, eigenSecImageRes, eigenSecSujeto =
Eigenfaces.main(args="", fileName=fileName)
    knnSujeto = "Sujeto " + str(knnSujeto)
```

```
    eigenfacesExecTime.append(eigenfacesExecutionTime)
    root.stifSubjectLbl.config(text=stifSujeto)
    root.knnSubjectLbl.config(text=knnSujeto)
    root.eigenSubjectLbl.config(text=eigenSujeto)
    eigenSubject.set(eigenSujeto)
    cv2.imwrite(stifResImgPath, stifImageRes)
    cv2.imwrite(stifSecResImgPath, stifSecondImage)
    cv2.imwrite(knnResImgPath, knnImageRes)
    cv2.imwrite(knnSecResImgPath, knnSecImgRes)
    cv2.imwrite(eigenfacesResImgPath, eigenfacesImageRes)
    cv2.imwrite(eigenSecResImgPath, eigenSecImageRes)
    updateResultImages()

# Este metodo se manda a llamar cuando se presiona el boton de mas detalles. Este se encarga de
# brindar mas detalles de los resultados haciendo modificaciones en las imagenes para poder comparar
# de manera visual lo que observa cada algoritmo.
def moreDetails():
    global detailsWindow
    detailsWindow = tk.Toplevel()
    detailsWindow.geometry('1200x800')
    detailsWindow.configure(background='#F0F8FF')
    detailsWindow.title('Mas informacion de resultados')
    stifResLbl = Label(detailsWindow, text='Resultado STIF', bg='#F0F8FF', font=('arial', 14, 'bold')).place(x=55,y=30)
    stifSubjectLbl = Label(detailsWindow, text='', bg='#F0F8FF', font=('arial', 14, 'bold')).place(x=55,y=630)
    detailsWindow.stifResCanva = Canvas(detailsWindow, height=200, width=200)
    detailsWindow.stifResCanva.place(x=55, y=50)
    detailsWindow.stifSecResCanva = Canvas(detailsWindow, height=200, width=200)
    detailsWindow.stifSecResCanva.place(x=55, y=300)
    stifSecResLbl = Label(detailsWindow, text='Segundo mejor resultado STIF', bg='#F0F8FF', font=('arial', 14,
'bold')).place(x=55,y=280)
    stifSecSubjectLbl = Label(detailsWindow, text='', bg='#F0F8FF', font=('arial', 14, 'bold')).place(x=55,y=510)
    knnResLbl = Label(detailsWindow, text='Resultado KNN', bg='#F0F8FF', font=('arial', 14,
'bold')).place(x=480,y=30)
    knnSubjectLbl = Label(detailsWindow, text='', bg='#F0F8FF', font=('arial', 14, 'bold')).place(x=55,y=630)
    knnSecResLbl = Label(detailsWindow, text='Segundo mejor resultado KNN', bg='#F0F8FF', font=('arial', 14,
'bold')).place(x=480,y=280)
    knnSecSubjectLbl = Label(detailsWindow, text='', bg='#F0F8FF', font=('arial', 14, 'bold')).place(x=55,y=630)
    detailsWindow.knnResCanva = Canvas(detailsWindow, height=200, width=200)
    detailsWindow.knnResCanva.place(x=480, y=50)
    detailsWindow.knnSecResCanva = Canvas(detailsWindow, height=200, width=200)
    detailsWindow.knnSecResCanva.place(x=480, y=300)
    eigenResLbl = Label(detailsWindow, text='Resultado Eigenfaces', bg='#F0F8FF', font=('arial', 14,
'bold')).place(x=905,y=30)
    eigenSubjectLbl = Label(detailsWindow, text='', bg='#F0F8FF', font=('arial', 14, 'bold')).place(x=55,y=630)
    eigenSecResLbl = Label(detailsWindow, text='Segundo mejor resultado Eigenfaces', bg='#F0F8FF', font=('arial',
14, 'bold')).place(x=905,y=280)
    eigenSecSubjectLbl = Label(detailsWindow, text='', bg='#F0F8FF', font=('arial', 14, 'bold')).place(x=55,y=630)
    detailsWindow.eigenResCanva = Canvas(detailsWindow, height=200, width=200)
```

```
    detailsWindow.eigenResCanva.place(x=905, y=50)
    detailsWindow.eigenSecResCanva = Canvas(detailsWindow, height=200, width=200)
    detailsWindow.eigenSecResCanva.place(x=905, y=300)
    print(eigenClassReport)
    #eigenClassRepLbl = Label(detailsWindow, text=, bg='#F0F8FF', font=('arial', 14, 'normal')).place(x=905,y=150)
    imageToEvaluate = cv2.imread(fileName)
    stifImage = cv2.imread(stifResImgPath)
    knnImage = cv2.imread(knnResImgPath)
    eigenfacesImage = cv2.imread(eigenfacesResImgPath)
    cv2.imwrite("./stifMatches.png", DetailedMatches.showDetailedMatches(imageToEvaluate, stifImage))
    cv2.imwrite("./knnMatches.png", DetailedMatches.showDetailedMatches(imageToEvaluate, knnImage))
    cv2.imwrite("./eigenMatches.png",DetailedMatches.showDetailedMatches(imageToEvaluate, eigenfacesImage))
    root.stifDetailsPicture = PhotoImage(file = 'stifMatches.png')
    root.stifSecPicture = PhotoImage(file = stifSecResImgPath)
    root.knnDetailsPicture = PhotoImage(file = 'knnMatches.png')
    root.knnSecDetailsPicture = PhotoImage(file = knnSecResImgPath)
    root.eigenDetailsPicture = PhotoImage(file = 'eigenMatches.png')
    root.eigenSecPicture = PhotoImage(file = eigenSecResImgPath)
    detailsWindow.stifResCanva.create_image(100,100,anchor=CENTER, image=root.stifDetailsPicture)
    detailsWindow.stifSecResCanva.create_image(100,100,anchor=CENTER, image=root.stifSecPicture)
    detailsWindow.knnResCanva.create_image(100, 100, anchor=CENTER, image=root.knnDetailsPicture)
    detailsWindow.knnSecResCanva.create_image(100, 100, anchor=CENTER, image=root.knnSecDetailsPicture)
    detailsWindow.eigenResCanva.create_image(100,100,anchor=CENTER, image=root.eigenDetailsPicture)
    detailsWindow.eigenSecResCanva.create_image(100,100,anchor=CENTER, image=root.eigenSecPicture)
    stifExpLbl = Label(detailsWindow, text=stifExp, bg='#F0F8FF', font=('arial', 12, 'normal')).place(x=40,y=550)
    knnExpLbl = Label(detailsWindow, text=knnExp, bg='#F0F8FF', font=('arial', 12, 'normal')).place(x=460,y=550)
    eigenExpLbl = Label(detailsWindow, text=eigenExp, bg='#F0F8FF', font=('arial', 12, 'normal')).place(x=880,y=550)
    detailsWindow.stifResCanva.update()
    detailsWindow.stifSecResCanva.update()
    detailsWindow.knnResCanva.update()
    detailsWindow.eigenResCanva.update()
    print("Getting more details")

# Dentro de esta seccion se realiza la ejecucion principal de la interfaz grafica.
# Aqui se especifica el tamaño de la ventana asi como todas las etiquetas y ventanas de imagen
# para el despliegue de la seleccion y de resultados.
global root
root = Tk()
root.geometry('1200x800')
root.configure(background='#F0F8FF')
root.title('Evaluacion de algoritmos')

# Aqui se colocan las etiquetas y botones tanto de inicio como de mas detalles de los resultados obtenidos.
title = Label(root, text='Sistema de evaluación de rendimiento de algoritmos de reconocimiento facial', bg='#F0F8FF',
font=('arial', 16, 'bold')).place(x=50, y=10)
author = Label(root, text="Desarrollado por Carlos Daniel Miranda Viloria", bg='#F0F8FF', font=('arial', 12,
'normal')).place(x=50, y=30)
```

```
instrucciones = Label(root, text="1.Seleccione una imagen para ser evaluada\n2.De Click en el boton de iniciar",
bg='#F0F8FF', font=('arial', 12, 'normal')).place(x=50, y=60)
seleccionImagenLbl = Label(root, text='Seleccione una imagen a evaluar', bg='#F0F8FF', font=('arial', 13,
'normal')).place(x=480, y=50)
seleccionImagen = Button(root, text='Seleccionar imagen', bg='#F0F8FF', font=('arial', 12, 'normal'),
command=uploadImage).place(x=480, y=70)
startButton = Button(root, text="Iniciar", bg='#F0F8FF', font=('arial', 14, 'bold'), command=(lambda :
startEvaluation())).place(x=555,y=330)

root.imageToEvaluate=Canvas(root, height=200, width=200)
root.imageToEvaluate.place(x=480, y=110)
root.resizable(True, True)

stifLabel = Label(root, text='Resultado STIF', bg='#F0F8FF', font=('arial', 14, 'bold')).place(x=55,y=390)
root.stifSubjectLbl = Label(root, text='', bg='#F0F8FF', font=('arial', 14, 'bold'))
root.stifSubjectLbl.place(x=55,y=630)
root.stifCanva = Canvas(root, height=200, width=200)
root.stifCanva.place(x=55, y=410)

knnLabel = Label(root, text='Resultado KNN', bg='#F0F8FF', font=('arial', 14, 'bold')).place(x=480,y=390)
root.knnSubjectLbl = Label(root, text='', bg='#F0F8FF', font=('arial', 14, 'bold'))
root.knnSubjectLbl.place(x=480,y=630)
root.knnCanva = Canvas(root, height=200, width=200)
root.knnCanva.place(x=480, y=410)

eigenSubject = StringVar('')
eigenfacesLabel = Label(root, text='Resultado Eigenfaces', bg='#F0F8FF', font=('arial', 14,
'bold')).place(x=905,y=390)
root.eigenSubjectLbl = Label(root, text='', bg='#F0F8FF', font=('arial', 14, 'bold'))
root.eigenSubjectLbl.place(x=905,y=630)
root.eigenfacesCanva = Canvas(root, height=200, width=200)
root.eigenfacesCanva.place(x=905, y=410)

detailsButton = Button(root, text="Mas detalles", bg='#F0F8FF', font=('arial', 14, 'bold'), command=(lambda :
moreDetails())).place(x=530,y=650)



root.mainloop()
```

DetailedMatches.py

```
import cv2
import time
import numpy as np

# Este metodo se encarga de realizar las relaciones visuales de cada una de las imagenes resultantes
```

```
# de los algoritmos ejecutados para brindar un resultado mas detallado y sencillo de comparar.
def showDetailedMatches(picture1, picture2):
    sift = cv2.SIFT_create()
    keypoints_1, descriptors_1 = sift.detectAndCompute(picture1, None)
    imagesMatches = []
    imageDescriptors = []
    imageKeypoints = []
    images = []
    keypoints_2, descriptors_2 = sift.detectAndCompute(picture2, None)
    bf = cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)
    imageMatches = bf.match(descriptors_1, descriptors_2)
    matched_img = cv2.drawMatches(picture1, keypoints_1, picture2, keypoints_2, imageMatches[:50], picture2,
flags=2)
    return matched_img


if __name__ == '__main__':
    showDetailedMatches()
```

testing.py

```
from sklearn.metrics import accuracy_score
import Sift
import Eigenfaces
import knn
import xlsxwriter
import time

distortions = ['Sujetos','Sin Distorsion','brightness(-30,10,all_channels)','affine1([10,10,5,5,70,10,75,5,60,60,60,65])',
'affine2([10,10,5,5,70,10,85,15,70,92,85,72])', 'arc1([15,])', 'arc2([18,])', 'scale(112x112,135%)']

lowerLimit = 0
upperLimit = 10
initTime = time.time()

# Este archivo se encarga de realizar la evaluacion con cada uno de los algoritmos,
# usa los metodos correspondientes para las pruebas automatizadas dentro de cada clase.
# De la misma manera este mismo se encarga de generar el archivo con el reporte general
# que contiene la precision asi como el rendimiento de manera general por cada uno de los algoritmos.

SiftGeneralScores, SiftSubjectScores = SIFT.testingMatrix()
print(SiftGeneralScores)

sujetosSift = []
prediccionesSift = []
for i in range(40):
    sujetosSift.append(['Sujeto ' + str(i+1)])
```

```
    prediccionesSift.append([''])

for (distortion, averageScore) in zip(distortions, SiftGeneralScores):
    for (i, sujeto, predicciones) in zip(range(40), sujetosSift, prediccionesSift):
        interval = SiftSubjectScores[lowerLimit:upperLimit]
        precisionSujeto = accuracy_score([i for j in range(10)], interval)
        print("Precision de sujeto %s es: %s" % (i+1, (precisionSujeto)))
        lowerLimit = upperLimit
        upperLimit += 10
        sujeto.append(precisionSujeto)
        stringToAdd = ''
        for(value) in interval:
            stringToAdd += str(value+1)
            stringToAdd += ','
        print(stringToAdd)
        predicciones.append(stringToAdd)
    print("Resultado promedio para distorsion %s es: %s" % (distortion, averageScore))
sujetosSift.append(['Promedio'] + SiftGeneralScores)

knnGeneralScores, knnSubjectScores = knn.testingMatrix()

sujetosKnn = []
prediccionesKnn = []
for i in range(40):
    sujetosKnn.append(['Sujeto ' + str(i+1)])
    prediccionesKnn.append([''])

lowerLimit = 0
upperLimit = 10

for (distortion, averageScore) in zip(distortions, knnGeneralScores):
    for (i, sujeto, predicciones) in zip(range(40), sujetosKnn, prediccionesKnn):
        interval = knnSubjectScores[lowerLimit:upperLimit]
        precisionSujeto = accuracy_score([i for j in range(10)], interval)
        print("Precision de sujeto %s es: %s" % (i+1, (precisionSujeto)))
        lowerLimit = upperLimit
        upperLimit += 10
        sujeto.append(precisionSujeto)
        stringToAdd = ''
        for(value) in interval:
            stringToAdd += str(value+1)
            stringToAdd += ','
        print(stringToAdd)
        predicciones.append(stringToAdd)
    print("Resultado promedio para distorsion %s es: %s" % (distortion, averageScore))
sujetosKnn.append(['Promedio'] + knnGeneralScores)
```

```
eigenfacesGeneralScores, eigenfacesSubjectScores = Eigenfaces.testingMatrix()
print(len(eigenfacesSubjectScores))

endTime = time.time()
print("Tiempo de evaluacion %s" % (endTime-initTime))

sujetosEigenfaces = []
prediccionesEigenfaces = []
for i in range(40):
    sujetosEigenfaces.append(['Sujeto ' + str(i+1)])
    prediccionesEigenfaces.append([''])

print(sujetosEigenfaces)

lowerLimit = 0
upperLimit = 10

for (distortion, averageScore) in zip(distortions, eigenfacesGeneralScores):
    for (i, sujeto, predicciones) in zip(range(40), sujetosEigenfaces, prediccionesEigenfaces):
        interval = eigenfacesSubjectScores[lowerLimit:upperLimit]
        precisionSujeto = accuracy_score([i for j in range(10)], interval)
        print("Precision de sujeto %s es: %s" % (i+1, (precisionSujeto)))
        lowerLimit = upperLimit
        upperLimit += 10
        sujeto.append(precisionSujeto)
        stringToAdd = ''
        for(value) in interval:
            stringToAdd += str(value+1)
            stringToAdd += ','
        print(stringToAdd)
        predicciones.append(stringToAdd)
    print("Resultado promedio para distorsion %s es: %s" % (distortion, averageScore))
sujetosEigenfaces.append(['Promedio'] + eigenfacesGeneralScores)

for sujeto in sujetosEigenfaces:
    print("Precisiones %s" % (sujeto))

# Dentro de esta seccion se crea el archivos xlsx con las precisiones por sujeto por cada algoritmos,
# asi como la precision general. De manera adicional y con el objetivo de brindar mas claridad a las
# pruebas se agregan hojas con las predicciones realizadas por cada algoritmo por cada sujeto.

workbook = xlsxwriter.Workbook('Resultados.xlsx')
worksheetSift = workbook.add_worksheet('Sift')
worksheetKnn = workbook.add_worksheet('KNN')
worksheetEigenfaces = workbook.add_worksheet('Eigenfaces')
worksheetPredSift = workbook.add_worksheet('Predicciones Sift')
worksheetPredKnn = workbook.add_worksheet('Predicciones KNN')
worksheetPredEigenfaces = workbook.add_worksheet('Predicciones Eigenfaces')
```

```
row = 1
column = 0

for distortion in distortions:
    worksheetSift.write(row, column, distortion)
    worksheetKnn.write(row, column, distortion)
    worksheetEigenfaces.write(row, column, distortion)
    worksheetPredSift.write(row, column, distortion)
    worksheetPredKnn.write(row, column, distortion)
    worksheetPredEigenfaces.write(row, column, distortion)
    column += 1

for (scoreSift, scoreKnn, scoreEigen) in zip(sujetosSift, sujetosKnn,sujetosEigenfaces):
    column = 0
    row += 1
    for (precisionSift, precisionKnn, precisionEigen) in zip(scoreSift, scoreKnn, scoreEigen):
        worksheetSift.write(row, column, precisionSift)
        worksheetKnn.write(row, column, precisionKnn)
        worksheetEigenfaces.write(row, column, precisionEigen)
        column += 1

column = 0
row = 1

for i in range(40):
    column = 0
    row += 1
    worksheetPredSift.write(row, column, 'Sujeto ' + str(i+1))
    worksheetPredKnn.write(row, column, 'Sujeto ' + str(i+1))
    worksheetPredEigenfaces.write(row, column, 'Sujeto ' + str(i+1))

row = 1

print("Agregando predicciones a excel")

for (prediccionSift, prediccionKnn,prediccionEigen) in zip(prediccionesSift,prediccionesKnn,prediccionesEigenfaces):
    column = 0
    row += 1
    print("Predicciones Eigenfaces")
    for(valSift, valKnn, valEigen) in zip(prediccionSift, prediccionKnn,prediccionEigen):
        worksheetPredSift.write(row, column, valSift[:-1])
        worksheetPredKnn.write(row, column, valKnn[:-1])
        worksheetPredEigenfaces.write(row, column, valEigen[:-1])
        column += 1

workbook.close()
```

Bibliografía

[1] Y. Moses, Y. Adini, and S. Ullman, "Face Recognition: The Problem of Compensating for Changes in Illumination Direction," European Conf. Computer Vision, 1994, pp. 286-296.

[2] J. Tanaka, and D. Simonyi, "The "parts and wholes" of face recognition: a review of the literature," Q J Exp Psychol (Hove), 2016 October; 69(10): 1876–1889, Oct, 2017, doi:10.1080/17470218.2016.1146780.

[3] (2022) OpenCV. "Introduction" [En línea]. Disponible: https://docs.opencv.org/4.6.0/d1/dfb/intro.html

[4] I. Jollife. and J. Cadima, "Principal component analysis: a review and recent developments". The Royal Society Publishing, 2016, doi: https://doi.org/10.1098/rsta.2015.0202

[5] C. Franco & C. Ospina & E. Cuevas & D. Capacho. "Reconocimiento facial basado en eigenfaces, lbhp y fisherfaces en la Beagleboard-xm". Revista colombiana de tecnologías de avanzada (rcta), 2017. 2. 10.24054/16927257.v26.n26.2015.2387.

[6] (2022) OpenCV. "Introduction to SIFT (Scale-Invariant Feature Transform)" [En línea]. Disponible: https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html

[7] (2022) OpenCV. "Understanding k-Nearest Neighbor" [En línea]. Disponible: https://docs.opencv.org/4.x/d5/d26/tutorial_py_knn_understanding.html

[8] (2022) Wand. [En línea]. Disponible: https://docs.wand-py.org/en/0.6.7/

[9] The Database of Faces, AT&T, 2002. [En línea]. Disponible: https://www.kaggle.com/datasets/kasikrit/att-database-of-faces

[10] (2022) Python. "Python Interface to Tcl/TK" [En línea]. Disponible: https://docs.python.org/es/3/library/tkinter.html

[11] (2022) Matplotlib. "Matplotlib: Visualization with Python" [En línea]. Disponible: https://matplotlib.org/

[12] (2022) NumPy. "NumPy Documentation" [En línea]. Disponible: https://numpy.org/doc/stable/

[13] (2022) ScikitLearn. "About Us" [En línea]. Disponible: https://scikit-learn.org/stable/about.html

[14] (2022) Pandas. "User Guide" [En línea]. Disponible: https://pandas.pydata.org/docs/user_guide/index.html

[15] (2022) Docker. “What is a container?” [En línea]. Disponible: https://www.docker.com/resources/what-container/

[16] (2022) A. Ahmadi. “Cascade Trainer GUI” [En línea]. Disponible: https://amin-ahmadi.com/cascade-trainer-gui/